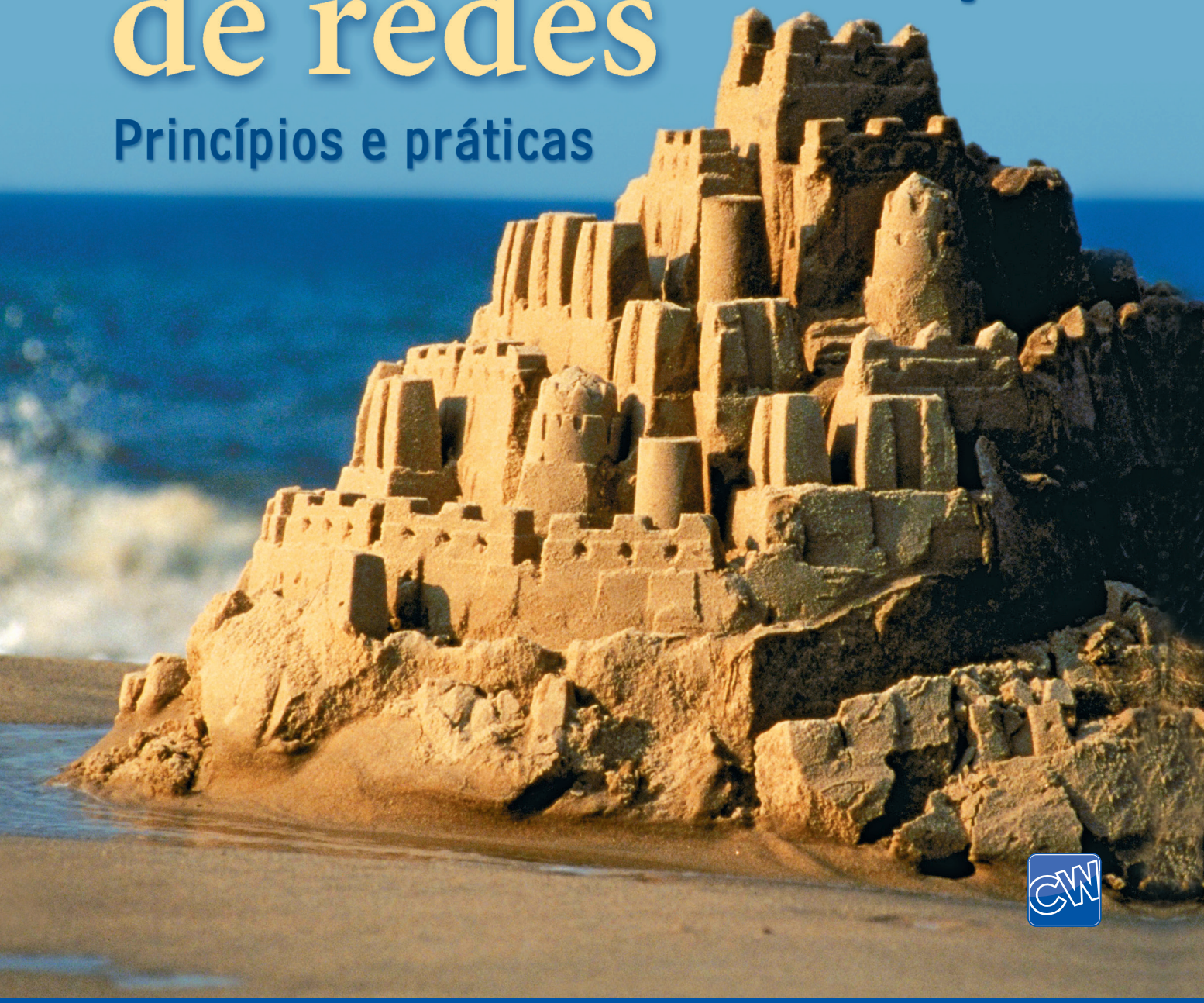
William Stallings

Criptografia e segurança de redes

4ª Edição

Princípios e práticas



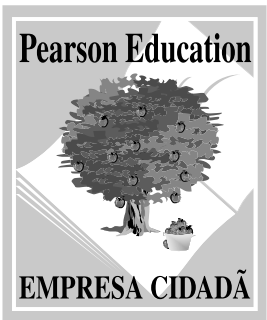
ALWAYS LEARNING

PEARSON

Criptografia e segurança de redes

4ª edição

Princípios e práticas



William Stallings

Criptografia e segurança de redes

4ª edição

Princípios e práticas

Tradução:
Daniel Vieira

Revisão Técnica:
Engenheiro Ákio Nogueira Barbosa
Mestre em Engenharia Elétrica pela Escola
Politécnica da Universidade de São Paulo

Engenheiro Marcelo Succi de Jesus Ferreira
Mestre em Engenharia Elétrica pela Escola
Politécnica da Universidade de São Paulo

PEARSON

abdr
ASSOCIAÇÃO BRASILEIRA DE DIREITOS REPROGRÁFICOS
Respeite o direito autoral

© 2008, Pearson Education do Brasil
© 2006 Pearson Education, Inc.

Tradução autorizada a partir da edição original em inglês, publicada pela Pearson Education, Inc. sob o selo Prentice Hall. Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Pearson Education do Brasil.

Gerente editorial: Roger Trimer
Editora sênior: Sabrina Cairo
Editora de desenvolvimento: Marco Pace
Editoras de texto: Eugênia Pessotti
Preparação: Paulo Roberto de Miguel
Revisão: Maria Aiko Nishijima e Sandra Scapin
Capa: Celso Blanes (sobre projeto original de Bruce Kenselaar)
Editoração Eletrônica: ERJ Composição Editorial e Artes Gráficas Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Stallings, William
Criptografia e segurança de redes / William Stallings; tradução Daniel Vieira; revisão técnica Ákio Barbosa e Marcelo Succi. – 4. ed. – São Paulo: Pearson Prentice Hall, 2008.

Título original: Criptography and Networking Security 4/E

ISBN 978-85-7605-119-0

1. Computadores – Segurança 2. Criptografia 3. Dados – Proteção I. Barbosa, Akio. II. Succi, Marcelo. III. Título.

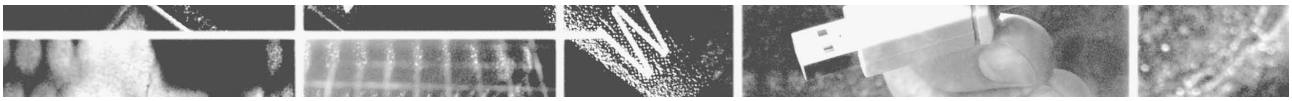
07-9195 CDD-005.82

Índices para catálogo sistemático:

1. Criptografia : Dados : Segurança : Computadores: Processamento de dados
005.82

6ª reimpressão – setembro 2013
Direitos exclusivos para a língua portuguesa cedidos à
Pearson Education do Brasil Ltda.,
uma empresa do grupo Pearson Education
Rua Nelson Francisco, 26
CEP 02712-100 – São Paulo – SP – Brasil
Fone: 11 2178-8686 – Fax: 11 2178-8688
vendas@pearson.com

Para Antígona.
nunca grosseiro,
nunca chato,
sempre um Sábio.



SUMÁRIO

Prefácio **XIII**

Guia do Leitor **XV**

1 Conteúdo deste livro XVI

2 Organização do conteúdo do livro XVI

3 Internet e recursos da Web XVII

Notação **1**

Capítulo 1 – Introdução **2**

1.1 Tendências de segurança 4

1.2 A arquitetura de segurança OSI 5

1.3 Ataques à segurança 6

1.4 Serviços de segurança 8

1.5 Mecanismos de segurança 10

1.6 Um modelo para segurança de rede 12

1.7 Leitura e sites Web recomendados 13

1.8 Principais termos, perguntas de revisão e problemas 14

PARTE UM **CIFRAS SIMÉTRICAS** **15**

Capítulo 2 Técnicas clássicas de criptografia **17**

2.1 Modelo de cifra simétrica 18

2.2 Técnicas de substituição 22

2.3 Técnicas de transposição 32

2.4 Máquinas de rotor 33

2.5 Esteganografia 34

2.6 Leitura e sites Web recomendados 35

2.7 Principais termos, perguntas de revisão e problemas 36

Capítulo 3 – Cifras de bloco e o Data Encryption Standard **40**

3.1 Princípios de cifra de bloco 41

3.2 O Data Encryption Standard 46

3.3 A força do DES 54

3.4 Criptoanálise diferencial e linear 56
3.5 Princípios de projeto de cifra de bloco 58
3.6 Leitura recomendada 60
3.7 Principais termos, perguntas de revisão e problemas 61

Capítulo 4 – Corpos finitos 64
4.1 Grupos, anéis e corpos 65
4.2 Aritmética modular 67
4.3 O algoritmo euclidiano 72
4.4 Corpos finitos na forma GF(p) 73
4.5 Aritmética polinomial 76
4.6 Corpos finitos na forma GF(2^n) 80
4.7 Leitura e sites Web recomendados 88
4.8 Principais termos, perguntas de revisão e problemas 88

Capítulo 5 Advanced Encryption Standard 91
5.1 Critérios de avaliação para AES 92
5.2 A cifra AES 95
5.3 Leitura e sites Web recomendados 110
5.4 Principais termos, perguntas de revisão e problemas 111
APÊNDICE 5A Polinômios com coeficiente em GF(2^8) 112
APÊNDICE 5B AES simplificado 114

Capítulo 6 – Mais sobre cifras simétricas 121
6.1 Criptografia múltipla e DES triplo 122
6.2 Modos de operação de cifra de bloco 126
6.3 Cifras de fluxo e RC4 132
6.4 Leitura e sites Web recomendados 135
6.5 Principais termos, perguntas de revisão e problemas 135

Capítulo 7 – Confidencialidade usando criptografia simétrica 139
7.1 Posicionamento da função de criptografia 140
7.2 Confidencialidade do tráfego 145
7.3 Distribuição de chaves 146
7.4 Geração de números aleatórios 152
7.5 Leitura e sites Web recomendados 157
7.6 Principais termos, perguntas de revisão e problemas 158

PARTE DOIS CRIPTOGRAFIA DE CHAVE PÚBLICA E FUNÇÕES DE HASH 163

Capítulo 8 Introdução à teoria dos números 165
8.1 Números primos 166

8.2 Teoremas de Fermat e Euler 168
8.3 Testando se o número é primo 171
8.4 O teorema chinês do resto 173
8.5 Logaritmos discretos 175
8.6 Leitura e sites Web recomendados 179
8.7 Principais termos, perguntas de revisão e problemas 179

Capítulo 9 – Criptografia de chave pública e RSA 181
9.1 Princípios de criptossistemas de chave pública 182
9.2 O Algoritmo RSA 188
9.3 Leitura e sites Web recomendados 198
9.4 Principais termos, perguntas de revisão e problemas 198
Apêndice 9A Prova do algoritmo RSA 202
Apêndice 9B A complexidade dos algoritmos 203

Capítulo 10 – Gerenciamento de chaves; outros criptosistemas de chave pública 205
10.1 Gerenciamento de chaves 206
10.2 Acordo de chaves Diffie-Hellman 211
10.3 Aritmética de curva elíptica 214
10.4 Criptografia de curva elíptica 221
10.5 Leitura e sites Web recomendados 223
10.6 Principais termos, perguntas de revisão e problemas 223

Capítulo 11 – Autenticação de mensagem e funções de hash 226
11.1 Requisitos de autenticação 227
11.2 Funções de autenticação 227
11.3 Códigos de autenticação de mensagem 236
11.4 Funções de hash 238
11.5 Segurança das funções de hash e MACs 243
11.6 Leitura recomendada 245
11.7 Principais termos, perguntas de revisão e problemas 245
Apêndice 11A Base matemática do ataque do aniversário 247

Capítulo 12 – Algoritmos de hash e de MAC 252
12.1 Algoritmo de hash seguro 253
12.2 Whirlpool 257
12.3 HMAC 265
12.4 CMAC 268
12.5 Leitura e sites Web recomendados 269
12.6 Principais termos, perguntas de revisão e problemas 269

Capítulo 13 – Assinaturas digitais e protocolos de autenticação **272**
13.1 Assinaturas digitais 273
13.2 Protocolos de autenticação 275
13.3 Padrão de assinatura digital 280
13.4 Leitura e sites Web recomendados 283
13.5 Principais termos, perguntas de revisão e problemas 283

PARTE TRÊS APLICAÇÕES DE SEGURANÇA DE REDE **287**

Capítulo 14 – Aplicações de autenticação **289**
14.1 Kerberos 290
14.2 Serviço de autenticação X.509 302
14.3 Infra-estrutura de chave pública 309
14.4 Leitura e sites Web recomendados 310
14.5 Principais termos, perguntas de revisão e problemas 311
Apêndice 14A Técnicas de criptografia do Kerberos 312

Capítulo 15 – Segurança de e-mail **315**
15.1 Pretty Good Privacy 316
15.2 S/MIME 330
15.3 Principais termos, perguntas de revisão e problemas 342
Apêndice 15A Compressão de dados usando ZIP 342
Apêndice 15B Conversão radix64 344
Apêndice 15C Geração de números aleatórios PGP 345

Capítulo 16 – Segurança de IP **348**
16.1 Visão geral da segurança de IP 349
16.2 Arquitetura da segurança de IP 350
16.3 Cabeçalho de autenticação 355
16.4 Encapsulamento de segurança do payload 358
16.5 Combinando associações de segurança 361
16.6 Gerenciamento de chaves 363
16.7 Leitura e sites Web recomendados 370
16.8 Principais termos, perguntas de revisão e problemas 371
Apêndice 16A Interligação de redes e protocolos da Internet 371

Capítulo 17 – Segurança na Web **378**
17.1 Considerações sobre segurança na Web 379
17.2 Secure Socket Layer e Transport Layer Security 381
17.3 Secure Electronic Transaction 393

17.4 Leitura e sites Web recomendados 401
17.5 Principais termos, perguntas de revisão e problemas 401

PARTE QUATRO SEGURANÇA DO SISTEMA 403

Capítulo 18 – Intrusos 405
18.1 Intrusos 406
18.2 Detecção de intrusão 408
18.3 Gerenciamento de senhas 416
18.4 Leitura e sites Web recomendados 422
18.5 Principais termos, perguntas de revisão e problemas 423
Apêndice 18A O mito da probabilidade de base 424

Capítulo 19 – Software malicioso 427
19.1 Vírus e ameaças relacionadas 428
19.2 Contramedidas para vírus 435
19.3 Ataques de negação de serviço distribuídos 438
19.4 Leitura e sites Web recomendados 441
19.5 Principais termos, perguntas de revisão e problemas 442

Capítulo 20 – Firewalls 443
20.1 Princípios de projeto de firewall 444
20.2 Sistemas confiáveis (trusted systems) 451
20.3 Critérios comuns para avaliação de segurança da tecnologia da informação 455
20.4 Leitura e sites Web recomendados 458
20.5 Principais termos, perguntas de revisão e problemas 459

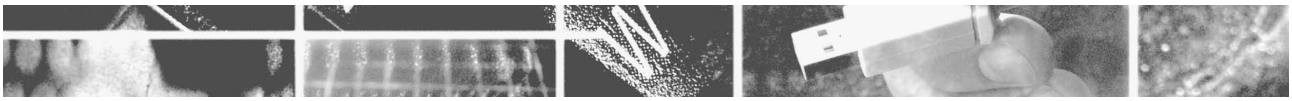
Apêndice – A Padrões e organizações de definição de padrões 461
A.1 A importância dos padrões 461
A.2 Padrões da Internet e a Internet Society 461
A.3 National Institute of Standards and Technology 464

Apêndice B – Projetos para ensino de criptografia e segurança de rede 465
B.1 Projetos de pesquisa 465
B.2 Projetos de programação 465
B.3 Exercícios de laboratório 466
B.4 Tarefas escritas 466
B.5 Tarefas de leitura/relatórios 466

Glossário 467

Referências 471

Índice remissivo 481



PREFÁCIO

"O nó, se eu pudesse sugerir, senhor, um pouco mais amarrado. É preciso dar o efeito borboleta perfeito. Se me permite."
"O que importa, Jeeves, em uma época como esta? Você não vê que a felicidade doméstica de Mr. Little está em risco?"
"Não haverá nunca, senhor, o tempo, em que os nós não importam."

—Very good, Jeeves! P. G. Wodehouse

Nesta época de conectividade eletrônica universal, de vírus e hackers, de espionagem eletrônica e fraude eletrônica, não há realmente um tempo em que a segurança não importa. Duas tendências se juntam para tornar o assunto deste livro de interesse vital. Primeiro, o crescimento explosivo dos sistemas de computador e suas interconexões através de redes tem aumentado a dependência de organizações e indivíduos da informação armazenada e comunicada por meio desses sistemas. Isso, por sua vez, tem levado a uma maior conscientização da necessidade de proteger dados e recursos contra divulgação, de garantir a autenticidade de dados e mensagens, e de proteger sistemas contra ataques baseados em redes. Em segundo lugar, as disciplinas de criptografia e segurança de redes amadureceram e levaram ao desenvolvimento de aplicações práticas e prontamente disponíveis para impor a segurança da rede.

Objetivos

O objetivo deste livro é oferecer um estudo conciso dos princípios e da prática da criptografia e da segurança de redes. Nas duas primeiras partes, as questões básicas a serem focalizadas por uma capacidade de segurança de redes são exploradas oferecendo-se um tutorial e um estudo da tecnologia de criptografia e segurança de redes. A última parte do livro trata da prática de segurança de redes: aplicações concretas que têm sido implementadas e estão sendo usadas para oferecer segurança de redes.

O assunto (e, portanto, este livro) se baseia em diversas disciplinas. Em particular, é impossível apreciar o significado de algumas das técnicas discutidas aqui sem um conhecimento básico da teoria dos números e alguns resultados da teoria da probabilidade. Apesar disso, tentamos tornar o livro uma obra estanque, que oferece ao leitor não apenas os resultados matemáticos básicos necessários, mas também um conhecimento intuitivo desses resultados. Esse material de base é apresentado conforme a necessidade. Essa técnica ajuda a motivar o material que é introduzido, e o autor considera isso preferível a simplesmente apresentar todo o material matemático de uma só vez no início do livro.

Público-alvo

Este livro é voltado para os estudantes universitários e também para profissionais da área. Como livro-texto, pode ser usado na disciplina de criptografia e segurança de redes dos cursos de graduação em ciência da computação, engenharia de sistemas e engenharia elétrica.

Também pode ser considerado bibliografia básica sobre o tema e é recomendado para estudo por autodidatas da área.

Sobre o livro

Criptografia e segurança de redes está organizado em quatro partes:

Parte Um — Criptografia convencional: apresenta um exame detalhado dos algoritmos de criptografia convencionais e os princípios de projeto, incluindo uma discussão sobre o uso da criptografia convencional para confidencialidade.

Parte Dois — Criptografia de chave pública e funções de hash: traz um exame detalhado dos algoritmos e os princípios de projeto da criptografia de chave pública. Essa parte também examina o uso de códigos de autenticação de mensagem e funções de hash, além de assinaturas digitais e certificados de chave pública.

Parte Três — Prática de segurança de rede: abrange ferramentas e aplicações importantes de segurança de redes, incluindo Kerberos, certificados X.509v3, PGP, S/MIME, IP Security, SSL/TLS e SET.

Parte Quatro — Segurança do sistema: examina as questões de segurança de sistemas — incluindo as ameaças e as contramedidas para intrusos e vírus —, e o uso de firewalls e sistemas confiáveis.

Além disso, o livro inclui um extenso glossário e, ao final de cada capítulo, uma lista de sugestões de problemas, perguntas de revisão, uma relação de de palavras-chave e sugestões de leitura adicional e sites Web recomendados.

No início de cada parte, o leitor encontra um pequeno resumo de cada capítulo a ela pertencente.

Companion Website

No site www.pearson.com.br/stallings, são oferecidos materiais de apoio aos professores e alunos. Os recursos incluem: apresentações em PowerPoint, links para sites úteis, respostas dos exercícios, biblioteca, no formato PDF (Adobe Acrobat), de figuras e tabelas do livro, sugestões de projeto e apêndices adicionais (em inglês). Além disso, no site Web do autor (http://WilliamStallings.com/StudentSupport.html), também estão disponíveis, em inglês, outros documentos e informações complementares.

Projetos para o ensino de criptografia e segurança de redes

Para muitos professores, um componente importante de um curso de criptografia ou segurança é um projeto, ou conjunto de projetos, por meio do qual o aluno seja capaz de pôr em prática os conceitos do aprendidos. Este livro oferece um leque de recursos de apoio sem igual para incluir um componente de projeto no curso. Na área do site de apoio reservada ao professor, estão disponíveis orientações sobre como estruturar e implementar os projetos e também um conjunto de projetos sugeridos que aborda uma grande variedade de tópicos, entre eles:

- Projetos de pesquisa: uma série de trabalhos de pesquisa que instrui o aluno a pesquisar um assunto específico na Internet e escrever um relatório.
- Projetos de programação: uma série de projetos de programação que aborda uma grande variedade de tópicos. Esses projetos podem ser implementados em uma linguagem adequada em qualquer plataforma.
- Exercícios de laboratório: um conjunto de projetos que envolvem conhecimento de programação e familiaridade por parte do aluno com os conceitos apresentados no livro.
- Tarefas escritas: para cada capítulo do livro, é oferecido um conjunto de sugestões de tarefas escritas.
- Tarefas de leitura/relalórios: uma lista de artigos, um para cada capítulo, que podem ser designados para o aluno ler e depois escrever um pequeno relatório a respeito.

Tópicos em destaque

Recomendamos aos professores e alunos atenção especial aos tópicos a seguir, que refletem as últimas atualizações na área:

- AES simplificado: essa é uma versão do AES (Advanced Encryption Standard), que permite as alunos entenderem com maior facilidade os fundamentos do AES.
- Whirlpool: um novo algoritmo de hash seguro, baseado no uso de uma cifra de bloco simétrica.
- CMAC: um novo modo de operação de cifra de bloco. O CMAC (Cipher-based Message Authentication Code) oferece autenticação de mensagem com base no uso de uma cifra de bloco simétrica.
- Public-Key Infrastructure (PKI): um assunto atual e de fundamental importância. Veja mais informações na Seção 14.3.
- Ataques de negação de serviço distribuídos (DDoS): ataques de DDoS assumiram maior significado nos últimos anos.
- Critérios comuns para avaliação de segurança da tecnologia da informação: os Common Criteria tornaram-se a estrutura internacional para expressar requisitos de segurança e avaliar produtos e implementações.
- Apêndices on-line: no site do livro, seis apêndices adicionais, em inglês, complementam o material de apoio.

Agradecimentos

As seguintes pessoas revisaram todo ou grande parte do manuscrito: Danny Krizanc (Wesleyan University), Breno de Medeiros (Florida State University), Roger H. Brown (Rensselaer em Hartford), Cristina Nita-Rotarul (Purdue University) e Jimmy McGibney (Waterford Institute of Technology).

Agradecemos também a muitas pessoas que ofereceram revisões técnicas detalhadas de um único capítulo: Richard Outerbridge, Jorge Nakahara, Jeroen van de Graaf, Philip Moseley, Andre Correa, Brian Bowling, James Muir, Andrew Holt, Décio Luiz Gazzoni Filho, Lucas Ferreira, Dr. Kemal Bicakci, Routo Terada, Anton Stiglic, Valery Pryamikov e Yongge Wang.

Joan Daemen gentilmente revisou o capítulo sobre AES. Vincent Rijmen revisou o material sobre Whirlpool. E Edward F. Schaefer revisou o material sobre AES simplificado.

As seguintes pessoas contribuíram com os 'trabalhos para casa': Joshua Brandon Holden (Rose-Hulman Institute if Technology), Kris Gaj (George Mason University) e James Muir (University of Waterloo).

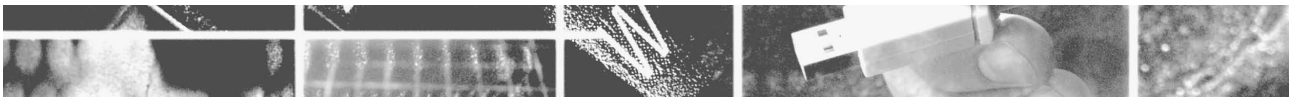
Sanjay Rao e Ruben Torres, da Purdue University, desenvolveram os exercícios de laboratório que estão disponíveis como material de apoio ao professor. As seguintes pessoas contribuíram com trabalhos de projeto também disponíveis como material de apoio ao professor: Henning Schulzrinne (Columbia University); Cetin Kaya Koc (Oregon State University); e David Balenson (Trusted Information Systems e George Washington University).

Finalmente, gostaria de agradecer a todas as pessoas envolvidas na publicação do livro, responsáveis por um excelente trabalho. Isso inclui o pessoal da Prentice Hall, particularmente a gerente de produção Rose Kernan; minha gerente de suplementos Sarah Parker; e minha nova revisora Tracy Dunkelberger. Além disso, Patricia M. Daly realizou a revisão de textos.

Com toda essa assistência, pouca coisa ficou para o meu próprio crédito. Porém, tenho o orgulho de dizer que, sem qualquer ajuda, eu mesmo selecionei todas as citações.

Agradecimentos dos editores brasileiros

Agradecemos à professora Graça Bressan, da Escola Politéca da Universidade de São Paulo, por sua colaboração em várias etapas no processo de desenvolvimento desta obra.



GUIA DO LEITOR

1 **Conteúdo deste livro**

2 **Organização do conteúdo do livro**
- Assunto
- Ordem dos tópicos

3 **Internet e recursos da Web**
- Sites Web deste livro
- Outros sites Web
- Grupos de discussão da Usenet

A arte da guerra nos ensina a contar não com a probabilidade de o inimigo não chegar, mas com nossa própria prontidão para recebê-lo; não com a chance de não ser atacado, mas com o fato de tornar nossa posição inatacável.

A arte da guerra, Sun Tzu

Este livro e o site Web que o acompanha dispõem de muito material. Nesta seção, o leitor poderá ter uma idéia geral.

1 CONTEÚDO DESTE LIVRO

Após um capítulo introdutório, o Capítulo 1, o livro é organizado em quatro partes:

Parte Um — Cifras simétricas: Oferece um estudo da codificação simétrica, incluindo algoritmos clássicos e modernos. A ênfase está nos algoritmos mais importantes, o Data Encryption Standard (DES) e o Advanced Encryption Standard (AES). Essa parte também trata da autenticação de mensagens e gerenciamento de chaves.
Parte Dois — Criptografia de chave pública e funções de hash: Oferece um estudo sobre algoritmos de chave pública, incluindo RSA (Rivest-Shamir-Adelman) e curva elíptica. Também abrange aplicações de chave pública, incluindo assinaturas digitais e trocas de chave.
Parte Três — Aplicações de segurança de rede: Examina o uso de algoritmos criptográficos e protocolos de segurança para oferecer segurança nas redes e na Internet. Os tópicos abordados incluem autenticação do usuário, e-mail, segurança de IP e segurança na Web.
Parte Quatro — Segurança do sistema: Trata dos recursos de segurança criados para proteger um sistema de computador contra ameaças à segurança, incluindo intrusos, vírus e vermes. Essa parte também examina a tecnologia de firewall.

Muitos dos algoritmos criptográficos e protocolos e aplicações de segurança de rede descritos neste livro foram especificados como padrões. Os mais importantes são os padrões Internet, definidos nas RFCs (Request for Comments) da Internet, e Federal Information Processing Standards (FIPS), emitidos pelo National Institute of Standards and Technology (NIST). O Apêndice A explica o processo de criação de padrões e lista os padrões citados neste livro.

2 ORGANIZAÇÃO DO CONTEÚDO DO LIVRO

Assunto

O material neste livro é organizado em três categorias gerais:

- **Criptologia:** Este é o estudo das técnicas para garantir o sigilo e/ou a autenticidade da informação. Os dois ramos principais da criptologia são a criptografia, que é o estudo do projeto dessas técnicas; e a criptoanálise, que trata das formas de reverter essas técnicas, recuperar informações ou forjar informações que serão aceitas como autênticas.
- **Segurança de rede:** Esta área abrange o uso dos algoritmos de criptografia nos protocolos de rede e aplicações de rede.
- **Segurança de computador:** Neste livro, usamos esse termo para nos referir à segurança dos computadores contra intrusos (por exemplo, hackers) e software malicioso (por exemplo, vírus). Normalmente, o computador a ser protegido está conectado a uma rede e as maiores ameaças surgem dela.

As duas primeiras partes do livro tratam de duas técnicas criptográficas distintas: algoritmos criptográficos simétricos e algoritmos criptográficos com chave pública, ou assimétricos. Os algoritmos simétricos utilizam uma única chave, compartilhada por duas partes interessadas. Os algoritmos com chave pública utilizam duas chaves: uma chave privada, conhecida apenas por uma das partes, e uma chave pública, disponível a outras partes.

Ordem dos tópicos

Este livro contém um material bastante extenso. Para o instrutor ou leitor que deseja uma abordagem mais resumida, há diversas oportunidades.

Para abordar de modo mais completo o material das duas primeiras partes, os capítulos deverão ser lidos em seqüência. Com a exceção do Advanced Encryption Standard (AES), nenhum material na **Parte Um** exige base matemática especial. Para entender o AES, é preciso algum conhecimento sobre corpos finitos. Por sua vez, um conhecimento sobre corpos finitos exige uma base sobre números primos e aritmética modular. Em conseqüência disso, o Capítulo 4 abrange todos os requisitos matemáticos imediatamente antes de seu uso no Capítulo 5, sobre AES. Assim, se o Capítulo 5 for ignorado, o Capítulo 4 também poderá ser ignorado sem comprometer o entendimento dos demais capítulos da Parte Um.

O Capítulo 2 apresenta alguns conceitos que serão úteis em capítulos posteriores da Parte Um. Porém, para o leitor cujo único interesse é a criptografia contemporânea, esse capítulo pode ser apenas examinado superficialmente. Os dois algoritmos criptográficos simétricos mais importantes são DES e AES, abordados nos capítulos 3 e 5, respectivamente. O Capítulo 6 abrange dois outros algoritmos interessantes, ambos com uso comercial. Esse capítulo pode ser ignorado se esses algoritmos não forem de interesse do leitor.

Para a **Parte Dois**, a única base matemática adicional necessária está na área de teoria dos números, que é abordada no Capítulo 8. O leitor que tiver ignorado os capítulos 4 e 5 deverá primeiro rever o material das seções 4.1 a 4.3.

Os dois algoritmos de chave pública para uso geral mais utilizados são RSA e curva elíptica, sendo que o RSA obteve muito mais aceitação. O leitor poderá ignorar o material sobre criptografia com curva elíptica no Capítulo 10, pelo menos em uma primeira leitura. No Capítulo 12, Whirlpool e CMAC têm menor importância.

A **Parte Três** e a **Parte Quatro** são relativamente independentes uma da outra, e podem ser lidas em qualquer ordem. Ambas as partes presumem um conhecimento básico do material contido nas Partes Um e Dois.

3 INTERNET E RECURSOS DA WEB

Existem inúmeros recursos disponíveis na Internet e na Web para dar suporte a este livro e ajudá-lo a manter-se atualizado com os desenvolvimentos nessa área.

Sites Web deste livro

Uma página Web específica foi preparada para este livro, em **WilliamStallings.com/Crypto/Crypto4e.html**. O site inclui o seguinte:

- **Sites Web úteis:** Existem links para outros sites Web relevantes, organizados por capítulo, incluindo os sites listados nesta seção e no decorrer deste livro.
- **Página de errata:** Uma lista de errata para este livro será mantida e atualizada conforme a necessidade. Por favor, envie quaisquer erros que você localizar para mim por e-mail. As páginas de errata referentes aos meus outros livros estão em **WilliamStallings.com**.
- **Figuras:** Todas as figuras deste livro em formato PDF (Adobe Acrobat).
- **Tabelas:** Todas as tabelas deste livro em formato PDF.
- **Slides:** Um conjunto de slides do PowerPoint, organizados por capítulo.
- **Cursos de criptografia e segurança de redes:** Há links para as home pages de cursos baseados neste livro; essas páginas podem ser úteis a outros instrutores, oferecendo idéias sobre como estruturar seus cursos.

Também mantenho um site de recursos para alunos de ciência da computação, em **WilliamStallings.com /StudentSupport.html**. A finalidade desse site é oferecer documentos, informações e links para alunos e profissionais de ciência da computação. Links e documentos são organizados em quatro categorias:

- **Matemática:** Inclui uma revisão da matemática básica, um manual de análise de filas, um manual de sistemas numéricos e links para diversos sites de matemática
- **Como fazer:** Recomendações e orientação para solucionar trabalhos de casa, redigir relatórios técnicos e preparar apresentações técnicas.
- **Fontes de pesquisa:** Links para coleções importantes de artigos, relatórios técnicos e bibliografias.
- **Diversos:** Uma série de outros documentos e links úteis.

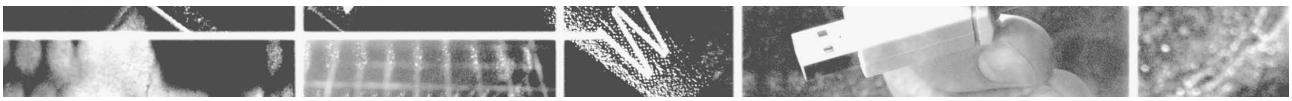
Outros sites Web

Existem muitos sites Web que oferecem informações relacionadas aos assuntos deste livro. Nos capítulos subseqüentes podem ser encontrados indicadores para sites Web específicos na seção "Leitura e sites Web recomendados". Como os endereços dos sites Web costumam mudar com freqüência, não incluí URLs neste livro. O link apropriado para todos os sites Web listados poderá ser encontrado no site Web deste livro. Ao longo do tempo outros links não mencionados neste livro serão acrescentados ao site Web.

Grupos de discussão da Usenet

Diversos grupos de discussão da Usenet são dedicados a algum aspecto da criptografia ou segurança de redes. Assim como em praticamente todos os grupos da Usenet, existe uma alta relação ruído—sinal, mas vale a pena experimentar e ver se alguns deles atendem as suas necessidades. Os mais relevantes são

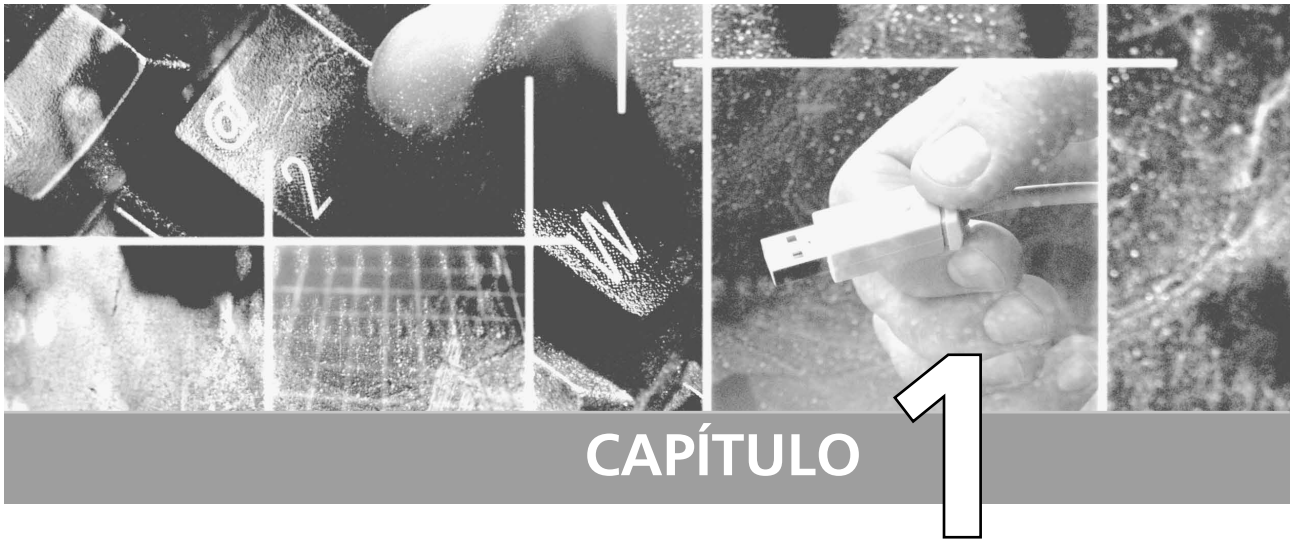
- **sci.crypt.research:** O melhor grupo para acompanhar. Esse é um grupo de discussão com moderador, que trata de assuntos de pesquisa; os artigos precisam ter alguma relação com os aspectos técnicos da criptologia.
- **sci.crypt:** Uma discussão geral da criptologia e assuntos relacionados.
- **sci.crypt.random-numbers:** Uma discussão sobre geradores de números aleatórios com importância em criptografia.
- **alt.security:** Uma discussão geral sobre assuntos de segurança.
- **comp.security.misc:** Uma discussão geral sobre assuntos de segurança de computador.
- **comp.security.firewalls:** Uma discussão geral sobre produtos e tecnologia de firewall.
- **comp.security.announce:** Notícias, anúncios da Cert.
- **comp.risks:** Uma discussão sobre riscos ao público por parte de computadores e usuários.
- **comp.virus:** Uma discussão sobre vírus de computador, com moderador.



NOTAÇÃO

Até mesmo os nativos têm dificuldade para dominar esse vocabulário peculiar.
—*The Golden Bough,* Sir James George Frazer

Símbolo	Expressão	Significado
D, K	D(K,Y)	Decriptografia simétrica do texto cifrado Y usando a chave secreta K.
D, PR_a	D(PR_a, Y)	Decriptografia assimétrica do texto cifrado Y usando a chave privada de A
D, PU_a	D(PU_a, Y)	Decriptografia assimétrica do texto cifrado Y usando a chave pública de A
E, K	E(K, X)	Criptografia simétrica do texto claro X usando a chave secreta K.
E, PR_a	E(PR_a, X)	Criptografia assimétrica do texto claro X usando a chave privada de A
E, PU_a	E(PU_a, X)	Criptografia assimétrica do texto claro X usando a chave pública de A
K		Chave secreta
PR_a		Chave privada do usuário A
PU_a		Chave pública do usuário A
C, K	C(K, X)	Código de autenticação de mensagem da mensagem X usando a chave secreta K.
GF(p)		O campo finito de ordem p, onde p é primo. O campo é definido como o conjunto Z_p junto com as operações aritméticas módulo p.
GF(2^n)		O campo finito de ordem 2^n.
Z_n		Conjunto de inteiros não negativos menores que n.
gcd	gcd(i, j)	Máximo divisor comum; o maior inteiro positivo que divide tanto i quanto j sem resto na divisão.
mod	a mod m	Resto após a divisão de a por m.
mod, $\equiv$	$a \equiv b \pmod{m}$	$a \bmod m = b \bmod m$
mod, $\not\equiv$	$a \not\equiv b \pmod{m}$	$a \bmod m \neq b \bmod m$
dlog	$\text{dlog}_{a,p}(b)$	Logaritmo discreto do número b para a base a (mod p)
ϕ	$\phi(n)$	O número de inteiros positivos menores que n e relativamente primos de n. Essa é a função totiente de Euler.
Σ	$\sum_{i=1}^{n} a_i$	$a_1 + a_2 + \cdots + a_n$
Π	$\prod_{i=1}^{n} a_i$	$a_1 \times a_2 \times \cdots \times a_n$
$\mid$	$i \mid j$	i divide j, o que significa que não existe resto quando j é dividido por i
$\mid, \mid$	$\lvert a \rvert$	Valor absoluto de a
$\parallel$	$x \parallel y$	x concatenado com y
$\approx$	$x \approx y$	x é aproximadamente igual a y
$\oplus$	$x \oplus y$	OR exclusivo de x e y para variáveis de único bit; OR exclusivo bit a bit de x e y para variáveis de múltiplos bits.
$\lfloor, \rfloor$	$\lfloor x \rfloor$	O maior inteiro menor ou igual a x
$\in$	$x \in$ S	O elemento x está contido no conjunto S.
$\leftrightarrow$	$A \leftrightarrow (a_1, a_2, \ldots, a_k)$	O inteiro A corresponde à seqüência de inteiros $(a_1, a_2, \ldots, a_k)$



CAPÍTULO 1

Introdução

1.1 **Tendências de segurança**

1.2 **A arquitetura de segurança OSI**

1.3 **Ataques à segurança**

1.4 **Serviços de segurança**

1.5 **Mecanismos de segurança**

1.6 **Um modelo para segurança de rede**

1.7 **Leitura e sites Web recomendados**

1.8 **Principais termos, perguntas de revisão e problemas**

A combinação de espaço, tempo e força que precisam ser considerados como os elementos básicos dessa teoria de defesa torna essa questão bastante complicada. Conseqüentemente, não é fácil encontrar um ponto fixo de partida

Da guerra, Carl von Clausewitz

PONTOS PRINCIPAIS

- A arquitetura de segurança OSI (Open Systems Interconnection) oferece uma estrutura sistemática para definir ataques à segurança, mecanismos e serviços.
- **Ataques à segurança** são classificados como ataques passivos, que incluem leitura não autorizada de uma mensagem de arquivo e análise de tráfego; e ataques ativos, como modificação de mensagens ou arquivos e negação de serviço.
- **Um mecanismo de segurança** é qualquer processo (ou um dispositivo incorporando tal processo) projetado para detectar, impedir ou permitir a recuperação de um ataque à segurança. Alguns exemplos de mecanismos são algoritmos de criptografia, assinaturas digitais e protocolos de autenticação.
- **Serviços de segurança** incluem autenticação, controle de acesso, confidencialidade de dados, integridade de dados, irretratabilidade e disponibilidade.

Os requisitos de **segurança da informação** dentro de uma organização passaram por duas mudanças importantes nas últimas décadas. Antes do uso generalizado de equipamentos de processamento de dados, a segurança da informação considerada valiosa para uma organização era fornecida principalmente por meios físicos e administrativos. Um exemplo do primeiro é o uso de robustos armários para arquivo com uma fechadura de segredo para armazenar documentos confidenciais. Um exemplo do último são os procedimentos de seleção de pessoal usados durante o processo de contratação.

Com a introdução do computador, tornou-se evidente a necessidade de ferramentas automatizadas para proteger arquivos e outras informações armazenadas no computador. Esse é especialmente o caso de um sistema compartilhado, como um sistema de time-sharing, e a necessidade é ainda mais premente para sistemas que podem ser acessados por meio de uma rede telefônica pública, rede de dados ou a Internet. O nome genérico para o conjunto de ferramentas projetadas para proteger dados e impedir hackers é **segurança de computador**.

A segunda mudança importante que afetou a segurança é a introdução de sistemas distribuídos e o uso de redes e recursos de comunicação para transmitir dados entre o usuário do terminal e o computador e entre computadores. As medidas de segurança de rede são necessárias para proteger dados durante sua transmissão. Na verdade, o termo **segurança de rede** é, de certa forma, incorreto, pois praticamente todas as empresas, governo e organizações acadêmicas interconectam seus equipamentos de processamento de dados com um conjunto de redes interconectadas. Esse conjunto normalmente é conhecido como inter-rede,[1] utilizando-se o termo **segurança de inter-rede**.

Não existem limites claros entre essas duas formas de segurança. Por exemplo, um dos tipos mais notórios de ataque a sistemas de informação é o vírus de computador. Um vírus pode ser introduzido fisicamente em um sistema ao chegar em um disquete ou disco óptico e ser carregado em um computador. Os vírus também podem chegar por uma inter-rede. De qualquer forma, após o vírus se instalar em um sistema de computador, são necessárias ferramentas de segurança internas do computador para detectar e recuperar-se do vírus.

Este livro enfoca a segurança de inter-rede, que consiste em medidas para desencorajar, impedir, detectar e corrigir violações de segurança que envolvam a transmissão de informações. Essa é uma afirmação ampla, que abrange inúmeras possibilidades. Para dar uma idéia das áreas abordadas neste livro, considere os seguintes exemplos de violações de segurança:

1. O usuário A transmite um arquivo ao usuário B. O arquivo contém informações confidenciais (por exemplo, registros de folha de pagamento) que devem ser protegidas contra divulgação. O usuário C, que não está autorizado a ler o arquivo, é capaz de monitorar a transmissão e obter uma cópia do arquivo durante sua transmissão.
2. Um gerente de rede, D, transmite uma mensagem a um computador E sob seu gerenciamento. A mensagem instrui o computador E a atualizar um arquivo de autorização para incluir as identidades de diversos novos usuários que deverão receber acesso a esse computador. O usuário F intercepta a mensagem, altera seu conteúdo para incluir ou excluir entradas e, depois, encaminha a mensagem para E, que aceita a mensagem como se tivesse vindo do gerente D e atualiza seu arquivo de autorização conforme solicitado.
3. Em vez de interceptar uma mensagem, o usuário F cria sua própria mensagem com as entradas desejadas e transmite essa mensagem para E como se tivesse vindo do gerente D. O computador E aceita a mensagem como vindo do gerente D e atualiza seu arquivo de autorização conforme solicitado.
4. Um funcionário é demitido sem aviso. O gerente de pessoal envia uma mensagem a um sistema servidor para invalidar a conta do funcionário. Quando a invalidação é realizada, o servidor deve enviar um aviso ao arquivo do funcionário como confirmação da ação. O funcionário é capaz de interceptar a mensagem e adiá-la pelo tempo necessário para fazer um último acesso ao servidor e obter informações confidenciais. A mensagem é então encaminhada, a ação tomada e a confirmação postada. A ação do funcionário pode passar despercebida por um tempo considerável.
5. Uma mensagem é enviada de um cliente a uma corretora com instruções para diversas transações. Depois disso, os investimentos perdem valor e o cliente nega ter enviado a mensagem.

Embora essa lista não esgote de forma alguma os tipos possíveis de violações de segurança, ela ilustra a gama de problemas de segurança de rede.

A segurança de inter-redes é fascinante e complexa. Alguns dos motivos são:

1. A segurança envolvendo comunicações e redes não é tão simples quanto pode parecer a princípio para o iniciante. Os requisitos parecem ser simples; na verdade, a maioria dos principais requisitos para serviços de segurança pode receber rótulos auto-explicativos de uma palavra: confidencialidade, autenticação, irretratabilidade, integridade. Mas os mecanismos usados para atender a esses requisitos podem ser muito complexos, e entendê-los pode envolver um raciocínio bastante sutil.

1. Usamos o termo *inter-rede* para nos referirmos a qualquer conjunto de redes interconectadas. Uma intranet corporativa é um exemplo de uma inter-rede. A Internet pode ser um dos recursos usados por uma organização para construir sua inter-rede.

2. Ao desenvolver um mecanismo ou algoritmo de segurança específico, é preciso considerar sempre os ataques em potencial a esses recursos de segurança. Em muitos casos, os ataques bem-sucedidos são planejados examinando-se o problema de um modo completamente diferente, explorando assim um ponto fraco inesperado no mecanismo.
3. Devido ao ponto 2, os procedimentos usados para fornecer serviços específicos normalmente não são intuitivos: não é óbvio, pela declaração de um determinado requisito, que essas medidas elaboradas sejam necessárias. É somente quando as diversas contramedidas são consideradas que as medidas utilizadas fazem sentido.
4. Tendo projetado diversos mecanismos de segurança, é preciso decidir onde usá-los. Isso é verdadeiro tanto em termos de posicionamento físico (por exemplo, em que pontos de uma rede certos mecanismos de segurança são necessários) quanto em um sentido lógico (por exemplo, em que camada ou camadas de uma arquitetura como TCP/IP — Transmission Control Protocol/Internet Protocol — os mecanismos devem ser colocados].
5. Mecanismos de segurança normalmente envolvem mais do que um algoritmo ou protocolo específico. Eles geralmente também exigem que os participantes possuam alguma informação secreta (por exemplo, uma chave de criptografia), que levanta questões sobre a criação, distribuição e proteção dessa informação secreta. Há também uma confiança em protocolos de comunicação cujo comportamento pode complicar a tarefa de desenvolver o mecanismo de segurança. Por exemplo, se o funcionamento apropriado do mecanismo de segurança exigir a definição de limites para o tempo de trânsito de uma mensagem do emissor ao receptor, então qualquer protocolo ou rede que introduza atrasos variáveis e imprevisíveis pode tornar esses limites de tempo sem significado.

Sendo assim, há muita coisa a considerar. Este capítulo oferece uma visão geral do assunto que serve de estrutura para o material no restante do livro. Começamos com uma explicação geral sobre serviços e mecanismos de segurança de rede e os tipos de ataques para os quais eles são projetados. Depois desenvolvemos um modelo geral no qual os serviços e mecanismos de segurança podem ser vistos.

1.1 TENDÊNCIAS DE SEGURANÇA

Em 1994, o Internet Architecture Board (IAB) emitiu um relatório intitulado "Security in the internet architecture" — segurança na arquitetura da Internet — (RFC 1636). O relatório estabelecia o consenso geral de que a Internet precisa de mais e melhor segurança, e identificava as principais áreas para mecanismos de segurança. Entre estas estavam a necessidade de proteger a infra-estrutura da rede contra monitoração e controle não autorizados do tráfego da rede, e a necessidade de proteger o tráfego de usuário final para usuário final usando mecanismos de autenticação e de criptografia.

Essas questões são totalmente justificadas. Como confirmação, considere as tendências relatadas pelo Computer Emergency Response Team (CERT) Coordination Center (CERT/CC). A Figura 1.1a mostra a tendência em vulnerabilidades relacionadas à Internet, relatadas ao CERT por um período de 10 anos. Estas incluem os pontos fracos da segurança nos sistemas operacionais dos computadores atacados (por exemplo, Windows, Linux), além das vulnerabilidades nos roteadores da Internet e outros dispositivos de rede. A Figura 1.1b mostra o número de incidentes relacionados à segurança informados ao CERT. Estes incluem ataques de negação de serviço; falsificação de IP, em que os intrusos criam pacotes com endereços IP falsos e exploram aplicações que usam a autenticação baseada em IP; e diversas formas de bisbilhotagem e farejamento de pacote, em que os atacantes lêem informações transmitidas, incluindo informações de logon e conteúdo de banco de dados.

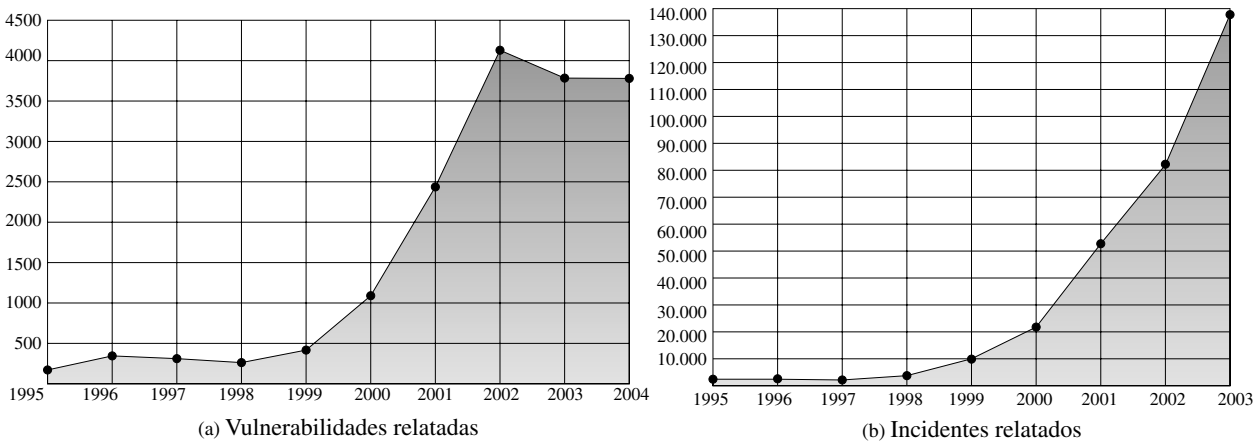
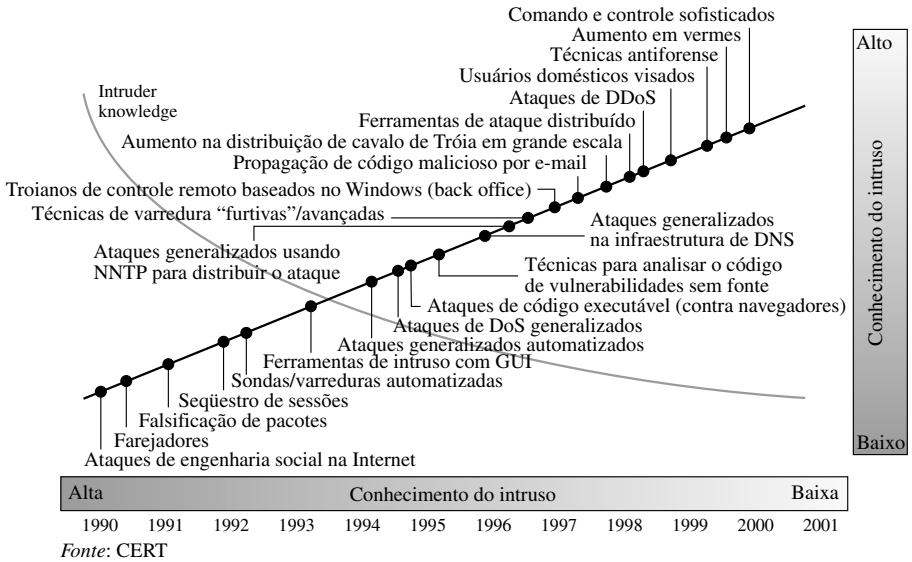


Figura 1.1 Estatísticas do CERT.



Fonte: CERT

Figura 1.2 Tendências na sofisticação do ataque e conhecimento do intruso.

Ao longo do tempo, os ataques na Internet e em sistemas conectados à Internet se tornaram mais sofisticados, enquanto a habilidade e o conhecimento exigidos para montar um ataque diminuíram (Figura 1.2). Os ataques se tornaram mais automatizados e podem causar mais danos.

Esse aumento nos ataques coincide com o aumento no uso da Internet e com aumentos na complexidade dos protocolos, aplicações e da própria Internet. Infra-estruturas críticas contam cada vez mais com a Internet para suas operações. Os usuários individuais contam cada vez mais com a segurança da Internet, de e-mail, da Web e das aplicações baseadas na Web. Assim, é preciso que haja uma grande gama de tecnologias e ferramentas para agir contra a ameaça crescente. Em um nível básico, os algoritmos criptográficos para confidencialidade e autenticação assumem maior importância. Além disso, os projetistas precisam focalizar os protocolos baseados na Internet e as vulnerabilidades dos sistemas operacionais e das aplicações atacadas. Este livro abrange todas essas áreas técnicas.

1.2 A ARQUITETURA DE SEGURANÇA OSI

Para avaliar efetivamente as necessidades de segurança de uma organização e avaliar e escolher diversos produtos e políticas de segurança, o gerente responsável precisa de algum meio sistemático de definir os requisitos de segurança e caracterizar as técnicas para satisfazer esses requisitos. Isso já é difícil em um ambiente de processamento de dados centralizado; com o uso de redes locais e de longa distância, os problemas são agravados.

A recomendação X.800 da ITU-T,[2] *Security architecture for OSI*, define tal técnica sistemática.[3] A arquitetura de segurança OSI é útil para os gerentes como um meio de organizar a tarefa de prover segurança. Além disso, como essa arquitetura foi desenvolvida como um padrão internacional, fornecedores de computador e comunicação desenvolveram recursos de segurança para seus produtos e serviços, que se relacionam com essa definição estruturada de serviços e mecanismos.

Para os nossos propósitos, a arquitetura de segurança OSI oferece uma visão geral útil, abstrata, de muitos dos conceitos dos quais este livro trata. A arquitetura de segurança OSI enfoca ataques, mecanismos e serviços de segurança. Estes podem ser definidos resumidamente da seguinte forma:

- **Ataque à segurança:** Qualquer ação que comprometa a segurança da informação pertencente a uma organização.
- **Mecanismo de segurança:** Um processo (ou um dispositivo incorporando tal processo) que é projetado para detectar, impedir ou permitir a recuperação de um ataque à segurança.
- **Serviço de segurança:** Um serviço de processamento ou comunicação que aumenta a segurança dos sistemas de processamento de dados e as transferências de informação de uma organização. Os serviços servem para frustrar ataques à segurança e utilizam um ou mais mecanismos de segurança para prover o serviço.

2. International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T) é uma agência patrocinada pelas Nações Unidas que desenvolve padrões, chamados de 'recomendações', relacionados a telecomunicações e à Open Systems Interconnection (OSI).

3. A arquitetura de segurança OSI foi desenvolvida no contexto da arquitetura de protocolo OSI. Porém, para nossos propósitos neste capítulo, um conhecimento da arquitetura de protocolos OSI não é obrigatório.

Na literatura, os termos *ameaça* e *ataque* normalmente são usados para designar mais ou menos a mesma coisa. A Tabela 1.1 oferece definições retiradas da RFC 2828, *Internet security glossary*.

Tabela 1.1 Ameaças e ataques (RFC 2828)

Ameaça
Potencial para violação da segurança quando há uma circunstância, capacidade, ação ou evento que pode quebrar a segurança e causar danos. Ou seja, uma ameaça é um possível perigo que pode explorar uma vulnerabilidade.
Ataque
Um ataque à segurança do sistema, derivado de uma ameaça inteligente, ou seja, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) de burlar os serviços de segurança e violar a política de segurança de um sistema.

1.3 ATAQUES À SEGURANÇA

Uma maneira útil de classificar os ataques à segurança, usada tanto na X.800 quanto na RFC 2828, é em termos de *ataques passivos* e *ataques ativos*. Um ataque passivo tenta descobrir ou utilizar informações do sistema, mas não afeta seus recursos. Um ataque ativo tenta alterar os recursos do sistema ou afetar sua operação.

Ataques passivos

Os ataques passivos possuem a natureza de bisbilhotar ou monitorar transmissões. O objetivo é obter informações que estão sendo transmitidas. Dois tipos de ataques passivos são liberação do conteúdo da mensagem e análise de tráfego.

A **liberação do conteúdo da mensagem** é facilmente compreendida (Figura 1.3a). Uma conversa telefônica, uma mensagem de correio eletrônico e um arquivo transferido podem conter informações importantes ou confidenciais. Desejamos impedir que alguém descubra o conteúdo dessas transmissões.

Um segundo tipo de ataque passivo, a **análise de tráfego**, é mais sutil (Figura 1.3b). Suponha que tivéssemos uma maneira de disfarçar o conteúdo das mensagens ou de outro tráfego de informações de modo que os oponentes,

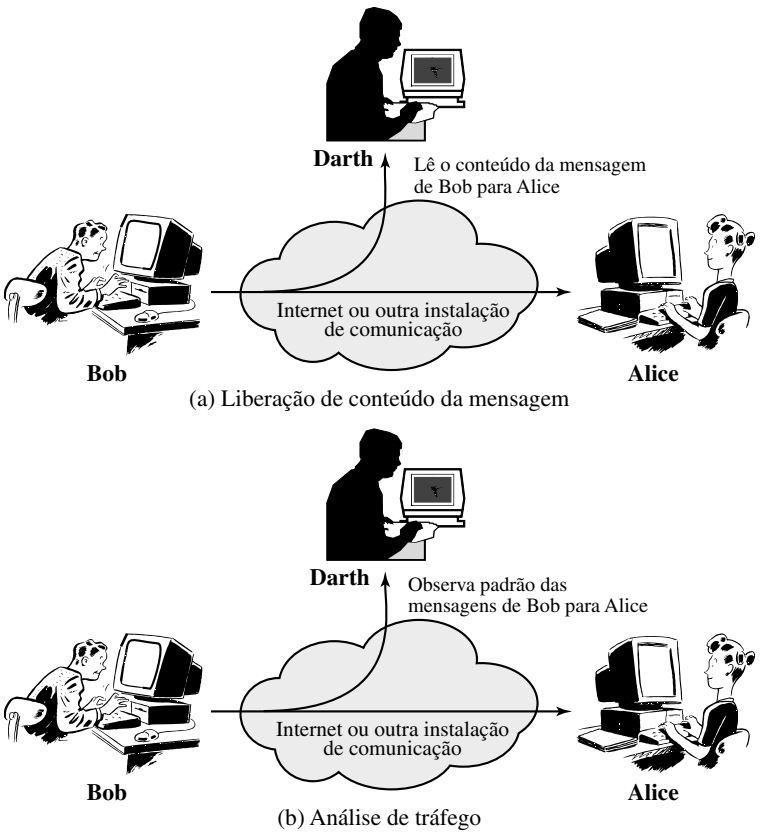


Figura 1.3 Ataques passivos.

mesmo que captassem a mensagem, não pudessem extrair suas informações. A técnica comum para disfarçar o conteúdo é a criptografia. Se tivéssemos proteção por criptografia, um oponente ainda poderia conseguir observar o padrão dessas mensagens. O oponente poderia determinar o local e a identidade dos hospedeiros envolvidos na comunicação e observar a freqüência e o tamanho das mensagens trocadas. Essa informação poderia ser útil para descobrir a natureza da comunicação que estava ocorrendo.

Ataques passivos são muito difíceis de detectar, pois não envolvem alteração dos dados. Normalmente, o tráfego de mensagens ocorre em um padrão aparentemente normal, e nem o emissor nem o receptor estão cientes de que um terceiro leu as mensagens ou observou o padrão de tráfego. Porém, normalmente, é viável impedir o sucesso desses ataques por meio da criptografia. Assim, a ênfase em lidar com ataques passivos está na prevenção, em vez de na detecção.

Ataques ativos

Ataques ativos envolvem alguma modificação do fluxo de dados ou a criação de um fluxo falso e podem ser subdivididos em quatro categorias: disfarce, repetição, modificação de mensagens e negação de serviço.

Um **disfarce** ocorre quando uma entidade finge ser uma entidade diferente (Figura 1.4a). Um ataque de disfarce normalmente inclui uma das outras formas de ataque ativo. Por exemplo, seqüências de autenticação podem ser captadas e reproduzidas depois que houver uma seqüência de autenticação válida, permitindo assim que uma entidade autorizada com poucos privilégios obtenha privilégios extras, imitando uma entidade que tenha esses privilégios.

A **repetição** envolve a captura passiva de uma unidade de dados e sua subseqüente retransmissão para produzir um efeito não autorizado (Figura 1.4b).

A **modificação de mensagens** simplesmente significa que alguma parte de uma mensagem legítima foi alterada ou que as mensagens foram adiadas ou reordenadas para produzir um efeito não autorizado (Figura 1.4c). Por exemplo, uma mensagem significando "Permitir que John Smith leia *contas* de arquivo confidenciais" é modificada para "Permitir que Fred Brown leia *contas* de arquivo confidenciais".

A **negação de serviço** impede ou inibe o uso ou gerenciamento normal das instalações de comunicação (Figura 1.4d). Esse ataque pode ter um alvo específico; por exemplo, uma entidade pode suprimir todas as mensagens dirigidas a determinado destino (por exemplo, o serviço de auditoria de segurança). Outra forma de negação de serviço é a interrupção de uma rede inteira, seja desativando a rede ou sobrecarregando-a com mensagens, a fim de prejudicar o desempenho.

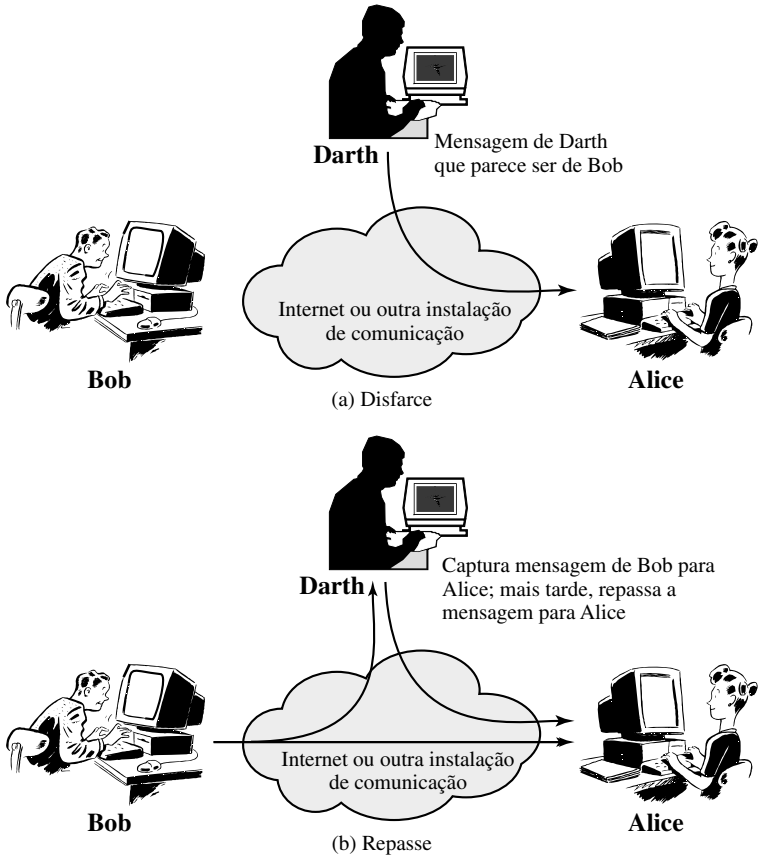


Figura 1.4 Ataques ativos (página 1 de 2).

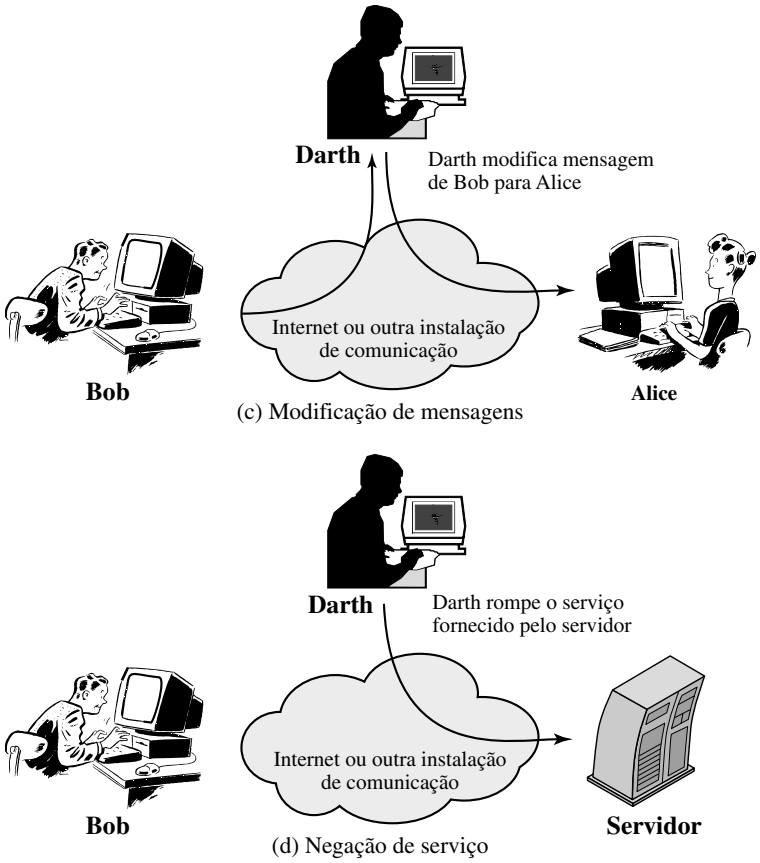


Figura 1.4 Ataques ativos (página 2 de 2).

Os ataques ativos apresentam as características opostas dos ataques passivos. Embora os ataques passivos sejam difíceis de detectar, existem medidas para impedir seu sucesso. Por outro lado, é muito difícil impedir ataques ativos absolutamente, devido à grande variedade de vulnerabilidades físicas, de software e de rede em potencial. Em vez disso, o objetivo é detectar ataques ativos e recuperar-se de qualquer interrupção ou atrasos causados por eles. Se a detecção tiver um efeito intimidador, ela também pode contribuir para a prevenção.

1.4 SERVIÇOS DE SEGURANÇA

A X.800 define um serviço de segurança como um serviço fornecido por uma camada de protocolo de comunicação de sistemas abertos, que garante a segurança adequada dos sistemas ou das transferências de dados. Talvez uma definição mais clara seja encontrada na RFC 2828, que oferece a seguinte definição: um serviço de processamento ou comunicação que é fornecido por um sistema para prover um tipo específico de proteção aos recursos do sistema; os serviços de segurança implementam políticas (ou diretrizes) de segurança e são implementados por mecanismos de segurança.

A X.800 divide esses serviços em cinco categorias e quatorze serviços específicos (Tabela 1.2). Vamos examinar cada categoria, uma por vez.[4]

Autenticação

O serviço de autenticação refere-se à garantia de que uma comunicação é autêntica. No caso de uma única mensagem, como uma advertência ou um sinal de alarme, a função do serviço de autenticação é garantir ao destinatário que a mensagem é proveniente de onde ela afirma ter vindo. No caso de uma interação de saída, como a conexão de um terminal com um hospedeiro, dois aspectos estão envolvidos. Primeiro, no momento do início da conexão, o ser-

4. Não existe um acordo universal sobre muitos dos termos usados na literatura de segurança. Por exemplo, o termo *integridade* às vezes é usado para se referir a todos os aspectos da segurança da informação. O termo *autenticação* às vezes é usado para se referir tanto à verificação da identidade quanto às diversas funções listadas sob integridade neste capítulo. Nosso uso aqui está de acordo com a recomendação X.800 e também com a RFC 2828.

Tabela 1.2 Serviços de segurança (X.800)

AUTENTICAÇÃO

A garantia de que a entidade se comunicando é aquela que ela afirma ser.

Autenticação de entidade par

Usada em associação com uma conexão lógica para confiabilidade da identidade das entidades conectadas.

Autenticação da origem de dados

Em uma transferência sem conexão, garante que a origem dos dados recebidos é a alegada.

CONTROLE DE ACESSO

O impedimento de uso não autorizado de um recurso (ou seja, esse serviço controla quem pode ter acesso a um recurso, sob que condições o acesso pode ocorrer e o que é permitido para aqueles que acessam o recurso).

CONFIDENCIALIDADE DOS DADOS

A proteção dos dados contra divulgação não autorizada.

Confidencialidade da conexão

A proteção de todos os dados do usuário em uma conexão.

Confidencialidade sem conexão

A proteção de todos os dados do usuário em um único bloco de dados.

Confidencialidade por campo selecionado

A confidencialidade dos campos selecionados dentro dos dados do usuário em uma conexão ou em um único bloco de dados.

Confidencialidade do fluxo de tráfego

A proteção das informações que poderiam ser derivadas da observação dos fluxos de tráfego.

INTEGRIDADE DE DADOS

A garantia de que os dados recebidos estão exatamente como foram enviados por uma entidade autorizada (ou seja, não contêm modificação, inserção, exclusão ou repetição).

Integridade da conexão com recuperação

Provê a integridade de todos os dados do usuário em uma conexão e detecta modificação, inserção, exclusão ou repetição de quaisquer dados dentro de uma seqüência, com tentativa de recuperação.

Integridade da conexão sem recuperação

Idem à anterior, mas oferece apenas detecção sem tentativa de recuperação.

Integridade da conexão com campo selecionado

Provê a integridade dos campos selecionados dentro dos dados do usuário de um bloco transferido por uma conexão e determina se os campos selecionados foram modificados, inseridos, excluídos ou replicados.

Integridade sem conexão

Provê a integridade de um único bloco de dados sem conexão e pode tomar a forma de detecção da modificação de dados. Além disso, pode haver uma forma limitada de detecção de repasse.

Integridade sem conexão com campo seletivo

Provê a integridade dos campos selecionados dentro de um único bloco de dados sem conexão; determina se os campos selecionados foram modificados.

IRRETRATABILIDADE

Oferece proteção contra negação, por parte de uma das entidades envolvidas em uma comunicação, de ter participado de toda ou parte da comunicação.

Irretratabilidade, Origem

Prova que a mensagem foi enviada pela parte especificada.

Irretratabilidade, Destino

Prova que a mensagem foi recebida pela parte especificada.

viço garante que as duas entidades são autênticas, ou seja, que cada uma é a entidade que afirma ser. Segundo, o serviço precisa garantir que a conexão não sofra interferência de modo que um terceiro possa fingir ser uma das duas partes legítimas, para fins de transmissão ou recepção não autorizada.

Dois serviços de autenticação específicos são definidos na X.800:

- **Autenticação da entidade par:** Provê a confirmação da identidade de uma entidade par de uma associação. O serviço é fornecido para uso no estabelecimento de uma conexão ou, às vezes, durante a fase de transferência de dados. Ele tem a intenção de garantir que uma entidade não está disfarçada ou realizando uma repetição não autorizada de uma conexão anterior.
- **Autenticação da origem de dados:** Provê a confirmação da origem de uma unidade de dados. Não oferece proteção contra a duplicação ou a modificação das unidades de dados. Esse tipo de serviço dá suporte a aplicações como correio eletrônico, nas quais não existem interações anteriores entre as entidades que se comunicam.

Controle de acesso

No contexto da segurança de redes, o controle de acesso é a capacidade de limitar e controlar o acesso aos sistemas e aplicações hospedeiras por meio de enlaces de comunicação. Para conseguir isso, cada entidade precisa ser identificada antes de obter acesso, ou autenticada, de modo que os direitos de acesso possam ser ajustados ao indivíduo.

Confidencialidade de dados

Confidencialidade é a proteção dos dados transmitidos contra ataques passivos. Com relação ao conteúdo de uma transmissão de dados, vários níveis de proteção podem ser identificados. O serviço mais amplo protege todos os dados transmitidos entre dois usuários por um período de tempo. Por exemplo, quando uma conexão TCP é estabelecida entre dois sistemas, essa ampla proteção impede a divulgação de quaisquer dados do usuário transmitidos pela conexão TCP. Formas mais estritas desse serviço também podem ser definidas, incluindo a proteção de uma única mensagem ou até mesmo campos específicos dentro de uma mensagem. Esses refinamentos são menos úteis do que a técnica ampla e podem ainda ser mais complexos e mais dispendiosos de implementar.

Outro aspecto da confidencialidade é a proteção do fluxo de tráfego contra análise. Isso exige que um atacante não consiga observar a origem e o destino, a freqüência, o tamanho ou outras características do tráfego em um sistema de comunicação.

Integridade de dados

Assim como a confidencialidade, a integridade pode se aplicar a um fluxo de mensagens, uma única mensagem ou campos selecionados dentro de uma mensagem. Novamente, a técnica mais útil e direta é a proteção total do fluxo.

Um serviço de integridade orientado à conexão, que lida com um fluxo de mensagens, garante que as mensagens são recebidas conforme enviadas, sem duplicação, inserção, modificação, reordenação ou repetição. A destruição dos dados também está incluída nesse serviço. Assim, o serviço de integridade orientada à conexão relaciona-se tanto à modificação do fluxo de mensagens quanto à negação de serviço. Por outro lado, um serviço de integridade sem conexão, que lida com mensagens individuais sem considerar qualquer contexto maior, geralmente oferece proteção apenas contra modificação da mensagem.

Podemos fazer uma distinção entre o serviço com e sem recuperação. Como o serviço de integridade está relacionado a ataques ativos, tratamos da detecção em vez da prevenção. Se uma violação de integridade for detectada, então o serviço poderá simplesmente informar essa violação, e alguma outra parte do software ou intervenção humana será necessária para recuperar-se da violação. Como alternativa, existem mecanismos disponíveis para recuperar-se da perda de integridade de dados, conforme veremos mais adiante. A incorporação de mecanismos de recuperação automatizados, em geral, é a alternativa mais atraente.

Irretratabilidade

A irretratabilidade impede que o emissor ou o receptor negue uma mensagem transmitida. Assim, quando uma mensagem é enviada, o receptor pode provar que o emissor alegado de fato enviou a mensagem. De modo semelhante, quando uma mensagem é recebida, o emissor pode provar que o receptor alegado de fato recebeu a mensagem.

Serviço de disponibilidade

Tanto a X.800 quanto a RFC 2828 definem a disponibilidade como sendo a propriedade de um sistema ou de um recurso do sistema ser acessível e utilizável sob demanda por uma entidade autorizada do sistema, de acordo com especificações de desempenho (ou seja, um sistema estará disponível se oferecer os serviços de acordo com o projeto do sistema sempre que os usuários os solicitarem). Diversos ataques podem resultar na perda ou na redução da disponibilidade. Alguns desses ataques são favoráveis a contramedidas automatizadas, como autenticação e criptografia, enquanto outros exigem algum tipo de ação física para impedir ou recuperar-se da perda de disponibilidade dos elementos de um sistema distribuído.

A X.800 trata a disponibilidade como uma propriedade a ser associada a vários serviços de segurança. Porém, faz sentido adotar especificamente um serviço de disponibilidade. Um serviço de disponibilidade é aquele que protege um sistema para garantir sua disponibilidade. Esse serviço trata das questões de segurança levantadas pelos ataques de negação de serviço. Depende do gerenciamento e controle apropriado dos recursos do sistema e, sendo assim, depende do serviço de controle de acesso e outros serviços de segurança.

1.5 MECANISMOS DE SEGURANÇA

A Tabela 1.3 lista os mecanismos de segurança definidos na recomendação X.800. Como podemos ver, os mecanismos são divididos nos implementados em uma camada específica de protocolo e nos específicos a qualquer camada de protocolo ou serviço de segurança em particular. Esses mecanismos serão abordados nos lugares apropriados no livro e, assim, não nos estenderemos agora, exceto para comentar sobre a definição da criptografia. A X.800 distingue mecanismos de criptografia reversíveis e mecanismos de criptografia irreversíveis. Um mecanismo de criptografia reversível é simplesmente um algoritmo de criptografia que permite que os dados sejam criptografados e subseqüentemente

decriptografados. Mecanismos de criptografia irreversíveis incluem algoritmos de hash e códigos de autenticação de mensagem, que são usados em aplicações de assinatura digital e autenticação de mensagens.

Tabela 1.3 Mecanismos de segurança (X.800)

MECANISMOS DE SEGURANÇA ESPECÍFICOS

Podem ser incorporados à camada de protocolo apropriada a fim de oferecer alguns dos serviços de segurança OSI.

Cifragem

O uso de algoritmos matemáticos para transformar os dados em um formato que não seja prontamente decifrável. A transformação e subseqüente recuperação dos dados depende de um algoritmo e zero ou mais chaves de criptografia.

Assinatura digital

Dados anexados a (ou uma transformação criptográfica de) uma unidade de dados que permite que um destinatário da unidade de dados comprove a origem e a integridade da unidade de dados e proteja-se contra falsificação (por exemplo, pelo destinatário).

Controle de acesso

Uma série de mecanismos que impõem direitos de acesso aos recursos.

Integridade de dados

Uma série de mecanismos utilizados para garantir a integridade de uma unidade de dados ou fluxo de unidades de dados.

Troca de informações de autenticação

Um mecanismo com o objetivo de garantir a identificação de uma entidade por meio da troca de informações.

Preenchimento de tráfego

A inserção de bits nas lacunas de um fluxo de dados para frustrar as tentativas de análise de tráfego.

Controle de roteamento

Permite a seleção de determinadas rotas fisicamente seguras para certos dados e permite mudanças de roteamento, especialmente quando existe suspeita de uma brecha de segurança.

Certificação

O uso de uma terceira entidade confiável para garantir certas propriedades de uma troca de dados.

MECANISMOS DE SEGURANÇA PERVASIVOS

Mecanismos que não são específicos a qualquer serviço de segurança OSI ou camada de protocolo específica.

Funcionalidade confiável

Aquela que é considerada como sendo correta em relação a alguns critérios (por exemplo, conforme estabelecido por uma política de segurança).

Rótulo de segurança

A marcação vinculada a um recurso (que pode ser uma unidade de dados) que nomeia ou designa os atributos de segurança desse recurso.

Detecção de evento

Detecção de eventos relevantes à segurança.

Registros de auditoria de segurança

Dados coletados e potencialmente utilizados para facilitar uma auditoria de segurança, que é uma revisão e exame independentes dos registros e atividades do sistema.

Recuperação de segurança

Lida com solicitações de mecanismos, como funções de tratamento e gerenciamento de eventos, e toma medidas de recuperação.

A Tabela 1.4 baseada na recomendação X.800, indica o relacionamento entre serviços de segurança e mecanismos de segurança.

Tabela 1.4 Relacionamento entre serviços e mecanismos de segurança

	Mecanismo							
Serviço	**Cifragem**	**Assinatura digital**	**Controle de acesso**	**Integridade de dados**	**Troca de informações**	**Preenchimento de tráfego**	**Controle de roteamento**	**Notarização**
Autenticação de entidade par	S	S			S			
Autenticação da origem de dados	S	S						
Controle de acesso			S					
Confidencialidade	S						S	
Confidencialidade do fluxo de tráfego	S					S	S	
Integridade de dados	S	S		S				
Irretratabilidade		S		S				S
Disponibilidade				S	S			

1.6 UM MODELO PARA SEGURANÇA DE REDE

Um modelo para grande parte do que explicaremos é mostrado, em termos muitos gerais, na Figura 1.5. Uma mensagem deve ser transferida de uma parte para outra por meio de algum tipo de inter-rede. As duas partes, que são as entidades *principais* nessa transação, precisam cooperar para que a troca ocorra. Um canal de informação lógico é estabelecido definindo-se uma rota pela inter-rede da origem ao destino e pelo uso cooperativo de protocolos de comunicação (por exemplo, TCP/IP) pelas entidades principais.

Os aspectos de segurança entram em cena quando é necessário ou desejável proteger a transmissão de informações de um oponente que pode representar uma ameaça à confidencialidade, autenticidade etc. Todas as técnicas para oferecer segurança possuem dois componentes:

- Uma transformação relacionada à segurança sobre a informação a ser enviada. Alguns exemplos incluem a criptografia da mensagem, que embaralha a mensagem de modo que fique ilegível para o oponente, e o acréscimo de um código com base no conteúdo da mensagem, que pode ser usado para verificar a identidade do emissor.
- Alguma informação secreta compartilhada pelos dois principais e, espera-se, desconhecida pelo oponente. Um exemplo é uma chave de criptografia usada em conjunto com a transformação para embaralhar a mensagem antes da transmissão e desembaralhá-la no recebimento.[5]

Um terceiro confiável pode ser necessário para se conseguir uma transmissão segura. Por exemplo, um terceiro pode ser responsável por distribuir a informação secreta aos dois principais enquanto a protege contra qualquer oponente. Ou então, um terceiro pode ser necessário para arbitrar disputas entre as duas entidades principais em relação à autenticidade de uma transmissão de mensagem.

Esse modelo geral mostra que existem quatro tarefas básicas no projeto de um serviço de segurança em particular:

1. Projetar um algoritmo para realizar a transformação relacionada à segurança. O algoritmo deverá ser tal que um oponente não possa violar sua finalidade.
2. Gerar a informação secreta a ser usada com o algoritmo.
3. Desenvolver métodos para a distribuição e compartilhamento da informação secreta.
4. Especificar um protocolo a ser usado pelas duas entidades principais que utilize o algoritmo de segurança e a informação secreta para conseguir um determinado serviço de segurança.

As Partes de I a III deste livro se concentram nos tipos de mecanismos e serviços de segurança que se encaixam no modelo mostrado na Figura 1.5. Porém, existem outras situações de interesse, relacionadas à segurança, que não se encaixam bem nesse modelo, mas que são consideradas neste livro. Um modelo geral dessas outras situações é ilustrado pela Figura 1.6, que reflete a preocupação de proteger um sistema de informações contra acesso indesejado. A maioria dos leitores está acostumada com os problemas causados pelos hackers, que tentam penetrar em sistemas que podem ser acessados por uma rede. Um hacker pode ser alguém que, sem má intenção, simplesmente se satisfaça em romper e entrar em um sistema de computador. Ou então, o intruso pode ser um funcionário aborrecido que deseje

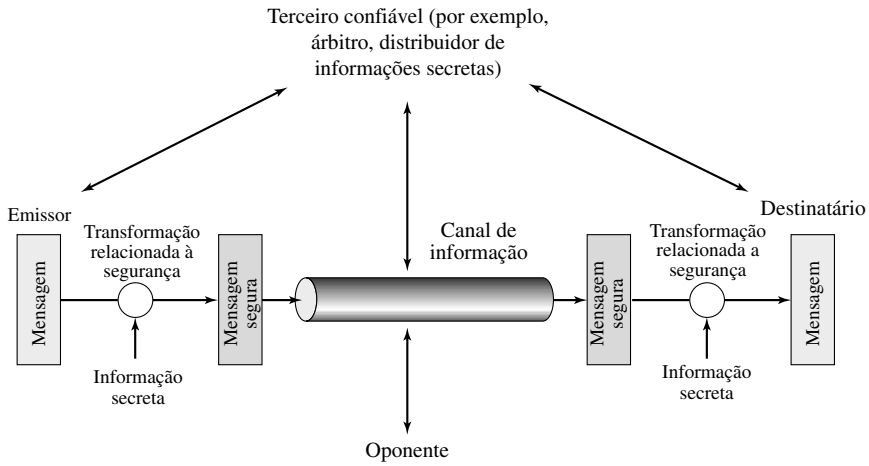


Figura 1.5 Modelo para segurança de rede.

5. A Parte II explica uma forma de criptografia, conhecida como criptografia por chave pública, em que somente uma das duas entidades principais precisa ter a informação secreta.

causar danos, ou um criminoso que busque explorar recursos do computador para obter lucro financeiro (por exemplo, obter números de cartão de crédito ou realizar transferências ilegais de dinheiro).

Outro tipo de acesso indesejado é a colocação, em um sistema computadorizado, de uma lógica que explore as vulnerabilidades do sistema e que possa afetar programas de aplicação e programas utilitários, como editores e compiladores. Os programas podem apresentar dois tipos de ameaças:

- Ameaças de acesso à informação, que interceptam ou modificam dados em favor de usuários que não deveriam ter acesso a esses dados.
- Ameaças de serviço, que exploram falhas de serviço nos computadores para inibir o uso por usuários autorizados.

Vírus e vermes (worms) são dois exemplos de ataques de software. Esses ataques podem ser introduzidos em um sistema por meio de um disco que contém a lógica indesejada escondida em um software útil. Eles também podem ser inseridos em um sistema por meio de uma rede; esse último mecanismo é mais preocupante na segurança de rede.

Os mecanismos de segurança necessários para lidar com o acesso indesejado estão em duas categorias amplas (veja a Figura 1.6). A primeira poderia ser chamada de função de portaria (gatekeeper). Ela inclui procedimentos de login baseados em senha, que são criados para negar o acesso a todos os usuários, menos os autorizados, e lógica de filtragem, que é elaborada para detectar e rejeitar vermes, vírus e outros ataques semelhantes. Quando um usuário indesejado ou software indesejado obtém acesso, a segunda linha de defesa consiste em uma série de controles internos que monitoram a atividade e analisam a informação armazenada em uma tentativa de detectar a presença de intrusos indesejados. Essas questões são exploradas na Parte IV.

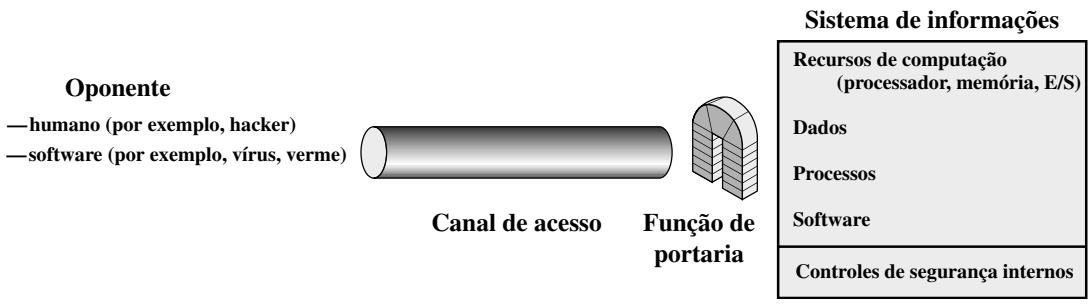


Figura 1.6 Modelo de segurança de acesso à rede.

1.7 LEITURA E SITES WEB RECOMENDADOS

[PFLE02] oferece uma boa introdução à segurança de computador e rede. Outros estudos excelentes são [PIEP03] e [BISH05]. [BISH03] abrange grande parte do material de [BISH05], mas com mais detalhes matemáticos e mais rigor. [SCHN00] é uma leitura valiosa para qualquer profissional no campo de segurança de computador ou rede: discute as limitações da tecnologia — e da criptografia em particular — no oferecimento de segurança e a necessidade de considerar o hardware, a implementação do software, as redes e as pessoas envolvidas como elementos no fornecimento e ataque à segurança.

BISH03 Bishop, M. *Computer security*: art and science. Boston: Addison-Wesley, 2003.
BISH05 Bishop, M. *Introduction to computer security*. Boston: Addison-Wesley, 2005.
PFLE02 Pfleeger, C. *Security in computing*. Upper Saddle River: Prentice Hall, 2002.
PIEP03 Pieprzyk, J.; Hardjono, T.; e Seberry, J. *Fundamentals of computer security*. New York: Springer-Verlag, 2003.
SCHN00 Schneier, B. *Secrets and lies*: digital security in a networked world. New York: Wiley, 2000.

Sites Web recomendados:

Os sites Web[6] a seguir são de interesse geral, relacionados à criptografia e à segurança de rede:

- **COAST:** Conjunto abrangente de links relacionados à criptografia e segurança de rede.
- **IETF Security Area:** Material relacionado aos esforços de padronização de segurança da Internet.

6. Como os URLs às vezes mudam, eles não estão incluídos. Para todos os sites Web listados neste e nos capítulos seguintes, o link apropriado está no site Web **williamstallings.com/Crypto/Crypto4e.html.**

- **Computer and Network Security Reference Index:** Um bom índice de produtos comerciais e seus fornecedores, FAQs, arquivos de grupos de discussão, artigos e outros sites Web.
- **The Cryptography FAQ:** FAQ extenso e valioso, abordando todos os aspectos da criptografia.
- **Tom Dunigan's Security Page:** Uma excelente lista de endereços de sites Web de criptografia e segurança de rede.
- **Helgar Lipma's Cryptology Pointers:** Outra excelente lista de endereços de sites Web de criptografia e segurança de redes.
- **IEEE Technical Committee on Security and Privacy:** Cópias de seus boletins, informações sobre atividades relacionadas ao IEEE.
- **Computer Security Resource Center:** Mantido pelo National Institute of Standards and Technology (NIST); contém uma grande variedade de informações sobre ameaças à segurança, tecnologia e padrões.
- **Security Focus:** Uma grande variedade de informações de segurança, com ênfase nos produtos de fornecedores e assuntos de interesse do usuário final.
- **SANS Institute:** Semelhante à Security Focus. Extensa coleção de artigos.

1.8 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

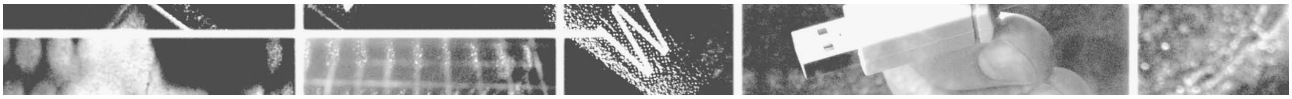
ameaça ativa
ameaça passiva
análise de tráfego
arquitetura de segurança OSI
ataque de repetição
ataques à segurança
autenticação
autenticidade
confidencialidade de dados
controle de acesso
criptografia
disfarce
disponibilidade
integridade
integridade de dados
intruso
mecanismos de segurança
negação de serviço
reconhecimento
serviços de segurança

Perguntas de revisão

1.1 O que é a arquitetura de segurança OSI?
1.2 Qual é a diferença entre ameaças à segurança passiva e à ativa?
1.3 Relacione e defina resumidamente as categorias de ataques passivos e ativos à segurança.
1.4 Relacione e defina resumidamente as categorias dos serviços de segurança.
1.5 Relacione e defina resumidamente as categorias dos mecanismos de segurança.

Problemas

1.1 Desenhe uma matriz semelhante à Tabela 1.4 que mostre o relacionamento entre serviços e ataques à segurança.
1.2 Desenhe uma matriz semelhante à Tabela 1.4 que mostre o relacionamento entre mecanismos e ataques à segurança.



PARTE UM

Cifras Simétricas

A criptografia provavelmente é o aspecto mais importante da segurança de comunicações e está se tornando cada vez mais importante como um componente básico para a segurança do computador.

Computers at risk: safe computing in the information age,
National Research Council, 1991

O crescente uso do computador e dos sistemas de comunicação pela indústria aumentou o risco de roubo de informações particulares. Embora essas ameaças possam exigir diversas contramedidas, a criptografia é um dos principais métodos para proteger informações eletrônicas valiosas.

Communications privacy: federal policy and actions,
General Accounting Office Report GAO/OSI-94-2, novembro de 1993

Com certeza, a ferramenta automatizada mais importante para a segurança da rede e das comunicações é a criptografia. Duas formas de criptografia são usadas normalmente: criptografia convencional, ou simétrica, e criptografia por chave pública, ou assimétrica. A Parte Um oferece um estudo sobre os princípios básicos da criptografia simétrica, examina os algoritmos mais usados e explica as aplicações da criptografia simétrica.

CONTEÚDO DA PARTE UM

Capítulo 2: Técnicas clássicas de criptografia
O Capítulo 2 descreve as técnicas de criptografia simétrica. Ele oferece uma apresentação suave e interessante da criptografia e criptoanálise, realçando os conceitos importantes.

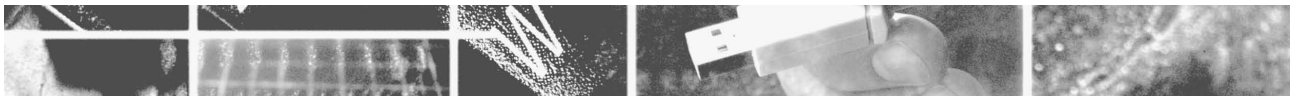
Capítulo 3: Cifras de bloco e o Data Encryption Standard
O Capítulo 3 apresenta os princípios da criptografia simétrica moderna, enfatizando a técnica de criptografia mais utilizada, Data Encryption Standard (DES). O capítulo inclui uma explicação sobre considerações de projeto e criptoanálise e apresenta a cifra de Feistel, que é a estrutura básicas dos esquemas de criptografia simétrica mais modernos.

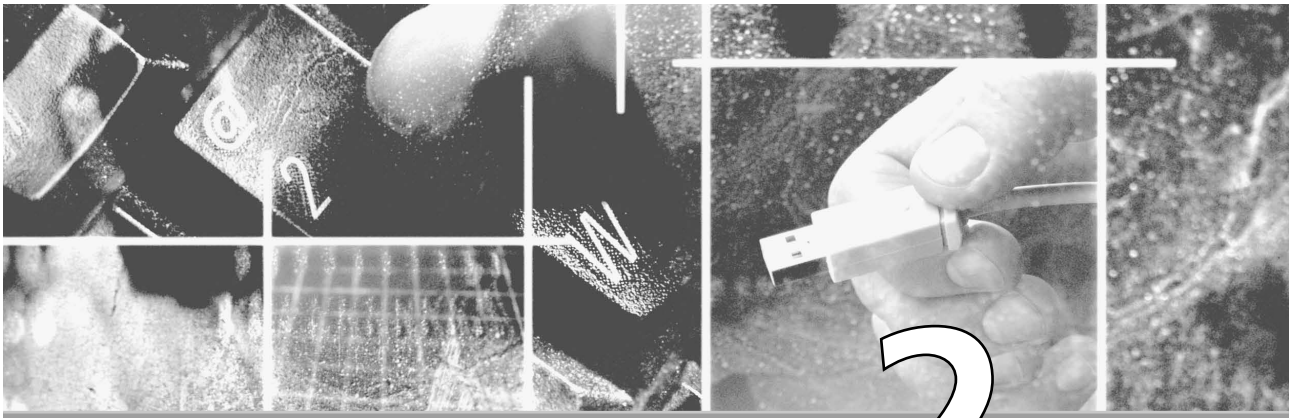
Capítulo 4: Corpos finitos
Os corpos finitos se tornaram cada vez mais importantes na criptografia. Diversos algoritmos criptográficos dependem fortemente das propriedades dos corpos finitos, principalmente o Advanced Encryption Standard (AES) e a criptografia por curva elíptica. Este capítulo está posicionado aqui para que os conceitos relevantes ao AES possam ser apresentados antes da explicação sobre o AES. O Capítulo 4 fornece a base necessária para a compreensão da aritmética dos corpos finitos na forma $GF(2^n)$.

Capítulo 5: Advanced Encryption Standard
O desenvolvimento mais importante na criptografia nos últimos anos é a adoção de um novo padrão de cifra simétrica, AES. O Capítulo 5 oferece uma explicação profunda sobre essa cifra.

Capítulo 6: Mais sobre cifras simétricas
O Capítulo 6 explora assuntos adicionais relacionados a cifras simétricas. O capítulo começa examinando a criptografia múltipla e, em particular, triple DES. Em seguida, examinamos o conceito dos modos de operação de cifra de bloco que lidam com as maneiras de tratar de um texto claro maior do que um único bloco. Finalmente, o capítulo explica as cifras de fluxo e descreve o RC4.

Capítulo 7: Confidencialidade usando criptografia simétrica
Além das questões que lidam com a construção real de um algoritmo de criptografia simétrica, diversas questões de projeto se relacionam ao uso da criptografia simétrica para oferecer confidencialidade. O Capítulo 7 estuda as mais importantes dessas questões. O capítulo inclui uma explicação sobre a criptografia fim-a-fim *versus* a criptografia de enlace, técnicas para conseguir confidencialidade de tráfego e técnicas de distribuição de chave. Um tópico relacionado importante, a geração de números aleatórios, também é explicado.





CAPÍTULO 2

Técnicas clássicas de criptografia

2.1 **Modelo de cifra simétrica**

2.2 **Técnicas de substituição**

2.3 **Técnicas de transposição**

2.4 **Máquinas de rotor**

2.5 **Esteganografia**

2.6 **Leitura e sites Web recomendados**

2.7 **Principais termos, perguntas de revisão e problemas**

Muitos bravos atualmente consideram o próprio nome como parte vital de si mesmos e, portanto, sofrem bastante para ocultar seu nome real, temendo que este dê um pretexto a pessoas mal-intencionadas para prejudicar seus donos.

The golden bough, Sir James George Frazer

PONTOS PRINCIPAIS

- A criptografia simétrica é uma forma de criptossistema em que a criptografia e a decriptografia são realizadas usando a mesma chave. Ela também é conhecida como criptografia convencional.
- A criptografia simétrica transforma o texto claro em texto cifrado, usando uma chave secreta e um algoritmo de criptografia. Usando a mesma chave e um algoritmo de decriptografia, o texto claro é recuperado a partir do texto cifrado.
- Os dois tipos de ataque em um algoritmo de criptografia são a criptoanálise, baseada nas propriedades do algoritmo de criptografia, e a força bruta, que envolve a tentativa de todas as chaves possíveis.
- As cifras simétricas tradicionais (anteriores ao computador) utilizam técnicas de substituição e/ou a transposição. As técnicas de substituição mapeiam elementos de texto claro (caracteres, bits) em elementos de texto cifrado. As técnicas de transposição transpõem sistematicamente as posições dos elementos do texto claro.
- As máquinas de rotor são dispositivos de hardware sofisticados, anteriores ao computador, que utilizam técnicas de substituição.
- A esteganografia é uma técnica para esconder uma mensagem secreta dentro de uma maior, de modo que outros não possam discernir a presença ou o conteúdo da mensagem oculta.

A criptografia simétrica, também chamada de criptografia convencional ou criptografia de chave única, era o único tipo de criptografia em uso antes do desenvolvimento da criptografia por chave pública na década de 1970. Esse continua sendo, de longe, o mais usado dos dois tipos de criptografia. A Parte 1 examina diversas cifras simétricas. Neste capítulo começamos verificando um modelo geral para o processo de criptografia simétrica; isso nos permitirá entender o contexto dentro do qual os algoritmos são usados. Em seguida, examinamos diversos algoritmos em uso antes da era do computador. Por fim, examinamos rapidamente uma técnica diferente, conhecida como esteganografia. O Capítulo 3 examina a cifra simétrica mais utilizada: DES.

Antes de começar, definimos alguns termos. Uma mensagem original é conhecida como **texto claro** (ou *plaintext*), enquanto a mensagem codificada é chamada de **texto cifrado** (ou *ciphertext*). O processo de converter texto claro em texto cifrado é conhecido como **cifragem** ou **criptografia**; restaurar o texto claro a partir do texto cifrado é a **decifragem** ou **decriptografia**. Os muitos esquemas utilizados para a criptografia constituem a área de estudo conhecida como **criptografia**. Esse esquema é conhecido como **sistema criptográfico** ou **cifra**. As técnicas empregadas para decifrar uma mensagem sem qualquer conhecimento dos detalhes de criptografia estão na área da **criptoanálise**. A criptoanálise é o que os leigos chamam de 'quebrar o código'. As áreas da criptografia e criptoanálise juntas são chamadas de **criptologia**.

2.1 MODELO DE CIFRA SIMÉTRICA

Um esquema de criptografia simétrica possui cinco ingredientes (Figura 2.1):

- **Texto claro:** Essa é a mensagem ou dados originais, inteligíveis, alimentados no algoritmo como entrada.
- **Algoritmo de criptografia:** O algoritmo de criptografia realiza diversas substituições e transformações no texto claro.
- **Chave secreta:** A chave secreta também é a entrada para o algoritmo de criptografia. A chave é um valor independente do texto claro e do algoritmo. O algoritmo produzirá uma saída diferente, dependendo da chave específica sendo usada no momento. As substituições e transformações exatas realizada pelo algoritmo dependem da chave.
- **Texto cifrado:** Essa é a mensagem embaralhada, produzida como saída. Ela depende do texto claro e da chave secreta. Para determinada mensagem, duas chaves diferentes produzirão dois textos cifrados diferentes. O texto cifrado é um fluxo de dados aparentemente aleatório e, nesse formato, é ininteligível.
- **Algoritmo de decriptografia:** Esse é basicamente o algoritmo de criptografia executado de modo inverso. Ele toma o texto cifrado e a chave secreta e produz o texto claro original.

Existem dois requisitos para o uso seguro da criptografia convencional:

1. Precisamos de um algoritmo de criptografia forte. No mínimo, queremos que um oponente que conheça o algoritmo e tenha acesso a um ou mais textos cifrados seja incapaz de decifrar o texto cifrado ou descobrir a chave. Esse requisito normalmente é indicado em uma forma mais forte: o oponente deverá ser incapaz de decriptografar o texto cifrado ou descobrir a chave mesmo que possua diversos textos cifrados juntamente com o texto claro que produziu cada texto cifrado.
2. Emissor e receptor precisam ter cópias da chave secreta de uma forma segura e precisam manter a chave protegida. Se alguém puder descobrir a chave e souber o algoritmo, toda a comunicação usando essa chave poderá ser lida.

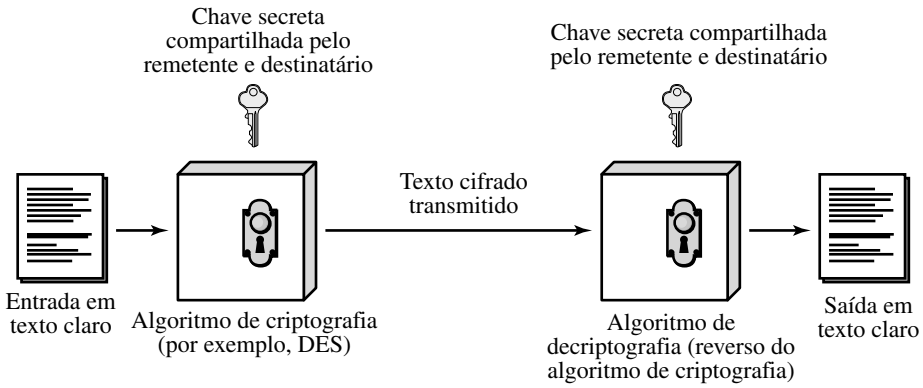


Figura 2.1 Modelo simplificado da criptografia convencional.

Consideramos que é impraticável decriptografar uma mensagem com base no texto cifrado *mais* o conhecimento do algoritmo de criptografia/decriptografia. Em outras palavras, não precisamos manter o algoritmo secreto; precisamos manter apenas a chave secreta. Essa característica da criptografia simétrica é o que a torna viável para uso generalizado. O fato de que o algoritmo não precisa ser mantido secreto significa que os fabricantes podem desenvolver, como realmente desenvolveram, implementações de chip de baixo custo com algoritmos de criptografia de dados. Esses chips são encontrados com facilidade e estão incorporados em diversos produtos. Com o uso da criptografia simétrica, o principal problema de segurança é manter o sigilo da chave.

Vejamos mais de perto os elementos essenciais de um esquema de criptografia simétrico, usando a Figura 2.2. Uma origem produz uma mensagem em texto claro, $X = [X_1, X_2, \ldots, X_M]$. Os M elementos de X são letras no alfabeto finito. Tradicionalmente, o alfabeto normalmente consiste em 26 letras maiúsculas. Hoje, normalmente é utilizado o alfabeto binário {0, 1}. Para a criptografia, uma chave na forma $K = [K_1, K_2, \ldots, K_J]$ é gerada. Se a chave for gerada na origem da mensagem, então ela também precisará ser fornecida ao destino por meio de algum canal seguro. Como alternativa, um terceiro poderia gerar a chave e oferecê-la com segurança à origem e ao destino.

Com a mensagem X e a chave de criptografia K como entrada, o algoritmo de criptografia forma o texto cifrado $Y = [Y_1, Y_2, \ldots, Y_N]$. Podemos escrever isso como

$$Y = \mathrm{E}(K, X)$$

Essa notação indica que Y é produzido usando-se o algoritmo de criptografia E como função de texto claro X, com a função específica determinada pelo valor da chave K.

O receptor pretendido, de posse da chave, é capaz de inverter a transformação:

$$X = \mathrm{D}(K, Y)$$

Um oponente, observando Y, mas não tendo acesso a K ou a X, pode tentar recuperar X ou K ou X e K. Presume-se que o oponente conheça os algoritmos de criptografia (E) e decriptografia (D). Se o oponente estiver interessado apenas nessa mensagem em particular, então o foco do esforço será recuperar X gerando uma estimativa de texto claro $X = \mathrm{D}(K, Y)$ Normalmente, porém, o oponente está interessado em ser capaz de ler também mensagens futuras, quando faz uma tentativa de recuperar K gerando uma estimativa $X = \mathrm{D}(K, Y)$.

Criptografia

Os sistemas criptográficos são caracterizados em três dimensões independentes:

1. **O tipo das operações usadas para transformar texto claro em texto cifrado.** Todos os algoritmos de criptografia são baseados em dois princípios gerais: substituição, em que cada elemento no texto claro (bit, letra, grupo de bits ou letras) é mapeado em outro elemento, e transposição, em que os elementos no texto claro são reorganizados. O requisito fundamental é que nenhuma informação seja perdida (ou seja, que todas as operações sejam reversíveis). A maioria dos sistemas, chamados de *sistemas de produto*, envolve vários estágios de substituições e transposições.

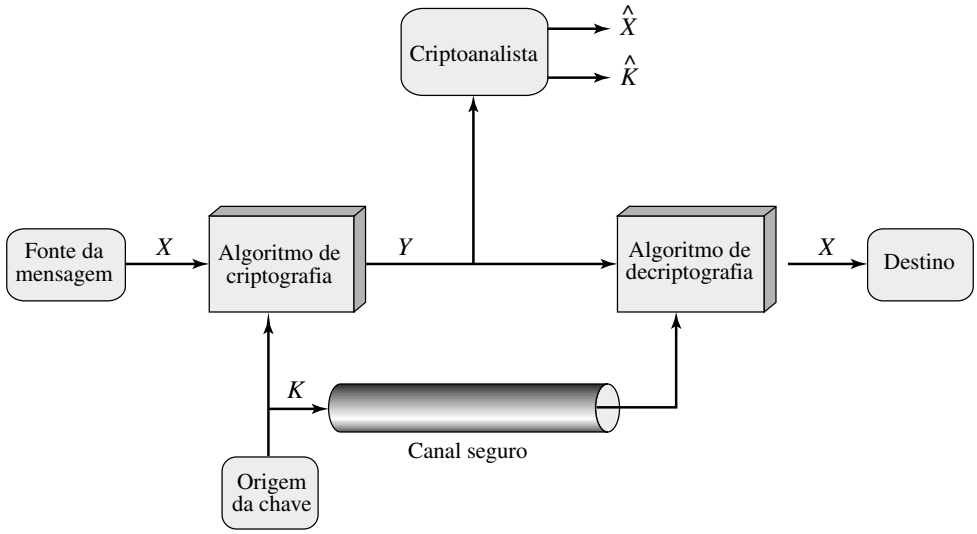


Figura 2.2 Modelo de criptossistema convencional.

2. **O número de chaves usadas.** Se tanto o emissor quanto o receptor utilizarem a mesma chave, o sistema é considerado como criptografia simétrica, de chave única, de chave secreta ou convencional. Se emissor e receptor usarem chaves diferentes, o sistema é considerado de criptografia assimétrica, de duas chaves ou de chave pública.
3. **O modo como o texto claro é processado.** Uma *cifra de bloco* processa a entrada de um bloco de elementos de cada vez, produzindo um bloco de saída para cada bloco de entrada. Uma *cifra em fluxo* processa os elementos da entrada continuamente, produzindo a saída de um elemento de cada vez, enquanto prossegue.

Criptoanálise

Normalmente, o objetivo de atacar um sistema de criptografia é recuperar a chave em uso, em vez de simplesmente recuperar o texto claro de um único texto cifrado. Existem duas técnicas gerais para o ataque a um esquema de criptografia convencional:

- **Criptoanálise:** Os ataques criptoanalíticos contam com a natureza do algoritmo e talvez mais algum conhecimento das características gerais do texto claro, ou ainda alguns pares de amostra de texto claro e texto cifrado. Esse tipo de ataque explora as características do algoritmo para tentar deduzir um texto claro específico ou deduzir a chave utilizada.
- **Ataque por força bruta:** O atacante experimenta cada chave possível em um trecho do texto cifrado, até obter uma tradução inteligível para texto claro. Na média, metade de todas as chaves possíveis precisam ser experimentadas para se obter sucesso.

Se qualquer tipo de ataque tiver sucesso na dedução da chave, o efeito será catastrófico: todas as mensagens futuras e passadas, codificadas com essa chave, estarão comprometidas.

Primeiro consideramos a criptoanálise e, depois, os ataques por força bruta.

A Tabela 2.1 resume os diversos tipos de ataques de criptoanálise, baseados na quantidade de informação conhecida pelo criptoanalista. O problema mais difícil é apresentado quando tudo o que está disponível é *apenas o texto cifrado*. Em alguns casos, nem sequer o algoritmo de criptografia é conhecido, mas em geral podemos considerar que o oponente conhece o algoritmo usado para a criptografia. Um ataque possível sob essas circunstâncias é a técnica de força bruta que tenta todas as chaves possíveis. Se o espaço da chave for muito grande, isso se tornará impraticável. Assim, o oponente precisa contar com uma análise do próprio texto cifrado, geralmente aplicando diversos testes estatísticos a ele. Para usar essa técnica, o oponente precisa ter alguma idéia geral do tipo de texto claro que está encoberto, como texto em inglês ou francês, um arquivo EXE, uma listagem-fonte em Java, um arquivo de contabilidade etc.

O ataque apenas com texto cifrado é o mais fácil de ser combatido, pois o oponente tem a menor quantidade de informações para trabalhar. Em muitos casos, porém, o analista tem mais informações. O analista pode ser capaz

Tabela 2.1 Tipos de ataques a mensagens criptografadas

Tipo de ataque	Conhecido ao criptoanalista
Apenas texto cifrado	• Algoritmo de criptografia • Texto cifrado
Texto claro conhecido	• Algoritmo de criptografia • Texto cifrado • Um ou mais pares de texto claro/texto cifrado formados com a chave secreta
Texto claro escolhido	• Algoritmo de criptografia • Texto cifrado • Mensagem de texto claro escolhida pelo criptoanalista, juntamente com seu texto cifrado correspondente, gerado com a chave secreta
Texto cifrado escolhido	• Algoritmo de criptografia • Texto cifrado • Texto cifrado pretendido, escolhido pelo criptoanalista, juntamente com seu texto claro decriptografado correspondente, gerado com a chave secreta
Texto escolhido	• Algoritmo de criptografia • Texto cifrado • Mensagem de texto claro escolhida pelo criptoanalista, juntamente com seu texto cifrado correspondente, gerado com a chave secreta • Texto cifrado pretendido, escolhido pelo criptoanalista, juntamente com seu texto claro decriptografado correspondente, gerado com a chave secreta

de obter uma ou mais mensagens de texto claro, além de suas criptografias. Ou então o analista pode saber que certos padrões de texto claro aparecerão em uma mensagem. Por exemplo, um arquivo codificado no formato Postscript sempre começando com o mesmo padrão, ou pode haver um cabeçalho ou banner padronizado para uma mensagem de transferência eletrônica de fundos etc. Todos esses exemplos são de *texto claro conhecido*. Com esse conhecimento, o analista pode ser capaz de deduzir a chave com base no modo como o texto claro conhecido é transformado.

Bastante relacionado ao ataque de texto claro conhecido é o que poderia ser chamado de ataque de palavra provável. Se o oponente estiver trabalhando com a criptografia de alguma mensagem de texto geral, ele pode ter pouco conhecimento do que está na mensagem. Porém, se o oponente estiver atrás de alguma informação muito específica, então partes da mensagem podem ser conhecidas. Por exemplo, se um arquivo de contabilidade inteiro estiver sendo transmitido, o oponente pode conhecer o posicionamento de certas palavras-chave no cabeçalho do arquivo. Como outro exemplo, o código-fonte de um programa desenvolvido pela Empresa X poderia incluir uma nota de direitos autorais em alguma posição padronizada.

Se o analista de alguma maneira for capaz de fazer com que o sistema de origem insira no sistema uma mensagem escolhida pelo analista, então o ataque de *texto claro escolhido* é possível. Um exemplo dessa estratégia é a criptoanálise diferencial, explicada no Capítulo 3. Em geral, se o analista for capaz de escolher as mensagens a serem criptografadas, ele poderá escolher deliberadamente padrões que presumivelmente revelarão a estrutura da chave.

A Tabela 2.1 lista dois outros tipos de ataque: texto cifrado escolhido e texto escolhido. Estes são menos empregados como técnicas criptoanalíticas, mas, apesar disso, são vias de ataque possíveis.

Somente algoritmos relativamente fracos não conseguem resistir a um ataque apenas de texto cifrado. Em geral, um algoritmo de criptografia é projetado para resistir a um ataque de texto claro conhecido.

Duas outras definições são dignas de nota. Um esquema de criptografia é **incondicionalmente seguro** se o texto cifrado gerado pelo esquema não tiver informações suficientes para determinar exclusivamente o texto claro correspondente, não importando quanto texto cifrado esteja à disposição. Ou seja, não importa quanto tempo um oponente tenha, é impossível que ele decodifique o texto cifrado, simplesmente porque a informação exigida não está lá. Com o escalonamento de um esquema conhecido como One-Time Pad (descrito mais adiante neste capítulo), não existe algoritmo de criptografia que seja incondicionalmente seguro. Portanto, tudo o que os usuários de um algoritmo de criptografia podem se esforçar para obter é um algoritmo que atenda a um ou a ambos os critérios a seguir:

- Custo para quebrar a cifra superior ao valor da informação codificada.
- Tempo exigido para quebrar a cifra superior ao tempo de vida útil da informação.

Um esquema de criptografia é considerado **computacionalmente seguro** se um desses dois critérios for atendido. O problema é que é muito difícil estimar a quantidade de esforço exigido para criptoanalisar o texto cifrado com sucesso.

Todas as formas de criptoanálise de esquemas de criptografia simétricos são projetadas para explorar o fato de que rastros da estrutura ou do padrão no texto claro podem sobreviver à criptografia e serem discerníveis no texto cifrado. Isso ficará mais claro quando examinarmos diversos esquemas de criptografia simétrica neste capítulo. Na Parte 2 veremos que a criptoanálise para esquemas de chave pública prosseguem a partir de uma premissa fundamentalmente diferente, a saber, de que as propriedades matemáticas do par de chaves podem possibilitar que uma das duas chaves seja deduzida a partir da outra.

Um **ataque por força bruta** envolve a tentativa de cada chave possível até que seja obtida uma tradução inteligível de texto cifrado para texto claro. Na média, metade de todas as chaves possíveis precisa ser experimentada para se conseguir sucesso. A Tabela 2.2 mostra o tempo envolvido para diversos espaços de chaves. Os resultados são mostrados para quatro tamanhos de chave binária. O tamanho de chave de 56 bits é usado com o algoritmo DES (Data Encryption Standard), e o tamanho de chave de 168 bits é usado para triple DES. O tamanho de chave mínimo especificado para AES (Advanced Encryption Standard) é de 128 bits. Os resultados também são mostrados para os chamados códigos de substituição que utilizam uma chave de 26 caracteres (explicada mais adiante), em que todas as permutações possíveis dos 26 caracteres servem como chaves. Para cada tamanho de chave, os resultados são mostrados

Tabela 2.2 Tempo médio exigido para busca completa da chave

Tamanho da chave (bits)	Número de chaves alternativas	Tempo necessário para 1 decriptografia/μs	Tempo necessário para 10^6 decriptografias/μs
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} μs = 35,8 minutos	2,15 milissegundos
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} μs = 1142 anos	10,01 horas
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ anos	$5{,}4 \times 10^{18}$ anos
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ anos	$5{,}9 \times 10^{30}$ anos
26 caracteres (permutação)	$26! = 4 \times 10^{26}$	2×10^{26} μs = $6{,}4 \times 10^{12}$ anos	$6{,}4 \times 10^{6}$ anos

considerando que é necessário 1 *m*s para realizar uma única decriptografia, que é uma ordem de grandeza razoável para as máquinas de hoje. Com o uso de arquiteturas de sistemas de microprocessadores maciçamente paralelas, pode ser possível alcançar taxas de processamento com muitas ordens de grandeza maiores. A última coluna da Tabela 2.2 considera os resultados para um sistema que pode processar 1 milhão de chaves por microssegundo. Como pode ser visto, nesse nível de desempenho, o DES não pode mais ser considerado computacionalmente seguro.

2.2 TÉCNICAS DE SUBSTITUIÇÃO

Nesta seção e na próxima, examinamos uma amostra do que poderíamos chamar de técnicas de criptografia clássicas. Um estudo dessas técnicas nos permite ilustrar as técnicas básicas da criptografia simétrica usadas hoje e os tipos de ataques criptoanalíticos que precisam ser antecipados.

Os dois componentes básicos de todas as técnicas de criptografia são substituição e transposição. Vamos examiná-los nas duas seções seguintes. Finalmente, explicamos um sistema que combina substituição e transposição.

Uma técnica de substituição é aquela em que as letras de texto claro são substituídas por outras letras ou por números ou símbolos.[1] Se o texto claro for visto como uma seqüência de bits, então a substituição envolve substituir padrões de bits de texto claro por padrões de bits de texto cifrado.

Cifra de César

O uso mais antigo que conhecemos de uma cifra de substituição, e o mais simples, foi feito por Júlio César. A cifra de César consiste em substituir cada letra do alfabeto pela letra que fica três posições adiante no alfabeto. Por exemplo:

```
claro: meet me after the toga party
cifra: PHHW PH DIWHU WKH WRJD SDUWB
```

Observe que o alfabeto recomeça no final, de modo que a letra após Z é A. Podemos definir a transformação listando todas as possibilidades, da seguinte forma:

```
claro :  a b c d e f g h i j k l m n o p q r s t u v w x y z
cifra:   D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Vamos atribuir um equivalente numérico a cada letra:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Então, o algoritmo pode ser expresso da seguinte forma. Para cada letra em texto claro p, substitua-a pela letra do texto cifrado C:[2]

$$C = \mathrm{E}(\mathrm{e}, p) = (p + 3) \bmod 26$$

Um deslocamento pode ser de qualquer quantidade, de modo que o algoritmo de César geral é

$$C = \mathrm{E}(k, p) = (p + k) \bmod 26$$

onde k assume um valor no intervalo de 1 a 25. O algoritmo de decriptografia é simplesmente

$$p = \mathrm{D}(k, C) = (C - k) \bmod 26$$

Se for conhecido que determinado texto cifrado é uma cifra de César, então uma criptoanálise pela força bruta será facilmente realizada. Basta experimentar todas as 25 chaves possíveis. A Figura 2.3 mostra os resultados da aplicação dessa estratégia ao texto cifrado do exemplo. Nesse caso, o texto claro aparece na terceira linha.

1. Quando são envolvidas letras, as convenções a seguir são usadas neste livro: texto claro em minúsculas; texto cifrado em maiúsculas; os valores de chave em minúsculas e itálico.
2. Definimos a mod n como sendo o resto quando a é dividido por n. Por exemplo, 11 mod 7 = 4. Veja, no Capítulo 4, uma explicação mais profunda sobre aritmética modular.

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figura 2.3 Criptoanálise pela força bruta da cifra de César.

Três características importantes desse problema nos permitiram usar uma criptoanálise pela força bruta:

1. Os algoritmos de criptografia e descriptografia são conhecidos.
2. Existem apenas 25 chaves a serem experimentadas.
3. A linguagem do texto claro é conhecida e facilmente reconhecível.

Na maioria das situações de rede, podemos considerar que os algoritmos são conhecidos. O que geralmente torna a criptoanálise pela força bruta impraticável é o uso de um algoritmo que emprega um grande número de chaves. Por exemplo, o algoritmo triple DES, examinado no Capítulo 6, utiliza uma chave de 168 bits, gerando um espaço de chave de 2^{168}, ou mais de $3{,}7 \times 10^{50}$ chaves possíveis.

A terceira característica também é significativa. Se a linguagem do texto claro for desconhecida, então a saída de texto claro pode não ser reconhecível. Além disso, a entrada pode ser abreviada ou compactada de alguma maneira, novamente dificultando o reconhecimento. Por exemplo, a Figura 2.4 mostra uma parte de um arquivo de texto compactada usando um algoritmo chamado ZIP. Se esse arquivo for então criptografado com uma cifra de substituição simples (expandida para incluir mais do que apenas 26 caracteres alfabéticos), então o texto claro não poderá ser reconhecido quando for descoberto na criptoanálise pela força bruta.

```
˜+Wµ"— Ω-O)≤4{∞‡¸ ë~Ω%ràu·¯Í ◊¯Z-
Ú≠2Ò#Åæ∂ œ«q7¸Ωn·®3N◊Ú Œz'Y-ƒ∞Í[±Û_ èΩ,<NO¬±«ˇxã  Åäƒèü3Å
x}ö§k°Â
_yÍ ˆΔÉ] ¸¤ J/˚iTê&ı 'c<uΩ-
 ÄD(G WÄC~y_ïõÄW PÔı«ÎÜ†ç],¤¡ˇÌ^üÑπ˜≈ˇLˇ9OgflOˇ&Œ≤ ¬≤ ØÔ§˝:
 ˇŒ!SGqèvoˆ ú\¸S>h<-*6ø‡%x´˝|fiÓ#≈˜my‰ˇ≥ñP<,ƒi Áj  Å◊¿˝Zù-
Ω¨Õ¯6Œÿ{% „ΩÊó ¸ï π÷ÁÂî˚úO2çSÿ´O-
2ÄflßÌ /@ˆ”∏Kªª PŒπ¸úéˆ´3∑˜öˇÔZÌ"Y¬ŸΩœY> Ω+eô/˙<K£¿*÷˜"≤û˜
B ZøK˜Qßÿüƒ¸!ÒflÎzsS/]>ÈQ ü
```

Figura 2.4 Exemplo de texto compactado.

Cifras monoalfabéticas

Com apenas 25 chaves possíveis, a cifra de César está longe de ser segura. Um aumento drástico no espaço de chave pode ser alcançado permitindo uma substituição arbitrária. Lembre-se da atribuição para a cifra de César:

```
claro:  a b c d e f g h i j k l m n o p q r s t u v w x y z
cifra:  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Se, em vez disso, a linha 'cifra' puder ser qualquer permutação dos 25 caracteres alfabéticos, então haverá 26 ou mais do que 4×10^{26} chaves possíveis. Isso significa 10 ordens de grandeza a mais do que o espaço de chave para DES, e pareceria eliminar as técnicas de força bruta da criptoanálise. Essa técnica é conhecida como **cifra por substituição monoalfabética**, pois um único alfabeto de cifra (mapeando do alfabeto claro para um alfabeto cifrado) é utilizado por mensagem.

Porém, existe outra linha de ataque. Se o criptoanalista souber a natureza do texto claro (por exemplo, texto em inglês não compactado), então ele poderá explorar as regularidades da linguagem. Para ver como essa criptoanálise poderia prosseguir, damos aqui um exemplo parcial adaptado de outro em [SINK66]. O texto cifrado a ser resolvido é

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

Como um passo inicial, a freqüência relativa das letras poderá ser determinada e comparada com uma distribuição de freqüência padrão para o inglês, como vemos na Figura 2.5 (com base em [LEWA00]). Se a mensagem fosse muito longa, apenas essa técnica poderia ser suficiente, mas, como essa é uma mensagem relativamente curta, não podemos esperar uma coincidência exata. De qualquer maneira, as freqüências relativas das letras no texto cifrado (em porcentagens) são as seguintes:

P 13,33	H 5,83	F 3,33	B 1,67	C 0,00
Z 11,67	D 5,00	W 3,33	G 1,67	K 0,00
S 8,33	E 5,00	Q 2,50	Y 1,67	L 0,00
U 8,33	V 4,17	T 2,50	I 0,83	N 0,00
O 7,50	X 4,17	A 1,67	J 0,83	R 0,00
M 6,67				

Comparando esse desmembramento com a Figura 2.5, parece provável que as letras cifradas P e Z sejam os equivalentes das letras do texto claro e e t, mas não se sabe ao certo quem é quem. As letras S, U, O, M e H são todas de freqüência relativamente alta e provavelmente correspondem às letras que estão no conjunto {a, h, i, n, o, r, s}. As letras com freqüências mais baixas (a saber, A, B, G, Y, I, J) provavelmente estão incluídas no conjunto {b, j, k, q, v, x, z}.

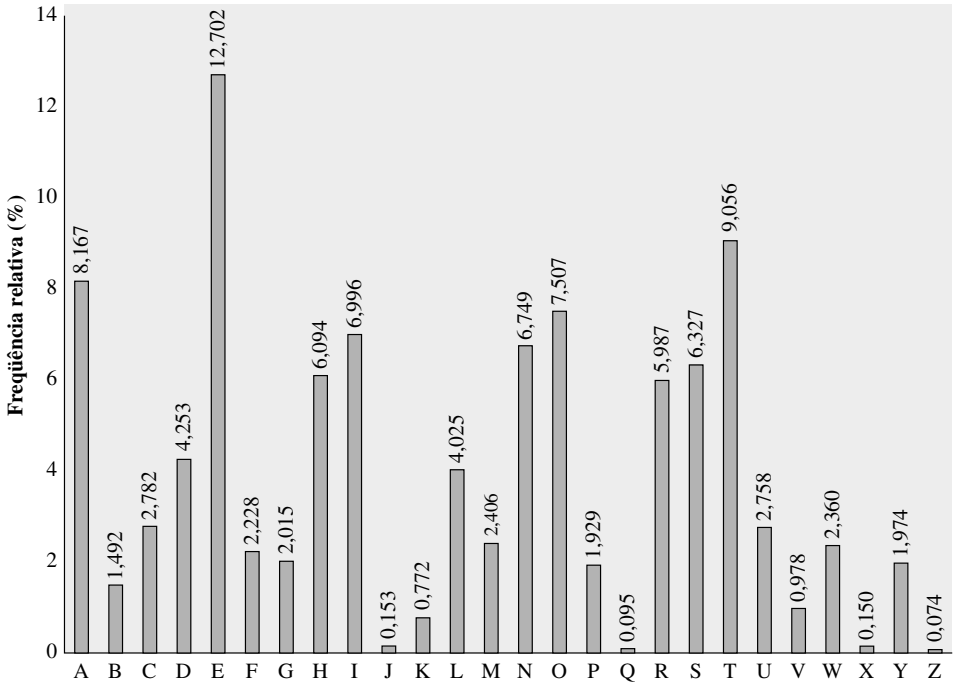


Figura 2.5 Freqüência relativa de letras no texto em inglês.

Existem diversas maneiras de prosseguir neste ponto. Poderíamos fazer algumas tentativas de atribuição e começar a preencher o texto claro para ver se ele tem um 'esqueleto' razoável de uma mensagem. Uma técnica mais sistemática é procurar outras regularidades. Por exemplo, é possível saber que certas palavras estão no texto. Ou então poderíamos procurar seqüências repetidas de letras cifradas e tentar deduzir seus equivalentes no texto claro.

Uma ferramenta poderosa é examinar a freqüência de combinações de duas letras, conhecidas como digramas. Uma tabela semelhante à Figura 2.5 poderia ser montada para mostrar a freqüência relativa dos digramas. O mais comum desses digramas é th. Em nosso texto cifrado, o digrama mais comum é ZW, que aparece três vezes. Assim, fazemos a correspondência de Z com t e de W com h. Depois, pela nossa hipótese anterior, podemos igualar P com e. Agora observe que a seqüência ZWP aparece no texto cifrado, sendo assim podemos traduzir essa seqüência como "the". Esse é o trigrama (combinação de três letras) mais freqüente em inglês, o que parece indicar que estamos no caminho certo.

Em seguida, observe a seqüência ZWSZ na primeira linha. Não sabemos se essas quatro letras formam uma palavra completa, mas, se formarem, ela tem a forma th_t. Nesse caso, S é igual a a.

Até aqui, então, temos:

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a        e  e te  a that e e a       a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
   e t   ta t ha e ee  a e  th    t  a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 e  e e tat  e   the   t
```

Somente quatro letras foram identificadas, mas já temos um bom pedaço da mensagem. A análise continuada das freqüências mais a tentativa e erro deverão facilmente gerar uma solução a partir desse ponto. O texto claro completo, com espaços incluídos entre as palavras, é o seguinte:

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

As cifras monoalfabéticas são fáceis de serem quebradas porque refletem os dados de freqüência do alfabeto original. Uma contramedida é oferecer vários substitutos, conhecidos como homófonos, para uma única letra. Por exemplo, a letra e poderia ser atribuída a diversos símbolos de cifra diferentes, como 16, 74, 35 e 21, com cada homófono usado em rotação, ou aleatoriamente. Se o número de símbolos atribuídos a cada letra for proporcional à freqüência relativa dessa letra, então a informação de freqüência de única letra será completamente extinta. O grande matemático Carl Friedrich Gauss acreditava ter criado uma cifra indecifrável usando homófonos. Porém, até mesmo com homófonos, cada elemento do texto claro afeta somente um elemento do texto cifrado, e padrões de múltiplas letras (por exemplo, freqüências de digrama) ainda sobrevivem no texto cifrado, tornando a criptoanálise relativamente simples.

Dois métodos principais são usados nas cifras de substituição para reduzir a extensão à qual a estrutura do texto claro sobrevive no texto cifrado. Uma técnica é criptografar várias letras do texto claro, e a outra é usar vários alfabetos de cifra. Examinemos rapidamente cada uma delas.

Cifra Playfair

A cifra de criptografia de múltiplas letras mais conhecida é a Playfair, que trata os digramas no texto claro como unidades isoladas e traduz essas unidades em digramas de texto cifrado.[3]

O algoritmo Playfair é baseado no uso de uma matriz 5 X 5 de letras construída usando uma palavra-chave. Aqui está um exemplo, solucionado por Lord Peter Wimsey em *Have his carcase*, de Dorothy Sayers:[4]

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Nesse caso, a palavra-chave é *monarchy*. A matriz é construída pelo preenchimento das letras da palavra-chave (menos duplicatas) da esquerda para a direita e de cima para baixo, e depois pelo preenchimento do restante da matriz com as letras restantes na ordem alfabética. As letras I e J contam como uma letra. O texto claro é criptografado com duas letras de cada vez, de acordo com as seguintes regras:

3. Essa cifra, na realidade, foi inventada pelo cientista britânico Sir Charles Wheatstone em 1854, mas recebeu o nome de seu amigo, Barão Playfair de St. Andrews, que defendeu a cifra no ministério de assuntos externos da Grã Bretanha.
4. O livro fornece um relato fascinante de um ataque de palavra provável.

1. Letras de texto claro repetidas no mesmo par são separadas por uma letra de preenchimento, como x, de modo que balloon seria tratado como ba lx lo on.
2. Duas letras de texto claro que se encontrem na mesma linha da matriz são substituídas pela letra à direita, com o primeiro elemento da linha vindo após o último, de forma circular. Por exemplo, ar é criptografado como RM.
3. Duas letras de texto claro que se encontrem na mesma coluna são substituídas pela letra abaixo, com o elemento de cima da coluna vindo após o último, de forma circular. Por exemplo, mu é criptografado como CM.
4. Caso contrário, cada letra de texto claro em um par é substituída pela letra que se encontra em sua própria linha e na coluna ocupada pela outra letra de texto claro. Assim, hs torna-se BP e ea torna-se IM (ou JM, a critério do criptografador).

A cifra Playfair é um grande avanço em relação às cifras monoalfabéticas simples. Por um lado, enquanto existem apenas 26 letras, existem $26 \times 26 = 676$ digramas, de modo que a identificação de digramas individuais é mais difícil. Além disso, as freqüências relativas das letras individuais exibem um intervalo muito maior do que o dos digramas, tornando a análise de freqüência muito mais difícil. Por esses motivos, a cifra Playfair foi, por muito tempo, considerada indecifrável. Ela foi usada como sistema de campo padrão pelo exército britânico na Primeira Guerra, e ainda era usada consideravelmente pelo exército dos EUA e outras forças aliadas durante a Segunda Guerra.

Apesar desse nível de confiança em sua segurança, a cifra Playfair é relativamente fácil de ser quebrada, pois ainda deixa intacta grande parte da estrutura da linguagem de texto claro. Algumas centenas de letras de texto cifrado geralmente são suficientes.

Um modo de revelar a eficácia da Playfair e outras cifras aparece na Figura 2.6, baseada em [SIMM93]. A linha rotulada com texto claro indica a distribuição de freqüência de mais de setenta mil caracteres alfabéticos no artigo da *Encyclopaedia Brittanica* sobre criptologia.[5] Essa também é a distribuição de freqüência de qualquer cifra por substituição monoalfabética. O gráfico foi desenvolvido da seguinte maneira. O número de ocorrência de cada letra no texto foi contado e dividido pelo número de ocorrências da letra e (a letra mais utilizada). Como resultado, e tem uma freqüência relativa de 1, t de aproximadamente 0,76, e assim por diante. Os pontos no eixo horizontal correspondem às letras na ordem de freqüência decrescente.

A Figura 2.6 também mostra a distribuição de freqüência que resulta quando o texto é criptografado usando a cifra Playfair. Para normalizar o gráfico, o número de ocorrências de cada letra no texto cifrado novamente foi dividido pelo número de ocorrências de e no texto claro. O gráfico resultante, portanto, ilustra em que extensão a distribuição de freqüência das letras, que torna trivial solucionar as cifras de substituição, é mascarada pela criptografia. Se a informação de distribuição de freqüência fosse totalmente ocultada no processo de criptografia, o gráfico de freqüências do texto cifrado seria plano, e a criptoanálise usando apenas o texto cifrado seria efetivamente impossível. Como mostra a figura, a cifra Playfair tem uma distribuição mais plana do que o texto claro, mas apesar disso ela revela muita estrutura para um criptoanalista trabalhar.

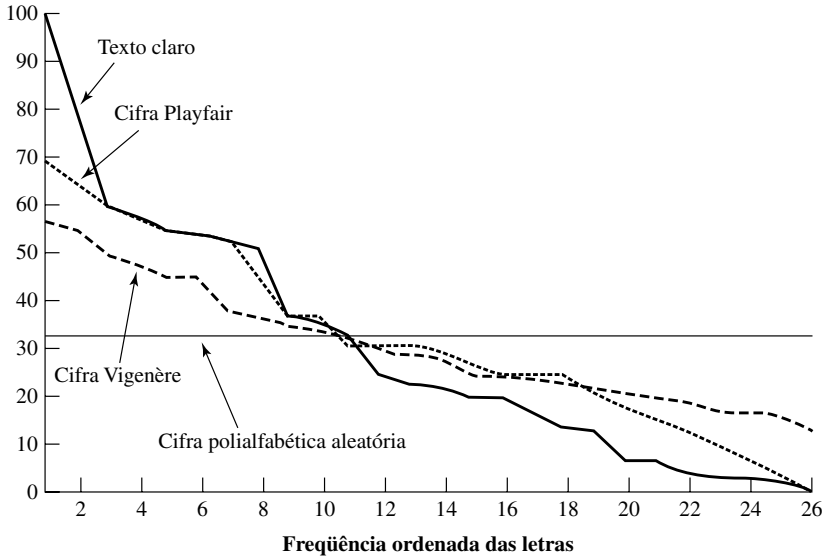


Figura 2.6 Freqüência relativa de ocorrência das letras.

5. Sou grato a Gustavus Simmons por fornecer os gráficos e explicar seu método de construção.

Cifra de Hill[6]

Outra cifra multiletras interessante é a cifra de Hill, desenvolvida pelo matemático Lester Hill em 1929. O algoritmo de criptografia utiliza m letras de texto claro sucessivas e as substitui por m letras de texto cifrado. A substituição é determinada por m equações lineares, em que cada caractere recebe um valor numérico (a = 0, b = 1, ... z = 25). Para $m = 3$, o sistema pode ser descrito da seguinte forma:

$$\begin{aligned} c_1 &= (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26 \\ c_2 &= (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26 \\ c_3 &= (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26 \end{aligned}$$

Isso pode ser expresso em termos de vetores coluna e matrizes:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

ou

$$\mathbf{C} = \mathbf{KP} \bmod 26$$

onde **C** e **P** são vetores de coluna de tamanho 3, representando o texto claro e o texto cifrado, e **K** é uma matriz 3×3, representando a chave de criptografia. As operações são realizadas em mod 26.

Por exemplo, considere o texto claro "paymoremoney" e use a chave de criptografia

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

As três primeiras letras do texto claro são representadas pelo vetor $\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$.

Then $\mathbf{K}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} = \text{LNS}$. Continuando dessa forma, o texto cifrado para o texto claro inteiro é LNSHDLEWMTRW.

A decriptografia exige o uso do inverso da matriz **K**. O inverso $\mathbf{K}^{-1}$ de uma matriz **K** é definido pela equação $\mathbf{KK}^{-1} = \mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$, onde **I** é a matriz contendo apenas zeros, exceto pelos 1s ao longo da diagonal principal do canto superior esquerdo ao inferior direito. O inverso de uma matriz nem sempre existe, mas, quando existe, satisfaz a equação dada. Nesse caso, o inverso é

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

Isso é demonstrado da seguinte maneira:

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

É fácil ver que, se a matriz $\mathbf{K}^{-1}$ for aplicada ao texto cifrado, então o texto claro será recuperado. Para explicar como é determinado o inverso de uma matriz, faremos uma incursão bem rápida pela álgebra linear.[7] Para qualquer matriz quadrada ($m \times m$), o determinante é igual à soma de todos os produtos que podem ser formados apanhando-se

6. Essa cifra é um pouco mais difícil de entender do que as outras deste capítulo, mas ilustra um ponto importante sobre a criptoanálise, que será útil mais adiante. Esta subseção pode ser ignorada em uma primeira leitura.

7. Os conceitos básicos da álgebra linear estão resumidos no documento *Math refresher*, no site Web "Computer science student resource", em WilliamStallings.com/StudentSupport.hmtl. O leitor interessado pode consultar qualquer texto sobre álgebra linear pra obter mais deta-

exatamente um elemento de cada linha e exatamente um elemento de cada coluna, com certos termos do produto precedidos por um sinal de subtração. Para uma matriz 2×2

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

o determinante é $k_{11}k_{22} - k_{12}k_{21}$. Para uma matriz 3×3, o valor do determinante é $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$. Se uma matriz quadrada **A** tiver um determinante diferente de zero, então o inverso da matriz será calculado como $[\mathbf{A}^{-1}]ij = (-1)^{i+j}(\mathrm{D}ji)/\det(\mathbf{A})$, onde $(\mathrm{D}ij)$ é o subdeterminante formado pela exclusão da linha i e coluna j de **A** e $\det(\mathbf{A})$ é o determinante de **A**. Para os nossos propósitos, toda a aritmética é feita com mod 26.

Em termos gerais, o sistema de Hill pode ser expresso da seguinte forma:

$$\mathbf{C} = \mathrm{E}(\mathbf{K}, \mathbf{P}) = \mathbf{KP} \bmod 26$$
$$\mathbf{P} = \mathrm{D}(\mathbf{K}, \mathbf{P}) = \mathbf{K}^{-1}\mathbf{C} \bmod 26 = \mathbf{K}^{-1}\mathbf{KP} = \mathbf{P}$$

Assim como na cifra Playfair, o ponto forte da cifra de Hill é que ela oculta completamente as freqüências de única letra. De fato, com Hill, o uso de uma matriz maior oculta mais informações de freqüência. Assim, uma cifra de Hill de 3×3 oculta não apenas informações de freqüência de uma única letra, mas também de duas letras.

Embora a cifra de Hill seja forte contra um ataque apenas de texto cifrado, ela é facilmente quebrada com um ataque de texto claro conhecido. Para uma cifra de Hill de $m \times m$, suponha que tenhamos M pares de texto claro/texto cifrado, cada um com tamanho m. Rotulamos os pares com $\mathbf{P}_j = \begin{pmatrix} p_{1j} \\ p_{2j} \\ \vdots \\ p_{mj} \end{pmatrix}$ e $\mathbf{C}_j = \begin{pmatrix} c_{1j} \\ c_{2j} \\ \vdots \\ c_{mj} \end{pmatrix}$, de modo que $\mathbf{C}_j = \mathbf{KP}_j$ para $1 \le j \le m$ e para alguma matriz de chave desconhecida K. Agora, defina duas matrizes $m \times m$, $\mathbf{X} = (p_{ij})$ e $\mathbf{Y} = (c_{ij})$. Depois, podemos formar a equação de matriz $\mathbf{Y} = \mathbf{KX}$. Se **X** tiver um inverso, então poderemos determinar $\mathbf{K} = \mathbf{YX}^{-1}$. Se **X** não puder ser invertida, então uma nova versão de **X** poderá ser formada com pares adicionais de texto claro/texto cifrado até que se obtenha uma matriz **X** que possa ser invertida.

Usamos um exemplo com base em [STIN02]. Suponha que o texto claro *friday* seja criptografado usando uma cifra de Hill 2×2 para gerar o texto cifrado PQCFKU. Assim, sabemos que $\mathbf{K}\begin{pmatrix} 5 \\ 17 \end{pmatrix} \bmod 26 = \begin{pmatrix} 15 \\ 16 \end{pmatrix}$; $\mathbf{K}\begin{pmatrix} 8 \\ 3 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$; e $\mathbf{K}\begin{pmatrix} 0 \\ 24 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$. Usando os dois primeiros pares de texto claro/texto cifrado, temos

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \mathbf{K}\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

O inverso de X pode ser calculado:

$$\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

de modo que

$$\mathbf{K} = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix}\begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

Esse resultado é verificado testando-se o par restante de texto claro/texto cifrado.

Cifras polialfabéticas

Outra forma de melhorar a técnica não alfabética simples é usar diferentes substituições não alfabéticas enquanto se prossegue pela mensagem de texto claro. O nome geral para essa técnica é cifra por substituição polialfabética. Todas essas técnicas têm as seguintes características em comum:

1. Um conjunto de regras de substituição monoalfabéticas relacionadas é utilizada.
2. Uma chave determina qual regra específica é escolhida para determinada transformação.

O mais conhecido, e um dos mais simples desses algoritmos, é denominado cifra de Vigenère. Nesse esquema, o conjunto de regras de substituição monoalfabéticas consiste nas 26 cifras de César, com deslocamentos de 0 a 25. Cada cifra é indicada por uma letra-chave, que é a letra do texto cifrado que substitui a letra do texto claro a. Assim, uma cifra de César com um deslocamento de 3 é indicada pelo valor de chave *d*.

Para ajudar na compreensão do esquema e auxiliar em seu uso, é construída uma matriz conhecida como tabela de Vigenère (Tabela 2.3). Cada uma das 25 cifras é disposta horizontalmente, com a letra-chave para cada cifra à sua esquerda. Um alfabeto normal para o texto claro aparece no topo. O processo de criptografia é simples: dada uma letra-chave *x* e uma letra de texto claro y, a letra do texto cifrado está na interseção da linha rotulada com *x* e a coluna rotulada com y; nesse caso, o texto cifrado é V.

Para criptografar uma mensagem, é preciso que haja uma chave tão longa quanto a mensagem. Normalmente, a chave é uma palavra-chave repetida. Por exemplo, se a palavra-chave for *deceptive*, a mensagem *we are discovered save yourself* é codificada desta forma:

```
chave:            deceptivedeceptivedeceptive
texto claro:      wearediscoveredsaveyourself
trxto cifrado:    ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

A decriptografia é igualmente simples. A letra-chave novamente identifica a linha. A posição da letra do texto cifrado nessa linha determina a coluna, e a letra do texto claro está no topo dessa coluna.

A força dessa cifra é que existem múltiplas letras de texto cifrado para cada letra do texto claro, uma para cada letra exclusiva da palavra-chave. Assim, a informação de freqüência de letra é ocultada. Porém, nem todo o conhecimento da estrutura do texto claro é perdido. Por exemplo, a Figura 2.6 mostra a distribuição de freqüência para uma cifra de Vigenère com uma palavra-chave de tamanho 9. Uma melhoria é obtida em relação à cifra Playfair, mas consideráveis informações de freqüência ainda permanecem.

É instrutivo esboçar um método para quebrar essa cifra, pois o método revela alguns dos princípios matemáticos que se aplicam à criptoanálise.

Primeiro, suponha que o oponente acredite que o texto cifrado foi criptografado usando a substituição monoalfabética ou uma cifra de Vigenère. Um teste simples pode ser feito para fazer uma determinação. Se uma substituição monoalfabética for usada, então as propriedades estatísticas do texto cifrado deverão ser iguais às da linguagem do texto claro. Assim, referenciando a Figura 2.5, deverá haver uma letra de cifra com uma freqüência relativa de ocorrência de aproximadamente 12,7%, uma com cerca de 9,06% e assim por diante. Se apenas uma única mensagem estiver disponível para análise, não esperaríamos uma coincidência exata dessa pequena amostra com um perfil estatístico da linguagem de texto claro. Apesar disso, se a correspondência for próxima, podemos considerar uma substituição monoalfabética.

Se, por outro lado, uma cifra de Vigenère for suspeita, então o progresso depende da determinação do tamanho da palavra-chave, como veremos em breve. Por enquanto, vamos nos concentrar em como podemos determinar o tamanho da palavra-chave. A idéia importante que leva a uma solução é a seguinte: se duas seqüências idênticas de letras de texto claro ocorrem a uma distância que seja um múltiplo inteiro do tamanho da palavra-chave, elas gerarão seqüências de texto cifrado idênticas. No exemplo anterior, duas ocorrências da seqüência *red* são separadas por nove posições de caractere. Conseqüentemente, em ambos os casos, *r* é criptografado usando a letra-chave *e*, e é criptografado usando a letra-chave *p*, e d é criptografado usando a letra-chave *t*. Assim, nos dois casos, a seqüência do texto cifrado é VTW.

Um analista examinando apenas o texto cifrado detectaria as seqüências repetidas VTW a uma distância de 9 e consideraria que a palavra-chave tem três ou nove letras de extensão. O surgimento de VTW duas vezes poderia ser por acaso, e não refletir letras de texto claro idênticas codificadas com letras-chave idênticas. Porém, se a mensagem for longa o suficiente, haverá diversas dessas seqüências de texto cifrado repetidas. Procurando fatores comuns nos deslocamentos das diversas seqüências, o analista deverá ser capaz de fazer uma boa escolha do tamanho da palavra-chave.

A solução da cifra agora depende de uma idéia importante. Se o tamanho da palavra-chave for *N*, então a cifra, por conseguinte, consiste em *N* cifras de substituição monoalfabéticas. Por exemplo, com a palavra-chave DECEPTIVE, as letras nas posições 1, 10, 19 e assim por diante são todas codificadas com a mesma cifra monoalfabética. Assim, podemos usar as características de freqüência conhecidas da linguagem do texto claro para atacar cada uma das cifras monoalfabéticas separadamente.

A natureza periódica da palavra-chave pode ser eliminada com o uso uma palavra-chave não repetida que seja tão grande quanto a própria mensagem. Vigenère propôs o que é conhecido como um **sistema de autochave**, em que uma palavra-chave é concatenada ao próprio texto claro para oferecer uma chave corrente. Para o nosso exemplo,

```
chave:           deceptivewearediscoveredsav
texto claro:     wearediscoveredsaveyourself
texto cifrado:   ZICVTWQNGKZEIIGASXSTSLVVWLA
```

Tabela 2.3 A tabela de Vigenère moderna

		Texto claro																								
		a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
Chave	*a*	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Até mesmo esse esquema é vulnerável à criptoanálise. Como a chave e o texto claro compartilham a mesma distribuição de freqüência das letras, uma técnica estatística poderá ser aplicada. Por exemplo, pode-se esperar que e criptografado por *e*, pela Figura 2.5, ocorra com uma freqüência de $(0{,}127)^2 \approx 0{,}016$, enquanto t criptografado por *t* ocorra apenas com a metade dessa freqüência. Essas regularidades podem ser exploradas para conseguir uma criptoanálise bem-sucedida.[8]

A principal defesa contra essa criptoanálise é escolher uma palavra-chave que seja tão longa quanto o texto claro e não possua relacionamento estatístico com ele. Esse sistema foi apresentado por um engenheiro da AT&T, chamado Gilbert Vernam, em 1918. Seu sistema funciona com base em dados binários, em vez de letras. O sistema pode ser expresso de forma sucinta da seguinte maneira:

$$c_i = p_i \oplus k_i$$

onde

$$p_i = i - \text{ésimo dígito binário do texto claro}$$
$$k_i = i - \text{ésimo dígito binário da chave}$$
$$c_i = i - \text{ésimo dígito binário do texto cifrado}$$
$$\oplus = \text{operação ou-exclusivo (XOR)}$$

Assim, o texto cifrado é gerado realizando-se o XOR bit a bit entre texto claro e a chave. Devido às propriedades do XOR, a decriptografia simplesmente envolve a mesma operação bit a bit:

$$p_i = c_i \oplus k_i$$

A essência dessa técnica é o meio de construção da chave. Vernam propôs o uso de uma fita circular em rotação que repetia periodicamente a chave, de modo que o sistema funcionava com uma palavra-chave muito longa, porém repetitiva. Embora esse esquema, com uma chave longa, apresente dificuldades de criptoanálise formidáveis, ele pode ser quebrado com texto cifrado suficiente, o uso de seqüências de texto claro conhecidas ou prováveis, ou ambos.

One-Time Pad (chave de uso único)

Um oficial do serviço de comunicação do Exército dos Estados Unidos, Joseph Mauborgne, propôs uma melhoria na cifra de Vernam, que gera o máximo em segurança. Mauborgne sugeriu o uso de uma chave aleatória que fosse tão grande quanto a mensagem, de modo que a chave não precisasse ser repetida. Além disso, a chave deve ser usada para criptografar e decriptografar uma única mensagem e depois ser descartada. Cada nova mensagem exige uma nova chave com o mesmo tamanho da nova mensagem. Esse esquema, conhecido como one-time pad, é inquebrável. Ele produz saída aleatória que não possui nenhum relacionamento estatístico com o texto claro. Como o texto cifrado não contém informação sobre o texto claro, simplesmente não existe um meio de quebrar o código.

Um exemplo deverá ilustrar nosso argumento. Suponha que estejamos usando um esquema de Vigenère com 27 caracteres, em que o vigésimo sétimo caractere é o caractere de espaço, mas com uma chave de uso único que é tão grande quanto a mensagem. Assim, a Tabela 2.3 precisa ser expandida para 27 X 27. Considere o texto cifrado

```
ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
```

Agora mostramos duas decodificações diferentes usando duas chaves diferentes:

```
texto cifrado:   ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
chave:           pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
texto claro:     mr mustard with the candlestick in the hall

texto cifrado:   ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
chave:           mfugpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt
texto claro:     miss scarlet with the knife in the library
```

Suponha que um criptoanalista conseguisse encontrar essas duas chaves. Dois textos claros plausíveis são produzidos. Como ele decide qual é a decriptografia correta (ou seja, qual é a chave correta)? Se a chave real fosse produzida de uma forma verdadeiramente aleatória, então o criptoanalista não poderia saber que uma dessas duas chaves seria mais provável que a outra. Assim, não existe um meio de decidir qual chave é a correta e, portanto, qual texto claro é o correto.

8. Embora as técnicas para quebrar uma cifra de Vigenère de forma alguma sejam complexas, uma edição de 1917 de *Scientific American* caracterizava esse sistema como "impossível de ser traduzido". Esse é um ponto que merece ser lembrado quando afirmações semelhantes são feitas para algoritmos modernos.

De fato, dado qualquer texto claro de tamanho igual ao do texto cifrado, existe uma chave que produz esse texto claro. Portanto, se você fizesse uma busca exaustiva de todas as chaves possíveis, acabaria com muitos textos claros legíveis, sem saber qual foi o texto claro pretendido. Portanto, o código é inquebrável.

A segurança da cifra de chave de uso único deve-se inteiramente à aleatoriedade da chave. Se o fluxo de caracteres que constitui a chave for verdadeiramente aleatório, então o fluxo de caracteres que constitui o texto cifrado será verdadeiramente aleatório. Assim, não existem padrões ou regularidades que um criptoanalista possa usar para atacar o texto cifrado.

Em teoria, não precisamos procurar mais uma cifra. A cifra de chave de uso único oferece segurança completa, mas, na prática, tem duas dificuldades fundamentais:

1. Existe o problema prático de criar grandes quantidades de chaves aleatórias. Qualquer sistema bastante utilizado poderia exigir milhões de caracteres aleatórios regularmente. O fornecimento de caracteres verdadeiramente aleatórios nesse volume é uma tarefa significativa.
2. Ainda mais assustador é o problema da distribuição e proteção de chaves. Para cada mensagem a ser enviada, uma chave de mesmo tamanho é necessária tanto pelo emissor quanto pelo receptor. Assim, existe o problema gigantesco de distribuição de chaves.

Por causa dessas dificuldades, a cifra de chave de uso único tem utilidade limitada, e é útil principalmente para canais de pouca largura de banda, exigindo segurança muito alta.

2.3 TÉCNICAS DE TRANSPOSIÇÃO

Todas as técnicas examinadas até aqui implicam a substituição de um símbolo de texto cifrado por um símbolo de texto claro. Uma espécie bem diferente de mapeamento é obtida pela realização de algum tipo de permutação nas letras do texto claro. Essa técnica é referenciada como uma cifra de transposição.

A cifra mais simples desse tipo é a técnica de *rail fence*, em que o texto claro é escrito como uma seqüência de diagonais e depois lido como uma seqüência de linhas. Por exemplo, para cifrar a mensagem *meet me after the toga party* com a cifra rail fence de profundidade 2, escrevemos o seguinte:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

A mensagem criptografada é

MEMATRHTGPRYETEFETEOAAT

Esse tipo de coisa seria trivial para a criptoanálise. Um esquema mais complexo é escrever a mensagem em um retângulo, linha por linha, e ler a mensagem coluna por coluna, mas permutar a ordem das colunas. A ordem das colunas, então, torna-se a chave para o algoritmo. Por exemplo,

```
Chave:          4 3 1 2 5 6 7
Texto claro:    a t t a c k p
                o s t p o n e
                d u n t i l t
                w o a m x y z
Texto cifrado:  TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Uma cifra de pura transposição é facilmente reconhecida, pois tem as mesmas freqüências de letra do texto claro original. Para o tipo de transposição por colunas mostrada, a criptoanálise é muito simples e envolve dispor o texto cifrado em uma matriz e trabalhar com as posições de coluna. As tabelas de freqüência de digrama e trigrama podem ser úteis.

A cifra de transposição pode se tornar muito mais segura realizando-se mais de um estágio de transposição. O resultado é uma permutação mais complexa, que não é facilmente reconstruída. Assim, se a mensagem anterior for recodificada usando o mesmo algoritmo

```
KChave:         4 3 1 2 5 6 7
Entrada:        t t n a a p t
                m t s u o a o
                d w c o i x k
                n l y p e t z
Saída:          NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

Para visualizar o resultado dessa dupla transposição, designe as letras na mensagem de texto claro original pelos números projetando sua posição. Assim, com 28 letras na mensagem, a seqüência original das letras é

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

Depois da primeira transposição, temos

```
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28
```

que tem uma estrutura um tanto regular. Mas, depois da segunda transposição, temos

```
17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28
```

Essa é uma permutação muito menos estruturada e é muito mais difícil de criptoanalisar.

2.4 MÁQUINAS DE ROTOR

O exemplo que acabamos de fornecer sugere que vários estágios de criptografia podem produzir um algoritmo significativamente mais difícil de criptoanalisar. Isso vale tanto para cifras de substituição quanto para cifras de transposição. Antes da introdução do DES, a aplicação mais importante do princípio de múltiplos estágios de criptografia era uma classe de sistemas conhecida como máquinas de rotor.[9]

O princípio básico da máquina de rotor é ilustrado na Figura 2.7. A máquina consiste em um conjunto de cilindros rotativos independentes, através dos quais pulsos elétricos podem fluir. Cada cilindro tem 26 pinos de entrada e 26 pinos de saída, com fiação interna que conecta cada pino de entrada a um único pino de saída. Para simplificar, mostramos somente três das conexões internas em cada cilindro.

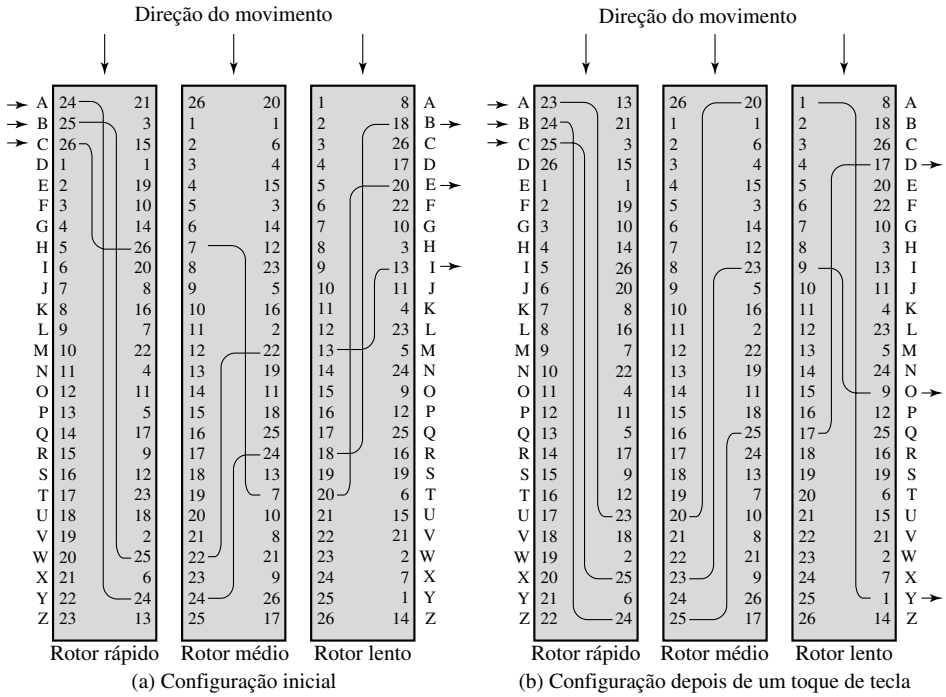


Figura 2.7 Máquina de três rotores com fiação representada por contatos numerados.

Se associarmos cada pino de entrada e saída a uma letra do alfabeto, então um único cilindro define uma substituição monoalfabética. Por exemplo, na Figura 2.7, se um operador pressionar uma tecla para a letra A, um sinal elétrico é aplicado ao primeiro pino do primeiro cilindro e flui pela conexão interna para o vigésimo quinto pino de saída.

9. Máquinas baseadas no princípio de rotor foram usadas pela Alemanha (Enigma) e pelo Japão (Purple) na Segunda Guerra Mundial. A quebra desses dois códigos pelos Aliados foi um fator significativo para o resultado da guerra.

Considere uma máquina com um único cilindro. Depois que cada tecla de entrada é pressionada, o cilindro gira uma posição, de modo que as conexões internas são deslocadas de acordo. Assim, é definida uma cifra de substituição monoalfabética diferente. Depois de 26 letras de texto claro, o cilindro estaria de volta à posição inicial. Assim, temos um algoritmo de substituição polialfabética com um período de 26.

Um sistema de cilindro único é trivial e não apresenta uma tarefa criptoanalítica formidável. O poder da máquina de rotor está no uso de múltiplos cilindros, em que os pinos de saída de um cilindro são conectados aos pinos de entrada do seguinte. A Figura 2.7 mostra um sistema de três cilindros. A metade esquerda da figura mostra uma posição em que a entrada do operador para o primeiro pino (letra a em texto claro) é direcionada pelos três cilindros para aparecer na saída do segundo pino (letra B em texto cifrado).

Com múltiplos cilindros, aquele mais próximo da entrada do operador gira uma posição de pino a cada toque de tecla. A metade direita da Figura 2.7 mostra a configuração do sistema depois de um único toque de tecla. Para cada rotação completa do cilindro interno, o cilindro do meio gira uma posição de pino. Finalmente, para cada rotação completa do cilindro do meio, o cilindro externo gira uma posição de pino. Esse é o mesmo tipo de operação vista em um odômetro de automóvel. O resultado é que existem 26 X 26 X 26 = 17.576 alfabetos de substituição diferentes usados antes que o sistema se repita. O acréscimo do quarto e quinto rotores resulta em períodos de 456.976 e 11.881.376 letras, respectivamente. Conforme David Kahn coloca eloqüentemente, referindo-se a uma máquina de cinco rotores [KAHN96, p. 413]:

> Um período desse tamanho impede qualquer possibilidade prática de uma solução direta com base na freqüência da letra. Essa solução geral precisaria de cerca de 50 letras por alfabeto cifrado, significando que todos os cinco rotores teriam de passar por seu ciclo combinado 50 vezes. O texto cifrado teria de ser tão longo quanto todos os discursos feitos no andar térreo do Senado e da Câmara em três sessões sucessivas do Congresso. Provavelmente nenhum criptoanalista levaria esse tipo de troféu durante seu tempo de vida; até mesmo os diplomatas, que podem ser tão eloqüentes quanto os políticos, raramente chegam a esse ponto.

O significado da máquina de rotor hoje é que ela aponta o caminho para a cifra mais utilizada de todos os tempos: Data Encryption Standard (DES). Esta será explicada no Capítulo 3.

2.5 ESTEGANOGRAFIA

Concluímos com uma explicação sobre uma técnica que, estritamente falando, não é criptografia, a saber, a esteganografia.

Uma mensagem em texto claro pode estar oculta de duas maneiras. Os métodos de esteganografia escondem a existência da mensagem, enquanto os métodos de criptografia tornam a mensagem ininteligível a estranhos por meio de várias transformações do texto.[10]

Uma forma simples de esteganografia, mas que é demorada de construir, é aquela em que um arranjo de palavras e letras dentro de um texto aparentemente inofensivo soletra a mensagem real. Por exemplo, a seqüência de primeiras letras de cada palavra da mensagem geral soletra a mensagem escondida. A Figura 2.8 mostra um exemplo em que um subconjunto das palavras da mensagem geral é usado para transmitir a mensagem oculta.

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your
letter and for the Summer examination package.
All Entry Forms and Fees Forms should be ready
for final despatch to the Syndicate by Friday
20th or at the very latest, I'm told. by the 21st.
Admin has improved here, though there's room
for improvement still; just give us all two or three
more years and we'll really show you! Please
don't let these wretched 16+ proposals destroy
your basis O and A pattern. Certainly this
sort of change, if implemented immediately,
would bring chaos.

Sincerely yours.

Figura 2.8 Um quebra-cabeças para o Inspetor Morse.
(The silent world of Nicholas Quinn, de Colin Dexter)

10. *Esteganografia* era uma palavra obsoleta que foi revivida por David Kahn e recebeu o significado que tem hoje [KAHN96].

Diversas outras técnicas têm sido usadas historicamente, sendo que alguns exemplos são:

- **Marcação de caractere:** Letras selecionadas do texto impresso ou datilografado são sobrescritas por lápis. As marcas normalmente não são visíveis, a menos que o papel seja mantido em ângulo com uma fonte de luz clara.
- **Tinta invisível:** Diversas substâncias podem ser usadas para a escrita, mas não deixam rastros visíveis, a menos que alguma química seja aplicada ao papel.
- **Perfurações:** Pequenos furos em letras selecionadas normalmente não são visíveis, a menos que o papel tenha uma fonte de luz no fundo.
- **Fita corretiva de máquina de escrever:** Usada entre as linhas digitadas com uma fita preta, os resultados da digitação com a fita corretiva são visíveis apenas sob uma luz forte.

Embora essas técnicas possam parecer arcaicas, elas possuem equivalentes contemporâneos. [WAYN93] propõe esconder uma mensagem usando os bits menos significativos dos frames em um CD. Por exemplo, a resolução máxima do formato Kodak Photo CD é de 2.048 por 3.072 pixels, com cada pixel contendo 24 bits de informações de cor RGB. O bit menos significativo de cada pixel de 24 bits pode ser alterado sem afetar muito a qualidade da imagem. O resultado é que você pode ocultar uma mensagem de 2,3 megabytes em um único instantâneo digital. Agora existem diversos pacotes de software disponíveis que levam esse tipo de técnica à esteganografia.

A esteganografia tem diversas desvantagens quando comparada à criptografia. Ela exige muito esforço para esconder relativamente poucos bits de informação, embora usando algum esquema como o proposto no parágrafo anterior possa torná-la mais eficaz. Além disso, quando o sistema é descoberto, ele se torna praticamente inútil. Esse problema também pode ser contornado se o método de inserção depender de algum tipo de chave (por exemplo, ver o Problema 2.11). Como alternativa, uma mensagem pode ser codificada e depois escondida usando a esteganografia.

A vantagem da esteganografia é que ela pode ser empregada pelas partes que têm algo a perder se o fato de sua comunicação secreta (não necessariamente o conteúdo) for descoberto. A criptografia sinaliza o tráfego como importante ou secreto, ou pode identificar o emissor ou o receptor como alguém com algo a esconder.

2.6 LEITURA E SITES WEB RECOMENDADOS

Para quem estiver interessado na história da criação de código e quebra de código, o livro a ser lido é [KAHN96]. Embora trate mais do impacto da criptologia do que de seu desenvolvimento técnico, essa é uma introdução excelente e uma leitura interessante. Outro relato histórico excelente é [SING99].

Uma abordagem curta, incluindo as técnicas deste capítulo e outras, é [GARD72]. Existem muitos livros que abordam a criptografia clássica em um estilo mais técnico; um dos melhores é [SINK66]. [KORN96] é um livro agradável de ler e contém uma extensa seção sobre técnicas clássicas. Dois livros de criptografia que contêm uma grande quantidade de material técnico sobre técnicas clássicas são [GARR01] e [NICH99]. Para o leitor verdadeiramente interessado, os dois volumes de [NICH96] abrangem diversas cifras clássicas com detalhes, e oferecem muitos textos cifrados a serem criptoanalisados, juntamente com as soluções.

Uma abordagem excelente das máquinas de rotor, incluindo uma explicação sobre sua criptoanálise, pode ser encontrado em [KUMA97].

[KATZ00] oferece uma abordagem completa da esteganografia. Outra fonte interessante é [WAYN96].

GARD72 Gardner, M. *Codes, ciphers, and secret writing*. Nova York: Dover, 1972.

GARR01 Garrett, P. *Making breaking codes: an introduction to cryptology*. Upper Saddle River: Prentice Hall, 2001.

KAHN96 Kahn, D. *The codebreakers: the story of secret writing*. Nova York: Scribner, 1996.

KATZ00 Katzenbeisser, S., ed. *Information Hiding techniques for steganography and digital watermarking*. Boston: Artech House, 2000.

KORN96 Korner, T. *The pleasures of counting*. Cambridge: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills: Aegean Park Press, 1997.

NICH96 Nichols, R. *Classical cryptography course*. Laguna Hills: Aegean Park Press, 1996.

NICH99 Nichols, R. ed. *ICSA Guide to cryptography*. Nova York: McGraw-Hill, 1999.

SING99 Singh, S. *The code book: the science of secrecy from ancient egypt to quantum cryptography*. Nova York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary cryptanalysis: a mathematical approach*. Washington: The Mathematical Association of America, 1966.

WAYN96 Wayner, P. *Disappearing cryptography*. Boston: AP Professional Books, 1996.

Sites Web recomendados:

- **American Cryptogram Association:** Uma associação de criptógrafos amadores. O site Web inclui informações e links para sites referentes à criptografia clássica.
- **Crypto Corner:** O site Web de Simon Singh. Muita informação boa e ferramentas interativas para aprender sobre criptografia.
- **Steganography:** Boa coleção de links e documentos.

2.7 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

ataque de força bruta
cifra
cifra de bloco
cifra de César
cifra de fluxo
cifra de Hill
cifra de transposição
cifra de Vigenère
cifra monoalfabética
cifra Playfair
cifra polialfabética
cifra rail fence
cifragem
computacionalmente seguro
criptografia
criptografia convencional
criptografia de única chave
criptografia simétrica
criptoanálise
criptografia
criptologia
decifragem
decriptografia
esteganografia
incondicionalmente seguro
one-time pad (cifra de chave de uso único)
sistema criptográfico
texto cifrado
texto claro

Perguntas de revisão

2.1 Quais são os ingredientes essenciais de uma cifra simétrica?
2.2 Quais são as duas funções básicas usadas nos algoritmos de criptografia?
2.3 Quantas chaves são necessárias para duas pessoas se comunicarem por meio de uma cifra?
2.4 Qual é a diferença entre uma cifra de bloco e uma cifra de fluxo?
2.5 Quais são as duas técnicas gerais para atacar uma cifra?
2.6 Liste e defina rapidamente os tipos de ataques criptoanalíticos com base naquilo que o atacante conhece.
2.7 Qual é a diferença entre uma cifra incondicionalmente segura e uma cifra computacionalmente segura?
2.8 Defina brevemente a cifra de César.
2.9 Defina brevemente a cifra monoalfabética.
2.10 Defina brevemente a cifra Playfair.
2.11 Qual é a diferença entre uma cifra monoalfabética e uma cifra polialfabética?
2.12 Quais são os dois problemas com a cifra de chave de uso único?
2.13 O que é uma cifra de transposição?
2.14 O que é esteganografia?

Problemas

2.1 Uma generalização da cifra de César, conhecida como cifra de César afim, tem a seguinte forma:

$$C = E([a, b], p) = (ap + b) \bmod 26$$

Substitua cada letra de texto claro p pela letra de texto cifrado C:
Um requisito básico de qualquer algoritmo de criptografia é que ele seja um-para-um. Ou seja, se $p \neq q$, então $E(k, p) \neq E(k, q)$. Caso contrário, a decriptografia será impossível, pois mais de um caractere de texto claro é mapeado no mesmo caractere de texto cifrado. A cifra de César afim não é um-para-um para todos os valores de a. Por exemplo, para $a = 2$ e $b = 3$, então $E([a, b], 0) = E([a, b], 13) = 3$.

a. Existem limitações sobre o valor de b? Explique por que sim ou não.
b. Determine quais valores de a não são permitidos.
c. Ofereça uma condição geral que indique quais valores de a são e não são permitidos. Justifique sua afirmação.

2.2 Quantas cifras de César afins existem?
2.3 Um texto cifrado foi gerado com uma cifra afim. A letra mais freqüente do texto cifrado é B, e a segunda letra mais freqüente do texto cifrado é U. Quebre esse código.
2.4 O texto cifrado a seguir foi gerado usando um algoritmo de substituição simples:

53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;;]8*;:‡*8†83
(88)5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956*2(5*-4)8‡8*
;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4)485†528806*81
(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;

Decodifique essa mensagem. *Dicas:*

1. Como você sabe, a letra que ocorre com mais freqüência em inglês é e. Portanto, o primeiro ou segundo (ou talvez terceiro?) caractere mais comum na mensagem provavelmente signifique e. Além disso, e normalmente é visto em pares (por exemplo, meet, fleet, speed, seen, been, agree, etc.). Tente encontrar um caractere no texto cifrado que seja decriptografado para e.
2. A palavra mais comum em inglês é *the*. Use esse fato para adivinhar os caracteres que representam t e h.
3. Decifre o restante da mensagem deduzindo outras palavras.

Aviso: a mensagem resultante está em inglês, mas pode não fazer muito sentido em uma primeira leitura.

2.5 Um modo de solucionar o problema de distribuição de chave é usar uma linha de um livro que o emissor e o receptor possuem. Normalmente, pelo menos em romances de espionagem, a primeira sentença de um livro serve como a chave. O esquema específico explicado neste problema é de um dos melhores romances de suspense envolvendo códigos secretos, *Talking to strange men*, de Ruth Rendell. Explique esse problema sem consultar esse livro!

Considere a mensagem a seguir:

SIDKHKDM AF HCRKIABIE
SHIMC KD LFEAILA

Esse texto cifrado foi produzido usando a primeira sentença de *The other side of silence* (um livro sobre o espião Kim Philby):

The snow lay thick on the steps and the snowflakes driven by the wind looked black in the headlights of the cars.

Uma cifra de substituição simples foi utilizada.

a. Qual é o algoritmo de criptografia?
b. Qual a sua segurança?
c. Para simplificar o problema de distribuição de chave, as duas partes podem combinar o uso da primeira ou da última sentença de um livro como chave. Para mudar a chave, eles simplesmente precisam combinar um novo livro. O uso da primeira sentença seria preferível ao uso da última. Por quê?

2.6 Em um de seus casos, Sherlock Holmes foi confrontado com a seguinte mensagem:

534 C2 13 127 36 31 4 17 21 41
DOUGLAS 109 293 5 37 BIRLSTONE
26 BIRLSTONE 9 127 171

Embora Watson estivesse confuso, Holmes foi capaz imediatamente de deduzir o tipo de cifra. Você consegue descobrir?

2.7 Este problema usa um exemplo do mundo real, de um antigo manual das Forças Especiais dos EUA (domínio público). Uma cópia está disponível em http://www.box.net/public/43yscu1ax2.

a. Usando as duas chaves (palavras de memória) *cryptographic* e *network security*, codifique a mensagem a seguir:
Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends.
Faça suposições razoáveis sobre como tratar letras redundantes e letras em excesso nas palavras de memória e como tratar os espaços e a pontuação. Indique quais são as suas suposições. *Nota:* a mensagem é do romance de Sherlock Holmes *The sign of four*.
b. Decriptografe o texto cifrado. Mostre seu trabalho.
c. Comente sobre quando seria apropriado usar essa técnica e quais são suas vantagens.

2.8 A desvantagem da cifra monoalfabética é que tanto o emissor quanto o receptor precisam confiar a seqüência da cifra permutada à memória. Uma técnica comum para evitar isso é usar uma palavra-chave a partir da qual a seqüência da cifra possa ser gerada. Por exemplo, usando a palavra-chave *CIPHER*, escreva a palavra-chave seguida por letras não usadas na ordem normal e combine com as letras do texto claro:

```
texto claro:     a b c d e f g h i j
k l m n o p q r s t u v w x y z
texto cifrado:   C I P H E R A B D F
G J K L M N O Q S T U V W X Y Z
```

Se esse processo não produzir uma mistura suficiente, escreva as letras restantes em linhas sucessivas e depois gere a seqüência lendo as colunas, de cima para baixo:

```
C I P H E R
A B D F G J
K L M N O Q
S T U V W X
Y Z
```

Isso gera a seqüência

```
C A K S Y I B L T Z P D M U H F N V E
G O W R J Q X
```

Esse sistema é usado no exemplo da Seção 2.2 (aquele que começa com "it was disclosed yesterday"). Determine a palavra-chave.

2.9 Quando o barco de patrulha americano PT-109, sob o comando do tenente John F. Kennedy, foi afundado por um destróier japonês, uma mensagem foi recebida na estação sem fio australiana em código Playfair:

```
KXJEY UREBE ZWEHE WRYTU HEYFS
KREHE GOYFI WTTTU OLKSY CAJPO
BOTEI ZONTX BYBNT GONEY CUZWR
GDSON SXBOU YWRHE BAAHY USEDQ
```

A chave usada foi *royal new zealand navy*. Decodifique a mensagem. Traduza TT como tt.

2.10 **a.** Construa uma matriz Playfair com a chave *largest*.

b. Construa uma matriz Playfair com a chave *occurrence*. Faça uma suposição razoável sobre como tratar letras redundantes na chave.

2.11 **a.** Usando esta matriz Playfair

M	F	H	I/J	K
U	N	O	P	Q
Z	V	W	X	Y
E	L	A	R	G
D	S	T	B	C

codifique esta mensagem:

Must see you over Cadogan West. Coming at once.

Nota: A mensagem é da história de Sherlock Holmes, *The Adventure of the Bruce-Partington plans*.

b. Repita a parte (a) usando a matriz Playfair do Problema 2.10a.

c. Como você justifica os resultados desse problema? Você poderia generalizar sua conclusão?

2.12 **a.** Quantas chaves possíveis a cifra Playfair possui? Ignore o fato de que algumas chaves podem produzir resultados de criptografia idênticos. Expresse sua resposta como uma potência aproximada de 2.

b. Agora leve em conta o fato de que algumas chaves Playfair produzem os mesmos resultados de criptografia. Quantas chaves efetivamente exclusivas a cifra Playfair possui?

2.13 Que sistema de substituição resulta quando usamos uma matriz Playfair de 25×1?

2.14 **a.** Decifre a mensagem YITJP GWJOW FAQTQ XCSMA ETSQU SQAPU SQGKC PQTYJ usando a cifra de Hill com a chave inversa $\begin{pmatrix} 5 & 1 \\ 2 & 7 \end{pmatrix}$. Mostre seus cálculos e o resultado. Use $a = 1, b = 2, ..., z = 26$.

b. Decifre a mensagem MWALO LIAIW WTGBH JNTAK QZJKA ADAWS SKQKU AYARN CSODN IIAES OQKJY B usando a cifra de Hill com a chave inversa $\begin{pmatrix} 2 & 23 \\ 21 & 7 \end{pmatrix}$. Mostre seus cálculos e o resultado.

2.15 **a.** Codifique a mensagem *meet me at the usual place at ten rather than eight oclock* usando a cifra de Hill com a chave $\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$. Mostre seus cálculos e o resultado.

b. Mostre os cálculos para a decriptografia correspondente do texto cifrado para recuperar o texto claro original.

2.16 Mostramos que a cifra de Hill não resiste a um ataque de texto claro conhecido se forem fornecidos pares de texto claro/texto cifrado suficientes. É ainda mais fácil de solucionar a cifra de Hill se um ataque de texto claro escolhido puder ser montado. Descreva esse ataque.

2.17 Pode-se mostrar que a cifra de Hill com a matriz $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ exige que (ad − bc) seja relativamente primo de 26; ou seja, o único fator positivo comum entre (ad − bc) e 26 é 1. Assim, se (ad − bc) = 13 ou se for par, a matriz não é permitida. Determine o número de chaves diferentes (boas) que existem para uma cifra de Hill 2×2 sem contá-las uma a uma, usando as seguintes etapas:

a. Descubra o número de matrizes cujo determinante seja par porque uma ou ambas as linhas são pares. (Uma linha será 'par' se as duas entradas na linha forem pares.)

b. Descubra o número de matrizes cujo determinante é par porque uma ou ambas as colunas são pares. (Uma coluna será 'par' se as duas entradas na coluna forem pares.)

c. Descubra o número de matrizes cujo determinante é par porque todas as entradas são ímpares.

d. Levando em conta as sobreposições, descubra o número total de matrizes cujo determinante é par.

e. Descubra o número de matrizes cujo determinante é um múltiplo de 13 porque a primeira coluna é um múltiplo de 13.

f. Descubra o número de matrizes cujo determinante é um múltiplo de 13, onde a primeira coluna não é um múltiplo de 13, mas a segunda coluna é um múltiplo da primeira módulo 13.

g. Descubra o número total de matrizes cujo determinante é um múltiplo de 13.

h. Descubra o número de matrizes cujo determinante é um múltiplo de 25 porque elas se ajustam ao caso (a) e (e). (b) e (e). (c) e (e). (a) e (f). E assim por diante...

i. Descubra o número total de matrizes cujo determinante não é um múltiplo de 2 nem um múltiplo de 13.

2.18 Usando a cifra de Vigenère, codifique a palavra *explanation* usando a chave *leg*.

2.19 Este problema explora o uso de uma versão do bloco de única vez da cifra de Vigenère. Nesse esquema, a chave é um fluxo de números aleatórios entre 0 e 26. Por exemplo, se a chave for 3 19 5..., então a primeira letra do texto claro é codificada com um deslocamento de 3 letras, a segunda com um deslocamento de 19 letras, a terceira com um deslocamento de 5 letras, e assim por diante.

a. Codifique o texto claro *sendmoremoney* com o fluxo de chaves 9 0 1 7 23 15 21 14 11 11 2 8 9.

b. Usando o texto cifrado produzido na parte a, encontre uma chave de modo que o texto cifrado seja decriptografado como o texto claro *cashnotneeded*.

2.20 Qual é a mensagem embutida na Figura 2.8?

2.21 Em um dos mistérios de Dorothy Sayers, Lord Peter é confrontado com a mensagem que aparece na Figura 2.9. Ele também descobre a chave da mensagem, que é uma seqüência de inteiros:

78765654343211234345656787887876565 4
34321123434565678788787656544332112 34

a. Decodifique a mensagem. *Dica:* qual é o maior valor inteiro?
b. Se o algoritmo for conhecido, mas não a chave, que segurança tem esse esquema?
c. Se a chave for conhecida, mas não o algoritmo, que segurança tem esse esquema?

I thought to see the fairies in the fields, but I saw only the evil elephants with their black backs. Woe! how that sight awed me! The elves danced all around and about while I heard voices calling clearly. Ah! how I tried to see-throw off the ugly cloud-but no blind eye of a mortal was permitted to spy them. So then came minstrels, having gold trumpets, harps and drums. These played very loudly beside me, breaking that spell. So the dream vanished, whereat I thanked Heaven. I shed many tears before the thin moon rose up, frail and faint as a sickle of straw. Now though the Enchanter gnash his teeth vainly, yet shall he return as the Spring returns. Oh, wretched man! Hell gapes, Erebus now lies open. The mouths of Death wait on thy end.

Figura 2.9 Um quebra-cabeças para Lord Peter.

Problemas de programação

2.22 Escreva um programa que possa criptografar e decriptografar usando a cifra de César geral, também conhecida como cifra aditiva.

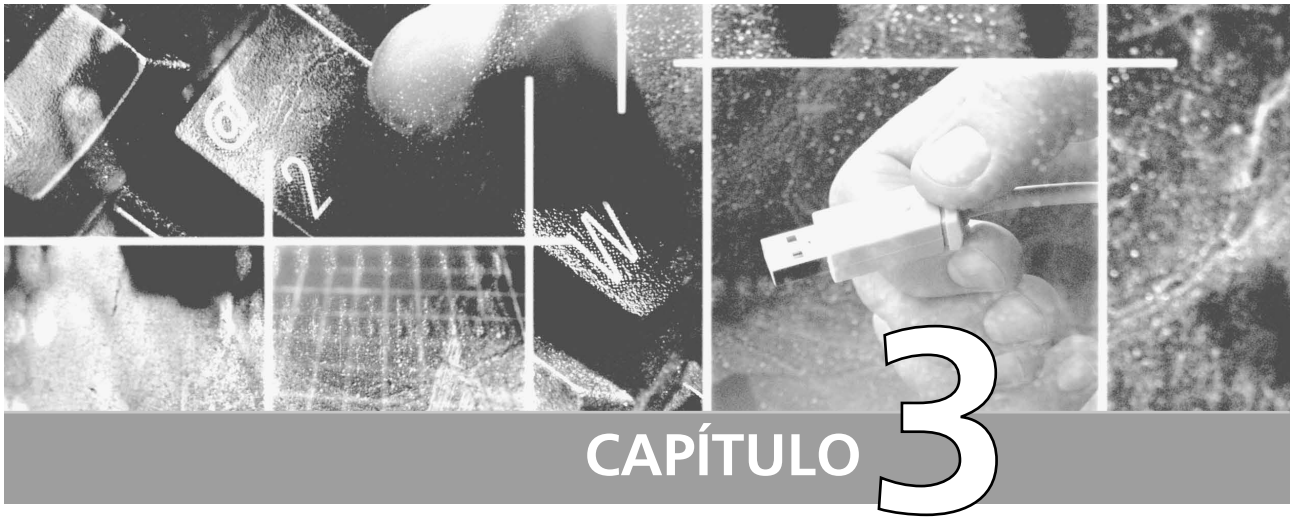
2.23 Escreva um programa que possa criptografar e decriptografar usando a cifra afim, descrita no Problema 2.1.

2.24 Escreva um programa que possa realizar um ataque de freqüência de letra em uma cifra aditiva sem intervenção humana. Seu software deverá produzir textos claros possíveis em ordem aproximada de probabilidade. Seria bom se a sua interface com o usuário permitisse que este especificasse *dê-me os 10 textos claros mais prováveis*.

2.25 Escreva um programa que possa realizar um ataque de freqüência de letra em qualquer cifra de substituição monoalfabética sem intervenção humana. Seu software deverá produzir textos claros possíveis em ordem aproximada de probabilidade. Seria bom se a sua interface com o usuário permitisse que este especificasse *dê-me os 10 textos claros mais prováveis*.

2.26 Crie um software que possa criptografar e decriptografar usando uma cifra de Hill 2×2.

2.27 Crie um software que possa realizar um ataque de texto claro conhecido rápido sobre uma cifra de Hill, dada a dimensão m. Qual a rapidez dos seus algoritmos, como uma função de m?



CAPÍTULO 3

Cifras de bloco e o Data Encryption Standard

3.1 **Princípios de cifra de bloco**

3.2 **O Data Encryption Standard**

3.3 **A força do DES**

3.4 **Criptoanálise diferencial e linear**

3.5 **Princípios de projeto de cifra de bloco**

3.6 **Leitura recomendada**

3.7 **Principais termos, perguntas de revisão e problemas**

Por toda a tarde, Mungo esteve trabalhando no código de Stern, principalmente com a ajuda das mensagens mais recentes que havia copiado na queda de Nevin Square. Stern estava muito confiante. Ele precisa estar bem ciente de que a Central de Londres sabia sobre essa queda. Era óbvio que eles não se importavam com a freqüência com que Mungo lia suas mensagens, de modo que estavam confiantes na inviolabilidade do código.

Talking to strange men, Ruth Rendell

PONTOS PRINCIPAIS

- Uma **cifra de bloco** é um esquema de criptografia/decriptografia em que um bloco de texto claro é tratado como um todo e usado para produzir um bloco de texto cifrado de mesmo tamanho.
- Muitas cifras de bloco possuem uma estrutura de Feistel, que consiste em diversas rodadas de processamento idênticas. Em cada rodada, uma substituição é realizada em metade dos dados sendo processados, seguida por uma permutação que troca as duas metades. A chave original é expandida de modo que uma chave diferente seja usada para cada rodada.
- O Data Encryption Standard (DES) tem sido o algoritmo de criptografia mais utilizado até o momento. Ele exibe a estrutura de Feistel clássica. O DES utiliza um bloco de 64 bits e uma chave de 56 bits.
- Dois métodos importantes de criptoanálise são a criptoanálise diferencial e a criptoanálise linear. O DES tem sido mostrado como altamente resistente a esses dois tipos de ataques.

O objetivo deste capítulo é ilustrar os princípios das cifras simétricas modernas. Com essa finalidade, focalizamos a cifra simétrica mais utilizada: o Data Encryption Standard (DES). Embora diversas cifras simétricas tenham sido desenvolvidas desde a introdução do DES e embora destinado a ser substituído pelo Advanced Encryption Standard (AES), o DES continua sendo o algoritmo mais importante. Além disso, um estudo detalhado do DES proporciona um conhecimento dos princípios usados em outras cifras simétricas. Examinaremos outras cifras simétricas importantes, incluindo AES, nos Capítulos 5 e 6.

Este capítulo começa com uma explicação sobre os princípios gerais das cifras de bloco simétrico, que são o tipo de cifra simétrica estudado neste livro (com a exceção da cifra de fluxo RC4 no Capítulo 6). Em seguida, explicamos o DES completo. Seguindo essa visão em um algoritmo específico, retornamos a uma explicação mais geral do projeto de cifra de bloco.

Em comparação com as cifras de chave pública, como RSA, a estrutura do DES e da maioria das cifras simétricas é muito complexa e não pode ser explicada tão facilmente quanto RSA e algoritmos semelhantes. Por conseguinte, o leitor pode querer começar com uma versão simplificada do DES, que é descrita no Apêndice C. Essa versão permite que o leitor realize a criptografia e decriptografia à mão, ganhando um bom conhecimento do funcionamento dos detalhes do algoritmo. A experiência em sala de aula indica que um estudo dessa versão simplificada melhora a compreensão do DES.[1]

3.1 PRINCÍPIOS DE CIFRA DE BLOCO

A maioria dos algoritmos de criptografia de bloco simétrico usados atualmente baseia-se em uma estrutura conhecida como cifra de bloco de Feistel [FEIS73]. Por esse motivo, é importante examinar os princípios de projeto da cifra de Feistel. Começamos com uma comparação das cifras de fluxo e das cifras de bloco. Depois, explicamos a motivação para a estrutura da cifra de bloco de Feistel. Finalmente, explicamos algumas de suas implicações.

Cifras de fluxo e cifras de bloco

Uma cifra de fluxo é aquela que codifica um fluxo de dados digital um bit ou um byte de cada vez. Alguns exemplos de cifras de fluxo clássicas são a cifra Vigenère autochaveada e a cifra Vernam. Uma cifra de bloco é aquela em que um bloco de texto claro é tratado como um todo e usado para produzir um bloco de texto cifrado com o mesmo tamanho. Normalmente, é utilizado um tamanho de bloco de 64 ou 128 bits. Usando alguns dos modos de operação explicados no Capítulo 6, uma cifra de bloco pode ser usada para conseguir o mesmo efeito de uma cifra de fluxo.

É preciso muito mais esforço para analisar as cifras de bloco. Em geral, elas parecem ser aplicáveis a uma gama maior de aplicações do que as cifras de fluxo. A grande maioria das aplicações de criptografia simétrica baseadas em rede utiliza cifras de bloco. Por conseguinte, a preocupação neste capítulo, e nas explicações seguintes no decorrer deste livro sobre criptografia simétrica, focalizará as cifras de bloco.

Motivação para a estrutura de cifra de Feistel

Uma cifra de bloco opera sobre um bloco de texto claro de n bits para produzir um bloco de texto cifrado de n bits. Existem 2^n diferentes blocos de texto claro possíveis e, para a criptografia ser reversível (ou seja, para a decriptografia ser possível), cada um produz um único bloco de texto cifrado. Essa transformação é chamada de reversível, ou não-singular. Os exemplos a seguir ilustram a transformação não-singular e singular para $n = 2$.

Mapeamento reversível		Mapeamento irreversível	
Texto claro	**Texto cifrado**	**Texto claro**	**Texto cifrado**
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

No segundo caso, um texto cifrado de 01 poderia ter sido produzido por um dos dois blocos de texto claro. Assim, se nos limitarmos aos mapeamentos reversíveis, o número de transformações diferentes é $2^n!$.

A Figura 3.1 ilustra a lógica de uma cifra de substituição geral para $n = 4$. Uma entrada de 4 bits produz um dos 16 estados de entrada possíveis, que é mapeado pela cifra de substituição para um único dos 16 estados de saída possíveis, cada um representado por 4 bits de texto cifrado. Os mapeamentos de criptografia e decriptografia podem ser definidos por uma tabulação, como mostra a Tabela 3.1. Essa é a forma mais geral de cifra de bloco e pode ser usada para definir

1. Você pode, porém, ignorar o Apêndice C, pelo menos em uma leitura inicial. Caso você se perca ou fique confuso com os detalhes do DES, poderá então voltar e começar com o DES simplificado

qualquer mapeamento reversível entre texto claro e texto cifrado. Feistel se refere a isso como a cifra de bloco ideal, pois permite o número máximo de mapeamentos de criptografia possíveis a partir do bloco de texto claro [FEIS75].

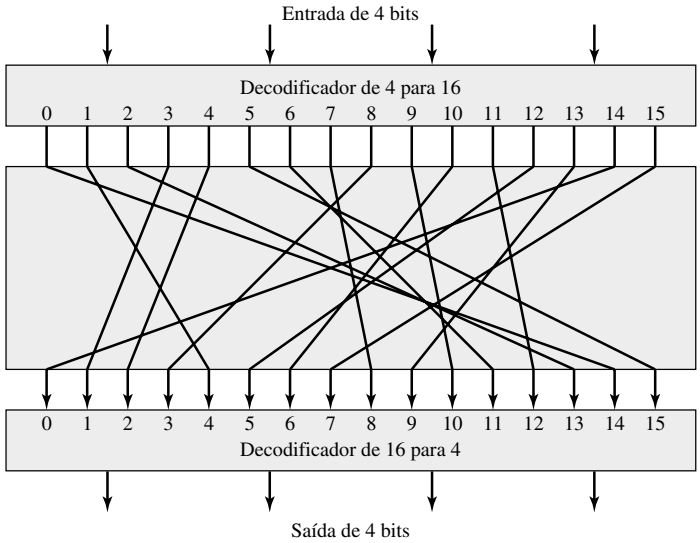


Figura 3.1 Substituição de bloco geral de *n* bits para *n* bits (mostrados com $n = 4$).

Mas existe um problema prático com a cifra de bloco ideal. Se ela for usada em um tamanho de bloco pequeno, como $n = 4$, então o sistema será equivalente a uma cifra de substituição clássica. Esses sistemas, como já vimos, são vulneráveis a uma análise estatística do texto claro. Esse ponto fraco não é inerente ao uso de uma cifra de substituição, mas resulta do uso de um tamanho de bloco pequeno. Se *n* for suficientemente grande e uma substituição reversível qualquer entre texto claro e texto cifrado for permitida, então as características estatísticas do texto claro de origem serão mascaradas a tal ponto que esse tipo de criptoanálise será inviável.

Tabela 3.1 Tabelas de criptografia e decriptografia para a cifra de substituição da Figura 3.1

Texto claro	Texto cifrado
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Texto cifrado	Texto claro
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Uma cifra de substituição reversível qualquer (a cifra de bloco ideal) para um grande tamanho de bloco não é prática, porém, do ponto de vista da implementação e do desempenho. Para tal transformação, o próprio mapeamento constitui a chave. Considere novamente a Tabela 3.1, que define um mapeamento reversível específico do texto claro para o texto cifrado, para $n = 4$. O mapeamento pode ser definido pelas entradas na segunda coluna, que mostram o valor do texto cifrado para cada bloco de texto claro. Essa é basicamente a chave que determina o mapeamento espe-

cífico entre todos os mapeamentos possíveis. Nesse caso, usando esse método simples de definição da chave, o tamanho de chave exigido é (4 bits) $\times$ (16 linhas) = 64. Em geral, para uma cifra de bloco ideal de n bits, o tamanho da chave, definido nesses moldes, é $n \times 2^n$ bits. Para um bloco de 64 bits, que é um tamanho desejável para afastar ataques estatísticos, o tamanho de chave exigido é $64 \times 2^{64} = 2^{70} \approx 10^{21}$ bits.

Considerando essas dificuldades, Feistel indica que é necessária uma aproximação do sistema de cifra de bloco ideal para um n grande, montado a partir de componentes facilmente observáveis [FEIS75]. Mas antes de passar para a técnica de Feistel, vamos fazer outra observação. Poderíamos usar a cifra de substituição de bloco geral, mas, para tornar essa implementação tratável, confinamo-nos a um subconjunto dos 2^n! possíveis mapeamentos reversíveis. Por exemplo, suponha que definamos o mapeamento em termos de um conjunto de equações lineares. No caso de $n = 4$, temos:

$$
\begin{aligned}
y_1 &= k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4 \\
y_2 &= k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4 \\
y_3 &= k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4 \\
y_4 &= k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4
\end{aligned}
$$

onde os x_i são os quatro dígitos binários do bloco de texto claro, os y_i são os quatro dígitos binários do bloco de texto cifrado, os k_{ij} são os coeficientes binários e a aritmética é mod 2. O tamanho de chave é apenas n^2, neste caso, 16 bits. O perigo com esse tipo de formulação é que ela pode ser vulnerável à criptoanálise por um atacante que saiba a estrutura do algoritmo. Neste exemplo, o que temos é basicamente a cifra de Hill, explicada no Capítulo 2, aplicada a dados binários em vez de caracteres. Como vimos no Capítulo 2, um sistema linear simples como esse é bastante vulnerável.

A cifra de Feistel

Feistel propôs [FEIS73] que podemos aproximar a cifra de bloco ideal utilizando o conceito de uma cifra de produto, que é a execução de duas ou mais cifras simples em seqüência, de tal forma que o resultado ou produto final seja criptograficamente mais forte do que qualquer uma das cifras componentes. A essência da técnica é desenvolver uma cifra de bloco com um tamanho de chave de k bits e um tamanho de bloco de n bits, permitindo um total de 2^k transformações possíveis, em vez de 2^n! transformações disponíveis com a cifra de bloco ideal.

Em particular, Feistel propôs o uso de uma cifra que alterna substituições e permutações. De fato, essa é uma aplicação prática de uma proposta de Claude Shannon para desenvolver uma cifra de produto que alterne funções de *confusão* e *difusão* [SHAN49]. Examinamos, em seguida, esses conceitos de difusão e confusão e, depois, apresentamos a cifra de Feistel. Mas primeiro, vale a pena comentar esse fato marcante: a estrutura da cifra de Feistel, que existe há mais de um quarto de século e que, por sua vez, é baseada na proposta de Shannon de 1945, é a estrutura utilizada por muitas cifras de bloco simétrico significativas atualmente em uso.

Difusão e confusão Os termos difusão e confusão foram introduzidos por Claude Shannon para captar os dois componentes básicos de qualquer sistema criptográfico [SHAN49].[2] A preocupação de Shannon foi impedir a criptoanálise baseada em análise estatística. O raciocínio é o seguinte. Considere que o atacante tem algum conhecimento das características estatísticas do texto claro. Por exemplo, em uma mensagem legível ao ser humano em alguma linguagem, a distribuição de freqüência das várias letras pode ser conhecida. Ou pode haver palavras ou frases que provavelmente aparecem na mensagem (palavras prováveis). Se essas estatísticas se refletirem de alguma maneira no texto cifrado, o criptoanalista poderá ser capaz de deduzir a chave de criptografia, ou parte da chave, ou pelo menos um conjunto de chaves que provavelmente contenha a chave exata. No que Shannon diz ser uma cifra ideal, todas as estatísticas do texto cifrado são independentes da chave específica utilizada. A cifra de substituição arbitrária que explicamos anteriormente (Figura 3.1) é uma cifra assim, mas, como vimos, não é prática.

Antes de recorrer a sistemas ideais, Shannon sugere dois métodos para frustrar a criptoanálise estatística: difusão e confusão. Na difusão, a estrutura estatística do texto claro é dissipada em estatísticas de longa duração do texto cifrado. Isso é obtido fazendo cada dígito do texto claro afetar o valor de muitos dígitos do texto cifrado. Geralmente, isso é equivalente a fazer cada dígito do texto cifrado ser afetado por muitos dígitos do texto claro. Um exemplo da difusão é criptografar uma mensagem $M = m_1, m_2, m_3, \ldots$ de caracteres com uma operação de média:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i}\right) \bmod 26$$

2. O artigo de Shannon, de 1949, apareceu originalmente como um relatório confidencial em 1945. Shannon tem de uma posição incrível e exclusiva na história do computador e da ciência da informação. Ele não apenas desenvolveu as primeiras idéias da criptografia moderna, mas também foi responsável pela invenção da disciplina da teoria da informação. Além disso, ele fundou outra disciplina, a aplicação da álgebra booleana ao estudo dos circuitos digitais; isso ele produziu rapidamente como uma tese de mestrado.

somando k letras sucessivas para obter uma letra de texto cifrado y_n. Pode-se mostrar que a estrutura estatística do texto claro foi dissipada. Assim, as freqüências das letras no texto cifrado serão mais próximas que no texto claro; as freqüências de digramas também serão mais próximas, e assim por diante. Em uma cifra de bloco binária, a difusão pode ser alcançada realizando-se repetidamente alguma permutação dos dados seguida pela aplicação de uma função a essa permutação; o efeito é que os bits de diferentes posições no texto claro original contribuem para um único bit de texto cifrado.[3]

Cada cifra de bloco envolve uma transformação de um bloco de texto claro em um bloco de texto cifrado, sendo que a transformação depende da chave. O mecanismo de difusão busca tornar o relacionamento estatístico entre o texto claro e o texto cifrado o mais complexo possível, a fim de frustrar tentativas de deduzir a chave. Por outro lado, a **confusão** busca tornar o relacionamento entre as estatísticas do texto cifrado e o valor da chave de criptografia o mais complexo possível, novamente para frustrar tentativas de descobrir a chave. Assim, mesmo que o atacante possa ter alguma idéia das estatísticas do texto cifrado, o modo como a chave foi usada para produzir esse texto cifrado é tão complexo que torna difícil deduzir a chave. Isso é obtido com o uso de um algoritmo de substituição complexo. Ao contrário, uma função de substituição linear simples incluiria pouca confusão.

Conforme [ROBS95b] indica, tão bem-sucedidas são a difusão e a confusão na captura da essência dos atributos desejados de uma cifra de bloco que elas se tornaram a base do projeto moderno de cifra de bloco.

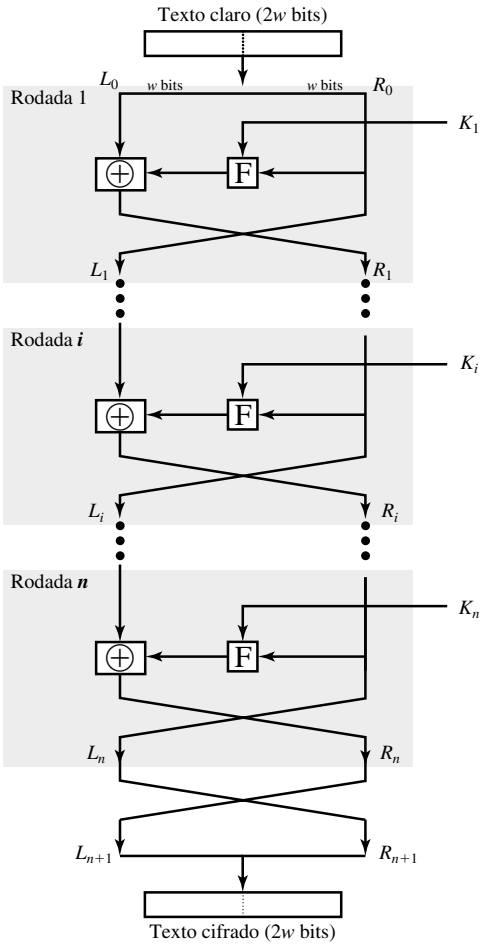


Figura 3.2 Rede de Feistel clássica.

3. Alguns livros sobre criptografia equiparam a permutação e a difusão. Isso é incorreto. A permutação, por si só, não muda as estatísticas do texto claro no nível de letras individuais ou blocos permutados. Por exemplo, no DES, a permutação inverte dois blocos de 32 bits, de modo que as estatísticas de cadeias de 32 bits ou menos são preservadas.

Estrutura da cifra de Feistel A Figura 3.2 representa a estrutura proposta por Feistel. As entradas do algoritmo de criptografia são um bloco de texto claro de tamanho $2w$ bits e uma chave K. O bloco do texto claro é dividido em duas metades, L_0 e R_0. As duas metades dos dados passam por n rodadas de processamento e depois se combinam para produzir o bloco do texto cifrado. Cada rodada i possui como entradas L_{i-1} e R_{i-1}, derivadas da rodada anterior, assim como uma subchave K_i derivada do K. Em geral, as subchaves K_i são diferentes de K e umas das outras.

Todas as rodadas têm a mesma estrutura. Uma substituição é realizada na metade esquerda dos dados. Isso é feito aplicando uma *função rodada* F à metade direita dos dados e depois aplicando OU exclusivo entre a saída dessa função e a metade esquerda dos dados. A função rodada tem a mesma estrutura geral para cada rodada, mas é parametrizada pela subchave da rodada K_i. Após essa substituição, é realizada uma permutação que consiste na troca das duas metades dos dados.[4] Essa estrutura é uma forma particular da rede de substituição—permutação (SPN – Substitution–Permutation Network) proposta por Shannon.

A realização exata de uma rede de Feistel depende da escolha dos seguintes parâmetros e características de projeto:

- **Tamanho do bloco:** Tamanhos de bloco maiores significam maior segurança (todas as outras coisas sendo iguais), mas reduzem a velocidade de criptografia/decriptografia para determinado algoritmo. A maior segurança é alcançada pela maior difusão. Tradicionalmente, um tamanho de bloco de 64 bits foi considerado uma escolha razoável e quase universal no projeto de cifra de bloco. Porém, o novo AES usa um tamanho de bloco de 128 bits.
- **Tamanho da chave:** Um tamanho de chave maior significa maior segurança, mas pode diminuir a velocidade de criptografia/decriptografia. A maior segurança é alcançada pela maior resistência a ataques de força bruta e maior confusão. Os tamanhos de chave de 64 bits ou menos agora são em grande parte considerados inadequados, e 128 bits tornou-se um tamanho comum.
- **Número de rodadas:** A essência da cifra de Feistel é que uma única rodada oferece segurança inadequada, mas que várias rodadas oferecem maior segurança. Um tamanho típico são 16 rodadas.
- **Algoritmo de geração de subchave:** Maior complexidade no algoritmo deverá levar a maior dificuldade de criptoanálise.
- **Função rodada:** Novamente, maior complexidade geralmente significa maior resistência à criptoanálise.

Há duas outras considerações no projeto de uma cifra de Feistel:

- **Criptografia/decriptografia rápida em software:** Em muitos casos, a criptografia é embutida nas aplicações ou funções utilitárias de tal maneira que impede uma implementação de hardware. Em consequência, a velocidade de execução do algoritmo torna-se uma preocupação.
- **Facilidade de análise:** Embora gostássemos de tornar nosso algoritmo o mais difícil possível de criptoanalisar, existe um grande benefício em tornar o algoritmo fácil de analisar. Ou seja, se o algoritmo puder ser explicado de forma concisa e clara, é mais fácil analisar esse algoritmo em busca de vulnerabilidades criptoanalíticas e, portanto, desenvolver um nível mais alto de garantia quanto à sua força. O DES, por exemplo, não tem uma funcionalidade facilmente analisável.

Algoritmo de decriptografia de Feistel O processo de decriptografia com uma cifra de Feistel é basicamente o mesmo processo de criptografia. A regra é a seguinte: use o texto cifrado como entrada para o algoritmo, mas use as subchaves K_i em ordem reversa. Ou seja, use K_n na primeira rodada, K_{n-1} na segunda rodada, e assim por diante até K_1 ser usado na última rodada. Essa é uma ótima característica, pois significa que não precisamos implementar dois algoritmos diferentes, um para criptografia e outro para decriptografia.

Para ver se o mesmo algoritmo com uma ordem de chave invertida produz o resultado correto, considere a Figura 3.3, que mostra o processo de criptografia descendo pelo lado esquerdo e o processo de decriptografia subindo pelo lado direito, para um algoritmo de 16 rodadas (o resultado seria o mesmo para qualquer número de rodadas). Por motivo de clareza, usamos a notação LE_i e RE_i para os dados atravessando pelo algoritmo de criptografia e LD_i e RD_i para os dados atravessando pelo algoritmo de decriptografia. O diagrama indica que, a cada rodada, o valor intermediário do processo de decriptografia é igual ao valor correspondente do processo de criptografia com as duas metades do valor trocadas. Colocando de outra maneira, considere que a última rodada de criptografia seja $\mathrm{LE}_i \| \mathrm{RE}_i$ (L_i concatenado com R_i). Então, a entrada correspondente à rodada de decriptografia $(16 - i)$ é $\mathrm{RE}_i \| \mathrm{LE}_i$ ou, de modo equivalente, $\mathrm{RD}_{16-i} \| LD_{16-i}$.

4. A rodada final é seguida por uma troca que reverte a troca que faz parte da rodada final. Alguém poderia simplesmente deixar as duas trocas fora do diagrama, sacrificando alguma consistência da apresentação. De qualquer maneira, a falta efetiva de uma troca na rodada final é feita para simplificar a implementação do processo de decriptografia, conforme veremos.

Vamos percorrer a Figura 3.3 para demonstrar a validade das afirmações anteriores.[5] Depois da última iteração do processo de criptografia, as duas metades da saída são trocadas, de modo que o texto cifrado é $RE_{16}\|LE_{16}$. A saída dessa rodada é o texto cifrado. Agora, tome esse texto cifrado e use-o como entrada para o mesmo algoritmo. A entrada para a primeira rodada é $RE_{16}\|LE_{16}$, que é igual à troca de 32 bits da saída da décima sexta rodada do processo de criptografia.

Agora, gostaríamos de mostrar que a saída da primeira rodada do processo de decriptografia é igual a uma troca de 32 bits da entrada com a décima sexta rodada do processo de criptografia. Primeiro, considere o processo de criptografia. Vemos que

$$\begin{aligned} LE_{16} &= RE_{15} \\ RE_{16} &= LE_{15} \oplus \end{aligned}$$

No lado da decriptografia,

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus F(RD_0, K_{16}) \\ &= RE_{16} \oplus F(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16}) \end{aligned}$$

O XOR tem as seguintes propriedades:

$$\begin{aligned} [A \times B] \oplus C &= A \oplus [B \oplus C] \\ D \oplus D &= 0 \\ E \oplus 0 &= E \end{aligned}$$

Assim, temos $LD_1 = RE_{15}$ e $RD_1 = LE_{15}$. Portanto, a saída da primeira rodada do processo de decriptografia é $LE_{15}\|RE_{15}$, que é a troca de 32 bits da entrada para a décima sexta rodada da criptografia. Essa correspondência se mantém por todas as 16 iterações, como é facilmente mostrado. Podemos converter esse processo em termos gerais. Para a *i*-ésima iteração do algoritmo de criptografia,

$$\begin{aligned} LE_i &= RE_{i-1} \\ RE_i &= LE_{i-1} \oplus F(RE_{i-1}, K_i) \end{aligned}$$

Reorganizando os termos,

$$\begin{aligned} RE_{i-1} &= LE_i \\ LE_{i-1} &= RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i) \end{aligned}$$

Assim, descrevemos as entradas para a i-ésima iteração como uma função das saídas, e essas equações confirmam as atribuições mostradas no lado direito da Figura 3.3.

Finalmente, vemos que a saída da última rodada do processo de decriptografia é RE0‖LE0. Uma troca de 32 bits recupera o texto claro original, demonstrando a validade do processo de decriptografia de Feistel.

Observe que a derivação não exige que F seja uma função reversível. Para ver isso, use um caso limitador em que F produz uma saída constante (por exemplo, tudo 1), independente dos valores de seus dois argumentos. As equações ainda são válidas.

3.2 O DATA ENCRYPTION STANDARD

O esquema de criptografia mais usado é baseado no Data Encryption Standard (DES) adotado em 1977 pelo National Bureau of Standards, hoje o National Institute of Standards and Technology (NIST), como Federal Information Processing Standard 46 (FIPS PUB 46). O próprio algoritmo é conhecido como Data Encryption

5. Para simplificar o diagrama, ele é retorcido, não mostrando a troca que ocorre ao final de cada iteração. Mas observe que o resultado intermediário ao final do i-ésimo estágio do processo de criptografia é a quantidade de $2w$ bits formada pela concatenação de LE_i e RE_i, e que o resultado intermediário ao final do *i*-ésimo estágio do processo de decriptografia é a quantidade de $2w$ bits formada pela concatenação de LD_i e RD_i.

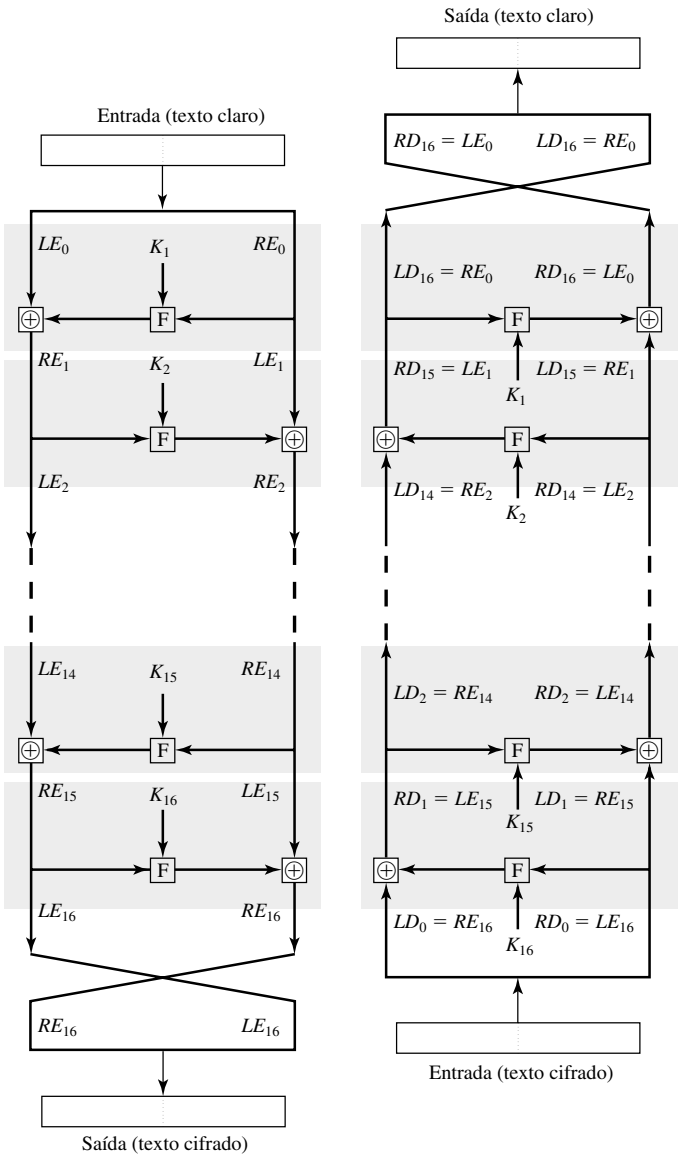


Figura 3.3 Criptografia e decriptografia de Feistel.

Algorithm (DEA).[6] Para DES, os dados são codificados em blocos de 64 bits usando uma chave de 56 bits. O algoritmo transforma a entrada de 64 bits em uma série de etapas em uma saída de 64 bits. As mesmas etapas, com a mesma chave, são usadas para reverter a criptografia.

O DES é bastante utilizado. Ele também tem sido tema de muita controvérsia com relação à sua segurança. Para apreciar a natureza da controvérsia, vamos rever rapidamente a história do DES.

No final da década de 1960, a IBM estabeleceu um projeto de pesquisa sobre criptografia de computador, liderado por Horst Feistel. O projeto foi concluído em 1971 com o desenvolvimento de um algoritmo com a designação LUCIFER [FEIS73], que foi vendido ao Lloyd's de Londres para uso em um sistema de caixa eletrônico, também desenvolvido pela IBM. LUCIFER é uma cifra de bloco de Feistel que opera em blocos de 64 bits, usando um tama-

6. A terminologia é um pouco confusa. Até há pouco tempo, os termos *DES* e *DEA* poderiam ser usados da mesma forma. Porém, a edição mais recente do documento DES inclui uma especificação do DEA, descrita aqui, mais o triple DEA (TDEA), descrito no Capítulo 6. Tanto DEA quanto TDEA fazem parte do Data Encryption Standard. Além disso, até a adoção recente do termo oficial *TDEA*, o algoritmo triple DEA normalmente era conhecido como *triple DES* e escrito como 3DES. Por questão de conveniência, usamos o termo 3DES.

nho de chave de 128 bits. Em razão dos resultados promissores produzidos pelo projeto LUCIFER, a IBM iniciou um esforço para desenvolver um produto de criptografia comercializável, que teoricamente poderia ser implementado em um único chip. O esforço foi chefiado por Walter Tuchman e Carl Meyer e envolvia não apenas pesquisadores da IBM, mas também consultores externos e orientação técnica da NSA. O resultado desse esforço foi uma versão refinada do LUCIFER, mais resistente à criptoanálise, mas que tinha um tamanho de chave reduzido de 56 bits, para caber em um único chip.

Em 1973, o National Bureau of Standards (NBS) emitiu uma solicitação de propostas para um padrão de cifra nacional. A IBM enviou os resultados de seu projeto Tuchman-Meyer. Esse foi certamente o melhor algoritmo proposto e foi adotado em 1977 como Data Encryption Standard.

Antes de sua adoção como um padrão, o DES proposto esteve sujeito a uma intensa crítica, que não diminuiu até hoje. Duas áreas atraíram o fogo dos críticos. Primeiro, o tamanho da chave no algoritmo LUCIFER original da IBM era de 128 bits, mas o do sistema proposto era de apenas 56 bits, uma redução muito grande no tamanho de chave, de 72 bits. Os críticos temiam que esse tamanho de chave fosse muito pequeno para agüentar os ataques de força bruta. Segundo, os critérios de projeto para a estrutura interna do DES, as caixas-S, eram confidenciais. Assim, os usuários não poderiam estar certos de que a estrutura interna do DES estava livre de quaisquer pontos vulneráveis que permitiriam à NSA decifrar as mensagens sem o benefício da chave. Eventos subseqüentes, principalmente o trabalho recente sobre criptoanálise diferencial, parece indicar que o DES tem uma estrutura interna muito forte. Além disso, de acordo com os participantes da IBM, as únicas mudanças feitas na proposta foram nas caixas-S, sugeridas pela NSA, que removiam as vulnerabilidades identificadas no decorrer do processo de avaliação.

Quaisquer que sejam os méritos do caso, o DES floresceu e é bastante utilizado, especialmente nas aplicações financeiras. Em 1994, o NIST reafirmou o DES para uso federal por outros cinco anos; o NIST recomendou o uso do DES para aplicações que não fossem de proteção de informações confidenciais. Em 1999, o NIST emitiu uma nova versão de seu padrão (FIPS PUB 46-3), indicando que o DES só deveria ser usado para sistemas legados e que o triple DES (que basicamente envolve a repetição do algoritmo DES três vezes sobre o texto claro usando duas ou três chaves diferentes para produzir o texto cifrado) fosse usado. Estudamos o triple DES no Capítulo 6. Como os algoritmos básicos de criptografia e decriptografia são iguais para DES e triple DES, continua sendo importante entender a cifra DES.

Criptografia DES

O esquema geral para a criptografia DES está ilustrado na Figura 3.4. Assim como em qualquer esquema de criptografia, existem duas entradas na função de criptografia: o texto claro a ser codificado e a chave. Nesse caso, o texto claro precisa ter 64 bits de extensão e a chave tem 56 bits de extensão.[7]

Examinando o lado esquerdo da figura, podemos ver que o processamento do texto claro prossegue em três fases. Primeiro, o texto claro de 64 bits passa por uma permutação inicial (IP), que reorganiza os bits para produzir a *entrada permutada*. Isso é seguido por uma fase consistindo em 16 rodadas da mesma função, que envolve funções de permutação e substituição. A saída da última (décima sexta) rodada consiste em 64 bits que são uma função do texto claro de entrada e da chave. As metades esquerda e direita da saída são trocadas para produzir a **pré-saída**. Finalmente, a pré-saída é passada por uma permutação (IP^{-1}) que é o inverso da função de permutação inicial, para produzir o texto cifrado de 64 bits. Com a exceção das permutações inicial e final, o DES tem a estrutura exata de uma cifra de Feistel, como mostra a Figura 3.2.

A parte da direita da Figura 3.4 mostra o modo como a chave de 56 bits é usada. Inicialmente, a chave é passada por uma função de permutação. Depois, para cada uma das 16 rodadas, uma *subchave* (K_i) é produzida pela combinação de um deslocamento circular à esquerda e uma permutação. A função de permutação é a mesma para cada rodada, mas uma subchave diferente é produzida, devido aos deslocamentos repetidos dos bits da chave.

Permutação inicial A permutação inicial e seu inverso são definidos por tabelas, como mostram as Tabelas 3.2a e 3.2b, respectivamente. As tabelas devem ser interpretadas da seguinte maneira. A entrada de uma tabela consiste em 64 bits numerados de 1 a 64. As 64 entradas na tabela de permutação contêm uma permutação dos números de 1 a 64. Cada entrada na tabela de permutação indica a posição de um bit de entrada numerado na saída, que também consiste em 64 bits.

7. Na realidade, a função espera uma chave de 64 bits como entrada. Porém, somente 56 desses bits são usados; outros 8 bits podem ser usados como bits de paridade ou simplesmente definidos arbitrariamente.

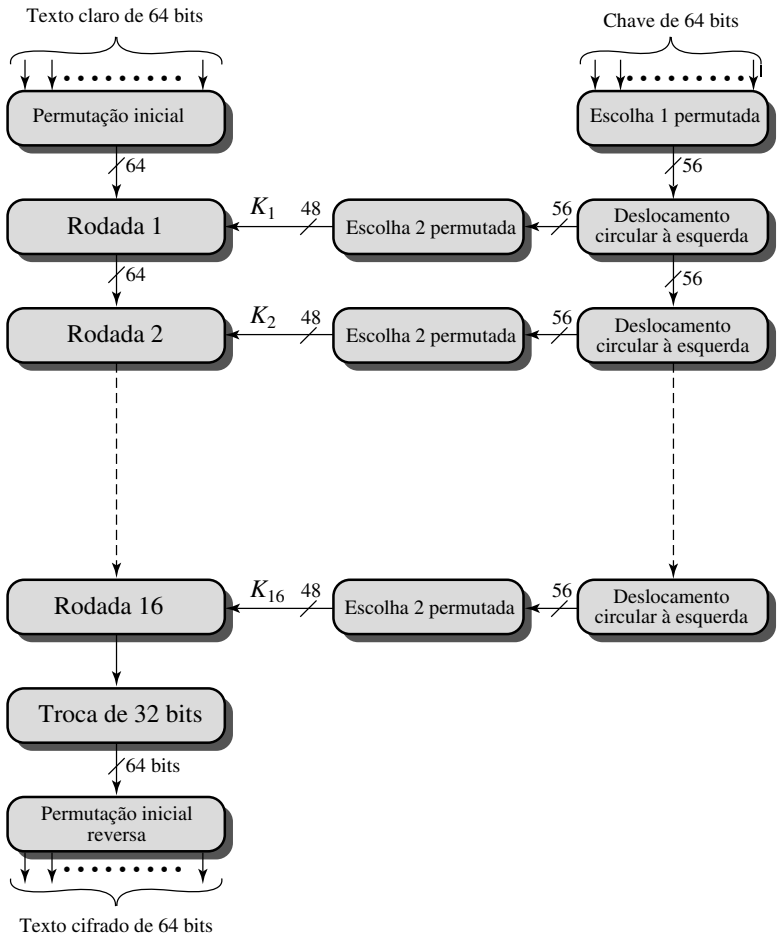


Figura 3.4 Representação geral do algoritmo de criptografia DES.

Tabela 3.2 Tabelas de permutação para DES

(a) Permutação inicial (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Permutação inicial inversa (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tabela 3.2 Tabelas de permutação para DES (*Continuação*)

(c) Permutação de expansão (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Função de permutação (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Para ver que essas duas funções de permutação são realmente o inverso uma da outra, considere a seguinte entrada M de 64 bits:

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

onde M_i é um dígito binário. Então, a permutação $X = \mathrm{IP}(M)$ é feita da seguinte forma:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Se depois calcularmos a permutação inversa $Y = \mathrm{IP}^{-1}(X) = \mathrm{IP}^{-1}(\mathrm{IP}(M))$, poderemos ver que a ordenação original dos bits é restaurada.

Detalhes de uma rodada individual A Figura 3.5 mostra a estrutura interna de uma única rodada. Novamente, comece focalizando o lado esquerdo do diagrama. As metades esquerda e direita de cada valor intermediário de 64 bits são tratadas como quantidades de 32 bits separadas, rotuladas com L (left — esquerda) e R (right — direita). Assim como na cifra de Feistel clássica, o processamento geral em cada rodada pode ser resumido nas seguintes fórmulas:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus \mathrm{F}(R_{i-1}, K_i)$$

A chave da rodada K_i é de 48 bits. A entrada R é de 32 bits. Essa entrada R é expandida inicialmente para 48 bits usando uma tabela que define uma permutação mais uma expansão que envolve duplicação de 16 dos bits de R (Tabela 3.2c). É realizado o XOR dos 48 bits resultantes com os bits de K_i. Esse resultado de 48 bits passa por uma função de substituição que produz uma saída de 32 bits, que é permutada conforme definido pela Tabela 3.2d.

O papel das caixas-S na função F está ilustrado na Figura 3.6. A substituição consiste em um conjunto de oito caixas-S, cada uma aceitando 6 bits como entrada e produzindo 4 bits como saída. Essas transformações são definidas na Tabela 3.3, que é interpretada da seguinte maneira: o primeiro e o último bits da entrada para a caixa S_i formam um número binário de 2 bits para selecionar uma das quatro substituições definidas pelas quatro linhas na tabela para S_i. Os quatro bits do meio selecionam uma das dezesseis colunas. O valor decimal na célula selecionada pela linha e coluna é então convertido em sua representação de 4 bits para produzir a saída. Por exemplo, em S_1, para a entrada 011001, a linha é 01 (linha 1) e a coluna é 1100 (coluna 12). O valor na linha 1, coluna 12, é 9, de modo que a saída é 1001.

Cada linha de uma caixa-S define uma substituição reversível geral. A Figura 3.1 pode ser útil para entender o mapeamento. A figura mostra a substituição para a linha 0 da caixa S_1.

A operação das caixas-S merece mais comentários. Ignore por enquanto a contribuição da chave (K_i). Se você examinar a tabela de expansão, verá que os 32 bits da entrada são divididos em grupos de 4 bits e, depois, se tornam grupos de 6 bits tomando os bits externos dos dois grupos adjacentes. Por exemplo, se parte da palavra de entrada for

... efgh ijkl mnop ...

esta se torna

... defghi hijklm lmnopq ...

Os dois bits mais externos de cada grupo selecionam uma de quatro substituições possíveis (uma linha de uma caixa-S). Depois, um valor de saída de 4 bits é substituído pela entrada específica de 4 bits (os quatro bits de entrada do meio). Depois, a saída de 32 bits das oito caixas-S é permutada, de modo que, na rodada seguinte, a saída de cada caixa-S afete imediatamente tantas outras quanto for possível.

Geração de chave Retornando às Figuras 3.4 e 3.5, vemos que uma chave de 64 bits é usada como entrada para o algoritmo. Os bits da chave são numerados de 1 até 64; cada oitavo bit é ignorado, conforme indicado pela falta de sombreamento na Tabela 3.4a. A chave é primeiro submetida a uma permutação controlada por uma tabela rotulada como Escolha Permutada Um (Tabela 3.4b). A chave de 56 bits resultante é então tratada como duas quantidades de 28 bits, rotuladas com C_0 e D_0. Em cada rodada, D_{i-1} e C_{i-1} estão sujeitos separadamente a um deslocamento esquerdo circular, ou rotação, de 1 ou 2 bits, conforme determinado pela Tabela 3.4d. Esses valores deslocados servem como

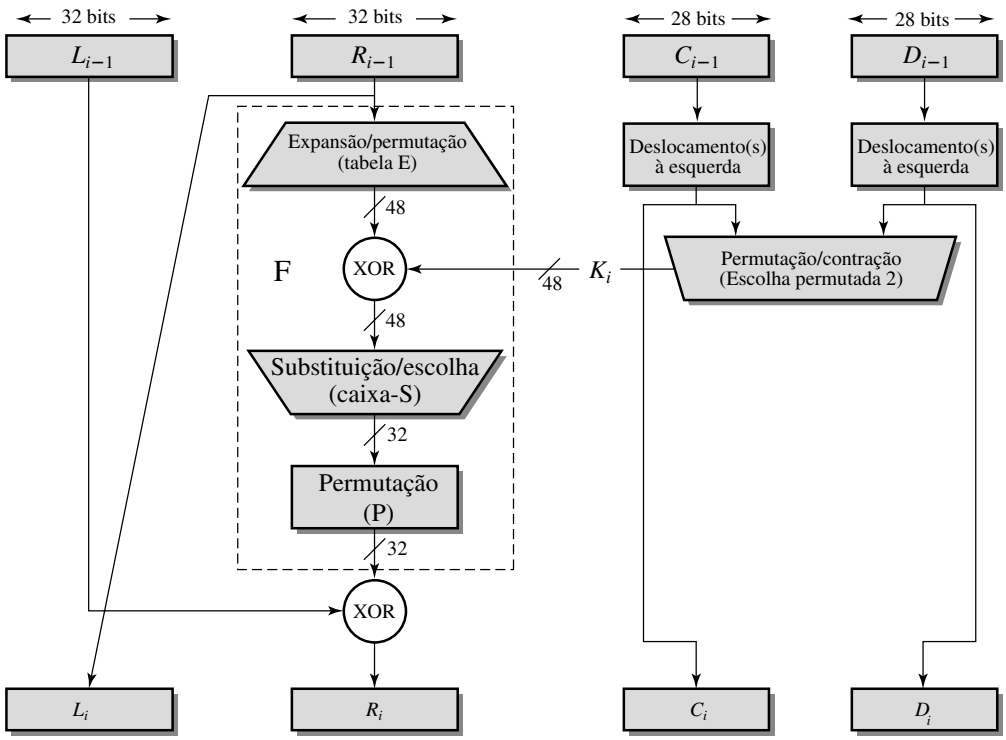


Figura 3.5 Rodada individual do algoritmo DES.

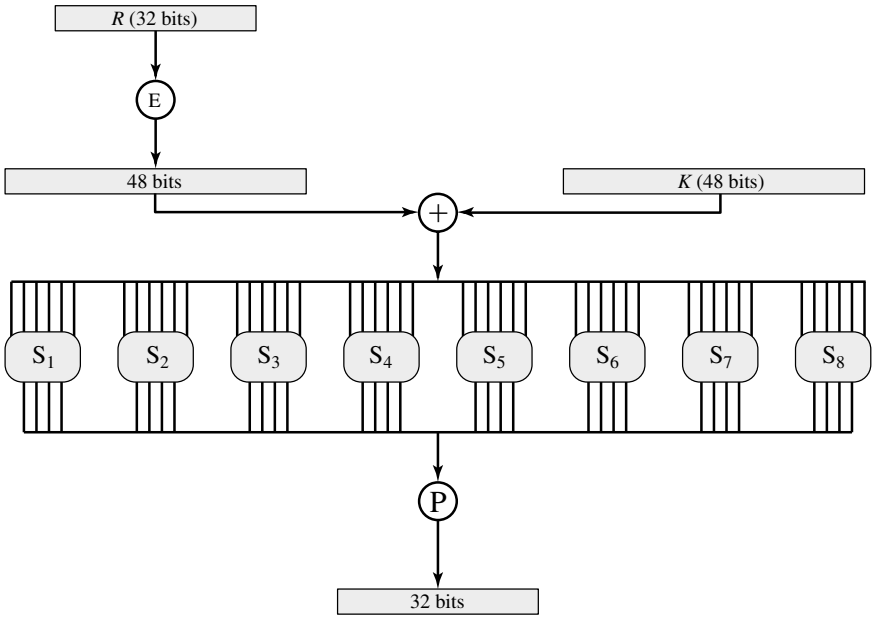


Figura 3.6 Cálculo de F(R, K).

entrada para a rodada a seguinte. Eles também servem como entrada para a Escolha Permutada Dois (Tabela 3.4c), que produz uma saída de 48 bits, que serve como entrada para a função $F(R_{i-1}, K_i)$.

Decriptografia DES

Assim como qualquer cifra de Feistel, a decriptografia usa o mesmo algoritmo da criptografia, exceto pela inversão da aplicação das subchaves.

O efeito avalanche

Uma propriedade desejável de qualquer algoritmo de criptografia é que uma pequena mudança no texto claro ou na chave produza uma mudança significativa no texto cifrado. Em particular, uma mudança em um bit do texto claro ou um bit da chave deverá produzir uma mudança em muitos bits do texto cifrado. Se a mudança fosse pequena, esta poderia oferecer um modo de reduzir o tamanho do texto claro ou o espaço de chave a ser pesquisado.

O DES exibe um forte efeito avalanche. A Tabela 3.5 mostra alguns resultados tirados de [KONH81]. Na Tabela 3.5a foram utilizados dois textos claros que diferem por um bit:

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

com a chave

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

A Tabela 3.5a mostra que, após apenas três rodadas, 21 bits diferem entre os dois blocos. Ao final, os dois textos cifrados diferem em 32 posições de bit.

A Tabela 3.5b mostra um teste semelhante em que um único texto claro é inserido:

01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

com duas chaves que diferem apenas em uma posição de bit:

1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100
0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Novamente, os resultados mostram que cerca de metade dos bits no texto cifrado diferem e que o efeito avalanche é pronunciado após apenas algumas rodadas.

Tabela 3.3 Definição de caixas-S do DES

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabela 3.4 Cálculo de escalonamento de chave do DES

(a) Chave de entrada

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Escolha Permutada Um (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Escolha Permutada Dois (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Escalonamento de Deslocamentos à Esquerda

Número da rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotacionados	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3.3 A FORÇA DO DES

Desde sua adoção como padrão federal nos Estados Unidos, tem havido grandes preocupações sobre o grau de segurança fornecido pelo DES. Essas preocupações, em grande parte, estão divididas em duas áreas: tamanho da chave e a natureza do algoritmo.

O uso de chaves de 56 bits

Com um tamanho de chave de 56 bits, existem 2^{56} chaves possíveis, o que é aproximadamente $7{,}2 \times 10^{16}$ chaves. Assim, um ataque de força bruta parece ser impraticável. Supondo que, na média, metade do espaço de chave tenha de ser pesquisado, uma única máquina realizando uma criptografia DES por microssegundo levaria mais de mil anos (ver Tabela 2.2) para quebrar a cifra.

Porém, a suposição de uma criptografia por microssegundo é bastante conservadora. Desde 1977, Diffie e Hellman postularam que existia tecnologia para montar uma máquina paralela com 1 milhão de dispositivos de criptografia, cada um podendo realizar uma criptografia por microssegundo [DIFF77]. Isso reduziria o tempo médio de busca para cerca de 10 horas. Os autores estimaram que o custo seria de aproximadamente 20 milhões de dólares em 1977.

Tabela 3.5 Efeito avalanche no DES

(a) Mudança no texto claro		(b) Mudança na chave	
Rodada	**Número de bits que diferem**	**Rodada**	**Número de bits que diferem**
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

O DES, finalmente e definitivamente, provou ser inseguro em julho de 1998, quando a Electronic Frontier Foundation (EFF) anunciou que tinha quebrado uma criptografia DES usando uma máquina "decifradora de DES" de uso especial, montada por menos de 250 mil dólares. O ataque levou menos de três dias. A EFF publicou uma descrição detalhada da máquina, permitindo que outros montassem seu próprio decifrador [EFF98]. E, naturalmente, os preços do hardware continuarão a cair enquanto as velocidades aumentam, tornando o DES praticamente inútil.

É importante observar que existe mais em um ataque de busca de chave do que simplesmente percorrer todas as chaves possíveis. A menos que um texto claro conhecido seja fornecido, o analista precisa ser capaz de reconhecer o texto claro como texto claro. Se a mensagem for apenas texto claro em inglês, então o resultado é obtido com facilidade, embora a tarefa de reconhecer o inglês tenha de ser automatizada. Se a mensagem de texto tiver sido compactada antes da criptografia, então o reconhecimento é mais difícil. E se a mensagem for algum tipo mais genérico de dados, como um arquivo numérico, e este tiver sido compactado, o problema se torna ainda mais difícil de automatizar. Assim, para suplementar a técnica de força bruta, algum grau de conhecimento sobre o texto claro esperado é necessário e algum meio de distinguir automaticamente o texto claro do lixo também é necessário. A técnica da EFF enfrenta esse problema também, e introduz algumas técnicas automatizadas que seriam eficazes em muitos contextos.

Felizmente, existem várias alternativas ao DES, sendo que as mais importantes são AES e triple DES, explicados nos Capítulos 5 e 6, respectivamente.

A natureza do algoritmo DES

Outra preocupação é que a criptoanálise seja possível por meio da exploração das características do algoritmo DES. O foco de preocupação tem sido nas oito tabelas de substituição, ou caixas-S, usadas em cada iteração. Como os critérios do projeto para essas caixas, e na realidade para o algoritmo inteiro, não se tornaram públicos, há uma suspeita de que elas foram construídas de modo que a criptoanálise seja possível para um oponente que conheça seus pontos fracos. Essa afirmação é torturante e, com o passar dos anos, diversas regularidades e comportamentos inesperados das caixas-S foram descobertos. Apesar disso, ninguém até aqui teve sucesso descobrindo o suposto ponto fraco fatal das caixas-S.[8]

Ataques de temporização

Discutiremos os ataques de temporização com mais detalhes na Parte Dois, pois eles se relacionam aos algoritmos de chave pública. Porém, a questão também pode ser relevante para criptografia simétrica. Basicamente, um ataque de

8. Pelo menos, ninguém confirmou publicamente essa descoberta.

temporização é aquele em que a informação sobre a chave ou sobre o texto claro é obtida observando-se quanto tempo foi gasto para determinada implementação realizar decriptografia em vários textos cifrados. Um ataque de temporização explora o fato de um algoritmo de criptografia ou decriptografia normalmente exigir quantidades ligeiramente diferentes de tempo com diferentes entradas. [HEVI99] descreve uma técnica que gera o peso de Hamming (número de bits iguais a um) da chave secreta. Isso está muito longe do conhecimento da chave real, mas é um primeiro passo interessante. Os autores concluem que o DES parece ser bastante resistente a um ataque de temporização bem-sucedido, mas sugerem algumas vias de exploração. Embora essa seja uma linha de ataque interessante, até aqui parece improvável que essa técnica sequer tenha sucesso contra o DES ou cifras simétricas mais poderosas, como triple DES e AES.

3.4 CRIPTOANÁLISE DIFERENCIAL E LINEAR

Na maior parte de sua vida, o principal problema com o DES tem sido sua vulnerabilidade ao ataque de força bruta, em razão do seu tamanho de chave relativamente curto (56 bits). Porém, também tem havido interesse em encontrar ataques criptoanalíticos no DES. Com a popularidade crescente das cifras de bloco com tamanhos de chave maiores, incluindo triple DES, os ataques por força bruta se tornaram cada vez mais impraticáveis. Assim, a ênfase tem sido maior nos ataques criptoanalíticos ao DES e outras cifras de bloco simétricas. Nesta seção, oferecemos uma rápida visão geral das duas técnicas mais poderosas e promissoras: criptoanálise diferencial e criptoanálise linear.

Criptoanálise diferencial

Um dos avanços mais significativos na criptoanálise nos últimos tempos é a criptoanálise diferencial. Nesta seção explicamos a técnica e sua aplicabilidade ao DES.

História A criptoanálise diferencial não foi relatada na literatura aberta antes de 1990. O primeiro trabalho publicado parece ter sido a criptoanálise de uma cifra de bloco chamada FEAL, por Murphy [MURP90]. Este foi seguido por diversos artigos de Biham e Shamir, que demonstraram essa forma de ataque em diversos algoritmos de criptografia e funções de hash; seus resultados estão resumidos em [BIHA93].

Os resultados mais conhecidos dessa técnica são aqueles que possuem aplicação ao DES. A criptoanálise diferencial é o primeiro ataque conhecido capaz de quebrar o DES em menos de 2^{55} criptografias. O esquema, relatado em [BIHA93], pode criptoanalisar com sucesso o DES com um esforço na ordem de 2^{47} criptografias, exigindo 2^{47} textos claros escolhidos. Embora 2^{47} certamente seja muito menos do que 2^{55}, a necessidade de que o adversário encontre 2^{47} textos claros escolhidos torna esse ataque de interesse apenas teórico.

Embora a criptoanálise diferencial seja uma ferramenta poderosa, ela não funciona muito bem contra o DES. O motivo, de acordo com um membro da equipe da IBM que projetou o DES [COPP94], é que a criptoanálise diferencial era conhecida da equipe desde 1974. A necessidade de fortalecer o DES contra ataques usando criptoanálise diferencial desempenhou um papel importante no projeto das caixas-S e da permutação P. Como evidência do impacto dessas mudanças, considere estes resultados comparáveis informados em [BIHA93]. A criptoanálise diferencial de um algoritmo LUCIFER de oito rodadas exige apenas 256 textos claros escolhidos, enquanto um ataque em uma versão de oito rodadas do DES exige 2^{14} textos claros escolhidos.

Ataque de criptoanálise diferencial O ataque de criptoanálise diferencial é complexo; [BIHA93] contém uma descrição completa. O raciocínio por trás da criptoanálise diferencial é observar o comportamento de pares de blocos de texto evoluindo a cada rodada da cifra, em vez de observar a evolução de um único bloco de texto. Aqui, oferecemos uma breve visão geral para que você possa ter uma idéia do ataque.

Começamos com uma mudança na notação para DES. Considere que o bloco de texto claro original m consiste em duas metades m_0, m_1. Cada rodada do DES mapeia a entrada da direita para a saída da esquerda e define a saída da direita como sendo uma função da entrada da esquerda e da subchave para essa rodada. Assim, a cada rodada, somente um novo bloco de 32 bits é criado. Se rotularmos cada novo bloco como m_i ($2 \leq i \leq 17$), então as metades da mensagem intermediária serão relacionadas da seguinte forma:

$$m_{i+1} = m_{i-1} \oplus \mathrm{f}(m_i, K_i), \qquad i = 1, 2, \ldots, 16$$

Em criptoanálise diferencial, começamos com duas mensagens, m e m', com uma diferença XOR conhecida $\Delta m = m \oplus m'$, e consideramos a diferença entre as metades da mensagem intermediária: $\Delta m_i = m_i \oplus m'_i$. Então temos:

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]\end{aligned}$$

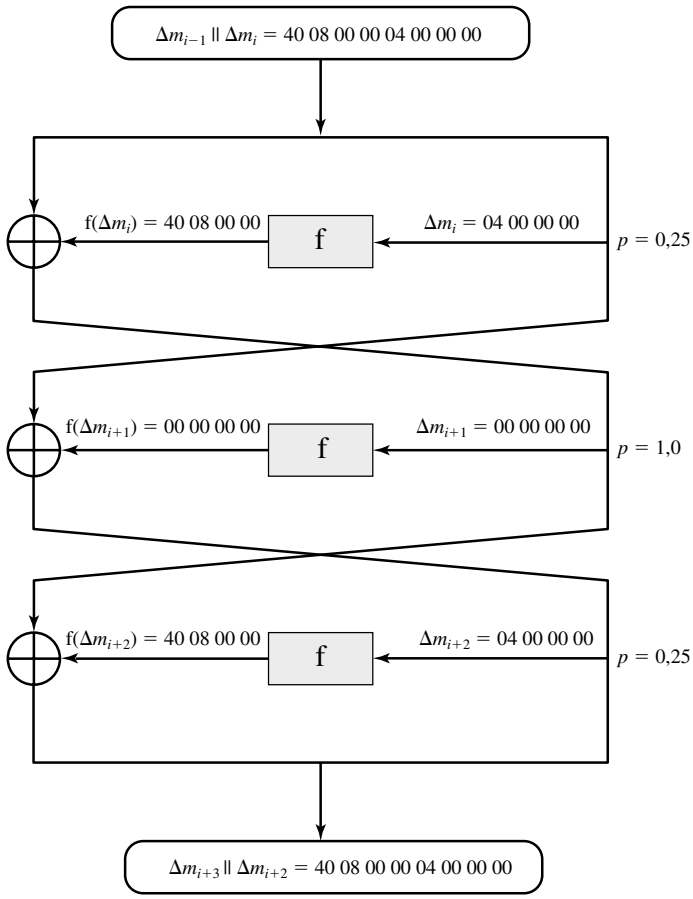


Figura 3.7 Propagação diferencial por meio de três rodadas de DES (números em hexadecimal).

Agora, suponha que muitos pares de entradas de f com a mesma diferença resultam na mesma diferença de saída caso seja usada a mesma subchave. Para ser mais preciso, vamos dizer que *X pode causar Y com probabilidade p*, se para uma fração p dos pares em que o XOR da entrada for X, o XOR da saída será igual a Y. Vamos supor que existam diversos valores de X com alta probabilidade de causar uma diferença de saída específica. Portanto, se soubermos Δm_{i-1} e Δm_i com alta probabilidade, então conhecemos Δm_{i+1} com alta probabilidade. Além disso, se diversas dessas referências forem determinadas, é viável determinar a subchave usada na função f.

A estratégia geral da criptoanálise diferencial baseia-se nessas considerações para uma única rodada. O procedimento é começar com duas mensagens de texto claro m e m' com determinada diferença e rastrear um provável padrão de diferenças após cada rodada gerar uma diferença provável para o texto cifrado. Na realidade, existem dois padrões prováveis de diferenças para as metades de 32 bits: $(\Delta m_{17} \| \Delta m_{16})$. Em seguida, submetemos m e m' à criptografia, para determinar a diferença real sob a chave desconhecida e comparar o resultado com a diferença provável. Se houver uma coincidência,

$$\mathrm{E}(K, m) \oplus \mathrm{E}(K, m') = (\Delta m_{17} \| \Delta m_{16})$$

então suspeitamos que todos os prováveis padrões em todas as rodadas intermediárias estão corretos. Com essa suposição, podemos fazer algumas deduções sobre os bits da chave. Esse procedimento precisa ser repetido muitas vezes para determinar todos os bits da chave.

A Figura 3.7, baseada em uma figura em [BIHA93], ilustra a propagação das diferenças por três rodadas do DES. As probabilidades mostradas à direita referem-se à probabilidade de que determinado conjunto de diferenças intermediárias apareça como uma função das diferenças na entrada. Em geral, depois de três rodadas, a probabilidade de que a diferença da saída seja conforme a mostrada é igual a $0{,}25 \times 1 \times 0{,}25 = 0{,}0625$.

Criptoanálise linear

Um desenvolvimento mais recente é a criptoanálise linear, descrita em [MATS93]. Esse ataque baseia-se em encontrar aproximações lineares para descrever as transformações realizadas no DES. Esse método pode encontrar uma

chave DES dados 2^{43} textos claros conhecidos, em comparação com os 2^{47} textos claros escolhidos para a criptoanálise diferencial. Embora essa seja uma pequena melhoria, pois pode ser mais fácil obter texto claro conhecido do que texto claro escolhido, ainda torna a criptoanálise linear inviável como um ataque ao DES. Até aqui, pouco trabalho foi feito por outros grupos para validar a técnica criptoanalítica linear.

Agora damos um rápido resumo do princípio no qual a criptoanálise linear é baseada. Para uma cifra com blocos de texto claro e texto cifrado de n bits e uma chave de m bits, considere que o bloco de texto claro seja rotulado com P[1],... P[n]; o bloco de texto cifrado, com C[1],... C[n]; e a chave, com K[1],... K[m]. Depois, defina

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

O objetivo da criptoanálise linear é encontrar uma equação linear eficaz na forma:

$$\mathrm{P}[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus \mathrm{C}[\beta_1, \beta_2, \ldots, \beta_b] = \mathrm{K}[\gamma_1, \gamma_2, \ldots, \gamma_c]$$

(onde $X = 0$ ou 1; $1 \leq a, b \leq n$, $1 \leq c \leq m$, e onde os termos α, β e γ representam posições fixas e únicas de bit) que apresente probabilidade $p \neq 0{,}5$. Quanto mais distante p estiver de 0,5, mais eficaz será a equação. Quando uma relação proposta é determinada, o procedimento é calcular os resultados do lado esquerdo da equação anterior para um grande número de pares de texto claro/texto cifrado. Se o resultado for 0 em mais da metade das vezes, considere que $\mathrm{K}[\gamma_1, \gamma_2, \ldots, \gamma_c] = 0$. Se ele for 1 na maior parte das vezes, considere que $\mathrm{K}[\gamma_1, \gamma_2, \ldots, \gamma_c] = 1$. Isso nos dá uma equação linear nos bits da chave. Tente obter mais dessas relações, de modo que possamos solucionar os bits da chave. Como estamos lidando com equações lineares, o problema pode ser enfrentado com uma rodada da cifra de cada vez, com os resultados sendo combinados.

3.5 PRINCÍPIOS DE PROJETO DE CIFRA DE BLOCO

Embora tenha havido muito progresso no projeto de cifras de bloco criptograficamente fortes, os princípios básicos não mudaram tanto desde o trabalho de Feistel e a equipe de projeto do DES, no início da década de 1970. É útil começar esta explicação examinando os critérios de projeto conhecidos usados no esforço DES. Depois, veremos três aspectos críticos do projeto da cifra de bloco: o número de rodadas, o projeto da função F e o escalonamento de chave.

Critérios de projeto do DES

Os critérios usados no projeto do DES, conforme relatados em [COPP94], focalizaram o projeto das caixas-S e a função P que recebe a saída das caixas-S (Figura 3.6). Os critérios para as caixas-S são os seguintes:

1. Nenhum bit de saída de qualquer caixa-S deverá estar muito próximo de uma função linear dos bits de entrada. Especificamente, se selecionarmos qualquer bit de saída e qualquer subconjunto dos seis bits de entrada, a fração das entradas para as quais esse bit de saída é igual aos XOR desses bits de entrada não deve ser próxima de 0 ou 1, mas deve estar perto de 1/2.
2. Cada linha de uma caixa-S (determinada por um valor fixo dos bits de entrada da esquerda e da direita) deve incluir todas as 16 combinações possíveis de bits de saída.
3. Se duas entradas de uma caixa-S diferirem em exatamente um bit, as saídas precisarão diferir em pelo menos dois bits.
4. Se duas entradas de uma caixa-S diferirem exatamente nos dois bits do meio, as saídas precisarão diferir em pelo menos dois bits.
5. Se duas entradas de uma caixa-S diferirem em seus dois primeiros bits e forem idênticas em seus dois últimos bits, as duas saídas não poderão ser iguais.
6. Para qualquer diferença não nula de 6 bits entre as entradas, não mais do que 8 dos 32 pares de entradas exibindo essa diferença poderão resultar na mesma diferença da saída.
7. Este é um critério semelhante ao anterior, mas para o caso de três caixas-S.

Coppersmith informou que o primeiro critério na lista anterior foi necessário porque as caixas-S são a única parte não linear do DES. Se as caixas-S fossem lineares (ou seja, cada bit de saída é uma combinação linear dos bits de entrada), o algoritmo inteiro seria linear e facilmente quebrado. Vimos esse fenômeno com a cifra de Hill, que é linear. Os critérios restantes visaram principalmente impedir a criptoanálise diferencial e fornecer boas propriedades de confusão.

Os critérios para a permutação P são os seguintes:

1. Os quatro bits de saída de cada caixa-S na rodada i são distribuídos de modo que dois deles afetam (oferecem entrada para) 'bits do meio' da rodada $(i + 1)$ e os outros dois afetam os bits do fim. Os dois bits do meio da entrada de uma caixa-S não são compartilhados com as caixas-S adjacentes. Os bits do fim são os dois bits da esquerda e os dois bits da direita, que são compartilhados com as caixas-S adjacentes.
2. Os quatro bits de saída de cada caixa-S afetam seis caixas-S diferentes na rodada seguinte, e nenhum grupo de dois afeta a mesma caixa-S.

3. Para duas caixas-S *j, k*, se um bit de saída de S_j afetar o bit do meio de S_k então, na rodada seguinte, o bit de saída de S_k não poderá afetar um bit do meio de S_j. Isso implica que, para $j = k$, um bit de saída de S_j não pode afetar um bit do meio de S_j.

Esses critérios servem para aumentar a difusão do algoritmo.

Número de rodadas

A força criptográfica de uma cifra de Feistel deriva de três aspectos do projeto: o número de rodadas, a função F e o algoritmo de escalonamento de chave. Vamos examinar primeiro a escolha do número de rodadas.

Quanto maior o número de rodadas, mais difícil é realizar a criptoanálise, mesmo para um F relativamente fraco. Em geral, o critério deverá ser de que o número de rodadas seja escolhido de modo que os esforços criptoanalíticos conhecidos exijam maior esforço do que um ataque de busca de chave por força bruta. Esse critério certamente foi usado no projeto do DES. Schneier [SCHN96] observa que, para o DES com 16 rodadas, um ataque de criptoanálise diferencial é ligeiramente mais eficiente do que a força bruta: o ataque de criptoanálise diferencial exige $2^{55.1}$ operações,[9] enquanto a força bruta exige 2^{55}. Se o DES tivesse 15 ou menos rodadas, a criptoanálise diferencial exigiria menos esforço do que a busca de chave por força bruta.

O critério é atraente porque facilita julgar a força de um algoritmo e comparar os algoritmos diferentes. Na ausência de uma descoberta criptoanalítica, a força de qualquer algoritmo que satisfaça o critério pode ser julgada unicamente pelo tamanho da chave.

Projeto da função F

O núcleo da cifra de bloco de Feistel é a função F. Como já vimos, no DES, essa função conta com o uso de caixas-S. Esse também é o caso da maioria das outras cifras de bloco simétrico, conforme veremos no Capítulo 4. Porém, podemos fazer alguns comentários gerais sobre os critérios para projeto de F. Depois disso, examinaremos especificamente o projeto da caixa-S.

Critérios de projeto para F A função F oferece o elemento de confusão em uma cifra de Feistel. Assim, é preciso que seja difícil 'desembaralhar' a substituição realizada por F. Um critério óbvio é que F seja não linear, como já explicamos. Quanto menos linear for F, mais difícil será qualquer tipo de criptoanálise. Existem várias medidas de não linearidade além do escopo deste livro. Em termos gerais, quanto mais difícil for aproximar F por um conjunto de equações lineares, mais não linear será F.

Vários outros critérios deverão ser considerados no projeto de F. Gostaríamos que o algoritmo tivesse boas propriedades de avalanche. Lembre-se de que, em geral, isso significa que uma mudança em um bit da entrada deverá produzir uma mudança em muitos bits da saída. Uma versão mais rigorosa disso é o critério de avalanche estrito (SAC – Strict Avalanche Criterion) [WEBS86], que afirma que qualquer bit de saída *j* de uma caixa-S deverá mudar com probabilidade 1/2 quando qualquer bit de entrada isolado *i* for invertido para todo *i, j*. Embora o SAC seja expresso em termos de caixas-S, um critério semelhante poderia ser aplicado a F como um todo. Isso é importante quando são considerados projetos que não incluem caixas-S.

Outro critério proposto em [WEBS86] é o critério de independência de bit (BIC – Bit Independence Criterion), que afirma que os bits de saída j e k devem mudar independentemente quando qualquer bit de entrada isolado i for invertido, para todo i, j e k. Os critérios SAC e BIC parecem fortalecer a eficácia da função de confusão.

Projeto de caixa-S Uma das áreas de pesquisa de maior interesse no campo de cifras de bloco simétricas é a do projeto de caixa-S. Os artigos são muito numerosos.[10] Aqui, mencionamos alguns princípios gerais. Basicamente, gostaríamos que qualquer mudança no vetor de entrada de uma caixa-S resultasse em mudanças de aparência aleatória na saída. O relacionamento deverá ser não linear e difícil de aproximar com funções lineares.

Uma característica óbvia da caixa-S é o seu tamanho. Uma caixa-S de $n \times m$ possui n bits de entrada e m bits de saída. O DES possui caixas-S de 6×4. Blowfish, descrito no Capítulo 6, possui caixas-S de 8×32. Caixas-S maiores, de modo geral, são mais resistentes à criptoanálise diferencial e linear [SCHN96]. Por outro lado, quanto maior a dimensão n, maior (exponencialmente) a tabela de pesquisa. Assim, por motivos práticos, normalmente é imposto um limite de n igual a cerca de 8 a 10. Outra consideração prática é que, quanto maior a caixa-S, mais difícil é projetá-la corretamente.

9. Lembre-se de que a criptoanálise diferencial do DES exige 2^{47} textos claros *escolhidos*. Se tudo o que você tiver para trabalhar for texto claro conhecido, então será necessário percorrer uma grande quantidade de pares de texto claro/texto cifrado conhecidos, procurando os úteis. Isso eleva o nível de esforço para $2^{55.1}$.

10. Um bom resumo dos estudos de projeto da caixa-S até o início de 1996 pode ser encontrado em [SCHN96].

Caixas-S normalmente são organizadas de maneira diferente da que é usada no DES. Uma caixa-S $n \times m$ normalmente consiste em 2^n linhas de m bits cada. Os n bits da entrada selecionam uma das linhas da caixa-S, e os m bits nessa linha são a saída. Por exemplo, em uma caixa-S de 8×32, se a entrada for 00001001, a saída consiste dos 32 bits na linha 9 (a primeira linha é rotulada como linha 0).

Mister e Adams [MIST96] propõem uma série de critérios para o projeto de caixa-S. Entre eles estão que a caixa-S deve satisfazer aos critérios SAC e BIC. Também sugerem que todas as combinações lineares das colunas da caixa-S devem ser *encurvadas* (bent). Funções Bent são uma classe especial de funções booleanas ligeiramente não lineares, de acordo com certos critérios matemáticos [ADAM90]. Tem havido um interesse crescente no projeto e análise de caixas-S usando funções Bent.

Um critério relacionado a caixas-S é proposto e analisado em [HEYS95]. Os autores definem o critério de avalanche garantida (GA — Garanteed Avalanche) da seguinte forma: uma caixa-S satisfaz o critério GA de ordem π se, para uma mudança de entrada de 1 bit, pelo menos π bits de saída mudarem. Os autores concluem que um GA no intervalo de ordem 2 a ordem 5 oferece fortes características de difusão para o algoritmo de criptografia geral.

Para caixas-S maiores, como 8×32, surge a questão quanto ao melhor método de seleção de entradas da caixa-S a fim de atender ao tipo de critério que explicamos. Nyberg, que escreveu muito sobre a teoria e a prática do projeto de caixa-S, sugere as seguintes técnicas (citadas em [ROBS95b]):

- **Aleatória:** Use alguma geração de número pseudo-aleatório ou alguma tabela de dígitos aleatórios para gerar as entradas nas caixas-S. Isso pode levar a caixas com características indesejáveis em tamanhos pequenos (por exemplo, 6×4), mas deverá ser aceitável para caixas-S grandes (por exemplo, 8×32).
- **Aleatória com teste:** Escolha entradas de caixa-S aleatoriamente, depois teste os resultados em relação a diversos critérios e descarte aqueles que não passarem.
- **Feita por humano:** Essa é uma técnica mais ou menos manual, apenas com uma matemática simples para ajudar. Essa é, aparentemente, a técnica usada no projeto do DES. É uma técnica difícil de ser transportada para caixas-S maiores.
- **Feita matematicamente:** Gere caixas-S de acordo com os princípios matemáticos. Por esse processo, podem-se construir caixas-S que oferecem segurança comprovada contra criptoanálise linear e diferencial, juntamente com uma boa difusão.

Uma variação na primeira técnica é usar caixas-S que sejam aleatórias e dependentes da chave. Um exemplo dessa técnica é Blowfish, descrito no Capítulo 6, que começa com caixas-S preenchidas com dígitos pseudo-aleatórios e depois altera o conteúdo usando a chave. Uma vantagem tremenda das caixas-S dependentes de chave é que, como não são fixas, é impossível analisá-las antecipadamente para procurar pontos fracos.

Algoritmo de escalonamento de chave

Uma última área de projeto de cifra em bloco, e uma que tem recebido menos atenção que o projeto de caixa-S, é o algoritmo de escalonamento de chave. Com qualquer cifra de bloco de Feistel, a chave é usada para gerar uma subchave para cada rodada. Em geral, gostaríamos de selecionar subchaves para maximizar a dificuldade de deduzir subchaves individuais e a dificuldade de voltar à chave principal. Nenhum princípio geral para isso foi promulgado até agora.

Hall sugere [ADAM94] que, no mínimo, o escalonamento de chave deve garantir o critério de avalanche estrita (SAC) e o critério de independência de bit (BIC) da chave/texto cifrado.

3.6 LEITURA RECOMENDADA

Há muitas informações sobre criptografia simétrica. Algumas das referências mais valiosas estão listadas aqui. Um trabalho de referência essencial é [SCHN96]. Esse trabalho incrível contém descrições de praticamente cada algoritmo criptográfico e protocolo publicado até o momento da escrita deste livro. O autor reúne resultados de jornais, procedimentos de conferência, publicações do governo e documentos de padrões, e os organiza em um estudo abrangente e compreensível. Outro estudo valioso e detalhado é [MENE97]. Uma abordagem matemática rigoroso é [STIN02].

As referências anteriores abrangem a criptografia de chave pública e também simétrica.

Talvez a descrição mais detalhada do DES seja [SIMO95]; o livro também contém uma explicação extensa da criptoanálise diferencial e linear do DES. [BARK91] oferece uma análise legível e interessante da estrutura e das técnicas criptoanalíticas em potencial para o DES. [EFF98] detalha o ataque de força bruta mais eficiente sobre o DES. [COPP94] examina a força inerente do DES e sua capacidade de resistir à criptoanálise.

BARK91 Barker, W. *Introduction to the analysis of the data encryption standard (DES)*. Laguna Hills: Aegean Park Press, 1991.

COPP94 Coppersmith, D. "The Data Encryption Standard (DES) and its strength against attacks". *IBM Journal of Research and Development*, maio de 1994.

EFF98 Electronic Frontier Foundation. *Cracking DES: secrets of encryption research, wiretap politics, and chip design*. Sebastopol: O'Reilly, 1998

MENE97 Menezes, A.; van Oorschot, P.; e Vanstone, S. *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997.

SCHN96 Schneier, B. *Applied cryptography*. New York: Wiley, 1996.

SIMO95 Simovits, M. *The DES: an extensive documentation and evaluation*. Laguna Hills: Aegean Park Press, 1995.

STIN02 Stinson, D. *Cryptography: theory and practice*. Boca Raton: CRC Press, 2002.

3.7 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

chave
cifra de bloco
cifra de Feistel
cifra de produto
confusão
criptoanálise diferencial
criptoanálise linear
Data Encryption Standard (DES)
difusão
efeito avalanche
função rodada
mapeamento irreversível
mapeamento reversível
permutação
rodada
subchave
substituição

Perguntas de revisão

3.1 Por que é importante estudar a cifra de Feistel?

3.2 Qual é a diferença entre uma cifra de bloco e uma cifra de fluxo?

3.3 Por que não é prático usar uma cifra de substituição reversível qualquer do tipo mostrado na Tabela 3.1?

3.4 O que é uma cifra de produto?

3.5 Qual é a diferença entre difusão e confusão?

3.6 Que parâmetros e escolhas de projeto determinam o algoritmo real de uma cifra de Feistel?

3.7 Qual é a finalidade das caixas-S no DES?

3.8 Explique o efeito de avalanche.

3.9 Qual é a diferença entre criptoanálise diferencial e linear?

Problemas

3.1 **a.** Na Seção 3.1, subseção sobre a motivação para a estrutura da cifra de Feistel, foi dito que, para um bloco de n bits, o número de mapeamentos reversíveis para a cifra de bloco ideal é $2^n!$ Justifique.

b. Nessa mesma discussão, foi dito que, para a cifra de bloco ideal, que permite todos os possíveis mapeamentos reversíveis, o tamanho da chave é $n \times 2^n$ bits. Mas, se houver $2n!$ mapeamentos possíveis, deverão ser necessários $\log_2 2^n!$ bits para discriminar entre os diferentes mapeamentos, e por isso o tamanho da chave deverá ser $\log_2 2^n!$. Porém, $\log_2 2^n! < n \times 2^n$. Explique a discrepância.

3.2 Considere uma cifra de Feistel composta de 16 rodadas com tamanho de bloco de 128 bits e tamanho de chave de 128 bits. Suponha que, para determinado k, o algoritmo de escalonamento de chave determine valores para as 8 primeiras chaves de rodada, $k_1, k_2, \ldots k_8$, e depois defina

$$k_9 = k_8, k_{10} = k_7, k_{11} = k_6, \ldots, k_{16} = k_1$$

Suponha que você tenha um texto cifrado c. Explique como, com acesso a um oráculo de criptografia, você pode decriptografar c e determinar m usando apenas uma única consulta ao oráculo. Isso mostra que tal cifra é vulnerável a um ataque de texto claro escolhido. (Um oráculo de cripto-

grafia pode ser imaginado como um dispositivo que, dado um texto claro, retorna o texto cifrado correspondente. Os detalhes internos do dispositivo não são conhecidos e você não pode abrir o dispositivo. Você só pode obter informações do oráculo fazendo consultas a ele e observando suas respostas.)

3.3 Considere um algoritmo de criptografia de bloco que codifique blocos de tamanho n, e considere $N = 2^n$. Digamos que temos t pares de texto claro/texto cifrado P_i, $C_i = \mathrm{E}(K, P_i)$, em que supomos que a chave K seleciona um dos $N!$ mapeamentos possíveis. Imagine que queremos encontrar K pela pesquisa exaustiva. Poderíamos gerar a chave K' e testar se $C_i = \mathrm{E}(K, P_i)$ para $1 \leq i \leq t$. Se K' criptografar cada P_i ao seu C_i apropriado, então temos evidência de que $K = K_1$. Porém, pode ser que os mapeamentos $\mathrm{E}(K, \cdot)$ e $\mathrm{E}(K', \cdot)$ combinem exatamente com os t pares de texto claro/texto cifrado P_i, C_i e não combinem com quaisquer outros pares.

a. Qual é a probabilidade de que $\mathrm{E}(K, \cdot)$ e $\mathrm{E}(K', \cdot)$ sejam de fato mapeamentos distintos?

b. Qual é a probabilidade de que $\mathrm{E}(K, \cdot)$ e $\mathrm{E}(K', \cdot)$ combinem com outros pares de texto claro/texto cifrado onde $0 \leq t' \leq N - t$?

3.4 Considere que π seja uma permutação dos inteiros $0, 1, 2, \ldots (2^n - 1)$, de modo que $\pi(m)$ forneça o valor permutado de $m, 0 \neq m < 2^n$. Em outras palavras, π é mapeado para o conjunto de inteiros de n bits para si mesmo e não existem dois inteiros mapeados para o mesmo inteiro. DES faz essa permutação para inteiros de 64 bits. Dizemos que π tem um ponto fixo em m se $\pi(m) = m$. Ou seja, se π for um mapeamento de criptografia, então um ponto fixo corresponde a uma mensagem que se codifica para si mesma. Estamos interessados na probabilidade de que π não tenha pontos fixos. Mostre o resultado um tanto inesperado de que mais de sessenta por cento dos mapeamentos terá pelo menos um ponto fixo.

3.5 Considere a substituição definida pela linha 1 da caixa-S S_1 na Tabela 3.3. Mostre um diagrama de bloco semelhante à Figura 3.1, que corresponda a essa substituição.

3.6 Calcule os bits número 1, 16, 33 e 48 na saída da primeira rodada da decriptografia DES, supondo que o bloco de texto cifrado seja composto somente de 1s e a chave externa seja composta somente de 1s.

3.7 Suponha que a função F do DES tenha mapeado cada entrada de 32 bits R, independente do valor da entrada K, para

a. uma cadeia de 32 bits de 1s,

b. complemento bit a bit de R.

Dica: Use as seguintes propriedades da operação XOR:

1. Que função o DES calcularia então?
2. Como ficaria a decriptografia?

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$A \oplus A = \mathbf{0}$$
$$A \oplus 0 = A$$
$$A \oplus \mathbf{1} = \text{complemento bit de } A$$

onde

A, B, C são cadeia de bits com n bits
$\mathbf{0}$ é uma cadeia de zeros com n bits
$\mathbf{1}$ é uma cadeia de uns com n bits

3.8 Este problema fornece um exemplo numérico da criptografia usando uma versão de uma rodada do DES. Começamos com o mesmo padrão de bits para a chave K e o texto claro, a saber:

em notação hexadecimal: 0 1 2 3 4 5 6 7 8 9 A B C D E F
em notação binária: 0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111

a. Derive K_1, a subchave da primeira rodada.

b. Derive L_0, R_0.

c. Expanda R_0 para obter $\mathrm{E}[R_0]$, onde $\mathrm{E}[\cdot]$ é a função de expansão da Figura 3.5.

d. Calcule $A = \mathrm{E}[R_0] \oplus K_1$.

e. Agrupe o resultado de 48 bits de (d) em conjuntos de 6 bits e avalie as substituições de caixa-S correspondentes.

f. Concatene os resultados de (e) para obter um resultado de 32 bits, B.

g. Aplique a permutação para obter $\mathrm{P}(B)$.

h. Calcule $R_1 = \mathrm{P}(B) \oplus L_0$.

i. Escreva o texto cifrado.

3.9 Mostre que a decriptografia DES é realmente o inverso da criptografia DES.

3.10 A troca de 32 bits após a décima sexta iteração do algoritmo DES é necessária para que o processo de criptografia possa ser invertido simplesmente executando-se o texto cifrado de volta pelo algoritmo com a ordem de chave revertida. Isso foi demonstrado no Problema 3.7. Porém, ainda pode não estar totalmente claro por que a troca de 32 bits é necessária. Para demonstrar por que, solucione os exercícios a seguir. Primeiramente, alguma notação:

$A \| B$ = a concatenação das cadeias de bits A e B

$\mathrm{T}_i(R \| L)$ = a transformação definida pela i-ésima iteração do algoritmo de criptografia, para $1 \neq I \neq 16$

$\mathrm{TD}_i(R \| L)$ = a transformação definida pela i-ésima iteração do algoritmo de decriptografia, para $1 \neq I \neq 16$

$T_{17}(R_7 \| L)$ = $L \| R$. Essa transformação ocorre após a décima sexta iteração do algoritmo de criptografia.

a. Mostre que a composição $\mathrm{TD}_1(\mathrm{IP}(\mathrm{IP}^{-1}(\mathrm{T}_{17}(\mathrm{T}_{16}(L_{15} \| R_{15})))))$ é equivalente à transformação que troca as metades de 32 bits, L_{15} e R_{15}. ou seja, mostre que

$$\mathrm{TD}_1(\mathrm{IP}(\mathrm{IP}^{-1}(\mathrm{T}_{17}(\mathrm{T}_{16}(L_{15}\|R_{15})))) = R_{15}\|L_{15}$$

b. Agora suponha que eliminamos a troca final de 32 bits no algoritmo de criptografia. Então, desejaríamos que a seguinte igualdade se mantivesse:

$$\mathrm{TD}_1(\mathrm{IP}(\mathrm{IP}^{-1}(\mathrm{T}_{16}(L_{15}\|R_{15})))) = L_{15}\|R_{15}$$

Isso acontece?

3.11 Compare a tabela de permutação inicial (Tabela 3.2a) com a tabela da escolha permutada um (Tabela 3.4b). As estruturas são semelhantes? Se forem, descreva as semelhanças. A que conclusões podemos chegar a partir dessa análise?

3.12 Ao usar o algoritmo DES para decriptografia, as 16 chaves (K_1, K_2,..., K_{16}) são usadas em ordem reversa. Portanto, o lado direito da Figura 3.5 não é mais válido. Projete um esquema de geração de chave com o escalonamento de deslocamento apropriado (semelhante à Tabela 3.4d) para o processo de decriptografia.

3.13 **a.** Considere que X' é o complemento bit a bit de X. Prove que, se o complemento do bloco de texto claro for obtido e o complemento de uma chave de criptografia for obtido, então o resultado da criptografia DES com esses valores será o complemento do texto cifrado original. Ou seja,

Se $Y = \mathrm{E}(K, X)$

Então $Y' = \mathrm{E}(K', X')$

Dica: Comece mostrando que, para duas cadeias de bits quaisquer de mesmo tamanho, A e B, $(A \oplus B)' = A' \oplus B$.

b. Foi dito que um ataque de força bruta no DES exige a pesquisa de um espaço de chave de 2^{56} chaves. O resultado da parte (a) muda isso?

3.14 Mostre que, no DES, os 24 primeiros bits de cada subchave vêm do mesmo subconjunto de 28 bits da chave inicial e que os 24 bits seguintes de cada subchave vêm de um subconjunto disjunto de 28 bits da chave inicial.

3.15 Para qualquer cifra de bloco, o fato de que ela é uma função não linear é fundamental para a sua segurança. Para ver isso, suponha que temos uma cifra de bloco linear EL que codifica blocos de 128 bits de texto claro em blocos de 128 bits de texto cifrado. Considere que EL(k, m) indique a criptografia de uma mensagem de 128 bits m sob uma chave k (o tamanho de bit real de k é irrelevante). Assim,

$$\mathrm{EL}(k, [m_1 \oplus m_2]) = \mathrm{EL}(k, m_1) \oplus \mathrm{EL}(k, m_2)$$

for all 128-bit patterns m_1, m_2

Descreva como, com 128 textos cifrados escolhidos, um adversário pode decriptografar qualquer texto cifrado sem conhecimento da chave secreta k. (Um 'texto cifrado escolhido' significa que um adversário tem a capacidade de escolher um texto cifrado e depois obter sua decriptografia. Aqui, você tem 128 pares de texto claro/texto cifrado para trabalhar e tem a capacidade de escolher o valor dos textos cifrados.)

3.16 Verifique a Figura C.2, que representa a geração de chave para S-DES.

a. Qual é a importância da função de permutação P10 inicial?

b. Qual é a importância das duas funções de deslocamento LS-1?

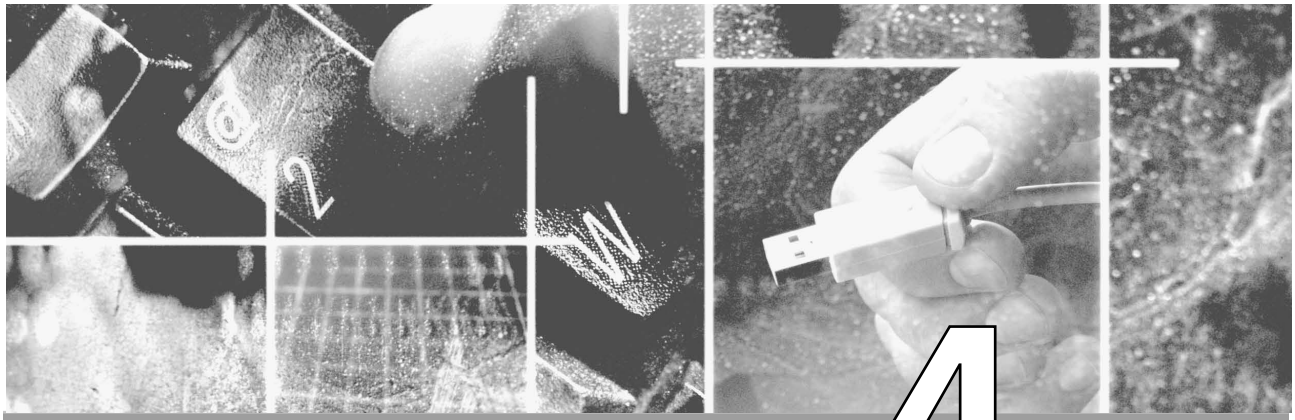
3.17 As equações para as variáveis q e r para S-DES são definidas na seção sobre análise S-DES. Ofereça as equações para s e r.

3.18 Usando S-DES, decodifique a cadeia (10100010) usando a chave (0111111101) manualmente. Mostre os resultados intermediários depois de cada função (IP, F_K, SW, F_K, IP^{-1}). Depois, codifique os primeiros 4 bits da cadeia de texto claro para uma letra e os próximos 4 bits para outra letra, onde codificamos de A até P na base 2 (ou seja, A = 0000, B = 0001, ..., P = 1111).
Dica: Como uma verificação intermediária, depois da aplicação de SW, a cadeia deverá ser (00010011).

Problemas de programação

3.19 Crie um programa que possa criptografar e decriptografar usando uma cifra de bloco de substituição geral.

3.20 Crie um programa que possa criptografar e decriptografar usando S-DES. Dados de teste: use texto claro, texto cifrado e a chave do Problema 3.18



CAPÍTULO 4

Corpos finitos

4.1 **Grupos, anéis e corpos**

4.2 **Aritmética modular**

4.3 **O algoritmo euclidiano**

4.4 **Corpos finitos na forma GF(p)**

4.5 **Aritmética polinomial**

4.6 **Corpos finitos na forma GF(2^n)**

4.7 **Leitura e sites Web recomendados**

4.8 **Principais termos, perguntas de revisão e problemas**

Na manhã seguinte, ao nascer do dia, Star entrou em casa, aparentemente ávida por uma lição. Eu disse, "Mostre oito". Ela fez uma exibição brilhante, primeiro batendo 4-4, depois batendo rapidamente 2-2-2-2, antes de entrar.

É incrível como Star aprendeu a contar até oito sem dificuldade, e do seu próprio jeito descobriu que cada número pode ser dado com diferentes divisões, isso não deixa dúvida de que estava conscientemente pensando em cada número. De fato, ela fez a aritmética mental, embora não possa, como os humanos, dar nome aos números. Mas ela aprendeu a reconhecer seus nomes falados quase imediatamente e lembrar dos sons dos nomes. Star é um pássaro silvestre único que, por vontade própria, buscou a ciência dos números com ávido interesse e incrível inteligência.

Living with birds, Len Howard

PONTOS PRINCIPAIS

- Um corpo é um conjunto de elementos em que duas operações aritméticas (adição e multiplicação) foram definidas e que possui as propriedades da aritmética comum, como fechamento, associatividade, comutatividade, distributividade e tendo inversos aditivo e multiplicativo.
- A aritmética modular é um tipo de aritmética de inteiros que reduz todos os números a um em um conjunto fixo $[0 \ldots n - 1]$ para algum número n. Qualquer inteiro fora dessa faixa é reduzido a um nessa faixa, tomando-se o resto após a divisão por n.
- O máximo divisor comum de dois inteiros é o maior inteiro positivo que divide exatamente os dois inteiros.
- Corpos finitos são importantes em diversas áreas da criptografia. Um corpo finito é simplesmente um corpo com um número finito de elementos. Pode ser mostrado que a ordem de um corpo finito (número de elementos no corpo) deve ser uma potência de um primo p^n, onde n é um inteiro positivo.
- Corpos finitos de ordem p podem ser definidos usando-se a aritmética mod p.
- Corpos finitos de ordem p^n, para $n > 1$, podem ser definidos usando-se a aritmética sobre polinômios.

Corpos finitos têm se tornado cada vez mais importantes na criptografia. Diversos algoritmos criptográficos contam fortemente com as propriedades dos corpos finitos, principalmente o Advanced Encryption Standard (AES) e a criptografia de curva elíptica.

O capítulo começa com uma rápida visão geral dos conceitos de grupo, anel e corpo. Esta seção é um tanto abstrata; o leitor poderá preferir passar os olhos rapidamente por esta seção em uma primeira leitura. Em seguida, precisamos de alguma base elementar em aritmética modular e em algoritmo euclidiano. Então, estaremos prontos para explicar os corpos finitos na forma GF(p), onde p é um número primo. Em seguida, precisamos de alguma base adicional, desta vez em aritmética de polinômio. O capítulo conclui com uma discussão dos corpos finitos na forma GF(2^n), onde n é um inteiro positivo.

Os conceitos e as técnicas da teoria dos números são muito abstratos e normalmente é difícil entendê-los intuitivamente sem exemplos [RUBI97]. De acordo com isso, este capítulo e o Capítulo 8 incluem diversos exemplos, cada um destacado em uma caixa sombreada.

4.1 GRUPOS, ANÉIS E CORPOS

Grupos, anéis e corpos são os elementos fundamentais de um ramo da matemática conhecido como álgebra abstrata, ou álgebra moderna. Na álgebra abstrata nos preocupamos com conjuntos em cujos elementos podemos operar algebricamente, ou seja, podemos combinar dois elementos do conjunto, talvez de várias maneiras, para obter um terceiro elemento do conjunto. Essas operações estão sujeitas a regras específicas, que definem a natureza do conjunto. Por convenção, a notação para as duas classes principais de operações sobre elementos do conjunto normalmente é a mesma notação para adição e multiplicação em números comuns. Porém, é importante observar que, na álgebra abstrata, não estamos limitados a operações aritméticas comuns. Tudo isso deverá ficar claro enquanto prosseguirmos.

Grupos

Um **grupo** G, às vezes indicado por $\{G, \bullet\}$, é um conjunto de elementos com uma operação binária, indicada por $\bullet$, que associa a cada par ordenado (a, b) de elementos em G um elemento $(a \bullet b)$ em G, de modo que os seguintes axiomas são obedecidos:[1]

(A1) Fechamento:	Se a e b pertencem a G, então $a \bullet b$ também está em G.
(A2) Associativo:	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ para todo a, b, c em G.
(A3) Elemento identidade:	Existe um elemento e em G, de modo que $a \bullet e = e \bullet a$ para todo a em G.
(A4) Elemento inverso:	Para cada a em G existe um elemento a' em G, de modo que $a \bullet a' = a' \bullet a = e$.

Considere que N_n indique um conjunto de n símbolos distintos que, por conveniência, representamos como $\{1, 2, ..., n\}$. Uma permutação de n símbolos distintos é um mapeamento de N_n a N_n. Defina S_n como o conjunto de todas as permutações de n símbolos distintos. Cada elemento de S_n é representado por uma permutação dos inteiros em $\{1, 2, ..., n\}$. É fácil demonstrar que S_n é um grupo:

A1: Se $\pi, \rho \in S_n$, então o mapeamento composto $\pi \bullet \rho$ é formado permutando-se os elementos de ρ de acordo com a permutação π. Por exemplo, $\{3, 2, 1\} \bullet \{1, 3, 2\} = \{2, 3, 1\}$. Claramente, $\pi, \rho \in S_n$.

A2: A composição de mapeamentos também é facilmente vista como sendo associativa.

A3: A o mapeamento de identidade é a permutação que não altera a ordem dos n elementos. Para S_n, o elemento de identidade é $\{1, 2, ..., n\}$.

A4: Para qualquer $\pi \in S_n$, o mapeamento que desfaz a permutação definida por π é o elemento inverso para π. Sempre haverá tal inverso. Por exemplo, $\{2, 3, 1\} \bullet \{3, 1, 2\} = \{1, 2, 3\}$.

Se um grupo tem um número finito de elementos, ele é referenciado como um **grupo finito**, e a **ordem** do grupo é igual ao número de elementos no grupo. Caso contrário, o grupo é um **grupo infinito**.

Um grupo é considerado **abeliano** se satisfizer à seguinte condição adicional:

(A5) Comutativo: $a \bullet b = b \bullet a$ para todo a, b em G.

1. O operador $\bullet$ é genérico e pode se referir a adição, a multiplicação ou a alguma outra operação matemática.

O conjunto de inteiros (positivos, negativos e 0) sob a adição é um grupo abeliano. O conjunto de números reais diferentes de zero sob a multiplicação é um grupo abeliano. O conjunto S_n do exemplo anterior é um grupo, mas não um grupo abeliano para $n > 2$.

Quando a operação em grupo for a adição, o elemento de identidade será 0; o elemento inverso de a é $-a$; e a subtração é definida com a seguinte regra: $a - b = a + (-b)$.

Grupo cíclico Definimos a exponenciação dentro de um grupo como a aplicação repetida do operador de grupo, de modo que $a^3 = a \bullet a \bullet a$. Além disso, definimos $a^0 = \text{e}$, o elemento de identidade; e $a^{-n} = (a')^n$. Um grupo G é cíclico se cada elemento de G for uma potência a^k (k é um inteiro) de um elemento fixo $a \in G$. Consideramos que o elemento a **gera** o grupo G, ou que é o **gerador** de G. Um grupo cíclico é sempre abeliano e pode ser finito ou infinito.

O grupo aditivo de inteiros é um grupo cíclico infinito gerado pelo elemento 1. Nesse caso, as potências são interpretadas aditivamente, de modo que n é a n-ésima potência de 1.

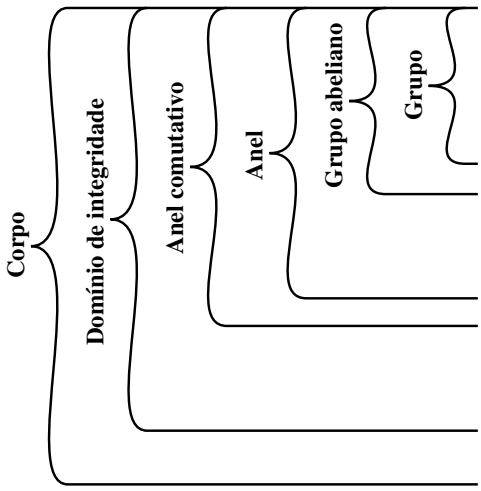
Anéis

Um **anel** R, às vezes indicado por $\{R, +, \times\}$, é um conjunto de elementos com duas operações binárias, chamadas adição e multiplicação,[2] de forma que para todo a, b, c em R, os seguintes axiomas são obedecidos:

(A1-A5) R é um grupo abeliano com relação à adição, ou seja, R satisfaz os axiomas de A1 a A5. Para o caso de um grupo aditivo, indicamos o elemento de identidade como zero e o inverso de a como $-a$.
(M1) Fechamento sob multiplicação: Se a e b pertencem a R, então ab também está em R.
(M2) Associatividade da multiplicação: $a(bc) = (ab)c$ para todo a, b, c em R.
(M3) Leis distributivas: $a(b + c) = ab + ac$ para todo a, b, c em R.
$(a + b)c = ac + ab$ para todo a, b, c em R.

Basicamente, um anel é um conjunto em que podemos realizar adição, subtração $[a - b = a + (-b)]$ e multiplicação sem sair do conjunto.

Com relação à adição e à multiplicação, o conjunto de todas as matrizes quadradas em n sobre os números reais é um anel.



(A1) Fechamento sob adição:	Se a e b pertencem a S, então $a + b$ também está em S
(A2) Associatividade da adição:	$a + (b + c) = (a + b) + c$ para todo a, b, c em S
(A3) Identidade aditiva:	Existe um elemento 0 em R tal que $a + 0 = 0 + a = a$ para todo a em S
(A4) Inverso aditivo:	Para cada a em S existe um elemento $-a$ em S tal que $a + (-a) = (-a) + a = 0$
(A5) Comutatividade da adição:	$a + b = b + a$ para todo a, b em S
(M1) Fechamento sob multiplicação:	Se a e b pertencem a S, então ab também está em S
(M2) Associatividade da multiplicação:	$a(bc) = (ab)c$ para todo a, b, c em S
(M3) Leis distributivas:	$a(b + c) = ab + ac$ para todo a, b, c em S
	$(a + b)c = ac + bc$ para todo a, b, c em S
(M4) Comutatividade da multiplicação:	$ab = ba$ para todo a, b em S
(M5) Identidade multiplicativa:	Existe um elemento 1 em S tal que $a1 = 1a = a$ para todo a em S
(M6) Sem divisores zero:	Se a, b em S e $ab = 0$, tentão $a = 0$ ou $b = 0$
(M7) Inverso multiplicativo:	Se a pertence a S e $a \neq 0$, existe um elemento a^{-1} em S tal que $aa^{-1} = a^{-1}a = 1$

Figura 4.1 Grupo, anel e corpo.

2. Generally, we do not use the multiplication symbol, $\times$, but denote multiplication by the concatenation of two elements.

Um anel é considerado **comutativo** se satisfizer a seguinte condição adicional:
(M4) Comutatividade da multiplicação: $ab = ba$ para todo a, b em R.

> Considere que S seja o conjunto de inteiros pares (positivos, negativos e 0) sob as operações normais de adição e multiplicação. S é um anel comutativo. O conjunto de todas as matrizes quadradas em n definidas no exemplo anterior não é um anel comutativo.

Em seguida, definimos um **domínio de integridade**, que é um anel comutativo que obedece ao seguinte axioma:

(M5) Identidade multiplicativa: Existe um elemento 1 em R de modo que $a1 = 1a = a$ para todo a em R.
(M6) Sem divisores zero: Se a, b em R e $ab = 0$, então $a = 0$ ou $b = 0$.

> Considere que S seja o conjunto de inteiros, positivos, negativos e 0, sob as operações normais de adição e multiplicação. S é um domínio de integridade.

Corpos

Um **corpo** F, às vezes indicado por $\{F, +, \times\}$, é um conjunto de elementos com duas operações binárias, chamadas de *adição* e *multiplicação*, de modo que para todo a, b, c em F, os seguintes axiomas são obedecidos:

(A1-M6) F é um domínio de integridade; ou seja, F satisfaz os axiomas de A1 a A5 e de M1 a M6.
(M7) Inverso multiplicativo: Para cada a em F, exceto 0, existe um elemento a^{-1} em F tal que $aa^{-1} = (a^{-1})a = 1$

Basicamente, um corpo é um conjunto em que podemos realizar adição, subtração, multiplicação e divisão sem sair do conjunto. A divisão é definida com a seguinte regra: $a/b = a(b^{-1})$

> Exemplos conhecidos de corpos são os números racionais, os números reais e os números complexos. Observe que o conjunto de todos os inteiros não é um corpo, pois nem todo elemento do conjunto tem um inverso multiplicativo; de fato, somente os elementos 1 e -1 possuem inversos multiplicativos nos inteiros.

A Figura 4.1 resume os axiomas que definem grupos, anéis e corpos.

4.2 ARITMÉTICA MODULAR

Dado qualquer inteiro positivo n e qualquer inteiro não negativo a, se dividirmos a por n, obteremos um quociente inteiro q e um resto inteiro r, que obedecem ao seguinte relacionamento:

$$a = qn + r \qquad 0 \le r < n; q = \lfloor a/n \rfloor \tag{4.1}$$

onde $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x.

A Figura 4.2 demonstra que, dados a e n positivos, sempre é possível encontrar q e r que satisfaçam o relacionamento anterior. Represente os inteiros na linha de números; a ficará em algum ponto nessa linha (mostramos a positivo, mas uma demonstração semelhante pode ser feita para a negativo). Começando com 0, prossiga para n, $2n$, até qn, de modo que qn £ a e $(q + 1)n > a$. A distância de qn para a é r, e descobrimos os valores exclusivos de q e r. O resto r normalmente é conhecido como **resíduo**.

$a = 11;$	$n = 7;$	$11 = 1 \times 7 + 4;$	$r = 4$	$q = 1$
$a = -11;$	$n = 7;$	$-11 = (-2) \times 7 + 3;$	$r = 3$	$q = -2$

Se a for um inteiro e n for um inteiro positivo, definimos a mod n como o resto quando a é dividido por n. O inteiro n é chamado de **módulo**. Assim, para qualquer inteiro a, sempre podemos escrever:

$$a = [a/n] \times n + (a \bmod n)$$

$11 \bmod 7 = 4;$	$-11 \bmod 7 = 3$

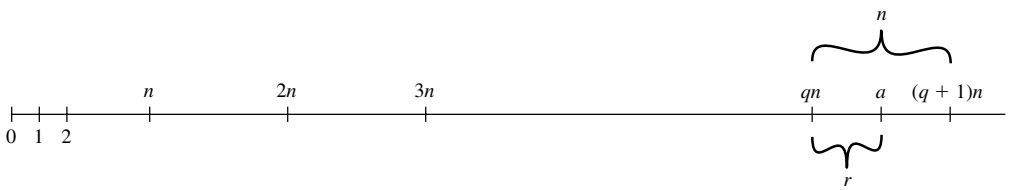


Figura 4.2 O relacionamento $a = qn + r$, $0 \leq r < n$.

Dois inteiros a e b são considerados **congruentes módulo *n***, se (a mod n) = (b mod n). Isso é escrito como $a \equiv b \pmod{n}$.[3]

$$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$$

Divisores

Dizemos que b diferente de zero divide a se $a = mb$ para algum m, onde a, b e m são inteiros. Ou seja, b divide a se não houver resto na divisão. A notação $b|a$ normalmente é usada para indicar que b divide a. Além disso, se $b|a$, dizemos que b é um **divisor** de a.

Os divisores positivos de 24 são 1, 2, 3, 4, 6, 8, 12 e 24.

Existem as seguintes relações:

- Se $a|1$, então $a = \pm 1$.
- Se $a|b$ e $b|a$, então $a = \pm b$.
- Qualquer $b \neq 0$ divide 0.
- Se $b|g$ e $b|h$, então $b|(mg + nh)$ para inteiros quaisquer m e n.

Para ver esse último ponto, observe que

Se $b|g$, então g está na forma $g = b \times g_1$ para algum inteiro g_1.
Se $b|h$, então h está na forma $h = b \times h_1$ para algum inteiro h_1.

Assim

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

e, portanto, b divide $mg + nh$.

$b = 7; g = 14; h = 63; m = 3; n = 2.$
$7|14$ e $7|63$. Para mostrar: $7|(3 \times 14 + 2 \times 63)$
Temos $(3 \times 14 + 2 \times 63) = 7(3 \times 2 + 2 \times 9)$
E é óbvio que $7|(7(3 \times 2 + 2 \times 9))$

Observe que se $a \equiv 0 \pmod{n}$ então $n|a$.

Propriedades das congruências

As congruências têm as seguintes propriedades:

1. $a \equiv b \pmod{n}$ se $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implica $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ e $b \equiv c \pmod{n}$ implica $a \equiv c \pmod{n}$.

3. Acabamos de usar o operador mod de duas maneiras diferentes: primeiro, como um operador binário que produz um resto, como na expressão a mod b; segundo, como uma relação de congruência que mostra a equivalência de dois binários, como na expressão $a = b \pmod{n}$. Para distinguir os dois usos, o termo mod é delimitado em parênteses para uma relação de congruência; isso é comum, mas não universal na literatura. Veja uma discussão mais profunda no Apêndice D.

Para demonstrar o primeiro ponto, se $n \mid (a - b)$, então $(a - b) = kn$ para algum k. Portanto, podemos escrever $a = b + kn$. Portanto, $(a \bmod n)$ = (resto quando $b + kn$ é dividido por n) = (resto quando b é dividido por n) = $(b \bmod n)$.

$23 \equiv 8 \pmod 5$	porque	$23 - 8 = 15 = 5 \times 3$
$-11 \equiv 5 \pmod 8$	porque	$-11 - 5 = -16 = 8 \times (-2)$
$81 \equiv 0 \pmod{27}$	porque	$81 - 0 = 81 = 27 \times 3$

Os pontos restantes podem ser facilmente provados.

Operações de aritmética modular

Observe que, por definição (Figura 4.2), o operador (mod n) mapeia todos os inteiros para o conjunto de inteiros $\{0, 1, \ldots (n - 1)\}$. Isso sugere a pergunta: como podemos realizar operações aritméticas dentro dos limites desse conjunto? Acontece que podemos; essa técnica é conhecida como **aritmética modular**.

A aritmética modular exibe as seguintes propriedades:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Demonstramos a primeira propriedade. Defina $(a \bmod n) = r_a$ e $(b \bmod n) = r_b$. Então, podemos escrever $a = r_a + jn$ para algum inteiro j e $b = r_b + kn$ para algum inteiro k. Então

$$\begin{aligned}(a + b) \bmod n &= (ra + jn + rb + kn) \bmod n \\ &= (ra + rb + (k + j)n) \bmod n \\ &= (ra + rb) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

As propriedades restantes também são facilmente comprovadas. Aqui estão exemplos de três propriedades:

$11 \bmod 8 = 3; 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \; mod \; 8) - (15 \; mod \; 8)] \; mod \; 8 = -4 \; mod \; 8 = 4$
$(11 - 15) \; mod \; 8 = -4 \; mod \; 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

A exponenciação é realizada pela multiplicação repetida, como na aritmética comum. (Teremos mais a dizer sobre a exponenciação no Capítulo 8.)

Para encontrar $11^7 \bmod 13$, podemos proceder da seguinte forma:
$11^2 = 121 \equiv 4 \pmod{13}$
$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$
$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$

Assim, as regras para a aritmética comum usando adição, subtração e multiplicação são transportadas para a aritmética modular.

A Tabela 4.1 oferece uma ilustração da adição e da multiplicação modulares módulo 8. Examinando a adição, os resultados são diretos e existe um padrão regular na matriz. As duas matrizes são simétricas sobre a diagonal principal, em conformidade com a propriedade comutativa da adição e da multiplicação. Como na adição normal, existe um inverso aditivo, ou negativo, a cada inteiro em aritmética modular. Nesse caso, o negativo de um inteiro x é o inteiro y, de modo que $(x + y) \bmod 8 = 0$. Para encontrar o inverso aditivo de um inteiro na coluna da esquerda, percorra a linha correspondente da matriz para encontrar o valor 0; o inteiro no topo dessa coluna é o inverso aditivo; assim, $(2 + 6) \bmod 8 = 0$. De modo semelhante, as entradas na tabela de multiplicação são diretas. Na aritmética comum, existe um inverso multiplicativo, ou recíproco, para cada inteiro. Na aritmética modular mod 8, o inverso multiplicativo de x é o inteiro y, de modo que $(x \times y) \bmod 8 = 1 \bmod 8$. Agora, para encontrar o inverso multiplicativo de um inteiro a partir da tabela de multiplicação, percorra a matriz na linha para esse inteiro, para encontrar o valor 1; o inteiro no topo dessa coluna é o inverso multiplicativo; assim, $(3 \times 3) \bmod 8 = 1$. Observe que nem todos os inteiros mod 8 têm um inverso multiplicativo; veja mais sobre isso adiante.

Tabela 4.1 Aritmética módulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Adição módulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplicação módulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Inversos aditivos e multiplicativos módulo 8

Propriedades da aritmética modular

Defina o conjunto Z_n como o conjunto de inteiros não negativos menores que n:

$$Z_n = \{0, 1, ..., (n - 1)\}$$

Isso é conhecido como o conjunto de resíduos, ou classes de resíduos módulo n. Para ser mais preciso, cada inteiro em Z_n representa uma classe de resíduo. Podemos rotular as classes de resíduo módulo n como [0], [1], [2], ..., [n − 1], onde

$$[r] = \{a: a \text{ é um inteiro}, a \equiv r \pmod{n}\}$$

As classes de resíduos módulo 4 são
[0] = { ... , −16, −12, −8, −4, 0, 4, 8, 12, 16, ...
[1] = { ... , −15, −11, −7, −3, 1, 5, 9, 13, 17, ... }
[2] = { ... , −14, −10, −6, −2, 2, 6, 10, 14, 18, ... }
[3] = { ... , −13, −9, −5, −1, 3, 7, 11, 15, 19, ... }

De todos os inteiros em uma classe de resíduo, o menor inteiro não negativo é aquele normalmente usado para representar a classe de resíduo. Encontrar o menor inteiro não negativo ao qual k é congruente módulo n é chamado de **reduzir *k* módulo *n***.

Se realizarmos a aritmética modular dentro de Z_n, as propriedades mostradas na Tabela 4.2 irão se manter para inteiros em Z_n. Assim, Z_n é um anel comutativo com um elemento de identidade multiplicativa (Figura 4.1).

Existe uma peculiaridade da aritmética modular que a distingue da aritmética comum. Primeiro, observe que, como na aritmética comum, podemos escrever o seguinte:

$$\textbf{se } (a + b) \equiv (a + c) \pmod{n} \textbf{ então } b \equiv c \pmod{n} \qquad \textbf{(4.2)}$$

$$(5 + 23) \equiv (5 + 7) \pmod{8}; \; 23 \equiv 7 \pmod{8}$$

Tabela 4.2 Propriedades da aritmética modular para inteiros em Z_n

Propriedade	Expressão
Leis comutativas	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Leis associativas	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Lei distributiva	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identidades	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Inverso aditivo $(-w)$	Para cada $w \in Z_n$, existe um z, de modo que $w + z \equiv 0 \bmod n$

A Equação (4.2) é coerente com a existência de um inverso aditivo. Acrescentando o inverso aditivo de a aos dois lados da Equação (4.2), temos:

$$((-a) + a + b) \equiv ((-a) + a + c)(\text{mod } n)$$
$$b \equiv c \ (\text{mod } n)$$

Porém, a seguinte declaração é verdadeira somente com a condição anexada:

se $(a \times b) \equiv (a \times c)(\text{mod } n)$ **então** $b \equiv c$ (mod n) **se** a é relativamente primo de n **(4.3)**

onde o termo *relativamente primo* é definido da seguinte forma: dois inteiros são **relativamente primos** se seu único fator inteiro positivo comum for 1. Semelhante ao caso da Equação (4.2), podemos dizer que a Equação (4.3) é coerente com a existência de um inverso multiplicativo. Aplicando o inverso multiplicativo de a aos dois lados da Equação (4.2), temos:

$$((a^{-1})ab) \equiv ((a^{-1})ac)(\text{mod } n)$$
$$b \equiv c \ (\text{mod } n)$$

Para ver isso, considere um exemplo em que a condição da Equação (4.3) não se mantém. Os inteiros 6 e 8 não são relativamente primos, pois têm o fator comum 2. Temos o seguinte:

$$6 \times 3 = 18 \equiv 2 \ (\text{mod } 8)$$
$$6 \times 7 = 42 \equiv 2 \ (\text{mod } 8)$$

Porém, $3 \not\equiv 7$ (mod 8).

O motivo para esse resultado estranho é que, para qualquer módulo n genérico, um multiplicador a aplicado por sua vez aos inteiros de 0 até $(n - 1)$ deixará de produzir um conjunto completo de resíduos se a e n tiverem quaisquer fatores em comum.

Com $a = 6$ e $n = 8$,

Z8	0	1	2	3	4	5	6	7
Multiplique por 6	0	6	12	18	24	30	36	42
Resíduos	0	6	4	2	0	6	4	2

Como não temos um conjunto completo de resíduos quando multiplicamos por 6, mais de um inteiro em Z_8 é mapeado para o mesmo resíduo. Especificamente, $6 \times 0 \bmod 8 = 6 \times 4 \bmod 8$; $6 \times 1 \bmod 8 = 6 \times 5 \bmod 8$; e assim por diante. Como esse é um mapeamento muitos-para-um, não existe um inverso exclusivo para a operação de multiplicação.

Contudo, se tomarmos $a = 5$ e $n = 8$, cujo único fator comum é 1,

Z8	0	1	2	3	4	5	6	7
Multiplique por 5	0	5	10	15	20	25	30	35
Resíduos	0	5	2	7	4	1	6	3

A linha de resíduos contém todos os inteiros em Z_8 em uma ordem diferente.

Em geral, um inteiro tem um inverso multiplicativo em Z_n se esse inteiro for relativamente primo de n. A Tabela 4.1c mostra que os inteiros 1, 3, 5 e 7 têm um inverso multiplicativo em Z_8, mas 2, 4 e 6 não.

4.3 O ALGORITMO EUCLIDIANO

Uma das técnicas básicas da teoria dos números é o algoritmo euclidiano, que é um procedimento simples para determinar o máximo divisor comum de dois inteiros positivos.

Máximo divisor comum

Lembre-se de que b diferente de zero é definido como sendo um divisor de a se $a = mb$ para algum m, onde a, b e m são inteiros. Usaremos a notação mdc(a, b) para significar o máximo divisor comum de a e b. O inteiro positivo c é considerado como o máximo divisor comum de a e b se

1. c é um divisor de a e de b;
2. qualquer divisor de a e b é um divisor de c.

Uma definição equivalente é a seguinte:

$$\text{mdc}(a, b) = \max[k, \text{ de modo que } k|a \text{ e } k|b]$$

Como exigimos que o máximo divisor comum seja positivo, $\text{mdc}(a, b) = \text{mdc}(a, -b) = \text{mdc}(-a, b) = \text{mdc}(-a, -b)$. Em geral, $\text{mdc}(a, b) = \text{mdc}(|a|, |b|)$.

$$\boxed{\text{mdc}(60, 24) = \text{mdc}(60, -24) = 12}$$

Além disso, como todos os inteiros diferentes de zero dividem 0, temos $\text{mdc}(a, 0) = |a|$.

Dissemos que dois inteiros a e b são relativamente primos se seu único fator inteiro positivo comum for 1. Isso é equivalente a dizer que a e b são relativamente primos se $\text{mdc}(a, b) = 1$.

> 8 e 15 são relativamente primos porque os divisores positivos de 8 são 1, 2, 4 e 8, e os divisores positivos de 15 são 1, 3, 5 e 15, de modo que 1 é o único inteiro nas duas listas.

Encontrando o máximo divisor comum

O algoritmo euclidiano é baseado no teorema a seguir: Para qualquer inteiro não negativo a e qualquer inteiro positivo b,

$$\text{mdc}(a, b) = \text{mdc}(b, a \bmod b) \tag{4.4}$$

$$\boxed{\text{mdc}(55, 22) = \text{mdc}(22, 55 \bmod 22) = \text{mdc}(22, 11) = 11}$$

Para ver que a Equação (4.4) funciona, considere $d = \text{mdc}(a,b)$. Então, pela definição do mdc, $d|a$ e $d|b$. Para qualquer inteiro positivo b, a pode ser expresso na forma

$$a = kb + r \equiv r \pmod b$$
$$a \bmod b = r$$

com inteiros k, r. Portanto, $(a \bmod b) = a - kb$ para algum inteiro k. Mas,como $d|b$, ele também divide kb. Também temos $d|a$. Portanto, $d|(a \bmod b)$. Isso mostra que d é um divisor comum de b e $(a \bmod b)$. Reciprocamente, se d é um divisor comum de b e $(a \bmod b)$, então $d|kb$ e, assim, $d|[kb + (a \bmod b]]$, que é equivalente a $d|a$. Assim, o conjunto de divisores comuns de a e b é igual ao conjunto dos divisores comuns de b e $(a \bmod b)$. Portanto, o mdc de um par é o mesmo que o mdc do outro par, provando o teorema.

A Equação (4.4) pode ser usada repetidamente para determinar o máximo divisor comum.

$$\boxed{\begin{array}{l}\text{mdc}(18, 12) = \text{mdc}(12, 6) = \text{mdc}(6, 0) = 6\\ \text{mdc}(11, 10) = \text{mdc}(10, 1) = \text{mdc}(1, 0) = 1\end{array}}$$

O algoritmo euclidiano faz uso repetitivo da Equação (4.4) para determinar o máximo divisor comum, da seguinte forma. O algoritmo supõe que $a > b > 0$. É aceitável restringir o algoritmo a inteiros positivos, pois $\text{mdc}(a, b) = \text{mdc}(|a|,|b|)$.

```
EUCLIDES(a, b)
1.  A ← a; B ← b
2.  if B = 0  return  A = mdc(a, b)
3.  R = A mod B
4.  A ← B
5.  B ← R
6.  goto 2
```

O algoritmo tem a seguinte progressão:

$$A_1 = B_1 \times Q_1 + R_1$$
$$A_2 = B_2 \times Q_2 + R_2$$
$$A_3 = B_3 \times Q_3 + R_3$$
$$A_4 = B_4 \times Q_4 + R_4$$

Para encontrar mdc(1970, 1066)

$1970 = 1 \times 1066 + 904$	mdc(1066, 904)
$1066 = 1 \times 904 + 162$	mdc(904, 162)
$904 = 5 \times 162 + 94$	mdc(162, 94)
$162 = 1 \times 94 + 68$	mdc(94, 68)
$94 = 1 \times 68 + 26$	mdc(68, 26)
$68 = 2 \times 26 + 16$	mdc(26, 16)
$26 = 1 \times 16 + 10$	mdc(16, 10)
$16 = 1 \times 10 + 6$	mdc(10, 6)
$10 = 1 \times 6 + 4$	mdc(6, 4)
$6 = 1 \times 4 + 2$	mdc(4, 2)
$4 = 2 \times 2 + 0$	mdc(2, 0)

Portanto, mdc(1970,1066) = 2

O leitor alerta poderá perguntar como podemos ter certeza de que esse processo termina. Ou seja, como podemos ter certeza de que, em algum ponto, B divide A? Se não, obteríamos uma seqüência infinita de inteiros positivos, cada um estritamente menor que o anterior, e isso é claramente impossível.

4.4 CORPOS FINITOS NA FORMA GF(*P*)

Na Seção 4.1, definimos um corpo como um conjunto que obedece a todos os axiomas da Figura 4.1 e demos alguns exemplos de corpos infinitos. Os corpos infinitos não são de interesse específico no contexto da criptografia. Porém, os corpos finitos desempenham um papel fundamental em muitos algoritmos criptográficos. Pode-se mostrar que a ordem de um corpo finito (número de elementos no corpo) precisa ser uma potência de um primo p^n, onde n é um inteiro positivo. Explicaremos os números primos com detalhes no Capítulo 8. Aqui só precisamos dizer que um número primo é um inteiro cujos únicos fatores inteiros positivos são ele próprio e 1. Ou seja, os únicos inteiros positivos que são divisores de p são p e 1.

O corpo finito de ordem p^n geralmente é escrito como GF(p^n); GF significa Galois field (corpo de Galois) em homenagem ao primeiro matemático a estudar os corpos finitos. Dois casos especiais são de interesse para nossos propósitos. Para $n = 1$, temos o corpo finito GF(p); esse corpo finito tem uma estrutura diferente daquela dos corpos finitos com $n > 1$, e é estudado nesta seção. Na Seção 4.6, examinamos os corpos finitos na forma GF(2^n).

Corpos finitos de ordem *p*

Para determinado primo p, o corpo finito de ordem p, GF(p), é definido como o conjunto Z_p de inteiros $\{0, 1, ..., p - 1\}$, juntamente com as operações aritméticas de módulo p.

Lembre-se de que mostramos na Seção 4.2 que o conjunto Z_n de inteiros $\{0, 1, ... , n - 1\}$, juntamente com as operações aritméticas de módulo n, é um anel comutativo (Tabela 4.2). Observamos ainda que qualquer inteiro em Z_n tem um inverso multiplicativo se e somente se esse inteiro for relativamente primo de n [ver explicação da Equação (4.3)].[4] Se n for primo, então todos os inteiros não nulos em Z_n serão relativamente primos de n e, portanto, existe um

4. Conforme indicado na explicação da Equação (4.3), dois inteiros são **relativamente primos** se seu único fator inteiro positivo comum for 1.

inverso multiplicativo para todos os inteiros não nulos em Z_n. Assim, podemos acrescentar as seguintes propriedades às que estão listadas na Tabela 4.2 para Z_p:

Inverso multiplicativo (w^{-1})	Para cada $w \in Z_p$, $w \neq 0$, existe um $z \in Z_p$ de modo que $w \times z \equiv 1 \pmod p$.

Como w é relativamente primo de p, se multiplicarmos todos os elementos de Z_p por w, os resíduos resultantes serão todos os elementos de Z_p permutados. Assim, exatamente um dos resíduos tem o valor 1. Portanto, existe algum inteiro em Z_p que, quando multiplicado por w, gera o resíduo 1. Esse inteiro é o inverso multiplicativo de w, designado como w^{-1}. Portanto, Z_p é, de fato, um corpo finito. Além disso, a Equação (4.3) é coerente com a existência de um inverso multiplicativo e pode ser reescrita sem a condição:

$$\textbf{se } (a \times b) \equiv (a \times c) \pmod p \textbf{ então } b \equiv c \pmod p \qquad \textbf{(4.5)}$$

Multiplicando os dois lados da Equação (4.5) pelo inverso multiplicativo de a, temos:

$$((a^{-1}) \times a \times b) \equiv ((a^{-1}) \times a \times c) \pmod p$$
$$b \equiv c \pmod p$$

O corpo finito mais simples é GF(2). Suas operações aritméticas são facilmente resumidas:

	0	1
0	0	1
1	1	0

Adição

	0	1
0	0	0
1	0	1

Multiplicação

w		
0	0	—
1	1	1

Inversos

Nesse caso, a adição é equivalente à operação ou-exclusive (XOR), e a multiplicação é equivalente à operação lógica AND.

A Tabela 4.3 mostra GF(7). Esse é um corpo de ordem 7 usando aritmética modular com módulo 7. Como podemos ver, ele satisfaz todas as propriedades exigidas de um corpo (Figura 4.1). Compare essa tabela com a Tabela 4.1. No último caso, vemos que o conjunto Z_8, usando aritmética modular com módulo 8, não é um corpo. Mais adiante neste capítulo, mostramos como definir operações de adição e multiplicação sobre Z_8 de modo a formar um corpo finito.

Encontrando o inverso multiplicativo em GF(p)

É fácil encontrar o inverso multiplicativo de um elemento em GF(p) para valores pequenos de p. Você simplesmente constrói uma tabela de multiplicação, como mostramos na Tabela 4.3b, e o resultado desejado pode ser lido diretamente. Porém, para valores grandes de p, essa técnica não é prática.

Se mdc(m, b) = 1, então b tem um inverso multiplicativo módulo m. Ou seja, para o inteiro positivo $b < m$, existe um $b^{-1} < m$ de modo que $bb^{-1} = 1 \bmod m$. O algoritmo euclidiano pode ser estendido de modo que, além de encontrar mdc(m, b), se o mdc for 1, o algoritmo retornará o inverso multiplicativo de b.

```
EUCLIDES ESTENDIDO(m, b)
1.  (A1, A2, A3) ← (1, 0, m); (B1, B2, B3) ← (0, 1, b)
2.  if B3 = 0 return  A3 = gcd(m, b);  não existe inverso
3.  if B3 = 1 return  B3 = gcd(m, b);  B2 = b^-1 mod m
4.  Q = ⌊A3/B3⌋
5.  (T1, T2, T3) ← (A1 - QB1, A2 - QB2, A3 - QB3)
6.  (A1, A2, A3) ← (B1, B2, B3)
7.  (B1, B2, B3) ← (T1, T2, T3)
8.  goto 2
```

Tabela 4.3 Aritmética em GF(7)

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) Adição módulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplicação módulo 7

w	$-w$	w^{-1}
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Inversos aditivo e multiplicativo módulo 7

No decorrer do cálculo, os seguintes relacionamentos permanecem:

$$m\text{T1} + b\text{T2} = \text{T3} \qquad m\text{A1} + b\text{A2} = \text{A3} \qquad m\text{B1} + b\text{B2} = \text{B3}$$

Para ver se esse algoritmo retorna corretamente o mdc(m, b), observe que, se igualarmos A e B no algoritmo euclidiano com A3 e B3 no algoritmo euclidiano estendido, então o tratamento das duas variáveis será idêntico. Em cada iteração do algoritmo euclidiano, A é definido como sendo igual ao valor anterior de B e B é definido como sendo igual ao valor anterior de A mod B. De modo semelhante, a cada etapa do algoritmo euclidiano estendido, A3 é definido como sendo igual ao valor anterior de B3, e B3 é definido como sendo igual ao valor anterior de A3 menos o quociente inteiro de A3 multiplicado por B3. Esse último valor é simplesmente o resto de A3 dividido por B3, que é A3 mod B3.

Observe também que, se mdc(m, b) = 1, então a etapa final teria B3 = 0 e A3 = 1. Portanto, na etapa anterior, B3 = 1. Mas, se B3 = 1, então podemos dizer o seguinte:

$$\begin{aligned} m\text{B1} + b\text{B2} &= \text{B3} \\ m\text{B1} + b\text{B2} &= \\ b\text{B2} &= 1 - m\text{B1} \\ b\text{B2} &\equiv 1 \pmod{m} \end{aligned}$$

E B2 é o inverso multiplicativo de b, módulo m.

A Tabela 4.4 é um exemplo da execução do algoritmo. Ela mostra que mdc(1750, 550) = 1 e que o inverso multiplicativo de 550 é 355, ou seja, $550 \times 355 \equiv 1 \pmod{1759}$.

Tabela 4.4 Encontrando o inverso multiplicativo de 550 em GF(1759).

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	−3	109
5	1	−3	109	−5	16	5
21	−5	16	5	106	−339	4
1	106	−339	4	−111	355	1

Resumo

Nesta seção, mostramos como construir um corpo finito de ordem p, em que p é primo. Especificamente, definimos GF(p) com as seguintes propriedades:

1. GF(p) consiste em p elementos.
2. As operações binárias + e × são definidas sobre o conjunto. As operações de adição, subtração, multiplicação e divisão podem ser realizadas sem sair do conjunto. Cada elemento do conjunto diferente de 0 tem um inverso multiplicativo.

Mostramos que os elementos de GF(p) são os inteiros $\{0, 1, ..., p - 1\}$ e que as operações aritméticas são adição e multiplicação mod p.

4.5 ARITMÉTICA POLINOMIAL

Antes de buscarmos nossa discussão de corpos finitos, precisamos apresentar o assunto interessante da aritmética de polinômios. Vamos nos referir aos polinômios em uma única variável x, e podemos distinguir três classes de aritmética de polinômios:

- Aritmética de polinômios ordinários, usando as regras básicas da álgebra
- Aritmética de polinômios em que a aritmética sobre os coeficientes é realizada módulo p; ou seja, os coeficientes estão em GF(p)
- Aritmética de polinômios em que os coeficientes estão em GF(p), e os polinômios são definidos módulo um polinômio $m(x)$, cuja potência mais alta é algum inteiro n

Esta seção examina as duas primeiras classes, e a próxima seção aborda a última classe.

Aritmética polinomial comum

Um polinômio de grau n (inteiro $n \geq 0$) é uma expressão na forma

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

onde os a_i são elementos de algum conjunto designado de números S, chamado **conjunto de coeficientes**, e $a_n \neq 0$. Dizemos que tais polinômios são definidos sobre o conjunto de coeficientes S.

Um polinômio de grau zero é chamado de **polinômio constante** e é simplesmente um elemento do conjunto de coeficientes. Um polinômio de grau n é considerado como um **polinômio mônico** se $a_n = 1$.

No contexto da álgebra abstrata, normalmente não nos interessamos por avaliar um polinômio para determinado valor de x [por exemplo, $f(7)$]. Para enfatizar esse ponto, a variável x às vezes é referenciada como **indeterminada**.

A aritmética polinomial inclui as operações de adição, subtração e multiplicação. Essas operações são definidas de uma maneira natural, como se a variável x fosse um elemento de S. A divisão é definida de modo semelhante, mas exige que S seja um corpo. Alguns exemplos de corpos incluem os números reais, números racionais e Z_p para o primo p. Observe que o conjunto de todos os inteiros não é um corpo e não admite divisão polinomial.

Adição e subtração são realizados somando-se ou subtraindo-se coeficientes correspondentes. Assim, se

$$f(x) = \sum_{i=0}^{n} a_i x^i; \quad g(x) = \sum_{i=0}^{m} b_i x^i; \quad n \geq m$$

então a adição é definida como

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i)x^i + \sum_{i=m+1}^{n} a_i x^i$$

e a multiplicação é definida como

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

onde

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0$$

Na última fórmula, tratamos a_i como zero, para $i > n$, e b_i como zero, para $i > m$. Observe que o grau do produto é igual à soma dos graus dos dois polinômios.

Como exemplo, considere $f(x) = x^3 + x^2 + 2$ e $g(x) = x^2 - x + 1$, onde S é o conjunto de inteiros. Então

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$
$$f(x) - g(x) = x^3 + x + 1$$
$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

$$\begin{array}{rrrrrr} & x^3 & +\,x^2 & & +\,2 & \\ + & & (x^2 & -\,x & +\,1) & \\ \hline & x^3 & +\,2x^2 & -\,x & +\,3 & \end{array}$$

(a) Adição

$$\begin{array}{rrrrrr} & x^3 & +\,x^2 & & +\,2 & \\ - & & (x^2 & -\,x & +\,1) & \\ \hline & x^3 & & +\,x & +\,1 & \end{array}$$

(b) Subtração

$$\begin{array}{rrrrrrr} & & x^3 & +\,x^2 & & +\,2 & \\ & & \times & (x^2 & -\,x & +\,1) & \\ \hline & & x^3 & +\,x^2 & & +\,2 & \\ & -\,x^4 & -\,x^3 & & -\,2x & & \\ x^5 & +\,x^4 & & +\,2x^2 & & & \\ \hline x^5 & & & +\,3x^2 & -\,2x & +\,2 & \end{array}$$

(c) Multiplicação

$$\begin{array}{r|rrrrr} & & x & +\,2 & & \\ \hline x^2 - x + 1 & x^3 & +\,x^2 & & +\,2 & \\ & x^3 & -\,x^2 & +\,x & & \\ \hline & & 2x^2 & -\,x & +\,2 & \\ & & 2x^2 & -\,2x & +\,2 & \\ \hline & & & x & & \end{array}$$

(d) Divisão

Figura 4.3 Exemplos de aritmética de polinômios.

Aritmética de polinômios com coeficientes em Z_p

Vamos agora considerar os polinômios em que os coeficientes são elementos de algum corpo F. Vamos nos referir a isso como um polinômio sobre o corpo F. Nesse caso, é fácil mostrar que o conjunto desses polinômios é um anel, conhecido como **anel polinomial**. Ou seja, se considerarmos que cada polinômio distinto é um elemento do conjunto, então esse conjunto será um anel.[5]

Quando a aritmética de polinômios é realizada em polinômios sobre um corpo, então a divisão é possível. Observe que isso não significa que a *divisão exata* é possível. Vamos esclarecer essa distinção. Dentro de um corpo, dados dois elementos a e b, o quociente a/b também é um elemento do corpo. Porém, dado um anel R que não é um corpo, em geral a divisão resultará em um quociente e um resto; essa não é divisão exata.

Considere a divisão 5/3 dentro de um conjunto S. Se S é o conjunto dos números racionais, que é um corpo, então o resultado é simplesmente expresso como 5/3 e é um elemento de S. Agora, suponha que S seja o corpo Z_7. Nesse caso, calculamos (usando a Tabela 4.3c):

$$5/3 = (5 \times 3^{-1}) \bmod 7 = (5 \times 5) \bmod 7 = 4$$

que é uma solução exata. Finalmente, suponha que S seja o conjunto de inteiros, que é um anel, mas não um corpo. Então, 5/3 produz um quociente de 1 e um resto de 2:

$$5/3 = 1 + 2/3$$
$$5 = 1 \times 3 + 2$$

Agora, se tentarmos realizar a divisão polinomial sobre um conjunto de coeficientes que não seja um corpo, descobriremos que a divisão nem sempre é definida.

Se o conjunto de coeficientes for de inteiros, então $(5x^2)/(3x)$ não teria uma solução, pois exigiria um coeficiente com um valor de 5/3, que não estaria no conjunto de coeficientes. Suponha que realizemos a mesma divisão polinomial sobre Z_7. Então, temos $(5x^2)/(3x) = 4x$, que é um polinômio válido sobre Z_7.

5. De fato, o conjunto de polinômios cujos coeficientes são elementos de um anel comutativo forma um anel polinomial, mas isso não nos interessa no contexto atual.

Porém, conforme demonstramos há pouco, mesmo que o conjunto de coeficientes seja um corpo, a divisão polinomial não será necessariamente exata. Em geral, a divisão produzirá um quociente e um resto:

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$$
$$f(x) = q(x)g(x) + r(x) \qquad \textbf{(4.6)}$$

Se o grau de $f(x)$ for n e o grau de $g(x)$ for m ($n \geq m$), então o grau do quociente $q(x)$ será $n - m$ e o grau do resto será no máximo $m - 1$. Com o conhecimento de que os restos são permitidos, podemos dizer que a divisão polinomial será possível se o conjunto de coeficientes for um corpo.

Em uma analogia com a aritmética de inteiros, podemos escrever $f(x)$ mod $g(x)$ para o resto $r(x)$ na Equação (4.6). Ou seja, $r(x) = f(x)$ mod $g(x)$. Se não houver resto [ou seja, $r(x) = 0$], então podemos dizer que $g(x)$ **divide** $f(x)$, escrito como $g(x)|f(x)$; de modo equivalente, podemos dizer que $g(x)$ é um **fator** de $f(x)$ ou $g(x)$ é um **divisor** de $f(x)$.

Para o exemplo anterior, [$f(x) = x^3 + x^2 + 2$ e $g(x) = x^2 - x + 1$], $f(x)/g(x)$ produz um quociente de $q(x) = x + 2$ e um resto $r(x) = x$, como mostra a Figura 4.3d. Isso é facilmente verificado observando-se que

$$q(x)g(x) + r(x) = (x + 2)(x^2 - x + 1) + x = (x^3 + x^2 - x + 2) + x$$
$$= x^3 + x^2 + 2 = f(x)$$

Para os nossos propósitos, os polinômios sobre GF(2) nos interessam mais. Lembre-se de que, conforme a Seção 4.4, em GF(2) a adição é equivalente à operação XOR e a multiplicação é equivalente à operação de AND lógico. Além disso, adição e subtração são equivalentes mod 2: $1 + 1 = 1 - 1 = 0$; $1 + 0 = 1 - 0 = 1$; $0 + 1 = 0 - 1 = 1$.

A Figura 4.4 mostra um exemplo de aritmética de polinômio sobre GF(2). Para $f(x) = (x^7 + x^5 + x^4 + x^3 + x + 1)$ e $g(x) = (x^3 + x + 1)$, a figura mostra $f(x) + g(x)$; $f(x) - g(x)$; $f(x) \times g(x)$; e $f(x)/g(x)$. Observe que $g(x)|f(x)$.

$$\begin{array}{lllll}
x^7 & + x^5 + x^4 + x^3 & & + x + 1 \\
 & & + (x^3 & + x + 1) \\
\hline
x^7 & + x^5 + x^4 & & \\
\end{array}$$

(a) Adição

$$\begin{array}{lllll}
x^7 & + x^5 + x^4 + x^3 & & + x + 1 \\
 & & - (x^3 & + x + 1) \\
\hline
x^7 & + x^5 + x^4 & & \\
\end{array}$$

(b) Subtração

$$\begin{array}{lllll}
 & x^7 & + x^5 + x^4 + x^3 & + x + 1 \\
 & & \times (x^3 & + x + 1) \\
\hline
 & x^7 & + x^5 + x^4 + x^3 & + x + 1 \\
x^8 & + x^6 + x^5 + x^4 & + x^2 + x & \\
x^{10} & + x^8 + x^7 + x^6 & + x^4 + x^3 & \\
\hline
x^{10} & & + x^4 \quad + x^2 & + 1
\end{array}$$

(c) Multiplicação

$$\begin{array}{lllll}
 & x^4 + 1 & & \\
\cline{2-4}
x^3 + x + 1 \,/ & x^7 + x^5 + x^4 + x^3 & & + x + 1 \\
 & x^7 + x^5 + x^4 & & \\
\cline{2-3}
 & & x^3 & + x + 1 \\
 & & x^3 & + x + 1 \\
\cline{3-4}
\end{array}$$

(d) Divisão

Figura 4.4 Exemplos de aritmética de polinômio sobre GF(2).

Um polinômio $f(x)$ sobre um corpo F é chamado de **irredutível** se e somente se $f(x)$ não puder ser expresso como um produto de dois polinômios, ambos sobre F, e ambos de grau menor que o de $f(x)$. Por analogia com inteiros, um polinômio irredutível também é chamado de **polinômio primo**.

O polinômio[6] $f(x) = x^4 + 1$ sobre GF(2) é redutível, pois $x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$.

Considere o polinômio $f(x) = x3 + x + 1$. Apenas com a inspeção, fica claro que x não é um fator de $f(x)$. Facilmente mostramos que $x + 1$ não é um fator de $f(x)$:

$$
\begin{array}{r|l}
 & x^2 + x \\
\hline
x + 1 & x^3 \qquad + x + 1 \\
 & \underline{x^3 + x^2} \\
 & \qquad x^2 + x \\
 & \qquad \underline{x^2 + x} \\
 & \qquad\qquad 1
\end{array}
$$

Assim, $f(x)$ não tem fatores de grau 1. Mas fica claro que, se $f(x)$ for redutível, ele precisará ter um fator de grau 2 e um fator de grau 1. Portanto, $f(x)$ é irredutível.

Encontrando o máximo divisor comum

Podemos estender a analogia entre a aritmética polinomial sobre um corpo e a aritmética de inteiros definindo o máximo divisor comum da seguinte forma. O polinômio $c(x)$ é considerado como o máximo divisor comum de $a(x)$ e $b(x)$ se

1. $c(x)$ dividir tanto $a(x)$ quanto $b(x)$;
2. qualquer divisor de $a(x)$ e $b(x)$ for um divisor de $c(x)$.

Uma definição equivalente é a seguinte: mdc$[a(x), b(x)]$ é o polinômio do grau máximo que divide tanto $a(x)$ quanto $b(x)$.

Podemos adaptar o algoritmo euclidiano para calcular o máximo divisor comum de dois polinômios. A igualdade na Equação (4.4) pode ser reescrita como o seguinte teorema:

$$\text{mdc}[a(x), b(x)] = \text{mdc}[b(x), a(x) \bmod b(x)] \qquad \textbf{(4.7)}$$

O algoritmo euclidiano para os polinômios pode ser indicado da seguinte maneira. O algoritmo presume que o grau de $a(x)$ é maior que o grau de $b(x)$. Então, para encontrar mdc$[a(x), b(x)]$,

```
EUCLIDES[a(x), b(x)]
1.  A(x) ← a(x); B(x) ← b(x)
2.  if B(x) = 0   return A(x) = mdc[a(x), b(x)]
3.  R(x) = A(x) mod B(x)
4.  A(x) ← B(x)
5.  B(x) ← R(x)
6.  goto 2
```

Encontre mdc$[a(x), b(x)]$ para $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ e $b(x) = x^4 + x^2 + x + 1$.

$$
\begin{array}{r|l}
 & x^2 + x \\
\hline
x^4 + x^2 + x + 1 & x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
 & \underline{x^6 \qquad + x^4 + x^3 + x^2} \\
 & \qquad x^5 \qquad\qquad\qquad + x + 1 \\
 & \qquad \underline{x^5 \qquad + x^3 + x^2 + x} \\
 & \qquad\qquad\qquad x^3 + x^2 \qquad + 1
\end{array}
$$

$\text{R}(x) = \text{A}(x) \bmod \text{B}(x) = x^3 + x^2 + 1$

(continua)

6. No restante deste capítulo, a menos que indicado de outra forma, todos os exemplos são de polinômios sobre GF(2).

$A(x) = x^4 + x^2 + x + 1; B(x) = x^3 + x^2 + 1$

$$
\begin{array}{r|l}
 & x + 1 \\
\hline
x^3 + x^2 + 1 & x^4 \qquad + x^2 + x + 1 \\
 & \underline{x^4 + x^3 \qquad + x} \\
 & \qquad x^3 + x^2 \qquad + 1 \\
 & \qquad \underline{x^3 + x^2 \qquad + 1}
\end{array}
$$

$R(x) = A(x) \bmod B(x) = 0$
$\gcd[a(x), b(x)] = A(x) = x^3 + x^2 + 1$

Resumo

Iniciamos esta seção com uma explicação sobre aritmética com polinômios comuns. Na aritmética de polinômios comuns a variável não é avaliada, ou seja, não conectamos um valor da variável dos polinômios. Em vez disso, as operações aritméticas são realizadas sobre polinômios (adição, subtração, multiplicação, divisão) usando as regras comuns da álgebra. A divisão de polinômios não é permitida, a menos que os coeficientes sejam elementos de um corpo.

Em seguida, explicamos a aritmética de polinômios em que os coeficientes são elementos de GF(p). Nesse caso, a adição, subtração, multiplicação e divisão de polinômios são permitidas. Porém, a divisão não é exata; ou seja, em geral, a divisão resulta em um quociente e um resto.

Finalmente, mostramos que o algoritmo euclidiano pode ser estendido para encontrar o máximo divisor comum de dois polinômios cujos coeficientes são elementos de um corpo.

Todo o material desta seção oferece um alicerce para a próxima seção, em que os polinômios são usados para definir corpos finitos de ordem p^n.

4.6 CORPOS FINITOS NA FORMA GF(2^n)

Anteriormente, neste capítulo, mencionamos que a ordem de um corpo finito precisa ser da forma p^n, onde p é um primo e n é um inteiro positivo. Na Seção 4.4, examinamos o caso especial de corpos finitos com ordem p. Descobrimos que, usando a aritmética modular em Z_p, todos os axiomas para um corpo (Figura 4.1) são satisfeitos. Para polinômios sobre p^n, com $n > 1$, as operações módulo p^n não produzem um corpo. Nesta seção, mostramos que a estrutura satisfaz os axiomas para um corpo em um conjunto com p^n elementos, e nos concentramos em GF(2^n).

Motivação

Praticamente todos os algoritmos de criptografia, tanto de chave simétrica quanto pública, envolvem operações aritméticas sobre inteiros. Se uma das operações utilizadas no algoritmo for a divisão, então temos de trabalhar na aritmética definida sobre um corpo. Por conveniência e por eficiência de implementação, também gostaríamos de trabalhar com inteiros que se encaixam exatamente em determinado número de bits, sem padrões de bit desperdiçados. Ou seja, queremos trabalhar com inteiros no intervalo de 0 até $2^n - 1$, que se encaixem em uma palavra de n bits.

Suponha que queiramos definir um algoritmo de criptografia convencional que opere sobre os dados 8 bits de cada vez, e queiramos realizar a divisão. Com 8 bits, podemos representar os inteiros no intervalo de 0 a 255. Porém, 256 não é um número primo, de modo que, se a aritmética for realizada em Z_{256} (aritmética módulo 256), esse conjunto de inteiros não será um corpo. O número primo mais próximo menor que 256 é 251. Assim, o conjunto Z_{251}, usando a aritmética módulo 251, é um corpo. Porém, nesse caso, os padrões de 8 bits representando os inteiros de 251 a 255 não seriam usados, resultando em um uso ineficaz do espaço de armazenamento.

Como o exemplo anterior indica, se todas as operações aritméticas tiverem de ser usadas, e se quisermos representar um intervalo completo de inteiros em n bits, então a aritmética módulo 2^n não funcionará; de modo equivalente, o conjunto de inteiros módulo 2^n, para $n > 1$, não é um corpo. Além disso, mesmo que o algoritmo de criptografia use apenas adição e multiplicação, mas não a divisão, o uso do conjunto Z_{2^n} é questionável, como ilustra o exemplo a seguir.

Suponha que queiramos usar blocos de 3 bits em nosso algoritmo de criptografia, e usemos apenas as operações de adição e multiplicação. Então, a aritmética módulo 9 é bem definida, como mostra a Tabela 4.1. Porém, observe que, na tabela de multiplicação, os inteiros diferentes de zero não aparecem em um número igual de vezes. Por exemplo, existem apenas quatro ocorrências de 3, mas doze ocorrências de 4. Por outro lado, conforme mencionamos, existem corpos finitos na forma $GF(2^n)$, de modo que existe especificamente um corpo finito de ordem $2^3 = 8$. A aritmética para esse corpo aparece na Tabela 4.5. Nesse caso, o número de ocorrências de inteiros diferentes de zero é uniforme para a multiplicação. Para resumir,

Inteiro	1	2	3	4	5	6	7
Ocorrências em Z_8	4	8	4	12	4	8	4
Ocorrências em $GF(2^3)$	7	7	7	7	7	7	7

Por enquanto, vamos deixar de lado a questão de como as matrizes da Tabela 4.5 foram construídas e, em vez disso, fazer algumas observações.

1. As tabelas de adição e multiplicação são simétricas sobre a diagonal principal, em conformidade com a propriedade comutativa da adição e da multiplicação. Essa propriedade também é exibida na Tabela 4.1, que usa a aritmética mod 8.
2. Todos os elementos diferentes de zero definidos pela Tabela 4.5 possuem um inverso multiplicativo, diferente do caso com a Tabela 4.1.
3. O esquema definido pela Tabela 4.5 satisfaz todos os requisitos para um corpo finito. Assim, podemos nos referir a esse esquema como $GF(2^3)$.
4. Por conveniência, mostramos a atribuição de 3 bits usada para cada um dos elementos de $GF(2^3)$.

Intuitivamente, pode parecer que um algoritmo que mapeia os inteiros uniformemente sobre si mesmos pode ser criptograficamente mais fraco do que um que ofereça um mapeamento uniforme. Assim, os corpos finitos na forma $GF(2^n)$ são atraentes para algoritmos criptográficos.

Para resumir, estamos procurando um conjunto consistente de 2^n elementos, com uma definição de adição e multiplicação sobre o conjunto que define um corpo. Podemos atribuir um inteiro exclusivo no intervalo de 0 até $2^n - 1$ a cada elemento do conjunto. Lembre-se de que não usaremos a aritmética modular, pois vimos que isso não resulta em um corpo. Em vez disso, mostraremos como a aritmética de polinômios oferece um meio para construir o corpo desejado.

Aritmética polinomial modular

Considere o conjunto S de todos os polinômios de grau $n - 1$ ou menos sobre o corpo Z_p. Assim, cada polinômio tem a forma

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

onde cada a_i assume um valor no conjunto $\{0, 1, ..., p - 1\}$. Existe um total de p^n polinômios diferentes em S.

Para $p = 3$ e $n = 2$, os $3^2 = 9$ polinômios no conjunto são

0	x	2
1	$x + 1$	$2x + 1$
2	$x + 2$	$2x + 2$

Para $p = 2$ e $n = 3$, os $2^3 = 8$ polinômios no conjunto são

0	$x + 1$	$x^2 + x$
1	x^2	$x^2 + x + 1$
x	$x^2 + 1$	

Com a definição apropriada das operações aritméticas, cada conjunto S é um corpo finito. A definição consiste nos seguintes elementos:

1. A aritmética segue as regras comuns da aritmética polinomial usando as regras básicas da álgebra com os dois refinamentos a seguir.
2. A aritmética sobre os coeficientes é realizada módulo p. Ou seja, usamos as regras da aritmética para o corpo finito Z_p.

Tabela 4.5 Aritmética em GF(2^3).

		000	001	010	011	100	101	110	111
	+	0	1	2	3	4	5	6	7
000	0	0	1	2	3	4	5	6	7
001	1	1	0	3	2	5	4	7	6
010	2	2	3	0	1	6	7	4	5
011	3	3	2	1	0	7	6	5	4
100	4	4	5	6	7	0	1	2	3
101	5	5	4	7	6	1	0	3	2
110	6	6	7	4	5	2	3	0	1
111	7	7	6	5	4	3	2	1	0

(a) Adição

		000	001	010	011	100	101	110	111
	×	0	1	2	3	4	5	6	7
000	0	0	0	0	0	0	0	0	0
001	1	0	1	2	3	4	5	6	7
010	2	0	2	4	6	3	1	7	5
011	3	0	3	6	5	7	4	1	2
100	4	0	4	3	7	6	2	5	1
101	5	0	5	1	4	2	7	3	6
110	6	0	6	7	1	5	3	2	4
111	7	0	7	5	2	1	6	4	3

(b) Multiplicação

w	$-w$	w^{-1}
0	0	—
1	1	1
2	2	5
3	3	6
4	4	7
5	5	2
6	6	3
7	7	4

(c) Inversos aditivos e multiplicativos

3. Se a multiplicação resultar em um polinômio de grau maior que $n - 1$, então o polinômio será reduzido módulo algum polinômio irredutível $m(x)$ de grau n. Ou seja, dividimos por $m(x)$ e mantemos o resto. Para um polinômio $f(x)$, o resto é expresso como $r(x) = f(x) \bmod m(x)$.

O Advanced Encryption Standard (AES) utiliza a aritmética no corpo finito GF(2^8), com o polinômio irredutível $m(x) = x^8 + x^4 + x^3 + x + 1$. Considere os dois polinômios $f(x) = x^6 + x^4 + x^2 + x + 1$ e $g(x) = x^7 + x + 1$. Então

$$\begin{aligned} f(x) + g(x) &= x^6 + x^4 + x^2 + x + 1 + x^7 + x + 1 \\ &= x^7 + x^6 + x^4 + x^2 \end{aligned}$$

$$\begin{aligned} f(x) \times g(x) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

$$\begin{array}{r|l} & x^5 + x^3 \\ \hline x^8 + x^4 + x^3 + x + 1 & x^{13} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + \quad + 1 \\ & \underline{x^{13} \qquad + x^9 + x^8 \qquad + x^6 + x^5} \\ & \qquad x^{11} \qquad\qquad\qquad\qquad + x^4 + x^3 \\ & \qquad \underline{x^{11} \qquad\qquad + x^7 + x^6 \qquad + x^4 + x^3} \\ & \qquad\qquad\qquad\qquad x^7 + x^6 \qquad\qquad\qquad + 1 \end{array}$$

Portanto, $f(x)$ X $g(x) \bmod m(x) = x^7 + x^6 + 1$.

Assim, como na aritmética modular comum, temos a noção de um conjunto de resíduos na aritmética polinomial modular. O conjunto de resíduos módulo $m(x)$, um polinômio de grau n, consiste em p^n elementos. Cada um desses elementos é representado por um dos p^n polinômios de grau $m < n$.

> A classe de resíduos $[x + 1]$, módulo $m(x)$, consiste em todos os polinômios $a(x)$, de modo que $a(x) \equiv (x + 1) \pmod{m(x)}$. De modo equivalente, a classe de resíduos $[x + 1]$ consiste em todos os polinômios $a(x)$ que satisfazem a igualdade $a(x) \bmod m(x) = x + 1$.

Podemos mostrar que o conjunto de todos os polinômios módulo um polinômio $m(x)$ de grau n irredutível satisfaz os axiomas da Figura 4.1 e, assim, forma um corpo finito. Além disso, todos os corpos finitos de determinada ordem são isomórficos, ou seja, duas estruturas de corpo finito quaisquer de determinada ordem têm a mesma estrutura, mas a representação, ou rótulos dos elementos, pode ser diferente.

> Para construir o corpo finito GF(2^3), precisamos escolher um polinômio irredutível de grau 3. Existem apenas dois polinômios assim: $(x^3 + x^2 + 1)$ e $(x^3 + x + 1)$. Usando o segundo, a Tabela 4.6 mostra as tabelas de adição e de multiplicação para GF(2^3). Observe que esse conjunto de tabelas tem a estrutura idêntica às da Tabela 4.5. Assim, conseguimos encontrar um meio de definir um corpo de ordem 2^3.

Encontrando o inverso multiplicativo

Assim como o algoritmo euclidiano pode ser adaptado para encontrar o máximo divisor comum de dois polinômios, o algoritmo euclidiano estendido pode ser adaptado para encontrar o inverso multiplicativo de um polinômio. Especificamente, o algoritmo encontrará o inverso multiplicativo de $b(x)$ módulo $m(x)$ se o grau de $b(x)$ for menor que o grau de $m(x)$ e $\text{mdc}[m(x), b(x)] = 1$. Se $m(x)$ for um polinômio irredutível, então ele não possui outro fator além de si mesmo ou 1, de modo que $\text{mdc}[m(x), b(x)] = 1$. O algoritmo é o seguinte:

```
EUCLIDES ESTENDIDO[m(x), b(x)]
1.  [A1(x), A2(x), A3(x)] ← [1, 0, m(x)]; [B1(x), B2(x),
    B3(x)] ← [0, 1, b(x)]
2.  if B3(x) = 0     return  A3(x) = mdc[m(x), b(x)];  no inverse
3.  if B3(x) = 1     return  B3(x) = mdc[m(x), b(x)];  B2(x) = b(x)^-1
    mod m(x)
4.  Q(x) = quociente de A3(x)/B3(x)
5.  [T1(x), T2(x), T3(x)] ← [A1(x) - Q(x)B1(x), A2(x) - Q(x)B2(x),
    A3(x) - Q(x)B3(x)]
6.  [A1(x), A2(x), A3(x)] ← [B1(x), B2(x), B3(x)]
7.  [B1(x), B2(x), B3(x)] ← [T1(x), T2(x), T3(x)]
8.  goto 2
```

> A Tabela 4.7 mostra o cálculo do inverso multiplicativo de $(x^7 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1)$. O resultado é que $(x^7 + x + 1)^{-1} = (x^7)$. Ou seja, $(x^7 + x + 1)(x^7) \equiv 1 \pmod{(x^8 + x^4 + x^3 + x + 1)}$.

Considerações computacionais

Um polinômio $f(x)$ em GF(2^n)

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

pode ser representado exclusivamente por seus n coeficientes binários $(a_{n-1}a_{n-2} \ldots a_0)$. Assim, cada polinômio em GF(2^n) pode ser representado por um número de n bits.

> As Tabelas 4.5 e 4.6 mostram as tabelas de adição e multiplicação para GF(2^3) módulo $m(x) = (x3 + x + 1)$. A Tabela 4.5 usa a representação binária e a Tabela 4.6 usa a representação polinomial.

Tabela 4.6 Aritmética polinomial módulo $(x^3 + x + 1)$

		000	001	010	011	100	101	110	111
	$+$	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
001	1	1	0	$x + 1$	x	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$
010	x	x	$x + 1$	0	1	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$
011	$x + 1$	$x + 1$	x	1	0	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2
100	x^2	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	x	$x + 1$
101	$x^2 + 1$	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	x
110	$x^2 + x$	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	$x + 1$	0	1
111	$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2	$x + 1$	x	1	0

(a) Adição

		000	001	010	011	100	101	110	111
	$\times$	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
010	x	0	x	x^2	$x^2 + x$	$x + 1$	1	$x^2 + x + 1$	$x^2 + 1$
011	$x + 1$	0	$x + 1$	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	x^2	1	x
100	x^2	0	x^2	$x + 1$	$x^2 + x + 1$	$x^2 + x$	x	$x^2 + 1$	1
101	$x^2 + 1$	0	$x^2 + 1$	1	x^2	x	$x^2 + x + 1$	$x + 1$	$x^2 + x$
110	$x^2 + x$	0	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	$x + 1$	x	x^2
111	$x^2 + x + 1$	0	$x^2 + x + 1$	$x^2 + 1$	x	1	$x^2 + x$	x^2	$x + 1$

(b) Multiplicação

Tabela 4.7 Euclides estendido [(x8 1 x4 1 x3 1 x 1 1), (x7 1 x 1 1)]

Inicialização	$A1(x) = 1; A2(x) = 0; A3(x) = x^8 + x^4 + x^3 + x + 1$ $B1(x) = 0; B2(x) = 1; B3(x) = x^7 + x + 1$
Iteração 1	$Q(x) = x$ $A1(x) = 0; A2(x) = 1; A3(x) = x^7 + x + 1$ $B1(x) = 1; B2(x) = x; B3(x) = x^4 + x^3 + x^2 + 1$
Iteração 2	$Q(x) = x^3 + x^2 + 1$ $A1(x) = 1; A2(x) = x; A3(x) = x^4 + x^3 + x^2 + 1$ $B1(x) = x^3 + x^2 + 1; B2(x) = x^4 + x^3 + x + 1; B3(x) = x$
Iteração 3	$Q(x) = x^3 + x^2 + x$ $A1(x) = x^3 + x^2 + 1; A2(x) = x^4 + x^3 + x + 1; A3(x) = x$ $B1(x) = x^6 + x^2 + x + 1; B2(x) = x^7; B3(x) = 1$
Iteração 4	$B3(x) = \gcd[(x^7 + x + 1), (x^8 + x^4 + x^3 + x + 1)] = 1$ $B2(x) = (x^7 + x + 1)^{-1} \bmod (x^8 + x^4 + x^3 + x + 1) = x^7$

Adição Vimos que a adição de polinômios é realizada somando-se coeficientes correspondentes e, no caso de polinômios sobre Z_2, a adição é apenas a operação XOR. Assim, a adição de dois polinômios em GF(2^n) corresponde a uma operação XOR bit a bit.

Considere os dois polinômios em GF(2^8) de nosso exemplo anterior: $f(x) = x^6 + x^4 + x^2 + x + 1$ e $g(x) = x^7 + x + 1$.

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1)$	$= x^7 + x^6 + x^4 + x^2$	(notação polinomial)
$(01010111) \oplus (10000011)$	$= (11010100)$	(notação binária)
$\{57\} \oplus \{83\}$	$= \{D4\}$	(notação hexadecimal)[7]

Multiplicação Não existe uma operação XOR simples que realizará multiplicação em GF(2^n). Porém, existe uma técnica razoavelmente simples e fácil de ser implementada, a qual discutiremos com referência a GF(2^8) usando $m(x) = x^8 + x^4 + x^3 + x + 1$, que é o corpo finito usado no AES. Essa técnica é facilmente generalizada para GF(2^n).

A técnica baseia-se na observação de que

$$x^8 \bmod m(x) = [m(x) - x^8] = (x^4 + x^3 + x + 1) \tag{4.8}$$

Reflita um pouco e você se convencerá de que a Equação (4.8) é verdadeira; se não, faça a divisão. Geralmente, em GF(2^n) com um polinômio $p(x)$ de grau n, temos $X^n \bmod p(x) = [p(x) - x^n]$.

Agora, considere um polinômio em GF(2^n) que tem a forma $f(x) = b^7x^7 + b^6x^6 + b^5x^5 + b^4x^4 + b^3x^3 + b^2x^2 + b^1x + b^0$. Se multiplicarmos por x, teremos

$$x \times f(x) = (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x) \tag{4.9}$$

Se $b_7 = 0$, então o resultado é um polinômio de grau menor que 8, que já está na forma reduzida, e nenhum outro cálculo é necessário. Se $b_7 = 1$, então a redução módulo $m(x)$ é obtida usando-se a Equação (4.8):

$$x \times f(x) = (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1)$$

Segue-se que a multiplicação por x (ou seja, 00000010) pode ser implementada como um deslocamento à esquerda por 1 bit seguido por um XOR bit a bit condicional com (00011011), que representa $(x^4 + x^3 + x + 1)$. Para resumir,

$$x \times f(x) = \begin{cases} (b_6b_5b_4b_3b_2b_1b_00) & \text{if } b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_00) \oplus (00011011) & \text{if } b_7 = 1 \end{cases} \tag{4.10}$$

7. Uma revisão básica sobre sistemas numéricos (decimal, binário, hexadecimal) pode ser encontrada no *Computer Science Student Resource Site*, em WilliamStallings.com/StudentSupport.html. Aqui, cada dois grupos de 4 bits em um byte é indicado por um único caractere hexadecimal, com dois caracteres delimitados por chaves.

A multiplicação por uma potência mais alta de x pode ser obtida pela aplicação repetida da Equação (4.10). Acrescentando resultados intermediários, podemos chegar à multiplicação por qualquer constante em GF(2^8).

Em um exemplo anterior, mostramos que, para $f(x) = x^6 + x^4 + x^2 + x + 1$, $g(x) = x^7 + x + 1$ e $m(x) = x^8 + x^4 + x^3 + x + 1$, $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$. Refazendo isso em aritmética binária, precisamos calcular (01010111) × (10000011). Primeiramente, determinamos os resultados da multiplicação por potências de x:

(01010111) × (00000010) = (10101110)
(01010111) × (00000100) = (01011100) ⊕ (00011011) = (01000111)
(01010111) × (00001000) = (10001110)
(01010111) × (00010000) = (00011100) ⊕ (00011011) = (00000111)
(01010111) × (00100000) = (00001110)
(01010111) × (01000000) = (00011100)
(01010111) × (10000000) = (00111000)

Assim,

(01010111) × (10000011) = (01010111) × [(00000001) ⊕ (00000010) ⊕ (10000000)]
= (01010111) ⊕ (10101110) ⊕ (00111000) = (11000001)

que é equivalente a $x^7 + x^6 + 1$.

Usando um gerador

Às vezes, é mais conveniente usar uma técnica equivalente para definir um corpo finito na forma GF(2^n), usando o mesmo polinômio irredutível. Para começar, precisamos de duas definições: um gerador g de um corpo finito F de ordem q (contém q elementos) é um elemento cujas primeiras $q - 1$ potências geram todos os elementos de F diferentes de zero. Ou seja, os elementos de F consistem em $0, g^0, g^1, \ldots, g^{q-2}$. Considere o corpo F definido por um polinômio $f(x)$. Um elemento b contido em F é chamado de raiz do polinômio se $f(b) = 0$. Finalmente, pode ser mostrado que uma raiz g de um polinômio irredutível é um gerador do corpo finito definido nesse polinômio.

Vamos considerar o corpo finito GF(2^3) definido sobre o polinômio irredutível $x^3 + x + 1$, explicado anteriormente. Assim, o gerador g precisa satisfazer $f(g) = g^3 + g + 1 = 0$. Lembre-se, conforme já explicamos, de que não precisamos encontrar uma solução numérica para essa igualdade. Em vez disso, lidamos com a aritmética de polinômios, em que a aritmética sobre os coeficientes é realizada módulo 2. Portanto, a solução para a igualdade anterior é $g^3 = -g - 1 = g + 1$. Agora, mostramos que g realmente gera todos os polinômios de grau menor que 3. Temos o seguinte:

$g^4 = g(g^3) = g(g + 1) = g^2 + g$
$g^5 = g(g^4) = g(g^2 + g) = g^3 + g^2 = g^2 + g + 1$
$g^6 = g(g^5) = g(g^2 + g + 1) = g^3 + g^2 + g = g^2 + g + g + 1 = g^2 + 1$
$g^7 = g(g^6) = g(g^2 + 1) = g^3 + g = g + g + 1 = 1 = g^0$

Vemos que as potências de g geram todos os polinômios diferentes de zero em GF(2^3). Além disso, deve ficar claro que $g^k = g^{k \bmod 7}$ para qualquer inteiro k. A Tabela 4.8 mostra a representação de potência, além das representações polinomial e binária.

Essa representação de potência facilita a multiplicação. Para multiplicar na notação de potência, some os expoentes módulo 7. Por exemplo, $g^4 + g^6 = g^{(10 \bmod 7)} = g^3 = g + 1$. O mesmo resultado é obtido usando-se aritmética polinomial, da seguinte forma: dados $g^4 = g^2 + g$ e $g^6 = g^2 + 1$. Então, $(g^2 + g) \times (g^2 + 1) = g^4 + g^3 + g^2 + 1$. Em seguida, precisamos determinar $(g^4 + g^3 + g^2 + 1) \bmod (g^3 + g + 1)$ pela divisão:

$$
\begin{array}{r|l}
 & g + 1 \\
\hline
g^3 + g^2 + 1 & g^4 + g^3 + g^2 + g \\
 & \underline{g^4 + \quad g^2 + g} \\
 & \quad g^3 + \quad g + 1 \\
 & \quad \underline{g^3 + \quad g + 1} \\
 & \qquad\qquad g + 1
\end{array}
$$

Obtemos um resultado $g + 1$, que combina com o resultado obtido usando a representação de potência.

A Tabela 4.9 mostra as tabelas de adição e multiplicação para GF(2^3) usando a representação de potência. Observe que isso resulta em resultados idênticos à representação polinomial (Tabela 4.6) com algumas das linhas e colunas trocadas.

Tabela 4.8 Gerador para $GF(2^3)$ usando $x^3 + x + 1$

Representação de potência	Representação polinomial	Representação binária	Representação decimal (Hex)
0	0	000	0
$g^0 (= g^7)$	1	001	1
g^1	g	010	2
g^2	g^2	100	4
g^3	$g + 1$	011	3
g^4	$g^2 + g$	110	6
g^5	$g^2 + g + 1$	111	7
g^6	$g^2 + 1$	101	5

Em geral, para $GF(2^n)$ com polinômio irredutível $f(x)$, determine $g^n = f(g) - g^n$. Depois, calcule todas as potências de g de g^{n+1} até g^{2^n-2}. Os elementos do corpo correspondem às potências de g de g^0 até g^{2^n-2}, mais o valor 0. Para a multiplicação de dois elementos no corpo, use a igualdade $g^k = g^{k \bmod (2^n-1)}$ para qualquer inteiro k.

Resumo

Nesta seção, mostramos como construir um corpo finito de ordem 2^n. Especificamente, definimos $GF(2^n)$ com as seguintes propriedades:

1. $GF(2^n)$ consiste em 2^n elementos.

Tabela 4.9 $GF(2^3)$ Arithmetic Using Generator for the Polynomial $(x^3 + x + 1)$

		000	001	010	100	011	110	111	101
	$+$	0	1	g	g^2	g^3	g^4	g^5	g^6
000	0	0	1	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$
001	1	1	0	$g + 1$	$g^2 + 1$	g	$g^2 + g + 1$	$g^2 + g$	g^2
010	g	g	$g + 1$	0	$g^2 + g$	1	g^2	$g^2 + 1$	$g^2 + g + 1$
100	g^2	g^2	$g^2 + 1$	$g^2 + g$	0	$g^2 + g + 1$	g	$g + 1$	1
011	g^3	$g + 1$	g	1	$g^2 + g + 1$	0	$g^2 + 1$	g^2	$g^2 + g$
110	g^4	$g^2 + g$	$g^2 + g + 1$	g^2	g	$g^2 + 1$	0	1	$g + 1$
111	g^5	$g^2 + g + 1$	$g^2 + g$	$g^2 + 1$	$g + 1$	g^2	1	0	g
101	g^6	$g^2 + 1$	g^2	$g^2 + g + 1$	1	$g^2 + g$	$g + 1$	g	0

(a) Adição

		000	001	010	100	011	110	111	101
	$\times$	0	1	g	g^2	g^3	g^4	g^5	g^6
000	0	0	0	0	0	0	0	0	0
001	1	0	1	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$
010	g	0	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1
100	g^2	0	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	g
011	g^3	0	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	G	g^2
110	g^4	0	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	g	g^2	$g + 1$
111	g^5	0	$g^2 + g + 1$	$g^2 + 1$	1	g	g^2	$g + 1$	$g^2 + g$
101	g^6	0	$g^2 + 1$	1	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$

(b) Multiplicação

2. As operações binárias + e × são definidas sobre o conjunto. As operações de adição, subtração, multiplicação e divisão podem ser realizadas sem sair do conjunto. Cada elemento do conjunto diferente de 0 tem um inverso multiplicativo.

Mostramos que os elementos de GF(2^n) podem ser definidos como o conjunto de todos os polinômios de grau menor ou igual a $n - 1$ ou com coeficientes binários. Cada polinômio desses pode ser representado por um único valor de n bits. A aritmética é definida como uma aritmética de polinômio módulo algum polinômio irredutível de grau n. Também vimos que uma definição equivalente de um corpo finito GF(2^n) utiliza um gerador e essa aritmética é definida usando potências do gerador.

4.7 LEITURA E SITES WEB RECOMENDADOS

[HERS75], ainda sendo impresso, é a abordagem clássica da álgebra abstrata; ele é legível e rigoroso; [DESK92] é outro bom recurso. [KNUT98] oferece boa abrangência da aritmética polinomial.

Um dos melhores tratamentos sobre os tópicos deste capítulo é [BERL84], ainda sendo impresso. [GARR01] também possui uma extensa cobertura. Uma abordagem completa e rigorosa dos corpos finitos é [LIDL94]. [HORO71] é uma boa visão geral dos tópicos deste capítulo.

BERL84 Berlekamp, E. *Algebraic coding theory*. Laguna Hills: Aegean Park Press, 1984.

DESK92 Deskins, W. *Abstract algebra*. New York: Dover, 1992.

GARR01 Garrett, P. *Making, breaking codes: an introduction to cryptology*. Upper Saddle River=: Prentice Hall, 2001.

HERS75 Herstein, I. *Topics in Algebra*. New York: Wiley, 1975.

HORO71 Horowitz, E. "Modular arithmetic and finite field theory: a tutorial." *Proceedings of the Second ACM Symposium and Symbolic and Algebraic Manipulation*, março de 1971.

KNUT98 Knuth, D. *The art of computer programming, volume 2: seminumerical algorithms*. Reading: Addison-Wesley, 1998.

LIDL94 Lidl, R. e Niederreiter, H. *Introduction to finite fields and their applications*. Cambridge: Cambridge University Press, 1994.

Site Web recomendado:

- **PascGalois Project:** Contém um conjunto inteligente de exemplos e projetos para auxiliar os alunos, fornecendo um conhecimento visual dos principais conceitos sobre álgebra abstrata.

4.8 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

algoritmo euclidiano
anel
anel comutativo
anel finito
anel infinito
anel polinomial
aritmética modular
aritmética polinomial
aritmética polinomial modular
associativo
comutativo
conjunto de coeficientes
corpo
corpo finito
corpo infinito
divisor
domínio de integridade
elemento de identidade
elemento inverso
gerador
grupo
grupo abeliano
grupo cíclico
grupo finito
grupo infinito
máximo divisor comum
módulo
número primo
operador de módulo
ordem
polinômio
polinômio irredutível
polinômio mônico
polinômio primo
relativamente primo
resíduo

Perguntas de revisão

4.1 Defina resumidamente um grupo.
4.2 Defina resumidamente um anel.
4.3 Defina resumidamente um corpo.
4.4 O que significa dizer que b é um divisor de a?
4.5 Qual é a diferença entre aritmética modular e aritmética comum?
4.6 Liste três classes de aritmética polinomial.

Problemas

4.1 Para o grupo S_n de permutações de n símbolos distintos,
a. Qual é o número de elementos em S_n?
b. Mostre que S_n não é abeliano para $n > 2$.

4.2 O conjunto de classes de resíduo módulo 3 forma um grupo
a. com relação à adição?
b. com relação à multiplicação?

4.3 Considere o conjunto $S = \{a, b\}$ com adição e multiplicação definidos pelas tabelas:

+	a	b
a	a	b
b	b	a

×	a	b
a	a	a
b	a	b

S é um anel? Justifique sua resposta.

4.4 Reformule a Equação (4.1), removendo a restrição de que a é um inteiro não negativo. Ou seja, considere que a seja qualquer inteiro.

4.5 Desenhe uma figura semelhante à Figura 4.2 para $a < 0$.

4.6 Encontre inteiros x de modo que
a. $5x \equiv 4 \pmod 3$
b. $7x \equiv 6 \pmod 5$
c. $9x \equiv 8 \pmod 7$

4.7 Neste texto, supusemos que o módulo é um inteiro positivo. Mas a definição da expressão $a \bmod n$ também faz sentido perfeitamente se n for negativo. Determine o seguinte:
a. $5 \bmod 3$
b. $5 \bmod -3$
c. $-5 \bmod 3$
d. $-5 \bmod -3$

4.8 Um módulo de 0 não se encaixa na definição, mas é definido por convenção da seguinte forma: $a \bmod 0 = a$. Com essa definição em mente, o que significa a seguinte expressão: $a \equiv b \pmod 0$?

4.9 Na Seção 4.2, definimos o relacionamento de congruência da seguinte forma: dois inteiros a e b são considerados congruentes módulo n se $(a \bmod n) = (b \bmod n)$. Então, provamos que $a \equiv b \pmod n$ se $n|(a - b)$. Alguns textos sobre teoria dos números utilizam esse último relacionamento como definição de congruência: dois inteiros a e b são considerados congruentes módulo n, se $n|(a - b)$. Usando essa última definição como ponto de partida, prove que, se $(a \bmod n) = (b \bmod n)$, então n divide $(a - b)$.

4.10 Qual é o menor inteiro positivo que tem exatamente k divisores, para $1 \leq k \leq 6$?

4.11 Prove o seguinte:
a. $a \equiv b \pmod n$ implica $b \equiv a \pmod n$
b. $a \equiv b \pmod n$ e $b \equiv c \pmod n$ implica $a \equiv c \pmod n$

4.12 Prove o seguinte:
a. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
b. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

4.13 Encontre o inverso multiplicativo de cada elemento diferente de zero em Z_5.

4.14 Mostre que um inteiro N é congruente módulo 9 com a soma de seus dígitos decimais. Por exemplo, $475 \equiv 4 + 7 + 5 = 16 \equiv 1 + 6 \equiv 7 \pmod 9$. Essa é a base para o procedimento familiar dos 'noves fora' quando se verificam os cálculos na aritmética.

4.15 **a.** Determine mdc(24140, 16762).
b. Determine mdc(4655, 12075).

4.16 A finalidade deste problema é definir um limite superior sobre o número de iterações do algoritmo euclidiano.
a. Suponha que $m = qn + r$ com $q > 0$ e $0 \leq r < n$. Mostre que $m/2 > r$.
b. Considere que A_i seja o valor de A no algoritmo euclidiano depois da i-ésima iteração. Mostre que

$$A_{i+2} < \frac{A_i}{2}$$

c. Mostre que, se m, n e N são inteiros com $1 \leq m, n, \leq 2^N$, então o algoritmo euclidiano usa no máximo $2N$ passos para encontrar mdc(m, n).

4.17 O algoritmo euclidiano já é conhecido há mais de 2.000 anos e sempre foi um favorito entre os teóricos de números. Depois de tantos anos, agora existe um concorrente em potencial, inventado por J. Stein em 1961. Mostramos o algoritmo de Stein a seguir. Determine mdc(A, B) com $A, B \geq 1$.

PASSO 1
Defina $A_1 = A$, $B_1 = B$, $C_1 = 1$

PASSO n
(1) Se $A_n = B_n$ pare. mdc$(A, B) = A_n C_n$
(2) Se A_n e B_n forem pares, defina $A_{n+1} = A_n/2$, $B_{n+1} = B_n/2$, $C_{n+1} = 2C_n$
(3) Se A_n for par e B_n for ímpar, defina $A_{n+1} = A_n/2$, $B_{n+1} = B_n$, $C_{n+1} = C_n$

(4) Se A_n for ímpar e B_n for par, defina
$A_{n+1} = A_n, B_{n+1} = Bn/2, C_{n+1} = C_n$
(5) Se A_n e B_n forem ímpares, defina
$A_{n+1} = |A_n - B_n|, B_{n+1} = \min(B_n, A_n), C_{n+1} = C_n$
Continue na etapa $n + 1$.

a. Para experimentar os dois algoritmos, calcule mdc(2152, 764) usando o algoritmo euclidiano e o algoritmo de Stein.

b. Qual é a vantagem aparente do algoritmo de Stein em relação ao algoritmo euclidiano?

4.18 **a.** Mostre que, se o algoritmo de Stein não terminar antes da n-ésima etapa, então

$$C_{n+1} \times \text{mdc}(A_{n+1}, B_{n+1}) = C_n \times \text{mdc}(A_n, B_n)$$

b. Mostre que, se o algoritmo não terminar antes da etapa $(n - 1)$, então

$$A_{n+2}B_{n+2} \leq \frac{A_nB_n}{2}$$

c. Mostre que, se $1 \leq A, B \leq 2^N$, então o algoritmo de Stein usa no máximo $4N$ passos para encontrar mdc(m, n). Assim, o algoritmo de Stein funciona aproximadamente com o mesmo número de passos do algoritmo euclidiano.

d. Demonstre que o algoritmo de Stein retorna realmente mdc(A, B).

4.19 Usando o algoritmo euclidiano estendido, encontre o inverso multiplicativo de

a. 1234 mod 4321
b. 24140 mod 40902
c. 550 mod 1769

4.20 Desenvolva um conjunto de tabelas semelhantes à Tabela 4.3 para GF(5).

4.21 Demonstre que o conjunto de polinômios cujos coeficientes formam um corpo é um anel.

4.22 Demonstre se cada uma dessas afirmações é verdadeira ou falsa para polinômios sobre um corpo:

a. O produto dos polinômios mônicos é mônico.
b. O produto dos polinômios de graus m e n tem grau $m + n$.
c. A soma dos polinômios de graus m e n tem grau $\max[m, n]$.

4.23 Para a aritmética de polinômio com coeficientes em Z_{10}, realize os seguintes cálculos:

a. $(7x + 2) - (x^2 + 5)$
b. $(6x^2 + x + 3) \times (5x^2 + 2)$

4.24 Determine quais dos seguintes são redutíveis sobre GF(2):

a. $x^3 + 1$
b. $x^3 + x^2 + 1$
c. $x^4 + 1$ (tenha cuidado)

4.25 Determine o mdc dos seguintes pares de polinômios:

a. $x^3 + x + 1$ e $x^2 + x + 1$ sobre GF(2)
b. $x^3 - x + 1$ e $x^2 + 1$ sobre GF(3)
c. $x^5 + x^4 + x^3 - x^2 - x + 1$ e $x^3 + x^2 + x + 1$ sobre GF(3)
d. $x^5 + 88x^4 + 73x^3 + 83x^2 + 51x + 67$ e $x^3 + 97x^2 + 40x + 38$ sobre GF(101)

4.26 Desenvolva um conjunto de tabelas semelhantes à Tabela 4.6 para GF(4) com $m(x) = x^2 + x + 1$.

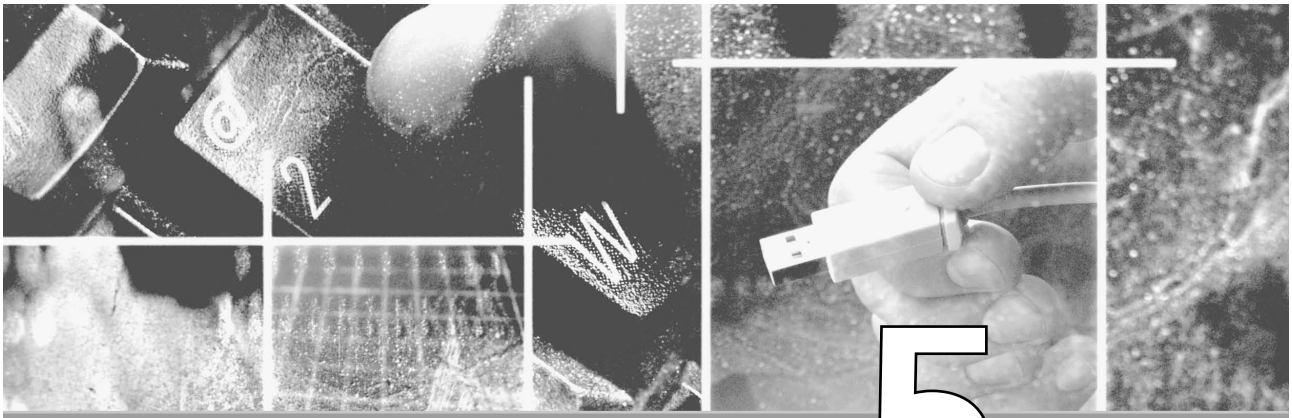
4.27 Determine o inverso multiplicativo de $x^3 + x + 1$ em GF(2^4), com $m(x) = x^4 + x + 1$.

4.28 Desenvolva uma tabela semelhante à Tabela 4.8 para GF(2^4) com $m(x) = x^4 + x + 1$.

Problemas de programação

4.29 Escreva uma calculadora simples de quatro funções em GF(2^4). Você pode usar pesquisas em tabela para os inversos multiplicativos.

4.30 Escreva uma calculadora de quatro funções simples em GF(2^8).



CAPÍTULO 5

Advanced Encryption Standard

5.1 **Critérios de avaliação para AES**

5.2 **A cifra AES**

5.3 **Leitura e sites Web recomendados**

5.4 **Principais termos, questões para revisão e problemas**

Apêndice 5A **Polinômios com coeficientes em GF(2^8)**

Apêndice 5B **AES simplificado**

"Parece muito simples."
"É muito simples. Mas, se você não souber qual é a chave, é praticamente indecifrável."

Talking to strange men, Ruth Rendell

PONTOS PRINCIPAIS

- O AES é uma cifra de bloco cujo objetivo é substituir o DES para aplicações comerciais. O AES usa um tamanho de bloco de 128 bits e um tamanho de chave de 128, 192 ou 256 bits.
- O AES não usa uma estrutura de Feistel. Em vez disso, cada rodada completa consiste em quatro funções distintas: substituição de bytes, permutação, operações aritméticas sobre um corpo finito e operação XOR com uma chave.

O padrão de criptografia avançado (AES – Advanced Encryption Standard) foi publicado pelo NIST (National Institute of Standards and Technology) em 2001. O AES é uma cifra em bloco simétrica cujo objetivo é substituir o DES como padrão aprovado para uma grande variedade de aplicações. Neste capítulo, examinamos primeiramente os critérios de avaliação utilizados pelo NIST para selecionar um candidato para o AES e depois examinamos a própria cifra.

Comparado a cifras de chave pública, como o RSA, a estrutura do AES, e da maioria das cifras simétricas, é muito complexa e não pode ser explicada tão facilmente quanto o RSA e os algoritmos semelhantes. Por conseguinte, o leitor pode começar com uma versão simplificada do AES, que é descrita no Apêndice 5B. Essa versão permite que ele realize a criptografia e decriptografia à mão e obtenha uma boa compreensão do funcionamento dos detalhes do algoritmo. A experiência em sala de aula indica que um estudo dessa versão simplificada melhora a compreensão do AES.[1]

5.1 CRITÉRIOS DE AVALIAÇÃO PARA AES

As origens do AES

Mencionamos no Capítulo 3 que, em 1999, o NIST lançou uma nova versão do seu padrão DES (FIPS PUB 46-3), que indicava que o DES só deveria ser utilizado em sistemas legados e que o DES triplo (3DES) fosse utilizado em seu lugar. Descreveremos o 3DES no Capítulo 6. O 3DES possui dois atrativos que garantem seu uso generalizado pelos próximos anos. Primeiro, com seu tamanho de chave de 168 bits, ele contorna a vulnerabilidade do ataque por força bruta ao DES. Segundo, o algoritmo de criptografia básico no 3DES é igual ao do DES. Esse algoritmo foi submetido a mais análises detalhadas que qualquer outro algoritmo de criptografia por um longo período, e nenhum ataque criptoanalítico eficaz contra o algoritmo, além da força bruta, teve sucesso. Por conseguinte, existe um alto nível de confiança de que o 3DES é muito resistente à criptoanálise. Se a segurança fosse a única consideração, então o 3DES seria uma escolha apropriada para um algoritmo de criptografia padronizado por mais algumas décadas.

A principal desvantagem do 3DES é que o algoritmo é relativamente lento em software. O DES original foi projetado para implementação em hardware em meados da década de 1970, e não produz um código de software eficiente. O 3DES, que tem três vezes mais rodadas que o DES, é correspondentemente mais lento. Uma desvantagem secundária é que tanto o DES quanto o 3DES utilizam um tamanho de bloco de 64 bits. Por motivos de eficiência e de segurança, um tamanho de bloco maior é desejável.

Por causa dessas desvantagens, o 3DES não é um candidato razoável para uso em longo prazo. Como alternativa, em 1997 o NIST pediu propostas para um novo Advanced Encryption Standard (AES), que deveria ter um grau de segurança igual ou superior ao 3DES e uma eficiência bem melhorada. Além desses requisitos gerais, o NIST especificou que o AES deveria ser uma cifra de bloco simétrica com um tamanho de bloco de 128 bits e suporte para tamanhos de chave de 128, 192 e 256 bits.

Em uma primeira rodada de avaliação, 15 algoritmos propostos foram aceitos. Uma segunda rodada estreitou a competição para 5 algoritmos. O NIST completou o processo de avaliação e publicou um padrão final (FIPS PUB 197) em novembro de 2001. O NIST selecionou o Rijndael como o algoritmo AES proposto. Os dois pesquisadores que desenvolveram e enviaram o Rijndael para o AES são criptógrafos belgas: o dr. Joan Daemen e dr. Vincent Rijmen.

Definitivamente, o AES está destinado a substituir o 3DES, mas esse processo levará alguns anos. O NIST antecipa que o 3DES permanecerá como um algoritmo aprovado (para uso do governo dos EUA) por prazo indeterminado.

Avaliação do AES

Vale a pena examinarmos os critérios utilizados pelo NIST para avaliar os candidatos em potencial. Esses critérios abrangem uma gama de preocupações para a aplicação prática de modernas cifras de bloco simétricas. Na verdade, dois conjuntos de critérios foram desenvolvidos. Quando o NIST emitiu sua solicitação original para propostas de algoritmos candidatos em 1997 [NIST97], a solicitação determinou que os algoritmos candidatos seriam comparados com base nos fatores mostrados na Tabela 5.1 (classificada em ordem decrescente de importância relativa). As três categorias de critérios são as seguintes:

- **Segurança:** Isso se refere ao esforço exigido para criptoanalisar um algoritmo. A ênfase da avaliação está na praticidade do ataque. Como o tamanho de chave mínimo para o AES é de 128 bits, os ataques por força bruta com tecnologia atual e projetada foram considerados impraticáveis. Portanto, a ênfase, com relação a esse ponto, é a criptoanálise que não seja um ataque pela força bruta.

1. Apesar disso, você pode ignorar o Apêndice 5B, pelo menos em uma primeira leitura. Se você se perder ou se confundir com os detalhes do AES, poderá então voltar e começar com o AES simplificado.

Tabela 5.1 Critérios de avaliação do NIST para o AES (12 de setembro de 1997)

SEGURANÇA

- **Segurança real:** em comparação com outros algoritmos enviados (na mesma chave e tamanho de bloco).
- **Aleatoriedade:** até que ponto a saída do algoritmo é indistinguível de uma permutação aleatória no bloco de entrada.
- **Solidez:** da base matemática para a segurança do algoritmo.
- **Outros fatores de segurança:** levantados pelo público durante o processo de avaliação, incluindo quaisquer ataques que demonstrem que a segurança real do algoritmo é menor que a resistência afirmada por quem o enviou.

CUSTO

- **Condições de licenciamento:** O NIST pretende, quando o AES for lançado, disponibilizar o algoritmo especificado no AES ao mundo inteiro, sem exclusividade e sem royalties.
- **Eficiência computacional:** A avaliação da eficiência computacional será aplicável para implementações tanto em hardware quanto em software. A análise da primeira rodada, feita pelo NIST, focará principalmente as implementações em software e uma combinação de tamanho de chave e bloco específica (128-128); será dada mais atenção às implementações em hardware e outras combinações de tamanho de chave e bloco suportadas durante a análise da segunda rodada. A eficiência computacional basicamente refere-se à velocidade do algoritmo. Os comentários públicos sobre a eficiência de cada algoritmo (especialmente para diversas plataformas e aplicações) também será levada em consideração pelo NIST.
- **Requisitos de memória:** A memória exigida para implementar um algoritmo candidato — para implementações tanto em hardware quanto em software do algoritmo – também será considerada durante o processo de avaliação. A análise da primeira rodada pelo NIST focará principalmente as implementações em software; será dada mais atenção às implementações em hardware durante a segunda rodada. Os requisitos de memória incluirão fatores como contagens de portas para implementações em hardware e tamanho de código e requisitos de RAM para implementações em software.

ALGORITMO E CARACTERÍSTICAS DE IMPLEMENTAÇÃO

- **Flexibilidade:** Os algoritmos candidatos com maior flexibilidade atenderão às necessidades de mais usuários do que aqueles menos flexíveis e, portanto, *inter alia*, são preferíveis. Porém, alguns excessos de funcionalidade possuem pouca aplicação prática (por exemplo, tamanhos de chave extremamente curtos); para esses casos não será dada preferência. Alguns exemplos de flexibilidade podem incluir (mas não são limitados) ao seguinte:
  - **a.** O algoritmo pode aceitar tamanhos adicionais de chave e bloco (por exemplo, tamanhos de bloco de 64 bits, tamanhos de chave diferentes daqueles especificados na seção de requisitos mínimos de aceitação — por exemplo, chaves entre 128 e 256 que sejam múltiplos de 32 bits etc.).
  - **b.** O algoritmo pode ser implementado de forma segura e eficiente em uma grande variedade de plataformas e aplicações (por exemplo, processadores de 8 bits, redes ATM, comunicações por voz e satélite, HDTV, B-ISDN etc.).
  - **c.** O algoritmo pode ser implementado como uma cifra de fluxo, gerador de código de autenticação de mensagem (MAC), gerador de número pseudo-aleatório, algoritmo de hashing etc.
- **Adequação de hardware e software:** Um algoritmo candidato não deverá ser restritivo no sentido de só poder ser implementado em hardware. Se alguém também puder implementá-lo de maneira eficiente em firmware, então essa será uma vantagem na área de flexibilidade.
- **Simplicidade:** Um algoritmo candidato será julgado de acordo com a simplicidade relativa do projeto.

- **Custo:** O NIST pretende que o AES seja de uso prático em uma grande variedade de aplicações. Por conseguinte, o AES precisa ter alta eficiência computacional, de modo a ser utilizável em aplicações de alta velocidade, como links de banda larga.
- **Algoritmo e características de implementação:** Essa categoria inclui uma variedade de considerações, entre as quais a flexibilidade, a conformidade com uma série de implementações em hardware e software e a simplicidade que tornarão uma análise da segurança mais clara.

Usando esses critérios, o grupo inicial de 21 algoritmos candidatos foi reduzido primeiro a 15 candidatos e, depois, a 5 candidatos. Durante a avaliação final, os critérios de avaliação, conforme descritos em [NECH00], haviam evoluído. Os critérios a seguir foram usados na avaliação final:

- **Segurança geral:** Para avaliar a segurança geral, o NIST contou com a análise de segurança pública, realizada pela comunidade criptográfica. Durante o decorrer do processo de avaliação de três anos, diversos criptógrafos publicaram suas análises dos pontos fortes e fracos dos diversos candidatos. Houve ênfase particular na análise dos candidatos com relação a ataques conhecidos, como a criptoanálise diferencial e linear. Porém, comparado com a análise do DES, a quantidade de tempo e o número de criptógrafos dedicados a analisar o Rijndael eram bastante limitados. Agora que uma única cifra AES foi escolhida, podemos esperar uma análise de segurança mais extensiva por parte da comunidade criptográfica.

- **Implementação em software:** As principais preocupações nessa categoria são velocidade de execução, desempenho por diversas plataformas e variação de velocidade com o tamanho da chave.
- **Ambientes de espaço restrito:** Em algumas aplicações, como cartões inteligentes, quantidades relativamente pequenas de memória de acesso aleatório (RAM) e/ou memória somente de leitura (ROM) estão disponíveis para finalidades como armazenamento de código (geralmente em ROM), representação de objetos de dados, como caixas-S (que podem ser armazenadas em ROM ou RAM, dependendo da utilização do pré-cálculo ou da representação booleana), e armazenamento de subchave (na RAM).
- **Implementações em hardware:** Assim como o software, as implementações em hardware podem ser otimizadas por velocidade ou por tamanho. Porém, no caso do hardware, o tamanho é traduzido muito mais diretamente em termos de custo do que normalmente acontece nas implementações em software. Duplicar o tamanho de um programa de codificação pode fazer pouca diferença em um computador de uso geral com muita memória, mas duplicar a área utilizada em um dispositivo de hardware geralmente aumenta o custo do dispositivo em mais que o dobro.
- **Ataques nas implementações:** O critério de segurança geral, explicado no primeiro item acima, trata dos ataques criptoanalíticos que exploram as propriedades matemáticas dos algoritmos. Existe outra classe de ataques que usa medições físicas realizadas durante a execução do algoritmo para colher informações sobre a quantidade de chaves, por exemplo. Esses ataques exploram uma combinação de características intrínsecas do algoritmo e recursos dependentes de implementação. Alguns exemplos desses ataques são aqueles de temporização e análise de potência. Os ataques de temporização são descritos no Capítulo 3. A idéia básica por trás da análise de potência [KOCH98, BIHA00] é a observação de que a potência consumida por um cartão inteligente, em qualquer período específico de tempo durante a operação criptográfica, está relacionada à instrução sendo executada e aos dados sendo processados. Por exemplo, a multiplicação consome mais potência que a adição, e a escrita de 1s consome mais potência do que a escrita de 0s.
- **Criptografia *versus* decriptografia:** Esse critério lida com várias questões relacionadas a considerações tanto de criptografia quanto de decriptografia. Se os algoritmos de criptografia e decriptografia forem diferentes, então um espaço extra será necessário para a decriptografia. Além disso, quer os dois algoritmos sejam iguais quer não, poderá haver diferenças de temporização entre criptografia e decriptografia.
- **Agilidade da chave:** A agilidade da chave refere-se à capacidade de alterar chaves rapidamente e com o mínimo de recursos. Isso inclui tanto o cálculo da subchave quanto a capacidade de troca entre diferentes associações de segurança em andamento quando as subchaves já estiverem disponíveis.
- **Outra versatilidade e flexibilidade:** [NECH00] indica duas áreas que se encaixam nessa categoria. A flexibilidade de parâmetro inclui facilidade para aceitar outros tamanhos de chave e bloco e facilidade para aumentar o número de rodadas a fim de lidar com os ataques recém-descobertos. A flexibilidade de implementação refere-se à possibilidade de otimizar os elementos da cifra para ambientes específicos.
- **Potencial para paralelismo em nível de instrução:** Esse critério refere-se à capacidade de explorar os recursos de ILP em processadores atuais e futuros.

A Tabela 5.2 mostra a avaliação que o NIST forneceu para o Rijndael com base nesses critérios.

Tabela 5.2 Avaliação final do NIST para o Rijndael (2 de outubro de 2000)

Segurança geral

O Rijndael não possui ataques de segurança conhecidos. Ele usa caixas-S como componentes não lineares. O Rijndael parece ter uma margem de segurança adequada, mas recebeu críticas sugerindo que sua estrutura matemática pode conduzir a ataques. Por outro lado, a estrutura simples pode ter facilitado sua análise de segurança durante o período de tempo do processo de desenvolvimento do AES.

Implementações em software

O Rijndael realiza criptografia e decriptografia muito bem em diversas plataformas, incluindo as de 8 bits e 64 bits, e DSPs. Porém, existe uma queda no desempenho com tamanhos de chave maiores, em razão do número maior de rodadas executadas. O alto paralelismo inerente ao Rijndael facilita o uso eficiente dos recursos do processador, resultando um desempenho de software muito bom, mesmo quando implementado em um modo sem capacidade de intercalação. O tempo de configuração de chaves do Rijndael é pequeno.

(continua)

Tabela 5.2 Avaliação final do NIST para o Rijndael (2 de outubro de 2000) (*Continuação*)

Ambientes de espaço restrito

Em geral, o Rijndael é muito bem adequado a ambientes de espaço restrito, nos quais a criptografia ou decriptografia (mas não ambas) são implementadas. Ele possui requisitos muito baixos de RAM e ROM. Uma desvantagem é que os requisitos de ROM aumentarão se a criptografia e a decriptografia forem implementadas simultaneamente, embora pareça continuar adequado a esses ambientes. O escalonamento de chaves para decriptografia é separado da criptografia.

Implementações em hardware

O Rijndael possui o throughput mais alto que qualquer um dos finalistas para os modos com feedback e o segundo mais alto para os modos sem feedback. Para os tamanhos de chave de 192 e 256 bits, o throughput cai nas implementações padrão e desdobrada, devido ao número adicional de rodadas. Para implementações totalmente em pipeline, o requisito de área aumenta, mas o throughput não é afetado.

Ataques nas implementações

As operações usadas pelo Rijndael estão entre as mais fáceis de serem defendidas contra ataques de consumo de potência e temporização. O uso de técnicas de máscara para defender o Rijndael contra esses ataques não causa degradação significativa no desempenho em comparação aos outros finalistas, e seu requisito de RAM continua sendo razoável. O Rijndael parece obter uma vantagem de velocidade importante em relação aos seus concorrentes quando essas proteções são consideradas.

Criptografia *versus* decriptografia

As funções de criptografia e decriptografia no Rijndael são diferentes. Um estudo do FPGA relata que a implementação conjunta de criptografia e decriptografia ocupa cerca de sessenta por cento mais espaço do que a implementação isolada da criptografia. A velocidade do Rijndael não varia significativamente entre a criptografia e a decriptografia, embora o desempenho da configuração das chaves na decriptografia seja mais lento do que na criptografia.

Agilidade da chave

O Rijndael admite cálculo de subchave sob demanda para a criptografia. Ele requer uma única execução do escalonamento de chaves para gerar todas as subchaves antes da primeira decriptografia com uma chave específica. Isso exige um pouco mais de recursos para a agilidade da chave do Rijndael.

Outras versatilidades e flexibilidades

O Rijndael admite qualquer combinação de tamanhos de bloco e de chave de 128 bits, 192 bits e 256 bits. A princípio, a estrutura do Rijndael pode acomodar quaisquer tamanhos de bloco e combinações de chave que sejam múltiplos de 32, além de mudanças no número de rodadas especificadas.

Potencial para paralelismo em nível de instrução

O Rijndael possui um excelente potencial para paralelismo em uma criptografia de bloco único.

5.2 A CIFRA AES[2]

A proposta do Rijndael para o AES definiu uma cifra em que o tamanho do bloco e o tamanho da chave podem ser especificados independentemente como 128, 192 ou 256 bits. A especificação do AES usa as mesmas três alternativas de tamanho de chave, mas limita o tamanho do bloco a 128 bits. Diversos parâmetros do AES dependem do tamanho da chave (Tabela 5.3). Na descrição desta seção consideramos um tamanho de chave de 128 bits, que provavelmente será o mais implementado.

2. Grande parte do material nesta seção apareceu originalmente em [STAL02].

Tabela 5.3 Parâmetros do AES

Tamanho da chave (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Tamanho do bloco de texto claro (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Número de rodadas	10	12	14
Tamanho da chave da rodada (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Tamanho da chave expandida (words/bytes)	44/176	52/208	60/240

O Rijndael foi projetado para ter as seguintes características:

- Resistência contra todos os ataques conhecidos
- Velocidade e compactação de código em uma grande variedade de plataformas
- Simplicidade de projeto

A Figura 5.1 mostra a estrutura geral do AES. A entrada dos algoritmos de criptografia e decriptografia é um único bloco de 128 bits. No FIPS PUB 197, esse bloco é representado como uma matriz quadrada de bytes e é copiado para o vetor **State**, que é modificado a cada estágio de criptografia ou decriptografia. Após o estágio final, **State** é copiado para uma matriz de saída. Essas operações são representadas na Figura 5.2a. De modo semelhante, a chave de 128 bits é representada como uma matriz quadrada de bytes. Essa chave é então expandida para um vetor de words para o escalonamento de chaves; cada word possui quatro bytes e o escalonamento de chaves total tem 44

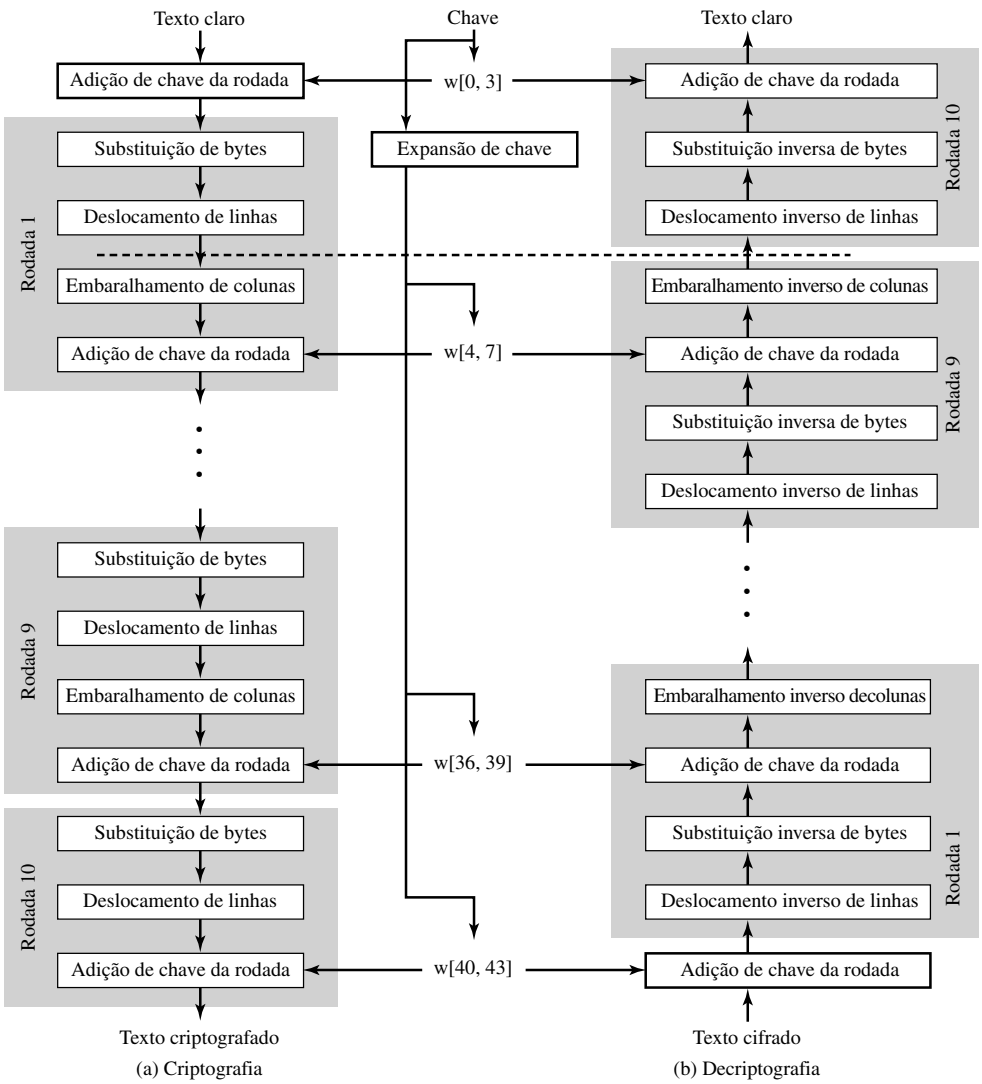


Figura 5.1 Criptografia e decriptografia AES.

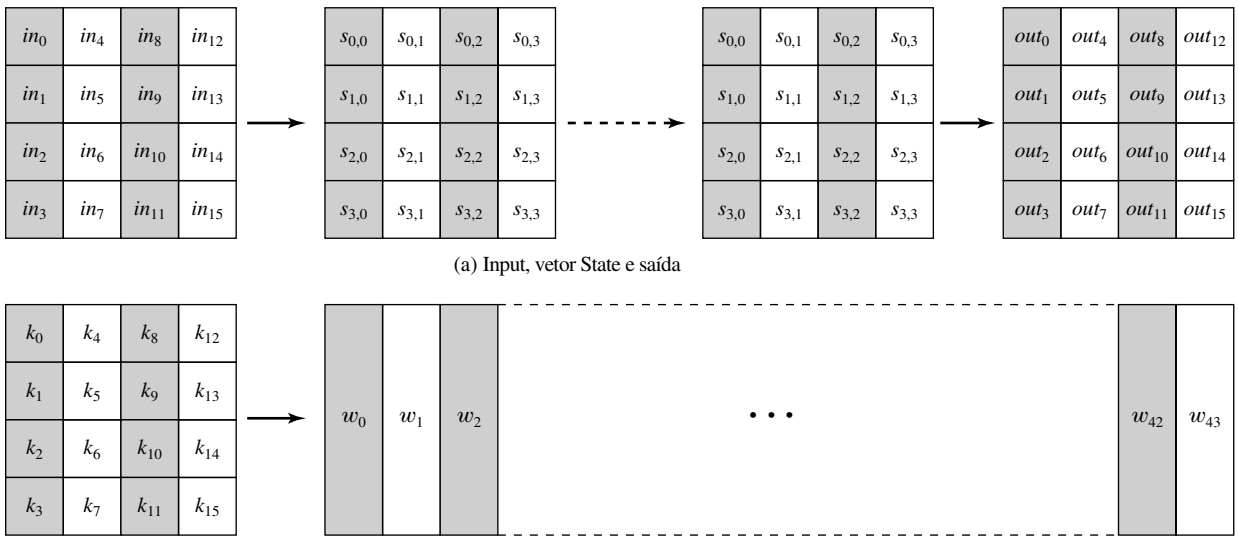


Figura 5.2 Estruturas de dados do AES.

words para a chave de 128 bits (Figura 5.2b). Observe que a ordenação dos bytes dentro de uma matriz ocorre por coluna. Assim, por exemplo, os quatro primeiros bytes de uma entrada de texto claro de 128 bits para a criptografia ocupam a primeira coluna da matriz **in**, os próximos quatro bytes ocupam a segunda coluna, e assim por diante. De modo semelhante, os quatro primeiros bytes da chave expandida, que formam uma word, ocupam a primeira coluna da matriz **w**.

Antes de nos aprofundarmos nos detalhes, podemos fazer vários comentários sobre a estrutura geral do AES:

1. Um recurso notável dessa estrutura é que ela não é uma estrutura de Feistel. Lembre-se de que, na estrutura de Feistel clássica, metade do bloco de dados é usada para modificar a outra metade do bloco de dados e, depois, as metades são trocadas. Dois dos finalistas do AES, incluindo o Rijndael, não usam uma estrutura Feistel, mas processam o bloco de dados inteiro em paralelo durante cada rodada, usando substituições e difusão linear.
2. A chave fornecida como entrada é expandida para um vetor de quarenta e quatro words de 32 bits, **w**[i]. Quatro words distintas (128 bits) servem como uma chave de para cada rodada; elas estão indicadas na Figura 5.1.
3. Quatro estágios diferentes são usados, um de permutação e três de substituição:
   - **SubBytes:** Utiliza uma caixa-S para realizar uma substituição byte a byte do bloco
   - **ShiftRows:** Uma permutação simples
   - **MixColumns:** Uma combinação linear que utiliza aritmética sobre GF(2^8)
   - **AddRoundKey:** Um XOR bit a bit simples do bloco atual com uma parte da chave expandida
4. A estrutura é muito simples. Para a criptografia e a decriptografia, a cifra começa com um estágio AddRoundKey, seguido por nove rodadas, sendo que cada uma inclui todos os quatro estágios, seguidos por uma décima rodada de três estágios. A Figura 5.3 representa a estrutura de uma rodada completa de criptografia.
5. Somente o estágio AddRoundKey faz uso da chave. Por esse motivo, a cifra começa e termina com um estágio AddRoundKey. Qualquer outro estágio, aplicado no início ou no fim, é reversível sem conhecimento da chave e, portanto, não agrega segurança.
6. O estágio AddRoundKey, com efeito, é uma forma de cifra de Vernam e, por si só, não seria suficiente. Os outros três estágios fornecem confusão, difusão e não-linearidade, mas por si sós não ofereceriam segurança, pois não usam a chave. Podemos ver a cifra como operações alternadas de criptografia XOR (AddRoundKey) de um bloco, seguido de embaralhamento do bloco (os outros três estágios), seguido de criptografia XOR, e assim por diante. Esse esquema é tanto eficiente quanto altamente seguro.
7. Cada estágio é facilmente reversível. Para os estágios SubByte, ShiftRows e MixColumns, a função inversa é usada no algoritmo de decriptografia. Para o estágio AddRoundKey, o inverso é obtido pelo XOR da mesma chave da rodada com o bloco, usando o resultado $A \oplus A \oplus B = B$.
8. Assim como na maioria das cifras de bloco, o algoritmo de decriptografia utiliza a chave expandida em ordem reversa. Porém, o algoritmo de decriptografia não é idêntico ao algoritmo de criptografia. Isso é uma conseqüência da estrutura específica do AES.
9. Uma vez estabelecido que todos os quatro estágios são reversíveis, é fácil verificar que a decriptografia recupera o texto claro. A Figura 5.1 mostra a criptografia e a decriptografia indo em direções verticais opostas. Em cada ponto horizontal (por exemplo, a linha tracejada na figura), **State** é o mesmo para a criptografia e a decriptografia.

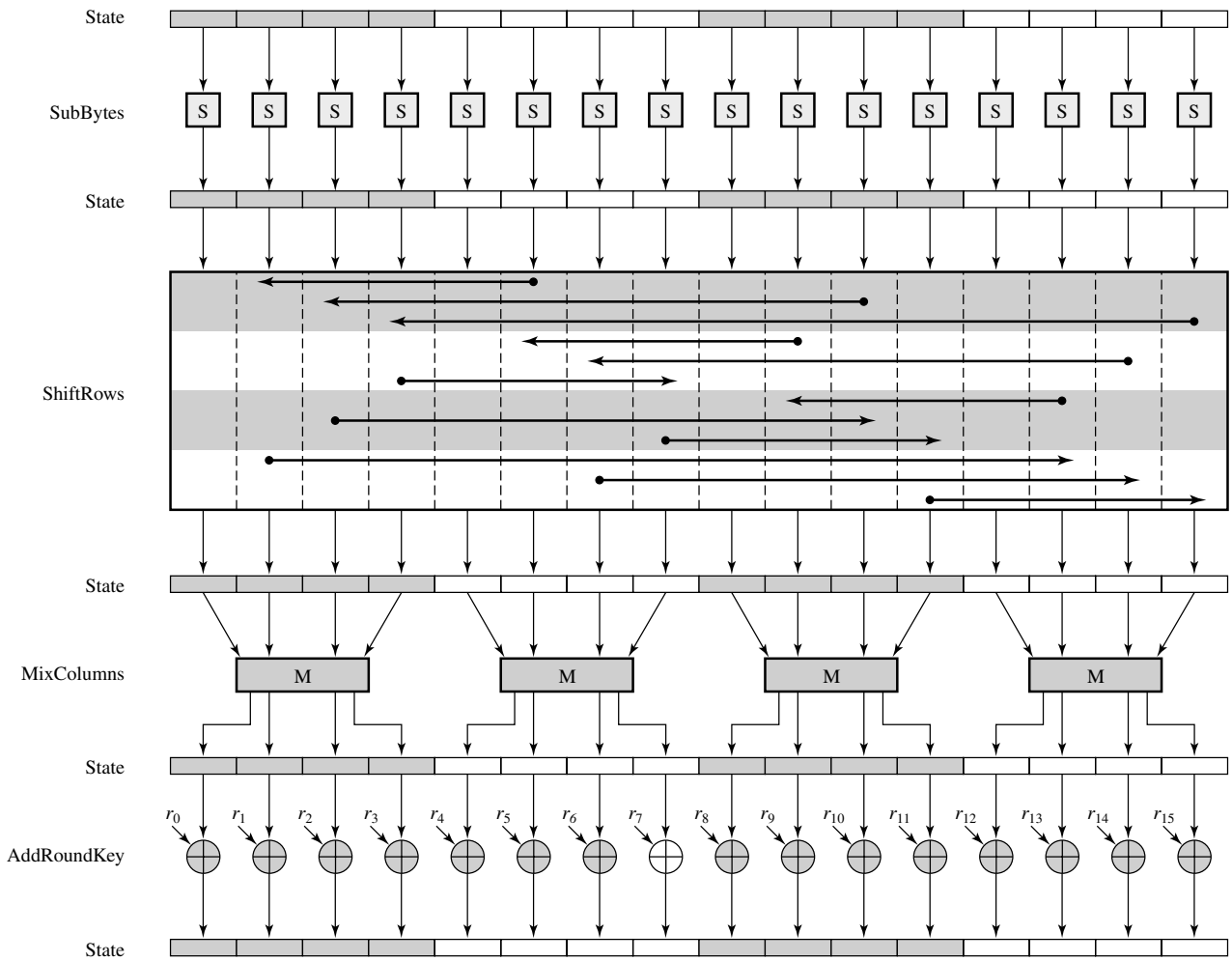


Figura 5.3 Uma rodada de criptografia

10. A rodada final da criptografia e da decriptografia consiste em apenas três estágios. Novamente, isso é uma conseqüência da estrutura específica do AES e é necessária para tornar a cifra reversível.

Agora, vamos passar para uma explicação de cada uma das quatro tranformações usadas nos quatro estágios do AES. Para cada tranformação, descrevemos o algoritmo direto (criptografia), o algoritmo inverso (decriptografia) e o raciocínio para a transformação. Isso é seguido por uma explicação sobre a expansão da chave.

Conforme mencionamos no Capítulo 4, o AES utiliza a aritmética no corpo finito GF(2^8), com o polinômio[3] irredutível $m(x) = x^8 + x^4 + x^3 + x + 1$. Os desenvolvedores do Rijndael indicam, como motivação para selecionar este dentre os 30 polinômios irredutíveis possíveis de grau 8, que ele é o primeiro na lista dada em [LIDL94].

Transformação SubBytes

Transformações direta e inversa A transformação substituição direta de bytes, chamada SubBytes, é uma tabela simples de busca (Figura 5.4a). O AES define uma matriz 16 × 16 de valores de byte, chamada de caixa-S (Tabela 5.4a), que contém uma permutação de todos os 256 valores possíveis de 8 bits. Cada byte de **State** é mapeado individualmente para um novo byte da seguinte maneira: os 4 bits mais à esquerda do byte são usados como um valor de linha e os 4 bits mais à direita são usados como um valor de coluna. Esses valores de linha e coluna servem como índices na caixa-S para selecionar um valor de saída de 8 bits. Por exemplo, o valor hexadecimal[4] {95} referencia a linha 9, coluna 5 da caixa-S, que contém o valor {2A}. De acordo com isso, o valor {95} é mapeado para o valor {2A}.

3. No restante desta explicação, as referências a GF(2^8) remetem ao corpo finito definido com esse polinômio.
4. Na FIPS PUB 197, um número hexadecimal é indicado pela delimitação por chaves. Usamos essa mesma convenção neste capítulo.

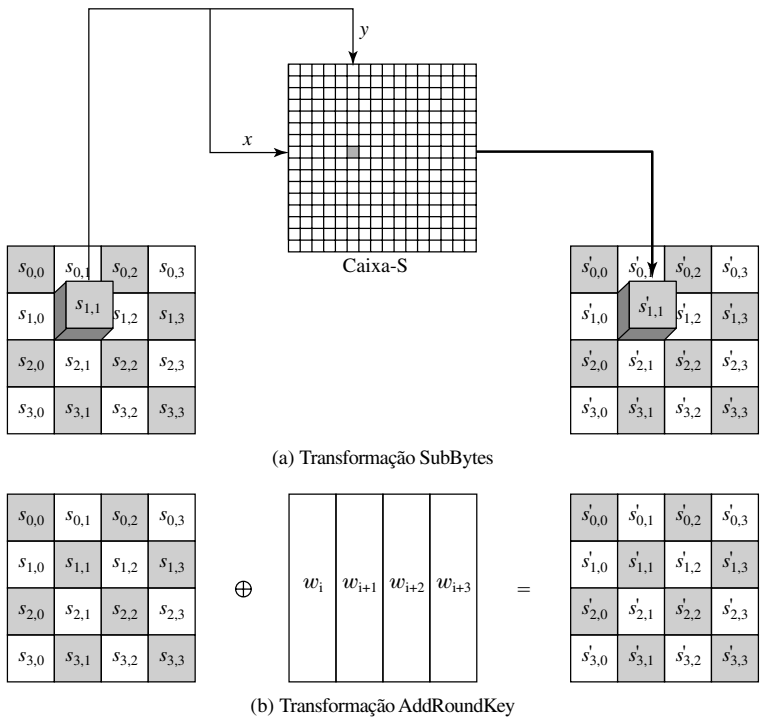


Figura 5.4 Operações em nível de byte do AES

Aqui está um exemplo da transformação SubBytes:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

A caixa-S é construída da seguinte forma:

1. Inicialize a caixa-S com os valores de byte em ordem crescente linha por linha. A primeira linha contém {00}, {01}, {02}, ..., {0F}; a segunda linha contém {10}, {11} etc.; e assim por diante. Assim, o valor do byte na linha x, coluna y é $\{xy\}$.
2. Mapeie cada byte da caixa-S para seu inverso multiplicativo no corpo finito GF(2^8); o valor {00} é mapeado para si mesmo.
3. Considere que cada byte na caixa-S é composto de 8 bits rotulados como $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$. Aplique a seguinte transformação a cada bit de cada byte na caixa-S:

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad \textbf{(5.1)}$$

onde c_i é o i-ésimo bit do byte c com o valor {63}; ou seja, $(c_7c_6c_5c_4c_3c_2c_1c_0) = (01100011)$. O apóstrofo (′) indica que a variável deve ser atualizada pelo valor à direita. O padrão AES representa essa transformação em forma de matriz da seguinte maneira:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \textbf{(5.2)}$$

Tabela 5.4 AES caixas-S

(a) Caixas-S

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
x	**7**	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) Caixa-S inversa

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
x	**7**	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

A Equação (5.2) precisa ser interpretada cuidadosamente. Na multiplicação normal de matrizes[5], cada elemento na matriz produto é a soma dos produtos dos elementos de uma linha e uma coluna. Nesse caso, cada elemento na matriz produto é o XOR bit a bit de produtos de elementos de uma linha e uma coluna. Além disso, a adição final mostrada na Equação (5.2) é um XOR bit a bit.

5. Para uma breve recapitulação das regras de multiplicação de matriz e vetor, consulte o documento Math Refresher e o site Computer Science Student Resource em *williamstallings.com/StudentSupport.html*.

Como um exemplo, considere o valor de entrada {95}. O inverso multiplicativo em GF(28) é $\{95\}^{-1} = \{8A\}$, que é 10001010 em binário. Usando a Equação (5.2),

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

O resultado é {2A}, que deverá aparecer na linha {09} coluna {05} da caixa-S. Isso pode ser verificado na Tabela 5.4a.

A **transformação substituição inversa de bytes**, chamada InvSubBytes, utiliza a caixa-S inversa mostrada na Tabela 5.4b. Observe, por exemplo, que a entrada {2A} produz a saída {95}, e a entrada {95} para a caixa-S produz {2A}. A caixa-S invertida é construída aplicando-se o inverso da transformação na Equação (5.1), seguido pelo inverso multiplicativo em GF(2^8). A transformação inversa é:

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

onde byte $d = \{05\}$, ou 00000101. Podemos representar essa transformação da seguinte forma:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Para confirmar que InvSubBytes é o inverso de SubBytes, rotule as matrizes de SubBytes e InvSubBytes como **X** e **Y**, respectivamente, e as versões vetoriais das constantes c e d como **C** e **D**, respectivamente. Para algum vetor de 8 bits **B**, a Equação (5.2) torna-se $\mathbf{B}' = \mathbf{XB} \oplus \mathbf{C}$. Precisamos mostrar que $\mathbf{Y}(\mathbf{XB} \oplus \mathbf{C}) \oplus \mathbf{D} = \mathbf{B}$. Efetuando a multiplicação, devemos mostrar que $\mathbf{YXB} \oplus \mathbf{YC} \oplus \mathbf{D} = \mathbf{B}$. Isso se torna:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

Demonstramos que **YX** é igual à matriz identidade, e **YC** = **D,** de modo que **YC** $\oplus$ **D** é igual ao vetor nulo.

Raciocínio A caixa-S é projetada para ser resistente a ataques criptoanalíticos conhecidos. Especificamente, os autores do Rijndael buscaram um projeto que tivesse uma baixa correlação entre bits de entrada e bits de saída, e a propriedade de que a saída não pode ser descrita como uma função matemática simples da entrada [DAEM01]. Além disso, a constante na Equação (5.1) foi escolhida de modo que a caixa-S não tenha pontos fixos [caixa-S(a) = a] nem 'pontos fixos opostos' [caixa-S(a) = $\overline{a}$], onde $\overline{a}$ é o complemento bit a bit de a.

Naturalmente, a caixa-S precisa ser reversível, ou seja, Icaixa-S[caixa-S(a)] = a. Porém, a caixa-S não é auto-inversa no sentido de que não é verdadeiro que caixa-S(a) = Icaixa-S(a). Por exemplo, caixa-S({95}) = {2A}, mas caixa-S({95}) = {AD}.

Transformação ShiftRows

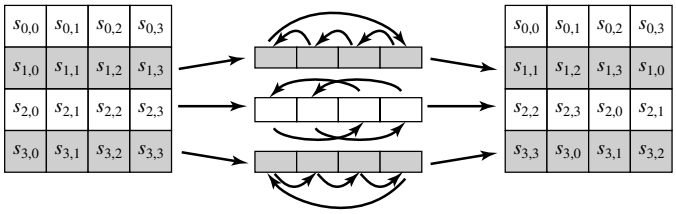
Transformações direta e inversa. A **transformação deslocamento direto de linhas**, chamada ShiftRows, é representada na Figura 5.5a. A primeira linha de **State** não é alterada. Para a segunda linha, é realizado um deslocamento circular de 1 byte à esquerda. Para a terceira linha, é realizado um deslocamento circular de 2 bytes à esquerda. Para a quarta linha, é realizado um deslocamento circular de 3 bytes à esquerda.

Veja um exemplo de ShiftRows:

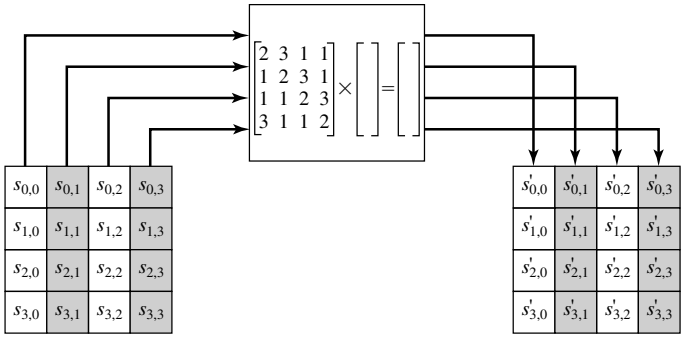
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95



(a) Transformação shift row



(b) Transformação mix column

Figura 5.5 Operações de linha e coluna do AES.

A **transformação deslocamento inverso de linhas**, chamada InvShiftRows, realiza os deslocamentos circulares na direção oposta para cada uma das três últimas linhas, com um deslocamento circular de um byte à direita para a segunda linha, e assim por diante.

Raciocínio A transformação deslocamento de linhas é mais substancial do que pode parecer a princípio. Isso porque o **State**, assim como a entrada e a saída da cifra, é tratado como um vetor de quatro colunas de 4 bytes. Assim, na criptografia, os 4 primeiros bytes do texto claro são copiados para a primeira coluna de **State**, e assim por diante. Além disso, conforme veremos, a chave de rodada é aplicada a **State** coluna por coluna. Assim, um deslocamento de linha move um byte individualmente de uma coluna para outra, que é uma distância linear de um múltiplo de 4 bytes. Observe também que a transformação garante que os 4 bytes de uma coluna são espalhados para quatro colunas diferentes. A Figura 5.3 ilustra o efeito.

Transformação MixColumns

Transformações direta e inversa A **transformação embaralhamento direto de colunas**, chamada Mix- Columns, opera sobre cada coluna individualmente. Cada byte de uma coluna é mapeado para um novo valor que é uma combinação linear de todos os quatro bytes nessa coluna. A transformação pode ser definida pela seguinte multiplicação de matriz sobre **State** (Figura 5.5b).

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.3}$$

Cada elemento na matriz produto é a soma dos produtos dos elementos de uma linha e uma coluna. Nesse caso, as adições e multiplicações[6] individuais são realizadas em GF(2^8). A transformação MixColumns sobre uma única coluna $j(0 \leq j \leq 3)$ de **State** pode ser expressa como

$$\begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned} \tag{5.4}$$

A seguir um exemplo de MixColumns:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Vamos verificar a primeira coluna deste exemplo. Lembre-se de que, conforme visto na Seção 4.6, em GF(2^8) a adição é a operação XOR bit a bit e a multiplicação pode ser realizada de acordo com a regra estabelecida na Equação (4.10). Em particular, a multiplicação de um valor por x (ou seja, por {02}), pode ser implementada como um deslocamento de 1 bit à esquerda por meio de um XOR bit a bit condicional (0001 1011) se o bit mais à esquerda do valor original (antes do deslocamento) for 1. Assim, para verificar a transformação MixColumns na primeira coluna, precisamos mostrar que

$$\begin{aligned} (\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\ \{87\} \oplus (\{02\} \bullet \{6E\}) \oplus (\{03\} \bullet \{46\}) \oplus \{A6\} &= \{37\} \\ \{87\} \oplus \{6E\} \oplus (\{02\} \bullet \{46\}) \oplus (\{03\} \bullet \{A6\}) &= \{94\} \\ (\{03\} \bullet \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \bullet \{A6\}) &= \{ED\} \end{aligned}$$

6. Seguimos a convenção do FIPS PUB 197 e usamos o símbolo $\cdot$ para indicar a multiplicação sobre o corpo finito GF(2^8) e $\oplus$ para indicar o XOR bit a bit, que corresponde à adição em GF(2^8).

Para a primeira equação, temos {02} • {87} = (0000 1110) ⊕ (0001 1011) = (0001 0101); e {03} • {6E} = {6E} ⊕ ({02} • {6E}) = (0110 1110) ⊕ (1101 1100) = (1011 0010). Então

$$\begin{aligned}
\{02\} \cdot \{87\} &= 0001\ 0101 \\
\{03\} \cdot \{6E\} &= 1011\ 0010 \\
\{46\} &= 0100\ 0110 \\
\{A6\} &= \underline{1010\ 0110} \\
& \quad 0100\ 0111 = \{47\}
\end{aligned}$$

As outras equações podem ser verificadas de modo semelhante.

A **transformação embaralhamento inverso de colunas**, chamada InvMixColumns, é definida pela seguinte multiplicação de matrizes:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.5}$$

Não fica imediatamente claro que a Equação (5.5) é o **inverso** da Equação (5.3). Precisamos mostrar que:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

que é equivalente a mostrar que:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

Ou seja, a matriz de transformação inversa multiplicada pela matriz de transformação direta é igual à matriz identidade. Para verificar a primeira coluna da Equação (5.6), precisamos mostrar que:

$$\begin{aligned}
(\{0E\} \bullet \{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\} \bullet \{03\}) &= \{01\} \\
(\{09\} \bullet \{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\} \bullet \{03\}) &= \{00\} \\
(\{0D\} \bullet \{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\} \bullet \{03\}) &= \{00\} \\
(\{0B\} \bullet \{02\}) \oplus \{0D\} \oplus \{09\} \oplus (\{0E\} \bullet \{03\}) &= \{00\}
\end{aligned}$$

Para a primeira equação, temos {0E} • {02} = 00011100; e {09} • {03} = {09} ⊕ ({09} • {02}) = 00001001 ⊕ 00010010 = 00011011. Então

$$\begin{aligned}
\{0E\} \bullet \{02\} &= 00011100 \\
\{0B\} &= 00001011 \\
\{0D\} &= 00001101 \\
\{09\} \bullet \{03\} &= \underline{00011011} \\
& \quad 00000001
\end{aligned}$$

As outras equações podem ser verificadas de modo semelhante.

O documento do AES descreve outra maneira de caracterizar a transformação MixColumns, que é em termos da aritmética de polinômios. No padrão, MixColumns é definida considerando cada coluna de **State** como sendo um polinômio de quarto grau com coeficientes em GF(2^8). Cada coluna é multiplicada módulo ($x^4 + 1$) pelo polinômio fixo $a(x)$, dado por

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad \textbf{(5.7)}$$

O Apêndice 5A demonstra que a multiplicação de cada coluna de **State** por $a(x)$ pode ser escrita como a multiplicação de matriz da Equação (5.3). De modo semelhante, podemos ver que a transformação na Equação (5.5) corresponde a tratar cada coluna como um polinômio de quarto grau e multiplicar cada coluna por $b(x)$, dado por

$$b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} \quad \textbf{(5.8)}$$

Podemos mostrar prontamente que $b(x) = a^{-1}(x) \bmod (x^4 + 1)$.

Raciocínio Os coeficientes da matriz na Equação (5.3) são baseados em um código linear com distância máxima entre as words de código, o que garante um bom embaralhamento entre os bytes de cada coluna. A transformação embaralhamento de colunas combinada com a transformação deslocamento de linhas garante que, após alguns passos, todos os bits da saída dependam de todos os bits da entrada. Veja uma explicação sobre isso em [DAEM99].

Além disso, a escolha dos coeficientes em MixColumns, que são todos {01}, {02} ou {03}, foi influenciada por considerações de implementação. Conforme explicamos, a multiplicação por esses coeficientes envolve no máximo um deslocamento e um XOR. Os coeficientes em InvMixColumns são mais difíceis de implementar. Porém, a criptografia foi julgada mais importante do que a decriptografia por dois motivos:

1. Para os modos de operação CFB e OFB (Figuras 6.5 e 6.6, descritas no Capítulo 6), somente a criptografia é usada.
2. Assim como qualquer cifra em bloco, o AES pode ser usado para construir um código de autenticação de mensagem (Parte Dois), e para isso apenas a criptografia é utilizada.

Transformação AddRoundKey

Transformações direta e inversa Na **transformação adição de chave de rodada direta**, chamada AddRoundKey, os 128 bits de **State** passam por um XOR bit a bit com os 128 bits da chave da rodada. Como mostrado na Figura 5.4b, a operação é vista como uma operação em nível de coluna entre os 4 bytes de uma coluna de **State** e uma word da chave da rodada; ela também pode ser vista como uma operação em nível de byte. A seguir um exemplo de AddRoundKey:

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

A primeira matriz é **State**, e a segunda matriz é a chave da rodada.

A transformação adição inversa de chave de rodada é idêntica à transformação adição direta de chave de rodada, pois a operação XOR é o seu próprio inverso.

Raciocínio A transformação adição de chave da rodada é a mais simples possível, e afeta cada bit de **State**. A complexidade da expansão de chave da rodada, mais a complexidade das outras transformações do AES, garantem a segurança.

Expansão de chave do AES

Algoritmo de expansão de chave O algoritmo de expansão de chave do AES utiliza como entrada uma chave de 4 words (16 bytes) e produz um vetor linear de 44 words (176 bytes). Isso é suficiente para oferecer uma chave de rodada de 4 words para o estágio AddRoundKey inicial e cada uma das 10 rodadas da cifra. O pseudocódigo a seguir descreve a expansão:

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i],
                                       key[4*i+1],
                                       key[4*i+2],
                                       key[4*i+3]);
    for (i = 4; i < 44; i++)
    {
    temp = w[i - 1];
    if (i mod 4 = 0)    temp = SubWord(RotWord(temp))
                               ⊕ Rcon[i/4];
    w[i] = w[i-4] ⊕ temp
    }
}
```

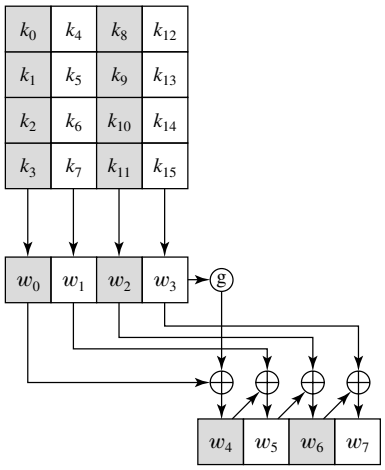


Figura 5.6 Expansão de chave do AES.

A chave é copiada para as quatro primeiras words da chave expandida. O restante da chave expandida é preenchido com quatro words de cada vez. Cada word incluída w[i] depende da word imediatamente anterior e da word quatro posições atrás. Em três dentre quatro casos, um simples XOR é usado. Para uma word cuja posição no vetor w é um múltiplo de 4, uma função mais complexa é usada. A Figura 5.6 ilustra a geração das oito primeiras words da chave expandida, usando o símbolo g para representar essa função complexa. A função g consiste nas seguintes subfunções:

1. RotWord realiza um deslocamento circular de um byte à esquerda em uma word. Isso significa que uma word de entrada [b0, b1, b2, b3] é transformada em [b1, b2, b3, b0].
2. SubWord realiza uma substituição em cada byte de sua word de entrada, usando a caixa-S (Tabela 5.4a).
3. O resultado das etapas 1 e 2 passa por um XOR com a constante de rodada, Rcon[j].

A constante de rodada é uma word na qual os três bytes mais à direita são sempre 0. Assim, o efeito de um XOR de uma word com Rcon é apenas para realizar um XOR no byte mais à esquerda da word. A constante de rodada é diferente para cada rodada, e é definida como Rcon[j] = (RC[j], 0, 0, 0), com (RC[1] = 1, RC[j] = 2 · RC[j − 1] e com a multiplicação definida sobre o corpo GF(2^8). Os valores de RC(j) em hexadecimal são

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Por exemplo, suponha que a chave para a rodada 8 seja

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Então, os 4 primeiros bytes (primeira coluna) da chave para a rodada 9 são calculados da seguinte forma:

i (decimal)	temp	Após RotWord	Após SubWord	Rcon (9)	Após XOR com Rcon	w[i − 4]	w[i] = temp ⊕ w[i − 4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Raciocínio Os autores do Rijndael criaram o algoritmo de expansão de chave para ser resistente a ataques criptoanalíticos conhecidos. A inclusão de uma constante de rodada dependente do rodada elimina a simetria, ou similaridade, entre os modos como as chaves de rodada são geradas em diferentes rodadas. Os critérios específicos usados são os seguintes [DAEM99]:

- O conhecimento de uma parte da chave de criptografia ou da chave de rodada não possibilita o cálculo de vários outros bits da chave de rodada
- Uma transformação reversível [ou seja, o conhecimento de quaisquer *Nk* words consecutivas da chave expandida possibilita a regeneração da chave expandida inteira (*Nk* = tamanho da chave, em words)]
- Velocidade em uma grande gama de processadores
- Uso de constantes de rodada para eliminar simetrias
- Propagação das diferenças da chave de criptografia nas chaves de rodada; ou seja, cada bit da chave de criptografia afeta muitos bits da chave de rodada
- Não-linearidade suficiente para impedir a determinação total das diferenças da chave de rodada somente a partir das diferenças da chave de criptografia
- Simplicidade de descrição

Os autores não quantificam o primeiro ponto da lista anterior, mas a idéia é que, se você souber menos do que *Nk* words consecutivas da chave de criptografia ou de uma das chaves de rodada, então será difícil reconstruir os bits desconhecidos restantes. Quanto menos bits forem conhecidos, mais difícil será reconstituir ou determinar outros bits na expansão da chave.

Cifra inversa equivalente

Conforme dissemos, a cifra de decriptografia do AES não é idêntica à cifra de criptografia (Figura 5.1). Ou seja, a seqüência de transformações para a decriptografia difere da seqüência para a criptografia, embora a forma dos escalonamentos de chave para criptografia e decriptografia seja a mesma. Isso tem a desvantagem de que dois módulos separados de software ou firmware são necessários para aplicações que exigem tanto criptografia quanto decriptografia. Porém, existe uma versão equivalente do algoritmo de decriptografia que tem a mesma estrutura do algoritmo de criptografia. A versão equivalente tem a mesma seqüência de transformações do algoritmo de criptografia (com transformações substituídas por seus inversos). Para conseguir essa equivalência, é preciso haver uma mudança no escalonamento de chave.

Duas mudanças separadas são necessárias para alinhar a estrutura de decriptografia com a estrutura de criptografia. Uma rodada de criptografia tem a estrutura SubBytes, ShiftRows, MixColumns, AddRoundKey. A rodada de decriptografia padrão tem a estrutura InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Assim, os dois primeiros estágios da rodada de decriptografia precisam ser trocados, assim como os dois estágios seguintes da rodada de decriptografia.

Trocando InvShiftRows e InvSubBytes InvShiftRows afeta a seqüência de bytes em **State**, mas não altera o conteúdo dos bytes e não depende desse conteúdo para realizar sua transformação. InvSubBytes afeta o conteúdo dos bytes em **State**, mas não altera a seqüência de bytes e não depende dessa seqüência para realizar sua transformação. Assim, essas duas operações comutam e podem ser trocadas. Para um determinado **State** S_i,

$$\text{InvShiftRows}\,[\text{InvSubBytes}\,(S_i)] = \text{InvSubBytes}\,[\text{InvShiftRows}\,(S_i)]$$

Trocando AddRoundKey e InvMixColumns As transformações AddRoundKey e InvMixColumns não alteram a seqüência de bytes em **State**. Se examinarmos a chave como uma seqüência de words, então AddRoundKey e InvMixColumns operam sobre **State** uma coluna de cada vez. Essas duas operações são lineares com relação à entrada de coluna. Ou seja, para determinado **State** S_i e determinada chave de rodada w_j:

$$\text{InvMixColumns}\,(S_i \oplus w_j) = [\text{InvMixColumns}\,(S_i)] \oplus [\text{InvMixColumns}\,(w_j]]$$

Para entender isso, suponha que a primeira coluna do **State** S_i seja a seqüência (y_0, y_1, y_2, y_3) e que a primeira coluna da chave de rodada w_j seja (k_0, k_1, k_2, k_3). Então, precisamos mostrar que

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

Vamos demonstrar isso para a entrada da primeira coluna. Precisamos mostrar que:

$$\begin{aligned} & [\{0E\} \bullet (y_0 \oplus k_0)] \oplus [\{0B\} \bullet (y_1 \oplus k_1)] \oplus [\{0D\} \bullet (y_2 \oplus k_2)] \oplus [\{09\} \bullet (y_3 \oplus k_3)] \\ = & [\{0E\} \bullet y_0] \oplus [\{0B\} \bullet y_1] \oplus [\{0D\} \bullet y_2] \oplus [\{09\} \bullet y_3] \\ \oplus & [\{0E\} \bullet k_0] \oplus [\{0B\} \bullet k_1] \oplus [\{0D\} \bullet k_2] \oplus [\{09\} \bullet k_3] \end{aligned}$$

Essa equação é válida por inspeção. Assim, podemos trocar AddRoundKey e InvMixColumns, desde que primeiro apliquemos InvMixCOlumns à chave da rodada. Observe que não precisamos aplicar InvMixColumns à chave de rodada para a entrada da primeira transformação AddRoundKey (precedendo a primeira rodada), nem da última transformação AddRoundKey (na rodada 10). Isso porque essas duas transformações AddRoundKey não são trocadas com InvMixColumns para produzir o algoritmo de decriptografia equivalente.

A Figura 5.7 ilustra o algoritmo de decriptografia equivalente.

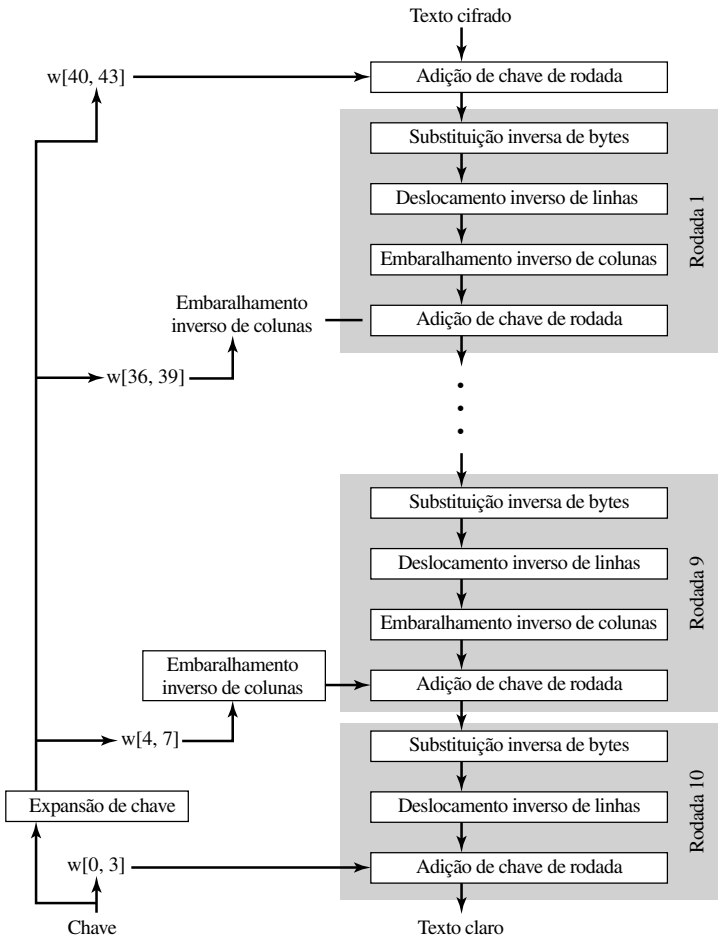


Figura 5.7 Cifra inversa equivalente

Aspectos de implementação

A proposta Rijndael [DAEM99] oferece algumas sugestões para a implementação eficiente em processadores de 8 bits, típicos de smart cards atuais, e em processadores de 32 bits, típicos de PCs.

Processador de 8 bits O AES pode ser implementado de modo muito eficiente em um processador de 8 bits. AddRoundKey é uma operação XOR byte a byte. ShiftRows é uma operação simples de deslocamento de bytes. SubBytes opera em nível de byte e só exige uma tabela de 256 bytes.

A transformação MixColumns exige multiplicação de matriz no corpo GF(2^8), o que significa que todas as operações são executadas sobre bytes. MixColumns só exige multiplicação por {02} e {03} que, como vimos, envolve deslocamentos simples, XORs condicionais e XORs. Isso pode ser implementado de uma maneira mais eficiente que elimina os deslocamentos e XORs condicionais. O conjunto de equações (5.4) mostra as equações para a transformação MixColumns em uma única coluna. Usando a identidade $\{03\} \cdot x = (\{02\} \cdot x) \oplus x$, podemos reescrever o conjunto de equações (5.4) como segue:

$$\begin{aligned} Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\ s'_{0,j} &= s_{0,j} \oplus Tmp \oplus [2 \cdot (s_{0,j} \oplus s_{1,j})] \\ s'_{1,j} &= s_{1,j} \oplus Tmp \oplus [2 \cdot (s_{1,j} \oplus s_{2,j})] \\ s'_{2,j} &= s_{2,j} \oplus Tmp \oplus [2 \cdot (s_{2,j} \oplus s_{3,j})] \\ s'_{3,j} &= s_{3,j} \oplus Tmp \oplus [2 \cdot (s_{3,j} \oplus s_{0,j})] \end{aligned} \tag{5.9}$$

O conjunto de equações (5.9) é verificado pela expansão e eliminação de termos.

A multiplicação por {02} envolve um deslocamento e um XOR condicional. Essa implementação pode ser vulnerável a um ataque de temporização do tipo descrito na Seção 3.4. Para evitar esse ataque e aumentar a eficiência de processamento, à custa de algum armazenamento, a multiplicação pode ser substituída por uma pesquisa de tabela. Defina a tabela de 256 bytes X2, de modo que $X2[i] = \{02\} \cdot i$. Então, o conjunto de equações (5.9) pode ser reescrito como

$$\begin{aligned} Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\ s'_{0,j} &= s_{0,j} \oplus Tmp \oplus X2[s_{0,j} \oplus s_{1,j}] \\ s'_{1,c} &= s_{1,j} \oplus Tmp \oplus X2[s_{1,j} \oplus s_{2,j}] \\ s'_{2,c} &= s_{2,j} \oplus Tmp \oplus X2[s_{2,j} \oplus s_{3,j}] \\ s'_{3,j} &= s_{3,j} \oplus Tmp \oplus X2[s_{3,j} \oplus s_{0,j}] \end{aligned}$$

Processador de 32 bits A implementação descrita na subseção anterior usa apenas operações de 8 bits. Para um processador de 32 bits, uma implementação mais eficiente pode ser conseguida se as operações forem definidas sobre words de 32 bits. Para mostrar isso, primeiro definimos as quatro transformações de uma rodada em formato algébrico. Suponha que comecemos com uma matriz **State** consistindo em elementos $a_{i,j}$ e uma matriz de chave de rodada consistindo em elementos $k_{i,j}$. Então, as transformações podem ser expressas da seguinte forma:

SubBytes	$b_{i,j} = \mathrm{S}[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

Na equação de ShiftRows, os índices de coluna são considerados mod 4. Podemos combinar todas essas expressões em uma única equação:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} \mathrm{S}[a_{0,j}] \\ \mathrm{S}[a_{1,j-1}] \\ \mathrm{S}[a_{2,j-2}] \\ \mathrm{S}[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

$$= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet \mathrm{S}[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet \mathrm{S}[a_{1,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet \mathrm{S}[a_{2,j-2}] \right)$$

$$\oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet \mathrm{S}[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Na segunda equação, estamos expressando a multiplicação de matrizes como uma combinação linear de vetores. Definimos quatro tabelas de 256 words (1024 bytes) da seguinte forma:

$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet \mathrm{S}[x] \right)$	$T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet \mathrm{S}[x] \right)$	$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet \mathrm{S}[x] \right)$	$T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet \mathrm{S}[x] \right)$

Assim, cada tabela utiliza um valor de byte como entrada e produz um vetor de coluna (uma word de 32 bits) que é uma função da entrada da caixa-S para esse valor de byte. Essas tabelas podem ser calculadas antecipadamente.

Podemos definir uma função de rodada operando sobre uma coluna da seguinte maneira:

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Como resultado, uma implementação baseada na equação anterior requer apenas quatro pesquisas de tabela e quatro XORs por coluna por rodada, mais 4 Kbytes para armazenar a tabela. Os autores do Rijndael acreditam que essa implementação compacta e eficiente provavelmente foi um dos fatores mais importantes na seleção do Rijndael para o AES.

5.3 LEITURA E SITES WEB RECOMENDADOS

A descrição mais completa do AES disponível até agora é o livro escrito pelos autores do AES, [DAEM02]. Os autores também oferecem uma rápida descrição e raciocínio do projeto em [DAEM01]. [LAND04] é um tratamento matemático rigoroso do AES e de sua criptoanálise.

DAEM01 Daemen, J., e Rijmen, V. "Rijndael: the Advanced Encryption Standard." *Dr. Dobb's Journal*, março de 2001.

DAEM02 Daemen, J. e Rijmen, V. *The design of Rijndael: AES – the Advanced Encryption Standard*. New York, Springer-Verlag, 2002.

LAND04 Landau, S. "Polynomials in the nation's service: using algebra to design the Advanced Encryption Standard." *American Mathematical Monthly*, fevereiro de 2004.

Sites Web recomendados:

- Home page do AES: Página do NIST sobre o AES. Contém o padrão além de vários outros documentos relevantes.
- The AES Lounge: Contém uma bibliografia abrangente de documentos e artigos sobre AES, com acesso a cópias eletrônicas.

5.4 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

Advanced Encryption Standard (AES)
análise de potência
caixa-S
National Institute of Standards and Technology (NIST)
Rijndael

Perguntas de revisão

5.1 Qual foi o conjunto original de critérios usados pelo NIST para avaliar as cifras AES candidatas?
5.2 Qual foi o conjunto final de critérios usados pelo NIST para avaliar as cifras AES candidatas?
5.3 O que é análise de potência?
5.4 Qual é a diferença entre Rijndael e AES?
5.5 Qual é a finalidade do vetor **State**?
5.6 Como é construída a caixa-S?
5.7 Descreva resumidamente o estágio SubBytes.
5.8 Descreva resumidamente o estágio ShiftRows.
5.9 Quantos bytes em **State** são afetados por ShiftRows?
5.10 Descreva resumidamente o estágio MixColumns.
5.11 Descreva resumidamente o estágio AddRoundKey.
5.12 Descreva resumidamente o algoritmo de expansão de chave.
5.13 Qual é a diferença entre SubBytes e SubWord?
5.14 Qual é a diferença entre ShiftRows e RotWord?
5.15 Qual é a diferença entre o algoritmo de decriptografia AES e a cifra inversa equivalente?

Problemas

5.1 Na explicação de MixColumns e InvMixColumns, foi dito que

$$b(x) = a^{-1}(x) \bmod (x^4 + 1)$$

onde $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ e $b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$. Mostre que isso é verdadeiro.

5.2 **a.** O que é $\{01\}^{-1}$ em GF(2^8)?
b. Verifique a entrada para {01} na caixa-S.

5.3 Mostre as primeiras oito words da expansão de chave para uma chave de 128 bits, todos com zero.

5.4 Dado o texto claro {00010203040506070809 0A0B0C0D0E0F} e a chave {0101010101010101010 10101 01010101},
a. Mostre o conteúdo original de **State**, exibido como uma matriz 4 × 4.
b. Mostre o valor de **State** após o AddRoundKey inicial.
c. Mostre o valor de **State** após SubBytes.
d. Mostre o valor de **State** após ShiftRows.
e. Mostre o valor de **State** após MixColumns.

5.5 Verifique a Equação (5.11). Ou seja, mostre que $x^i \bmod (x^4 + 1) = x^{i \bmod 4}$.

5.6 Compare AES com DES. Para cada um dos seguintes elementos do DES, indique o elemento análogo no AES ou explique por que ele não é necessário no AES.
a. XOR do conteúdo da subchave com a entrada da função f.
b. XOR da saída da função f com a metade esquerda do bloco
c. A função f
d. Permutação P
e. Troca de metades do bloco

5.7 Na subseção sobre aspectos de implementação, mencionamos que o uso de tabelas ajuda a impedir ataques de temporização. Sugira uma técnica alternativa.

5.8 Na subseção sobre aspectos de implementação, uma única equação algébrica é desenvolvida para descrever os quatro estágios de uma rodada típica do algoritmo de criptografia. Forneça a equação equivalente para a décima rodada.

5.9 Calcule a saída da transformação MixColumns para a seguinte seqüência de bytes de entrada "67 89 AB CD". Aplique a transformação InvMix Columns ao resultado obtido para verificar seus cálculos. Mude o primeiro byte da entrada de '67' para '77', aplique a transformação MixColumns novamente para a nova entrada e determine quantos bits mudaram na saída. *Nota:* Você pode realizar todos os cálculos à mão ou escrever um programa de apoio a esses cálculos. Se você preferir escrever um programa, ele deverá ser escrito intei-

ramente por você; nesta tarefa, não use bibliotecas ou código-fonte de domínio público.

5.10 Use a chave 1010 0111 0011 1011 para criptografar o texto claro "ok" conforme expresso em ASCII, ou seja, 0110 1111 0110 1011. Os projetistas do S-AES obtiveram o texto cifrado 0000 0111 0011 1000. E você?

5.11 Mostre que a matriz a seguir, com entradas em GF(2^4), é o inverso da matriz usada na etapa MixColumns do S-AES.

$$\begin{pmatrix} x^3 + 1 & x \\ x & x^3 + 1 \end{pmatrix}$$

5.12 Descreva minuciosamente uma decriptografia completa do texto cifrado 0000 0111 0011 1000, usando a chave 1010 0111 0011 1011 e o algoritmo S-AES. Você deverá obter o texto claro com o qual começamos o Problema 5.10. Observe que o inverso das caixas-S pode ser feito com uma pesquisa de tabela reversa. O inverso da etapa MixColumns é dado pela matriz do problema anterior.

Problemas de programação

5.13 Crie um software que possa criptografar e decriptografar usando S-AES. Dados de teste: um texto claro binário 0110 1111 0110 1011 criptografado com uma chave binária 1010 0111 0011 1011 deverá resultar no texto cifrado binário 0000 0111 0011 1000 less ecb $$$). A decriptografia deverá funcionar da mesma maneira.

5.14 Implemente um ataque de criptoanálise diferencial em uma rodada do S-AES.

APÊNDICE 5A: POLINÔMIOS COM COEFICIENTE EM GF(2^8)

Na Seção 4.5, explicamos a aritmética polinomial, na qual os coeficientes estão em Z_p e os polinômios são definidos módulo um polinômio $M(x)$, cuja potência mais alta é algum inteiro n. Nesse caso, adição e multiplicação de coeficientes ocorrem dentro do corpo Z_p, ou seja, adição e multiplicação são realizadas módulo p.

O documento do AES define a aritmética polinomial para polinômios de grau 3 ou menor, com coeficientes em GF(2^8). As regras a seguir se aplicam:

1. A adição é realizada somando-se os coeficientes correspondentes em GF(2^8). Conforme indicado na Seção 4.5, se tratarmos os elementos de GF(2^8) como cadeias de 8 bits, então a adição será equivalente à operação XOR. Assim, se tivermos

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad \textbf{(5.8)}$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \quad \textbf{(5.9)}$$

então

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

2. A multiplicação é realizada como na multiplicação normal de polinômios, com dois refinamentos:
   a. Os coeficientes são multiplicados em GF(2^8).
   b. O polinômio resultante é reduzido mod $(x^4 + 1)$.

Precisamos esclarecer a respeito de qual polinômio estamos falando. Lembre-se da Seção 4.6, em que cada elemento de GF(2^8) é um polinômio de grau 7 ou menor com coeficientes binários, e a multiplicação é executada módulo um polinômio de grau 8. De modo equivalente, cada elemento de GF(2^8) pode ser visto como um byte de 8 bits cujos valores dos bits correspondem aos coeficientes binários do polinômio correspondente. Para os conjuntos definidos nesta seção, estamos definindo um anel polinomial em que cada elemento do anel é um polinômio de grau 3 ou menor, com coeficientes em GF(2^8), e a multiplicação é executada módulo um polinômio de grau 4. De modo equivalente, cada elemento desse anel pode ser visto como uma word de 4 bytes cujos valores dos bytes são elementos de GF(2^8) que correspondem aos coeficientes de 8 bits do polinômio correspondente.

Denotamos o produto modular de $a(x)$ e $b(x)$ por $a(x) \oplus b(x)$. Para calcular $d(x) = a(x) \otimes b(x)$, o primeiro passo é realizar uma multiplicação sem a operação de módulo e juntar os coeficientes de potências semelhantes. Vamos expressar isso como $c(x) = a(x) \times b(x)$. Então,

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \quad \textbf{(5.10)}$$

onde

$$c_0 = a_0 \cdot b_0$$
$$c_1 = (a_1 \cdot b_0) \oplus (a_0 \cdot b_1)$$
$$c_2 = (a_2 \cdot b_0) \oplus (a_1 \cdot b_1)\ [+]\ (a_0 \cdot b_2)$$
$$c_3 = (a_3 \cdot b_0) \oplus (a_2 \cdot b_1)\ [+]\ (a_1 \cdot b_2)\ [+]\ (a_0 \cdot b_3)$$
$$c_4 = (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$$
$$c_5 = (a_3 \cdot b_2) \oplus (a_2 \cdot b_3)$$
$$c_6 = a_3 \cdot b_3$$

A última etapa é realizar a operação de módulo:

$$d(x) = c(x) \bmod (x^4 + 1)$$

Ou seja, $d(x)$ deve satisfazer a equação

$$c(x) = [(x^4 + 1) \times q(x)] \oplus d(x)$$

de modo que o grau de $d(x)$ seja 3 ou menor.

Uma técnica prática para realizar a multiplicação por esse anel polinomial é baseada na observação de que

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4} \tag{5.11}$$

Se agora combinarmos as Equações (5.10) e (5.11), acabaremos com

$$\begin{aligned} d(x) &= c(x) \bmod (x^4 + 1) = [c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0] \bmod (x^4 + 1) \\ &= c_3x^3 + (c_2 \oplus c_6)x^2 + (c_1 \oplus c_5)x + (c_0 \oplus c_4) \end{aligned}$$

Expandindo os coeficientes, temos as seguintes equações para os coeficientes de $d(x)$:

$$\begin{aligned} d_0 &= (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\ d_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\ d_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\ d_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3) \end{aligned}$$

Isso pode ser escrito em forma de matriz:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{5.12}$$

Transformação MixColumns

Na explicação sobre MixColumns, foi determinado que existem duas maneiras equivalentes de definir a transformação. A primeira é a multiplicação de matriz, mostrada na Equação (5.3) e repetida aqui:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

A segunda método é tratar cada coluna de **State** como um polinômio de quatro termos com coeficientes em GF(2^8). Cada coluna é multiplicada módulo $(x^4 + 1)$ pelo polinômio fixo $a(x)$, dado por

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Pela Equação (5.8), temos $a_3 = \{03\}$; $a_2 = \{01\}$; $a_1 = \{01\}$; $a_0 = \{02\}$. Para a j-ésima coluna de **State**, temos o polinômio $\text{col}_j(x) = s_{3,j}x^3 + s_{2,j}x^2 + s_{1,j}x + s_{0,j}$. Substituindo na Equação (5.12), podemos expressar $d(x) = a(x) \times \text{col}j(x)$ como

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

que é equivalente à Equação (5.3).

Multiplicação por x

Considere a multiplicação de um polinômio no anel por x: $c(x) = x \otimes b(x)$. Temos

$$\begin{aligned} c(x) &= x \otimes b(x) = [x \times (b_3x^3 + b_2x^2 + b_1x + b_0)] \bmod (x^4 + 1) \\ &= (b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod (x^4 + 1) \\ &= b_2x^3 + b_1x^2 + b_0x + b_3 \end{aligned}$$

Assim, a multiplicação por x corresponde a um deslocamento circular de 1 byte à esquerda dos 4 bytes na word que representa o polinômio. Se representarmos o polinômio como um vetor de colunas de 4 bytes, então temos

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

APÊNDICE 5B: AES SIMPLIFICADO

O AES simplificado (S-AES) foi desenvolvido pelo professor Edward Schaefer da Universidade de Santa Clara e vários de seus alunos [MUSA03]. Ele é um algoritmo de criptografia educacional, não um algoritmo seguro. Ele possui propriedades e estrutura semelhantes ao AES, com parâmetros muito menores.

O leitor poderia achar útil trabalhar com um exemplo à mão enquanto acompanha a discussão neste apêndice. Uma boa compreensão do S-AES facilitará ao aluno apreciar a estrutura e o funcionamento do AES.

Visão geral

A Figura 5.8 ilustra a estrutura geral do S-AES. O algoritmo de criptografia utiliza um bloco de 16 bits de texto claro como entrada e uma chave de 16 bits, e produz um bloco de 16 bits de texto cifrado como saída. O algoritmo de decriptografia do S-AES utiliza um bloco de 16 bits de texto cifrado e a mesma chave de 16 bits usada para produzir esse texto cifrado como entrada, produzindo como saída o bloco de 16 bits de texto claro original.

O algoritmo de criptografia envolve o uso de quatro funções diferentes ou transformações: adição de chave (A_K), substituição de nibbles (NS), deslocamento de linhas (SR) e embaralhamento de colunas (MC), cujas operações são explicadas mais adiante.

Podemos expressar o algoritmo de criptografia de maneira concisa como uma composição[7] de funções: de modo que

$$A_{k_0} \circ SR \circ NS \circ A_{k_1} \circ MC \circ SR \circ NS \circ A_{k_0}$$

de modo que A_{k_0} é aplicado primeiro.

O algoritmo de criptografia é organizado em três rodadas. A rodada 0 é simplesmente uma rodada de adição de chave, a rodada 1 é uma rodada completo de quatro funções e a rodada 2 contém apenas 3 funções. Cada rodada engloba a função adição de chave, a qual utiliza os 16 bits da chave. A chave inicial de 16 bits é expandida para 48 bits, de modo que cada rodada utiliza uma chave de rodada distinta de 16 bits.

Cada função opera sobre um estado de 16 bits, tratado como uma matriz 2×2 de nibbles, onde um nibble é igual a 4 bits. O valor inicial da matriz de estado é o texto claro de 16 bits; a matriz de estado é modificada por cada função subseqüente no processo de criptografia, produzindo após a última função o texto cifrado de 16 bits. Como mostra a Figura 5.9a, a ordenação dos nibbles dentro da matriz é por coluna. Assim, por exemplo, os primeiros oito bits de uma entrada de texto claro de 16 bits para a criptografia ocupa a primeira coluna da matriz, e os próximos

7. **Definição:** Se f e g são duas funções, então a função F com a equação $y = F(x) = g[f(x)]$ é chamada de **composição** de f e g e é denotada como $F = g \circ f$.

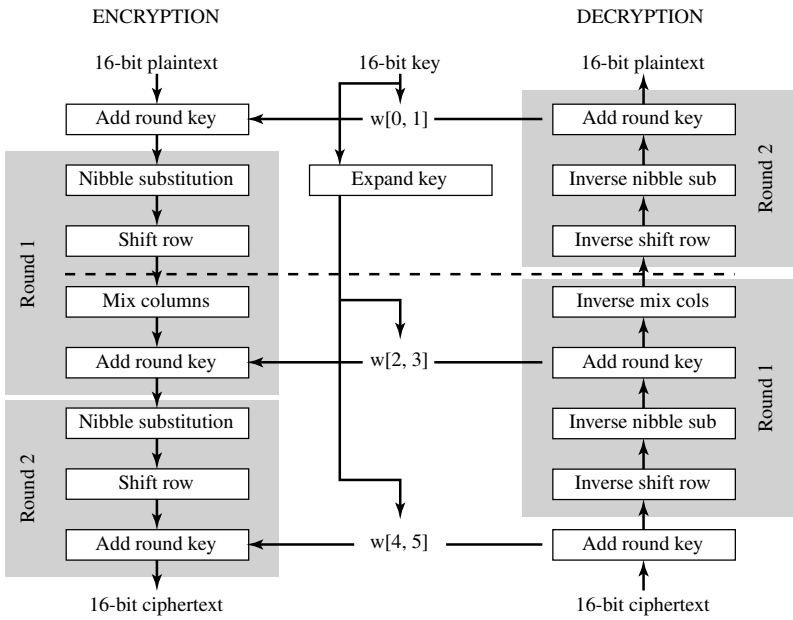


Figura 5.8 Criptografia e decriptografia do S-AES

oito bits ocupam a segunda coluna. A chave de 16 bits é organizada de modo semelhante, mas é um pouco mais conveniente ver a chave como dois bytes, em vez de quatro nibbles (Figura 5.9b). A chave expandida de 48 bits é tratada como três chaves de rodada, cujos bits são rotulados da seguinte forma: $K_0 = k_0 \ldots k_{15}$; $K_1 = k_{16} \ldots k_{31}$; $K_2 = k_{32} \ldots k_{47}$.

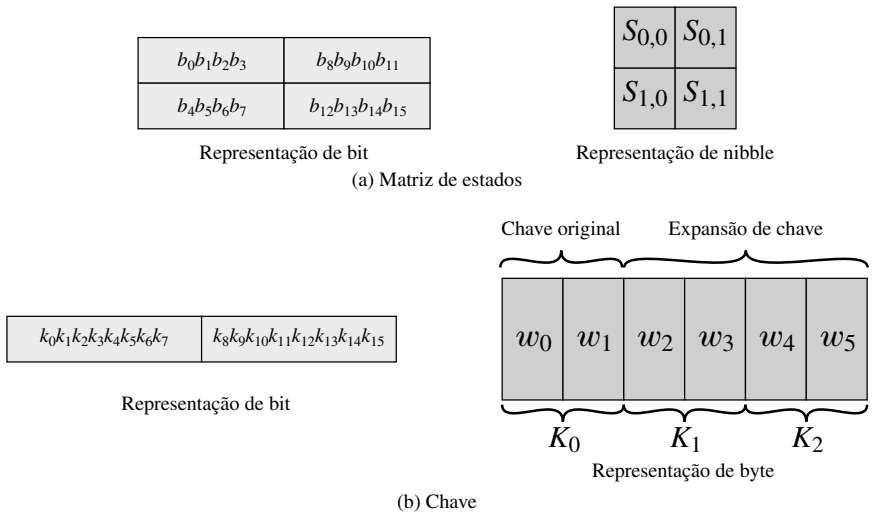


Figura 5.9 Estruturas de dados do S-AES

A Figura 5.10 mostra os elementos essenciais de uma rodada completa do S-AES.

A decriptografia também aparece na Figura 5.8 e é basicamente o inverso da criptografia:

$$A_{k0} \circ \text{INS} \circ \text{ISR} \circ \text{IMC} \circ A_{k1} \circ \text{INS} \circ \text{ISR} \circ A_{k2}$$

na qual três das funções têm uma função inversa correspondente: substituição inversa de nibbles (INS), deslocamento inverso de linhas (ISR) e embaralhamento inverso de colunas (IMC).

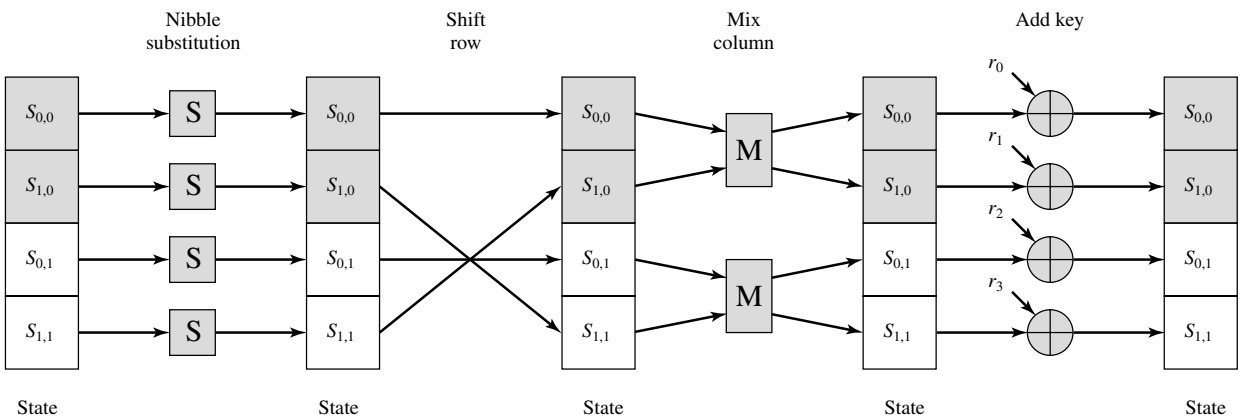


Figura 5.10 Rodada de criptografia do S-AES

Criptografia e decriptografia S-AES

Agora, vejamos as funções individuais que fazem parte do algoritmo de criptografia.

Adição de chave A função de adição de chave consiste no XOR bit a bit entre a matriz de estado de 16 bits e a chave de rodada de 16 bits. A Figura 5.11 representa isso como uma operação coluna a coluna, mas também pode ser vista como uma operação nibble a nibble ou bit a bit. Veja um exemplo a seguir:

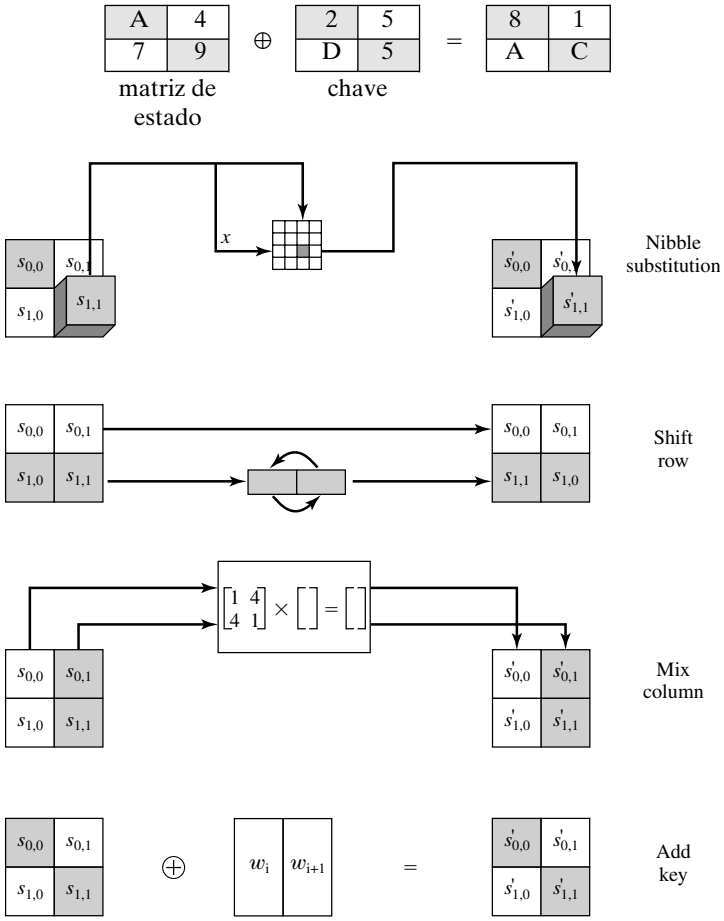


Figura 5.11 Transformações S-AES

O inverso da função adição de chave é idêntico à função adição de chave, pois a operação XOR é o seu próprio inverso.

Tabela 5.5 Caixas-S do S-AES

		j			
		00	**01**	**10**	**11**
i	00	9	4	A	B
	01	D	1	8	5
	10	6	2	0	3
	11	C	E	F	7

(a) Caixa-S

		j			
		00	**01**	**10**	**11**
i	00	A	5	9	B
	01	1	7	8	F
	10	6	0	2	3
	11	C	4	D	E

(b) Caixa-S invertida

Nota: Números hexadecimais em caixas sombreadas; números binários em caixas não sombreadas.

Substituição de nibbles A função de substituição de nibbles é uma pesquisa simples de tabela (Figura 5.11). O S-AES define uma matriz 4 X 4 de valores de nibbles, chamada caixa-S (Tabela 5.5a), que contém uma permutação de todos os valores de 4 bits possíveis. Cada nibble da matriz de estado é mapeado para um novo nibble da seguinte maneira: os 2 bits mais à esquerda do nibble são usados como um valor de linha e os 2 bits mais à direita são usados como um valor de coluna. Esses valores de linha e coluna servem como índices na caixa-S para selecionar um valor de saída exclusivo de 4 bits. Por exemplo, o valor hexadecimal A referencia a linha 2, coluna 2 da caixa-S, que contém o valor 0. Por conseguinte, o valor A é mapeado para o valor 0.

Aqui está um exemplo da transformação de substituição de nibble:

8	1
A	C

→

6	4
0	C

A função de substituição inversa de nibbles utiliza a caixa-S invertida, mostrada na Tabela 5.5b. Observe, por exemplo, que a entrada 0 produz a saída A, e a saída A para a caixa-S produz 0.

Deslocamento de linhas A função de deslocamento de linhas realiza o deslocamento circular de um nibble da segunda linha da matriz de estado; a primeira linha não é alterada (Figura 5.11). A seguir um exemplo:

6	4
0	C

→

6	4
C	0

A função de deslocamento inverso de linhas é idêntica à função de deslocamento de linhas, pois desloca a segunda linha de volta à sua posição original.

Embaralhamento de colunas A função embaralhamento de colunas opera sobre cada coluna individualmente. Cada nibble de uma coluna é mapeado para um novo valor que é uma função de ambos os nibbles nessa coluna. A transformação pode ser definida pela seguinte multiplicação de matriz sobre a matriz de estado (Figura 5.11):

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} \\ S'_{1,0} & S'_{1,1} \end{bmatrix}$$

Realizando a multiplicação de matriz, obtemos:

$$\begin{aligned} S'_{0,0} &= S_{0,0} \oplus (4 \cdot S_{1,0}) \\ S'_{1,0} &= (4 \cdot S_{0,0}) \oplus S_{1,0} \\ S'_{0,1} &= S_{0,1} \oplus (4 \cdot S_{1,1}) \\ S'_{1,1} &= (4 \cdot S_{0,1}) \oplus S_{1,1} \end{aligned}$$

Onde a aritmética é realizada em GF(2^4), e o símbolo · refere-se à multiplicação em GF(2^4). O Apêndice E fornece as tabelas de adição e multiplicação. A seguir um exemplo:

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 7 & 3 \end{bmatrix}$$

A função embaralhamento inverso de colunas é definida da seguinte forma:

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} \\ S'_{1,0} & S'_{1,1} \end{bmatrix}$$

Demonstramos que realmente definimos o inverso da seguinte maneira:

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix}$$

A multiplicação de matriz anterior utiliza os seguintes resultados em GF(2^4): $9 + (2 \cdot 4) = 9 + 8 = 1$; $(9 \cdot 4) + 2 = 2 + 2 = 0$. Essas operações podem ser verificadas por meio de tabelas aritméticas no Apêndice E ou pela aritmética polinomial.

A função embaralhamento de colunas é a mais difícil de visualizar. Por conseguinte, oferecemos uma perspectiva adicional para ela no Apêndice E.

Expansão de chave

Para a expansão de chave, os 16 bits da chave inicial são agrupados em uma fila de duas words de 8 bits. A Figura 5.12 mostra a expansão para 6 words, pelo cálculo de 4 novas words a partir das 2 words iniciais. O algoritmo é o seguinte:

$$\begin{aligned} w_2 &= w_0 \oplus g(w_1) = w_0 \oplus \text{RCON}(1) \oplus \text{SubNib}(\text{RotNib}(w_1)) \\ w_3 &= w_2 \oplus w_1 \\ w_4 &= w_2 \oplus g(w_3) = w_2 \oplus \text{RCON}(2) \oplus \text{SubNib}(\text{RotNib}(w_3)) \\ w_5 &= w_4 \oplus w_3 \end{aligned}$$

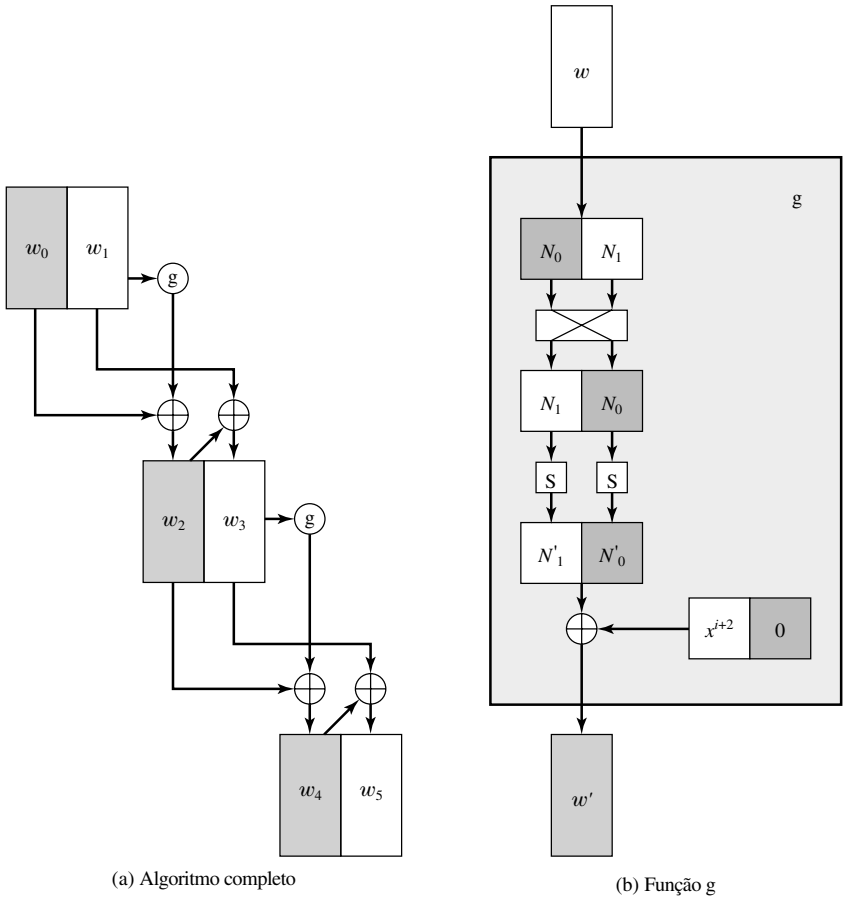


Figura 5.12 Expansão de chave do S-AES

RCON é uma constante de rodada, definida da seguinte forma: RC[i] = x^{i+2}, de modo que RC[1] = x^3 = 1000 e RC[2] = x^4 mod ($x^4 + x + 1$) = $x + 1$ = 0011. RC[i] forma o nibble mais à esquerda de um byte, com o nibble mais à direita sendo composto apenas de zeros. Assim, RCON(1) = 10000000 e RCON(2) = 00110000.

Por exemplo, suponha que a chave seja 2D55 = 0010 1101 0101 0101 = w_0w_1. Então

$$\begin{aligned}
w_2 &= 00101101 \oplus 10000000 \oplus \text{SubNib}(01010101) \\
&= 00101101 \oplus 10000000 \oplus 00010001 = 10111100 \\
w_3 &= 10111100 \oplus 01010101 = 11101001 \\
w_4 &= 10111100 \oplus 00110000 \oplus \text{SubNib}(10011110) \\
&= 10111100 \oplus 00110000 \oplus 00101111 = 10100011 \\
w_5 &= 10100011 \oplus 11101001 = 01001010
\end{aligned}$$

A caixa-S

A caixa-S é construída da seguinte forma:

1. Inicialize a caixa-S com os valores de nibble em ordem crescente linha por linha. A primeira linha contém os valores hexadecimais 0, 1, 2, 3; a segunda linha contém 4, 5, 6, 7; e assim por diante. Assim, o valor do nibble na linha i, coluna j é $4i + j$.
2. Trate cada nibble como um elemento do corpo finito GF(2^4) módulo $x^4 + x + 1$. Cada nibble $a_0a_1a_2a_3$ representa um polinômio de grau 3.
3. Mapeie cada byte da caixa-S em seu inverso multiplicativo no corpo finito GF(2^4) módulo $x^4 + x + 1$; o valor 0 é mapeado em si mesmo.
4. Considere que cada byte na caixa-S consiste em 4 bits rotulados com (b_0, b_1, b_2, b_3). Aplique a seguinte transformação a cada bit de cada byte na caixa-S: o padrão do AES representa essa transformação no formato de matriz da seguinte maneira:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

O apóstrofo (') indica que a variável deve ser atualizada pelo valor à direita. Lembre-se de que adição e multiplicação estão sendo calculados com módulo 2.

A Tabela 5.5a mostra a caixa-S resultante. Esta é uma matriz não linear e reversível. A caixa-S inversa aparece na Tabela 5.5b.

Estrutura do S-AES

Agora, podemos examinar vários aspectos interessantes com relação à estrutura do S-AES. Primeiro, observe que os algoritmos de criptografia e decriptografia começam e terminam com a função adição de chave. Qualquer outra função, no início ou no fim, é facilmente reversível sem conhecimento da chave e, por isso, não incluiria segurança, mas apenas um overhead de processamento. Assim, existe uma rodada 0 consistindo apenas na função de adição de chave.

O segundo ponto a observar é que a rodada 2 não inclui a função de embaralhamento de colunas. A explicação para isso na verdade está relacionada a uma terceira observação de que, embora o algoritmo de decriptografia seja o inverso do algoritmo de criptografia, como podemos ver claramente na Figura 5.8, ele não segue a mesma seqüência de funções. Assim,

$$\begin{aligned}
&\text{Criptografia} && A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0} \\
&\text{Decriptografia:} && A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}
\end{aligned}$$

Do ponto de vista da implementação, seria desejável que a função de decriptografia seguisse a mesma seqüência de funções da criptografia. Isso permite que o algoritmo de decriptografia seja implementado da mesma maneira que o algoritmo de criptografia, criando oportunidades para eficiência.

Observe que, se fôssemos capazes de trocar a segunda e a terceira funções, a quarta e a quinta funções e a sexta e a sétima funções na seqüência de decriptografia, teríamos a mesma estrutura do algoritmo de criptografia. Vejamos

se isso é possível. Primeiro, considere a troca de INS e ISR. Dado um estado N consistindo nos nibbles (N_0, N_1, N_2, N_3), a transformação INS(ISR(N)) prossegue da seguinte forma:

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} N_0 & N_2 \\ N_3 & N_1 \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_3] & \mathrm{IS}[N_1] \end{pmatrix}$$

Onde IS refere-se à caixa-S inversa. Invertendo as operações, a transformação ISR(INS(N)) prossegue da seguinte forma:

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_1] & \mathrm{IS}[N_3] \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_3] & \mathrm{IS}[N_1] \end{pmatrix}$$

que é o mesmo resultado. Assim, INS(ISR(N)) = ISR(INS(N)).

Agora, considere a operação embaralhamento inverso de colunas, seguida pela de adição de chave: IMC([+](N)), onde a chave da rodada K_1 consiste nos nibbles ($k_{0,0}$, $k_{1,0}$, $k_{0,1}$, $k_{1,1}$). Então:

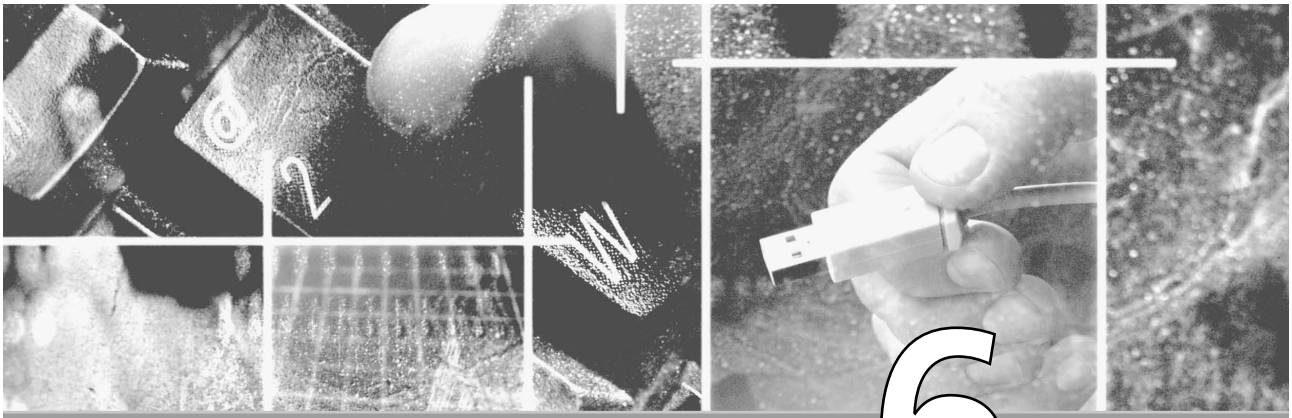
$$\begin{aligned}
\begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\left(\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}\right) &= \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} \oplus N_0 & k_{0,1} \oplus N_2 \\ k_{1,0} \oplus N_1 & k_{1,1} \oplus N_3 \end{pmatrix} \\
&= \begin{pmatrix} 9(k_{0,0} \oplus N_0) \oplus 2(K_{1,0} \oplus N_1) & 9(k_{0,1} \oplus N_2) \oplus 2(K_{1,1} \oplus N_3) \\ 2(k_{0,0} \oplus N_0) \oplus 9(K_{1,0} \oplus N_1) & 2(k_{0,1} \oplus N_2) \oplus 9(K_{1,1} \oplus N_3) \end{pmatrix} \\
&= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) \oplus (9N_0 \oplus 2N_1) & (9k_{0,1} \oplus 2k_{1,1}) \oplus (9N_2 \oplus 2N_3) \\ (2k_{0,0} \oplus 9k_{1,0}) \oplus (2N_0 \oplus 9N_1) & (2k_{0,1} \oplus 9k_{1,1}) \oplus (2N_2 \oplus 9N_3) \end{pmatrix} \\
&= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) & (9k_{0,1} \oplus 2k_{1,1}) \\ (2k_{0,0} \oplus 9k_{1,0}) & (2k_{0,1} \oplus 9k_{1,1}) \end{pmatrix} \oplus \begin{pmatrix} (9N_0 \oplus 2N_1) & (9N_2 \oplus 2N_3) \\ (2N_0 \oplus 9N_1) & (2N_2 \oplus 9N_3) \end{pmatrix} \\
&= \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}
\end{aligned}$$

Todas as etapas acima utilizam as propriedades da aritmética de corpo finito. O resultado é que $\mathrm{IMC}(\mathrm{A}_{K_1}(N)) = \mathrm{IMC}(K_1) \oplus \mathrm{IMC}(N)$. Agora, vamos definir a chave da rodada inversa para a rodada 1 como sendo $\mathrm{IMC}(K_1)$ e a operação de adição inversa de chave IA_{K_1} como sendo o XOR bit a bit da chave da rodada inversa com o vetor de estado. Então, temos $\mathrm{IMC}(\mathrm{A}_{K_1}(N)) = \mathrm{IA}_{K_1}(\mathrm{IMC}(N))$. Como resultado, podemos escrever o seguinte:

Criptografia:	$\mathrm{A}_{K_2} \circ \mathrm{SR} \circ \mathrm{NS} \circ \mathrm{A}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ \mathrm{A}_{K_0}$
Decriptografia:	$\mathrm{A}_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ \mathrm{A}_{K_1} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{A}_{K_2}$
Decriptografia:	$\mathrm{A}_{K_0} \circ \mathrm{ISR} \circ \mathrm{INS} \circ \mathrm{A}_{\mathrm{IMC}(K_1)} \circ \mathrm{IMC} \circ \mathrm{ISR} \circ \mathrm{INS} \circ \mathrm{A}_{K_2}$

Tanto a criptografia quanto a decriptografia agora seguem a mesma seqüência. Observe que essa derivação não funcionaria de modo tão eficaz se a rodada 2 do algoritmo de criptografia incluísse a função MC. Nesse caso, teríamos

Criptografia:	$\mathrm{A}_{K_2} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ \mathrm{A}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ \mathrm{A}_{K_0}$
Decriptografia:	$\mathrm{A}_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ \mathrm{A}_{K_1} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ \mathrm{A}_{K_2}$



CAPÍTULO 6

Mais sobre cifras simétricas

6.1 **Criptografia múltipla e DES triplo**

6.2 **Modos de operação de cifra de bloco**

6.3 **Cifras de fluxo e RC4**

6.4 **Leitura e sites Web recomendados**

6.5 **Principais termos, perguntas de revisão e problemas**

"Estou familiarizado com todas as formas de escritas secretas, sendo eu mesmo o autor de uma monografia superficial sobre o assunto, em que analiso 160 cifras diferentes", disse Holmes.

The adventure of the dancing men, Sir Arthur Conan Doyle

PONTOS PRINCIPAIS

- A criptografia múltipla é uma técnica em que um algoritmo criptográfico é usado repetidas vezes. Na primeira instância, um texto claro é convertido em texto cifrado usando-se o algoritmo criptográfico. Esse texto cifrado é então usado como entrada e o algoritmo é aplicado novamente. Esse processo pode ser repetido por qualquer número de estágios.
- O DES triplo utiliza por três vezes o algoritmo DES, usando um total de duas ou três chaves distintas.
- Um modo de operação é uma técnica para aprimorar o efeito de um algoritmo criptográfico ou adaptar o algoritmo para uma determinada aplicação, como aplicar uma cifra de bloco a uma seqüência de blocos de dados ou a um fluxo de dados.
- Cinco modos de operação foram padronizados para uso com cifras de bloco simétricas, como DES e AES: codebook eletrônico, encadeamento de blocos cifrados, realimentação de cifra, realimentação de saída e modo contador.
- Uma cifra de fluxo é um algoritmo criptográfico simétrico em que a saída de texto cifrado é produzida bit a bit ou byte a byte com base em um fluxo de entrada em texto claro. A cifra desse tipo mais utilizada atualmente é o RC4.

Este capítulo continua nossa explicação sobre cifras simétricas. Começaremos com o tópico da criptografia múltipla, examinando em particular o esquema de criptografia múltipla mais utilizado: o DES triplo.

Em seguida, passaremos para o assunto de modos de operação de cifras de bloco. Mostramos que existem diversas maneiras diferentes de aplicar uma cifra de bloco ao texto claro, cada uma com suas próprias vantagens e destinadas a um conjunto de aplicações em particular.

Finalmente, passaremos à explicação de cifras de fluxo simétricas, que diferem de forma significativa das cifras de bloco simétricas. Também examinaremos a cifra mais importante dessa família na atualidade, o RC4.

6.1 CRIPTOGRAFIA MÚLTIPLA E DES TRIPLO

Dada a vulnerabilidade em potencial do DES a um ataque por força bruta, tem havido um grande interesse na busca por uma alternativa. Uma possível abordagem é projetar um algoritmo completamente novo, sendo que o AES é o principal exemplo. Outra opção, que preservaria o investimento existente em software e equipamentos, é usar a criptografia múltipla com DES e chaves múltiplas. Começamos examinando o exemplo mais simples dessa segunda opção. Depois, examinaremos a estratégia bastante aceita de utilização do DES triplo (3DES).

DES duplo

A forma mais simples de criptografia múltipla consiste em dois estágios de criptografia e no uso de duas chaves (Figura 6.1a). Dado o texto claro P e duas chaves criptográficas, K_1 e K_2, o texto cifrado C é gerado da seguinte forma:

$$C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$$

A decriptografia exige que as chaves sejam aplicadas em ordem reversa:

$$P = \mathrm{D}(K_1, \mathrm{D}(K_2, C))$$

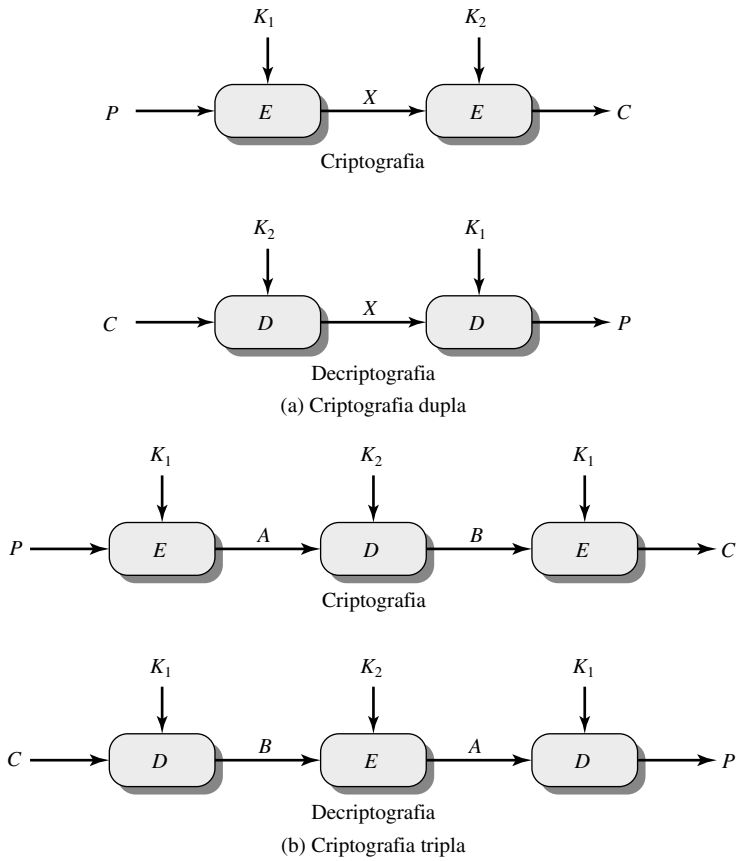


Figura 6.1 Criptografia múltipla.

Para o DES, esse esquema aparentemente envolve um tamanho de chave de $56 \times 2 = 112$ bits, resultando em um aumento bastante considerável em sua resistência a ataques. Entretanto, é necessário examinar o algoritmo mais de perto.

Redução a um único estágio Suponha que, para todos os valores de chave de 56 bits do DES, dadas duas chaves quaisquer, K_1 e K_2, fosse possível encontrar uma chave K_3, de modo que

$$\mathrm{E}(K_2, \mathrm{E}(K_1, P)) = \mathrm{E}(K_3, P) \tag{6.1}$$

Se fosse esse o caso, então a criptografia dupla, e na verdade qualquer número de estágios da criptografia múltipla com DES, seria inútil, pois o resultado seria equivalente a uma criptografia única com uma única chave de 56 bits.

Tendo isso em vista, não parece que a Equação (6.1) seja válida. Considere que a criptografia com DES seja uma transformação de blocos de 64 bits em blocos de 64 bits. De fato, o mapeamento pode ser visto como uma permutação, ou seja, se considerarmos todos os 2^{64} blocos de entrada possíveis, a criptografia usando DES com uma chave em particular irá mapear cada bloco em um bloco de 64 bits específico. Caso contrário, se, digamos, dois blocos de entrada fossem mapeados para o mesmo bloco de saída, então a decriptografia para recuperar o texto claro original seria impossível. Com 2^{64} entradas possíveis, quantos mapeamentos diferentes existem que geram uma permutação dos blocos de entrada? O valor pode ser facilmente calculado como sendo

$$\left(2^{64}\right)! = 10^{34738000000000000000} > \left(10^{10^{20}}\right)$$

Por outro lado, o DES define um mapeamento para cada chave diferente, com um número total de mapeamentos:

$$2^{56} < 10^{17}$$

Portanto, é razoável considerar que, se DES for usado duas vezes com chaves diferentes, ele produzirá um dos muitos mapeamentos que não são contemplados por uma única aplicação do DES. Embora houvesse muitas evidências apoiando essa suposição, apenas em 1992 isso foi comprovado [CAMP92].

Ataque meet-in-the-middle Assim, o uso do DES duplo resulta em um mapeamento que não é equivalente a uma única criptografia usando o DES simples. Entretanto, existe um modo de atacar esse esquema que não depende de qualquer propriedade específica do DES, mas que funcionará em qualquer cifra de criptografia de bloco.

O algoritmo, conhecido como ataque meet-in-the-middle (encontro no meio), foi descrito inicialmente em [DIFF77]. Ele é baseado na observação de que, se temos

$$C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$$

então (ver Figura 6.1a)

$$X = \mathrm{E}(K_1, P) = \mathrm{D}(K_2, C)$$

Dado um par conhecido, (P, C), o ataque prossegue da seguinte maneira. Primeiro, criptografe P para todos os 2^{56} valores possíveis de K_1. Armazene esses resultados em uma tabela e organize-a de acordo como os valores de X. Em seguida, decriptografe C usando todos os 2^{56} valores possíveis de K_2. À medida que cada texto decriptografado for gerado, compare o resultado com a tabela, em busca de uma correspondência a esse valor. Se houver uma correspondência, então compare as duas chaves resultantes com um novo par de texto claro/texto cifrado conhecido. Se as duas chaves produzirem o texto cifrado correto, aceite-as como sendo as chaves corretas.

Para qualquer texto claro P, existem 2^{64} valores de texto cifrado possíveis que poderiam ser produzidos pelo DES duplo. Por conseguinte, o DES duplo utiliza uma chave de 112 bits, de modo que existem 2^{112} chaves possíveis. Portanto, em média, para determinado texto claro P, o número de chaves diferentes de 112 bits que produzirão determinado texto cifrado C é $2^{112}/2^{64} = 2^{48}$. Assim, o procedimento anterior produzirá cerca de 2^{48} alarmes falsos no primeiro par (P, C). Um argumento semelhante indica que, com 64 bits adicionais de texto claro e texto cifrado conhecidos, a taxa de alarmes falsos é reduzida para $2^{48-64} = 2^{-16}$. Em outras palavras, se o ataque do encontro no meio for realizado sobre dois blocos de texto claro/texto cifrado conhecidos, a probabilidade de que as chaves corretas sejam determinadas é de $1 - 2^{-16}$. O resultado é que um ataque de texto claro conhecido terá sucesso contra o DES duplo, que tem um tamanho de chave de 112 bits, com um esforço da ordem de 2^{56}, não muito mais do que os 2^{55} exigidos para o DES simples.

DES triplo com duas chaves

Uma contramedida óbvia para o ataque do encontro no meio é usar três estágios de criptografia com três chaves diferentes. Isso aumenta o custo do ataque de texto claro conhecido para 2^{112}, que está além do praticável atualmente ou mesmo num futuro distante. Porém, isso tem a desvantagem de exigir um tamanho de chave de $56 \times 3 = 168$ bits, o que pode ser muito pesado.

Como alternativa, Tuchman propôs um método de criptografia triplo, que utiliza apenas duas chaves [TUCH79]. A função segue uma seqüência criptografar-decriptografar-criptografar (Figura 6.1b).

$$C = \mathrm{E}(K_1, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

Não existem razões de ordem criptográfica para o uso da decriptografia no segundo estágio. Sua única vantagem é permitir que os usuários do 3DES decriptografem dados criptografados pelos usuários do DES simples mais antigo:

$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}(K_1, P)$$

3DES com duas chaves é uma alternativa relativamente popular ao DES, e tem sido adotada para uso nos padrões de gerenciamento de chave ANS X9.17 e ISO 8732.[1]

Atualmente, não existem ataques criptoanalíticos práticos contra o 3DES. Coppersmith [COPP94] observa que o custo de uma busca de chave por força bruta no 3DES é da ordem de $2^{112} \approx (5 \times 10^{33})$, e estima que o custo da criptoanálise diferencial sofre um crescimento exponencial, em comparação com o DES simples, ultrapassando 10^{52}.

Vale a pena examinar diversos ataques propostos sobre o 3DES que, embora não sendo práticos, dão uma idéia dos tipos de ataques que foram considerados e que poderiam formar a base para ataques futuros mais eficientes.

A primeira proposta séria veio de Merkle e Hellman [MERK81]. Sua estratégia consiste em encontrar os valores de texto claro que produzem um primeiro valor intermediário de $A = 0$ (Figura 6.1b) e depois usar o ataque do encontro no meio para determinar as duas chaves. O nível de esforço é 2^{56}, mas a técnica exige 2^{56} pares de texto claro/texto cifrado escolhidos, um número que provavelmente não será fornecido pelo dono das chaves.

Um ataque de texto claro conhecido é esboçado em [VANO90]. Esse método é uma melhoria em relação à técnica de texto claro escolhido, mas exige mais esforço. O ataque é baseado na observação de que, se conhecermos A e C (Figura 6.1b), então o problema se reduz ao de um ataque sobre o DES duplo. Naturalmente, o atacante não conhece A, mesmo que P e C sejam conhecidos, desde que as duas chaves sejam desconhecidas. Porém, o atacante pode escolher um valor em potencial de A e depois tentar encontrar um par (P, C) conhecido que produza A. O ataque prossegue da seguinte maneira:

1. Obtenha pares $n(P,C)$. Esse é o texto claro conhecido. Coloque-os em uma tabela (Tabela 1) organizada de acordo com os valores de P (Figura 6.2b).
2. Escolha um valor qualquer a para A, e crie uma segunda tabela (Figura 6.2c) com entradas definidas da seguinte maneira. Para cada uma das 2^{56} chaves possíveis $K_1 = i$, calcule o valor de texto claro P_i que produz a:

   $$P_i = \mathrm{D}(i, a)$$

   Para cada P_i que corresponde a uma entrada na Tabela 1, crie uma entrada na Tabela 2 consistindo no valor K_1 e no valor de B que é produzido para o par (P, C) da Tabela 1, considerando esse valor de K_1:

   $$B = \mathrm{D}(i, C)$$

   Ao final dessa etapa, organize a Tabela 2 de acordo com os valores de B.
3. Agora temos uma série de valores candidatos de K_1 na Tabela 2 e estamos em posição de procurar um valor de K_2. Para cada uma das 2^{56} chaves possíveis $K_2 = j$, calcule o segundo valor intermediário para nosso valor escolhido de a:

   $$B_j = \mathrm{D}(j, a)$$

1. (ANS) American National Standard: Financial Institution Key Management (Wholesale). Por seu título, o X9.17 parece ser um padrão um tanto vago. Mesmo assim, diversas técnicas especificadas nesse padrão têm sido adotadas para uso em outros padrões e aplicações, conforme veremos no decorrer deste livro.

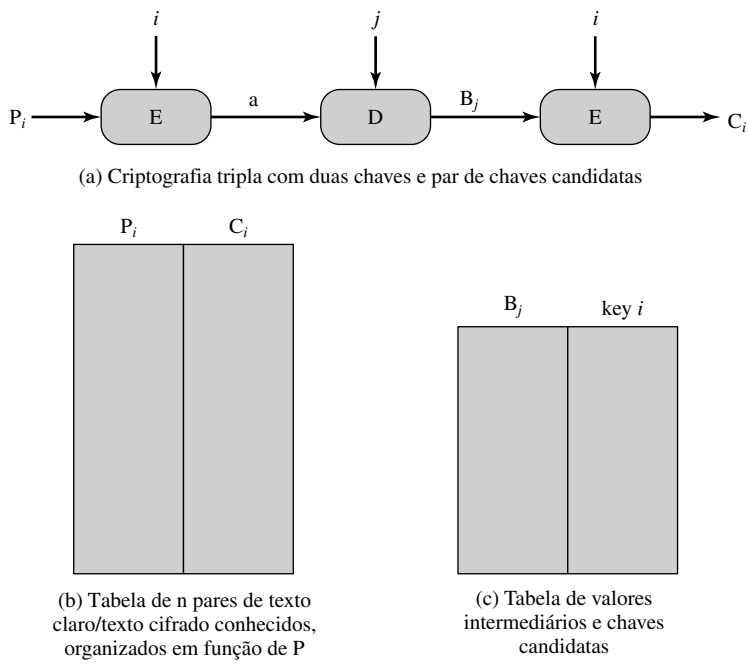


Figura 6.2 Ataque de texto claro conhecido sobre DES triplo.

A cada passo, procure por B_j na Tabela 2. Se houver uma combinação, então a chave correspondente i da Tabela 2 mais esse valor de j são os valores candidatos para as chaves desconhecidas (K_1, K_2). Por quê? Porque descobrimos um par de chaves (i, j) que produzem um par conhecido (P, C) (Figura 6.2a).

4. Teste cada par de chaves candidatas (i, j) em alguns outros pares de texto claro/texto cifrado. Se um par de chaves produzir o texto cifrado desejado, a tarefa está terminada. Se nenhum par tiver sucesso, repita o processo a partir da etapa 1 com um novo valor de a.

Para determinado par (P, C) conhecido, a probabilidade de selecionar o único valor de a que leva ao sucesso é de $1/2^{64}$. Assim, dados n pares (P, C), a probabilidade de sucesso para um único valor selecionado de a é $n/2^{64}$. Um resultado básico da teoria da probabilidade é que o número esperado de tentativas exigidas para retirar uma bola vermelha de uma urna contendo n bolas vermelhas e $N - n$ bolas verdes é $(N + 1)/(n + 1)$ se as bolas não forem substituídas. Assim, o número esperado de valores de a que precisam ser testados é, para um n grande,

$$\frac{2^{64} + 1}{n + 1} \approx \frac{2^{64}}{n}$$

Desse modo, o tempo de execução esperado do ataque é da ordem de

$$\left(2^{56}\right)\frac{2^{64}}{n} = 2^{120-\log_2 n}$$

DES triplo com três chaves

Embora os ataques descritos pareçam ser impraticáveis, qualquer um, usando o 3DES com duas chaves, poderá experimentar uma certa preocupação. Por isso, muitos pesquisadores agora acham que o 3DES com três chaves é a alternativa preferida (por exemplo, [KALI96a]). O 3DES com três chaves possui um tamanho de chave efetivo de 168 bits e é definido da seguinte maneira:

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

A compatibilidade com o DES é fornecida colocando-se $K_3 = K_2$ ou $K_1 = K_2$.

Diversas aplicações baseadas na Internet adotaram o 3DES com três chaves, incluindo PGP e S/MIME, ambos explicados no Capítulo 15.

6.2 MODOS DE OPERAÇÃO DE CIFRA DE BLOCO

Um algoritmo de cifra de bloco é um elemento básico para fornecer segurança aos dados. Para aplicar uma cifra de bloco em diversas aplicações, quatro 'modos de operação' foram definidos pelo NIST (FIPS 81). Basicamente, um modo de operação é uma técnica para aprimorar o efeito de um algoritmo criptográfico ou adaptar o algoritmo para uma aplicação, como a de uma cifra de bloco a uma seqüência de blocos de dados ou fluxo de dados. Os quatro modos abrangem praticamente todas as aplicações possíveis da criptografia, para as quais uma cifra de bloco poderia ser usada. À medida que novas aplicações e requisitos apareceram, o NIST expandiu a lista dos modos recomendados a cinco na Special Publication 800-38A. Esses modos são usados com qualquer cifra de bloco simétrica, incluindo DES triplo e AES. Os modos estão resumidos na Tabela 6.1 e descritos rapidamente no restante desta seção.

Modo Electronic Codebook

O modo mais simples é o Electronic Codebook (ECB), em que o texto claro é tratado um bloco de cada vez e cada bloco de texto claro é criptografado usando-se a mesma chave (Figura 6.3). O termo *codebook* é usado porque, para uma determinada chave, existe um texto cifrado exclusivo para cada bloco de b bits de texto claro. Portanto, podemos imaginar uma lista de códigos gigantesca, em que existe uma entrada para cada texto claro de b bits possível, mostrando seu texto cifrado correspondente.

Para uma mensagem maior do que b bits, o procedimento é simplesmente desmembrar a mensagem em blocos de b bits, complementando o último bloco, se necessário. A decriptografia é realizada sobre um bloco de cada vez, sempre usando-se a mesma chave. Na Figura 6.3, o texto claro (preenchido conforme a necessidade) consiste em uma seqüência de blocos de b bits, $P_1, P_2, \ldots, P_N$; a seqüência correspondente de blocos de texto cifrado é $C_1, C_2, \ldots, C_N$.

O método ECB é ideal para uma pequena quantidade de dados, como uma chave criptográfica. Assim, se você quiser transmitir uma chave DES com segurança, o ECB é o modo apropriado.

A característica mais significativa do ECB é que um mesmo bloco de b bits de texto claro, caso apareça mais de uma vez na mensagem, sempre produzirá o mesmo texto cifrado.

Para mensagens mais longas, o modo ECB pode não ser seguro. Se a mensagem for altamente estruturada, pode ser possível que um criptoanalista explore essas regularidades. Por exemplo, se for sabido que a mensagem sempre começa com certos campos predefinidos, então o criptoanalista pode ter diversos pares de texto claro/texto cifrado conhecido para trabalhar. Se a mensagem tiver elementos repetitivos, com um período de repetição múltiplo de b bits, então esses elementos podem ser identificados pelo analista. Isso pode ajudar na análise ou pode oferecer uma oportunidade para substituir ou reorganizar os blocos.

Tabela 6.1 Modos de operação de cifra de bloco.

Modo	Descrição	Aplicação típica
Electronic Codebook (ECB)	Cada bloco de 64 bits de texto claro é codificado independentemente, usando-se a mesma chave.	• Transmissão segura de valores isolados (por exemplo, uma chave criptográfica)
Cipher Block Chaining (CBC)	A entrada do algoritmo criptográfico é o XOR dos próximos 64 bits de texto claro e os 64 bits anteriores de texto crifrado.	• Transmissão de uso geral orientada a bloco • Autenticação
Cipher Feedback (CFB)	A entrada é processada j bits de cada vez. O texto crifrado anterior é usado como entrada para o algoritmo criptográfico de forma a produzir uma saída pseudo-aleatória, que é XORada com o texto claro para produzir a próxima unidade de texto cifrado.	• Transmissão de uso geral orientada a fluxo • Autenticação
Output Feedback (OFB)	Semelhante ao CFB, exceto que a entrada do algoritmo criptográfico é a saída DES anterior.	• Transmissão orientada a fluxo por canal com ruído (por exemplo, comunicação por satélite)
Counter (CTR)	Cada bloco de texto claro é XORado com um contador criptografado. O contador é incrementado para cada bloco subseqüente.	• Transmissão orientada a bloco de uso geral • Útil para requisitos de alta velocidade

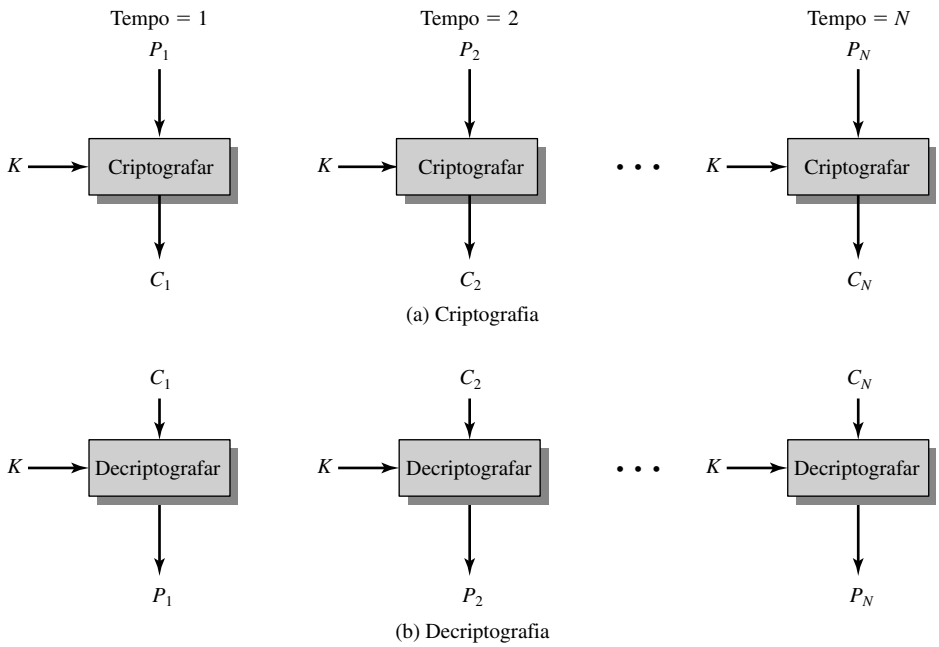


Figura 6.3 Modo Electronic Codebook (ECB).

Modo Cipher Block Chaining

Para contornar as deficiências de segurança do ECB, precisaríamos de uma técnica em que o mesmo bloco de texto claro, se repetido, produzisse diferentes blocos de texto cifrado. Um modo simples de satisfazer esse requisito é o modo de encadeamento de blocos cifrados (CBC — Cipher Block Chaining) (Figura 6.4). Nesse esquema, a entrada do algoritmo criptográfico é o XOR do bloco de texto claro atual e o bloco de texto cifrado anterior; a mesma chave é usada para cada bloco. Dessa maneira, encadeamos o processamento da seqüência de blocos de texto claro. A entrada da função de criptografia para cada bloco de texto claro não possui qualquer relacionamento fixo com o bloco de texto claro; portanto, padrões repetitivos de *b* bits não são expostos.

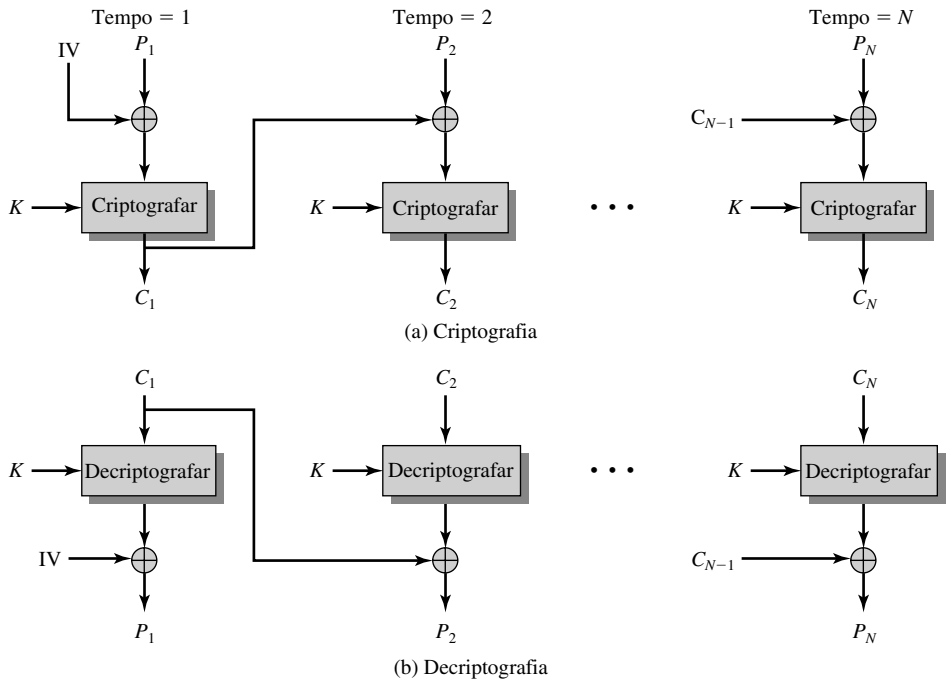


Figura 6.4 Modo Cipher Block Chaining (CBC).

Para a decriptografia, cada bloco de cifra é passado pelo algoritmo de decriptografia. O resultado é XORado com o bloco de texto cifrado anterior para produzir o bloco de texto claro. Para ver se isso funciona, podemos escrever

$$C_j = \mathrm{E}(K, [C_{j-1} \oplus P_j])$$

Então,

$$\begin{aligned} \mathrm{D}(K, C_j) &= \mathrm{D}(K, \mathrm{E}(K, [C_{j-1} \oplus P_j])) \\ \mathrm{D}(K, C_j) &= C_{j-1} \oplus P_j \\ C_{j-1} \oplus \mathrm{D}(K, C_j) &= C_{j-1} \oplus C_{j-1} \oplus P_j = P_j \end{aligned}$$

Para produzir o primeiro bloco de texto cifrado, um vetor de inicialização (IV) passa por um XOR com o primeiro bloco de texto claro. Na decriptografia, o IV passa por um XOR com a saída do algoritmo de decriptografia para recuperar o primeiro bloco de texto claro. O IV é um bloco de dados que tem o mesmo tamanho do bloco de cifra.

O IV precisa ser conhecido pelo emissor e pelo receptor, mas não previsível por um terceiro. Para obter o máximo de segurança, o IV deve ser protegido contra mudanças não autorizadas. Isso poderia ser feito pelo envio do IV por meio da criptografia ECB. Um motivo para proteger o IV é o seguinte: se um oponente for capaz de enganar o receptor, convencendo-o a usar um valor diferente para o IV, então o oponente será capaz de inverter os bits selecionados no primeiro bloco de texto claro. Para ver isso, considere o seguinte:

$$\begin{aligned} C_1 &= \mathrm{E}(K, [\mathrm{IV} \oplus P_1]) \\ P_1 &= \mathrm{IV} \oplus \mathrm{D}(K, C_1) \end{aligned}$$

Agora use a notação em que X[i] indica o i-ésimo bit da quantidade de b bits X. Então,

$$P_1[i] = \mathrm{IV}[i] \oplus \mathrm{D}(K, C_1)[i]$$

Depois, usando as propriedades de XOR, podemos dizer que

$$P_1[i]' = \mathrm{IV}[i]' \oplus \mathrm{D}(K, C_1)[i]$$

onde a notação de linha (′) indica complementação de bit. Isso significa que, se um oponente puder mudar previsivelmente os bits em IV, os bits correspondentes do valor recebido de P_1 podem ser mudados.

Para saber mais informações sobre outros ataques possíveis com base no conhecimento do IV, consulte [VOYD83].

Concluindo, em razão do mecanismo de encadeamento do CBC, esse é um modo apropriado para criptografar mensagens de tamanho maiores que b bits.

Além do seu uso para conseguir a confidencialidade, o modo CBC pode ser usado para autenticação. Esse uso é descrito na Parte Dois.

Modo Cipher Feedback

O algoritmo DES é basicamente uma cifra de bloco que utiliza blocos de b bits. Porém, é possível converter o DES em uma cifra de fluxo, usando o modo Cipher Feedback (CFB) ou Output Feedback. Uma cifra de fluxo elimina a necessidade de preencher uma mensagem para que ela apresente um número inteiro de blocos. Ela também pode operar em tempo real. Assim, se um fluxo de caracteres estiver sendo transmitido, cada caractere pode ser criptografado e transmitido imediatamente usando-se uma cifra de fluxo orientada a caractere.

Uma propriedade desejável de uma cifra de fluxo é que o texto cifrado tenha o mesmo tamanho do texto claro. Assim, se os caracteres de 8 bits estiverem sendo transmitidos, cada caractere deve ser criptografado para produzir uma saída de texto cifrado de 8 bits. Se mais de 8 bits forem produzidos, a capacidade de transmissão será desperdiçada.

A Figura 6.5 representa o esquema CFB. Na figura, considera-se que a unidade de transmissão é de s bits; um valor comum é $s = 8$. Assim como o CBC, as unidades de texto claro são encadeadas, de modo que o texto cifrado de qualquer unidade de texto claro é uma função de todo o texto claro anterior. Nesse caso, em vez de unidades de b bits, o texto claro é dividido em *segmentos* de s bits.

Primeiro, considere a criptografia. A entrada da função de criptografia é um registrador de deslocamento de b bits, que é definido inicialmente com algum vetor de inicialização (IV). Os s bits mais à esquerda (mais significativos) da saída da função de criptografia passam por um XOR com o primeiro segmento de texto claro para produzir a

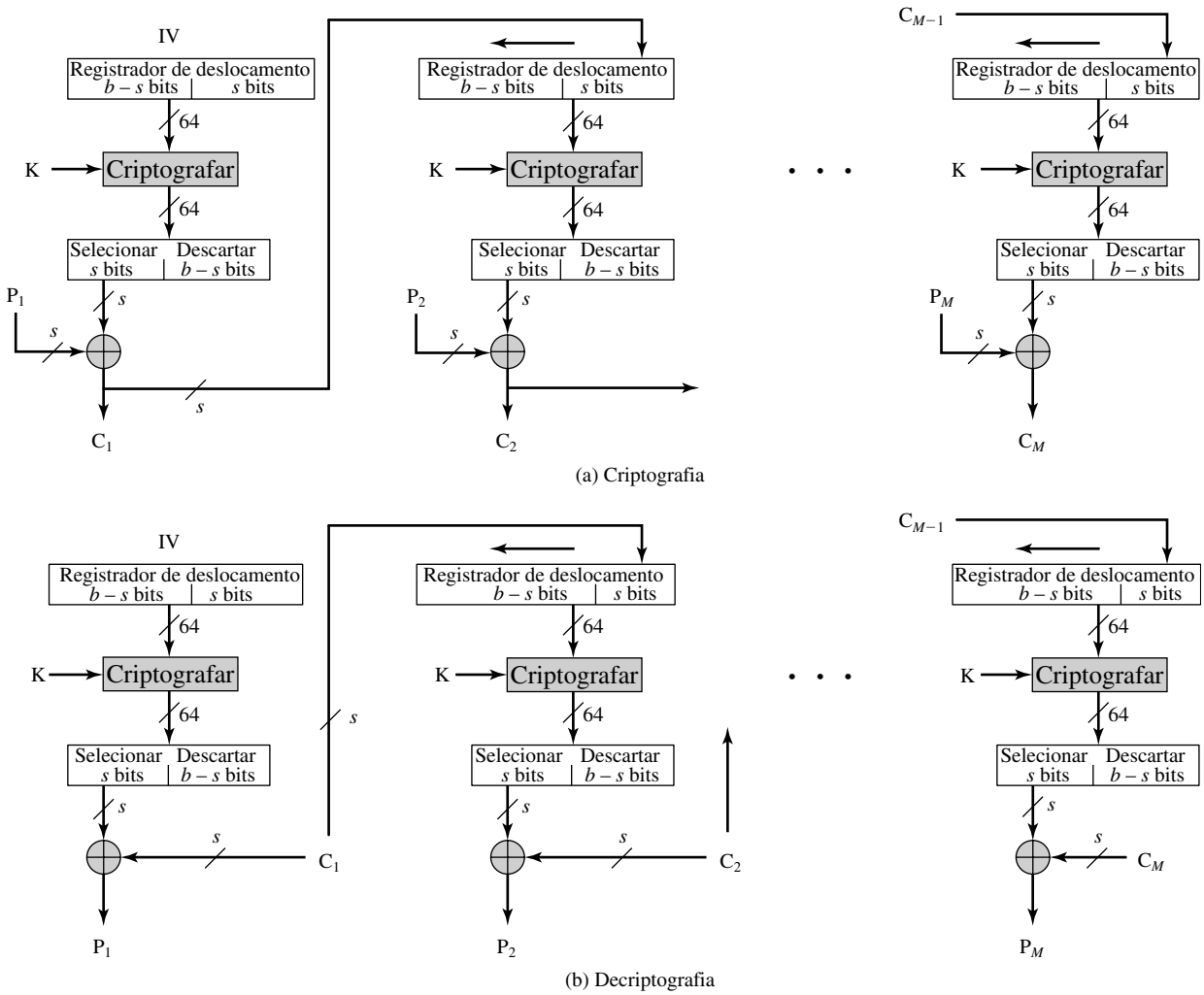


Figura 6.5 Modo Cipher Feedback (CFB) com *s* bits.

primeira unidade de texto cifrado C_1, que é então transmitida. Além disso, o conteúdo do registrador de deslocamento é deslocado para a esquerda de *s* bits e C_1 é colocado nos *s* bits mais à direita (menos significativos) do registrador de deslocamento. Esse processo continua até que todas as unidades de texto claro tenham sido criptografadas.

Para a decriptografia, o mesmo esquema é usado, exceto que a unidade de texto cifrado recebida passa por um XOR com a saída da função de criptografia para produzir a unidade de texto claro. Observe que é a função de *criptografia* que é utilizada, e não a função de decriptografia. Isso pode ser explicado com facilidade. Considere que $S_S(X)$ seja definido como os *s* bits mais significativos de *X*. Então,

$$C_1 = P_1 \oplus S_s[\mathrm{E}(K, \mathrm{IV})]$$

Portanto,

$$P_1 = C_1 \oplus S_s[\mathrm{E}(K, \mathrm{IV})]$$

O mesmo raciocínio é aplicado a etapas subseqüentes do processo.

Modo Output Feedback

O modo Output Feedback (OFB) é semelhante em estrutura ao CFB, conforme ilustramos na Figura 6.6. Como podemos ver, é a saída da função de criptografia que é realimentada no registrador de deslocamento no OFB, enquanto, no CFB, a unidade de texto cifrado é realimentada no registrador de deslocamento.

Uma vantagem do método OFB é que os erros de bit na transmissão não se propagam. Por exemplo, se acontecer um erro de bit em C_1, somente o valor recuperado de P_1 será afetado; as unidades de texto claro subseqüentes não serão modificadas. Com o CFB, C_1 serve como entrada para o registrador de deslocamento e, portanto, causa uma modificação adicional mais adiante.

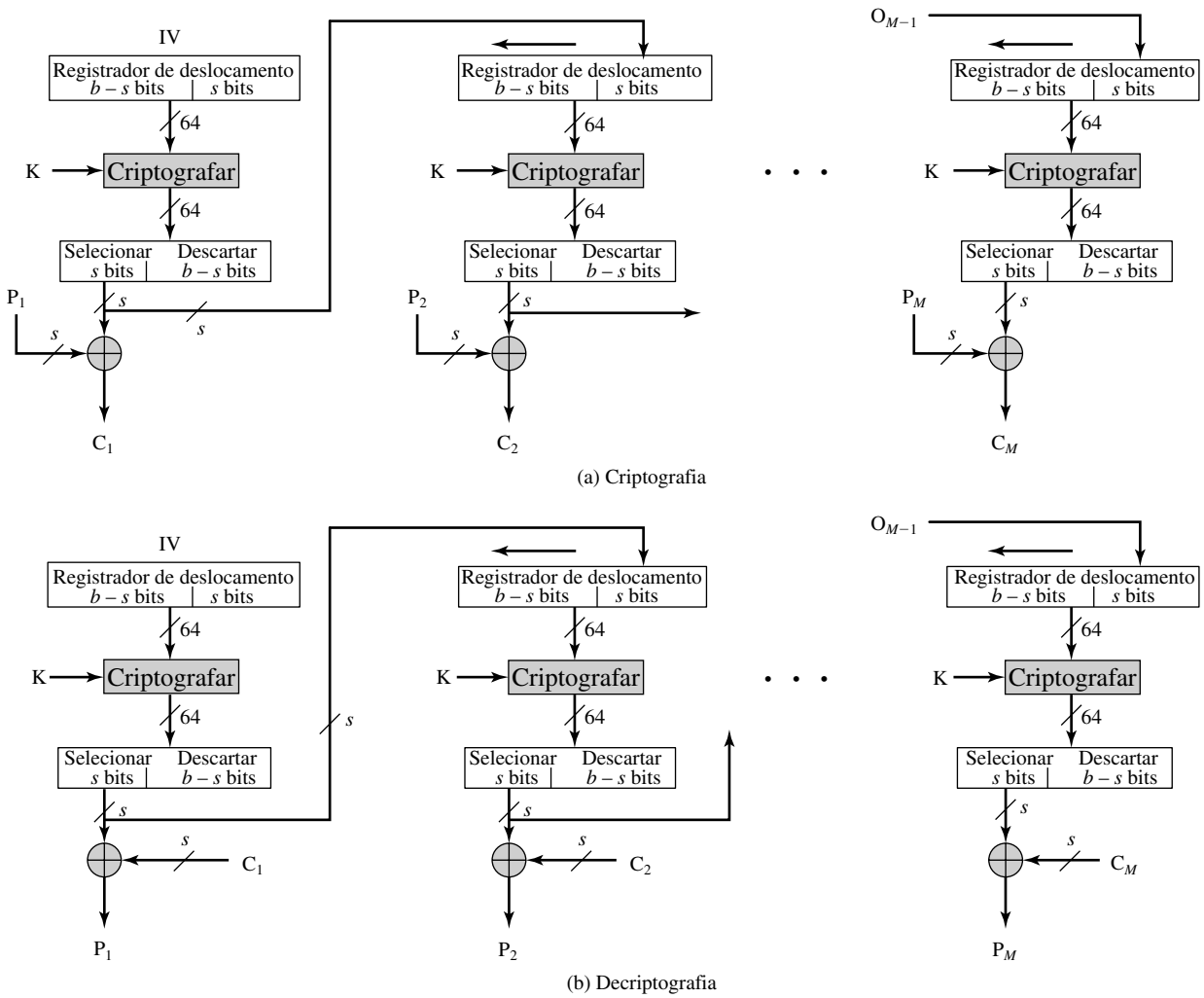


Figura 6.6 Modo Output Feedback (OFB) com *s* bits.

A desvantagem do OFB é que ele é mais vulnerável a um ataque por modificação de fluxo de mensagem do que o CFB. Considere que o fato de complementar um bit no texto cifrado complementa o bit correspondente no texto claro recuperado. Assim, podem ser feitas mudanças controladas no texto claro. Isso pode permitir que um oponente, fazendo as mudanças necessárias no campo de soma de verificação da mensagem, além da parte de dados, altere o texto cifrado de um modo que não seja detectado por um código de correção de erros. Para uma explicação mais profunda, consulte [VOYD83].

Modo Counter

Embora o interesse pelo modo counter (CTR) tenha aumentado recentemente, com aplicações na segurança de redes ATM (Asynchronous Transfer Mode) e IPSec (IP security), esse modo foi proposto há mais tempo (veja, por exemplo, [DIFF79]).

A Figura 6.7 representa o modo CTR. Usa-se um contador para geração de valores de tamanho igual àquele do bloco de texto claro. O único requisito indicado no SP 800-38A é que o valor do contador deve ser diferente para cada bloco de texto claro criptografado. Normalmente, o contador é inicializado com algum valor e depois incrementado em 1 (módulo 2^b, sendo b o tamanho do bloco) para cada bloco subseqüente. Na criptografia, o contador é inicialmente criptografado e depois passa por um XOR com o bloco de texto claro para produzir o bloco de texto cifrado; não existe encadeamento. Por exemplo, a mesma seqüência de valores de contador é utilizada, com cada contador criptografado passando por um XOR com um bloco de texto cifrado para recuperar o bloco de texto claro correspondente.

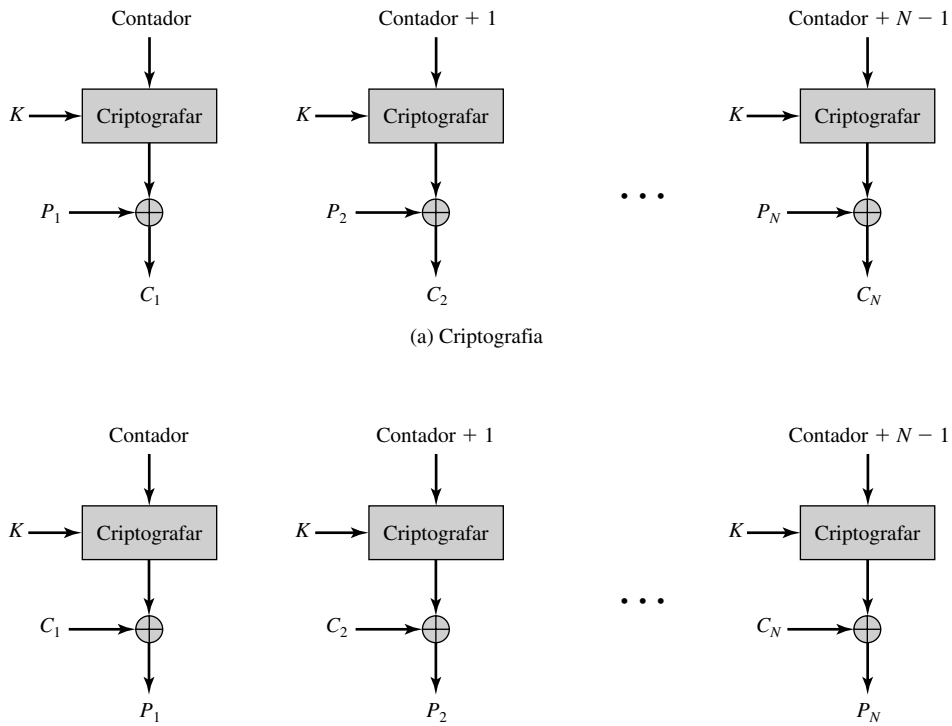


Figura 6.7 Modo Counter (CTR).

[LIPM00] relaciona as seguintes vantagens do modo CTR:

- **Eficiência do hardware:** Diferentemente dos três modos de encadeamento, a criptografia (ou decriptografia) no modo CTR pode ser feita em paralelo sobre múltiplos blocos de texto claro ou cifrado. Para os modos de encadeamento, o algoritmo precisa completar o cálculo em um bloco antes de passar para o bloco seguinte. Isso limita o throughput máximo do algoritmo para o inverso do tempo necessário para efetuar uma operação de criptografia ou decriptografia do bloco. No modo CTR, o throughput só é limitado pela quantidade de paralelismo alcançada.
- **Eficiência do software:** De modo semelhante, devido à possibilidade de execução paralela no modo CTR, os processadores que dão suporte aos recursos paralelos, como pipelining agressivo, despacho de múltiplas instruções por ciclo de clock, um grande número de registradores e instruções SIMD, podem ser utilizados de modo eficaz.
- **Pré-processamento:** A execução do algoritmo criptográfico básico não depende da entrada do texto claro ou do texto cifrado. Portanto, se memória suficiente estiver disponível e a segurança for mantida, o pré-processamento pode ser utilizado para preparar a saída das caixas de criptografia que alimentam as funções XOR na Figura 6.7. Quando a entrada em texto claro ou texto cifrado estiver presente, o único cálculo a ser efetuado será uma série de XORs. Essa estratégia melhora bastante o throughput.
- **Acesso aleatório:** O i-ésimo bloco de texto claro ou texto cifrado pode ser processado no modo padrão de acesso aleatório. Com os modos de encadeamento, o bloco C_i não pode ser calculado até que o $i - 1$ bloco anterior seja calculado. Pode haver aplicações em que um texto cifrado seja armazenado e seja desejável decriptografar apenas um bloco; para essas aplicações, o recurso de acesso aleatório é atraente.
- **Segurança demonstrável:** Pode-se demonstrar que o CTR é, pelo menos, tão seguro quanto os outros modos explicados nesta seção.
- **Simplicidade:** Diferentemente dos modos ECB e CBC, o modo CTR exige apenas a implementação do algoritmo de criptografia, e não do algoritmo de decriptografia. Isso importa mais quando o algoritmo de decriptografia difere substancialmente do de criptografia, como acontece com o AES. Além disso, o escalonamento das chaves de decriptografia não precisa ser implementado.

6.3 CIFRAS DE FLUXO E RC4

Nesta seção, examinaremos a cifra de fluxo simétrica provavelmente mais popular, RC4. Começaremos com uma visão geral da estrutura de cifras de fluxo, e depois examinaremos o RC4.

Estrutura de cifra de fluxo

A cifra de fluxo típica criptografa o texto claro um byte de cada vez, embora uma cifra de fluxo possa ser projetada para operar sobre um bit de cada vez ou sobre unidades maiores do que um byte de cada vez. A Figura 6.8 é um diagrama representativo da estrutura de cifras de fluxo. Nessa estrutura, uma chave é inserida em um gerador de bits pseudo-aleatório, que produz um fluxo de números de 8 bits que são aparentemente aleatórios. Explicamos os geradores de números pseudo-aleatórios no Capítulo 7. Por enquanto, simplesmente dizemos que um fluxo pseudo-aleatório é aquele que é imprevisível sem conhecimento da chave de entrada. A saída do gerador, chamada **fluxo de chave**, é combinada um byte de cada vez com o fluxo de texto claro, usando a operação XOR bit a bit. Por exemplo, se o próximo byte gerado pelo gerador for 01101100 e o próximo byte de texto claro for 11001100, então o byte de texto cifrado resultante será

```
  11001100   texto claro
⊕ 01101100   fluxo de chave
  ---------
  10100000   texto cifrado
```

A decriptografia requer o uso da mesma seqüência pseudo-aleatória:

```
  10100000   texto cifrado
⊕ 01101100   fluxo de chave
  ---------
  11001100   texto claro
```

A cifra de fluxo é semelhante ao one-time pad explicado no Capítulo 2. A diferença é que um one-time pad usa um fluxo de números aleatórios genuíno, enquanto uma cifra de fluxo usa um fluxo de números pseudo-aleatórios.

[KUMA97] relaciona as seguintes considerações de projeto importantes para uma cifra de fluxo:

1. A seqüência de criptografia deverá ter um período grande. Um gerador de números pseudo-aleatórios usa uma função que produz um fluxo determinístico de bits que se repete em algum ponto. Quanto maior o período de repetição, mais difícil será realizar a criptoanálise. Isso é basicamente a mesma consideração explicada com referência à cifra de Vigenère, na qual, quanto maior a palavra-chave, mais difícil é a criptoanálise.
2. O fluxo de chave deverá se aproximar ao máximo de um fluxo de números aleatórios verdadeiro. Por exemplo, deverá haver um número aproximadamente igual de 1s e 0s. Se o fluxo de chave for tratado como um fluxo de bytes, então todos os 256 valores de byte possíveis deverão aparecer aproximadamente com a mesma freqüência. Quanto mais aleatória for a aparência do fluxo de chave, mais aleatório será o texto cifrado, tornando a criptoanálise mais difícil.
3. Observe, pela Figura 6.8, que a saída do gerador de números pseudo-aleatórios é condicionada ao valor da chave de entrada. Para proteger-se contra ataques de força bruta, a chave precisa ser suficientemente longa. As mesmas considerações aplicadas a cifras de bloco são válidas aqui. Assim, com a tecnologia atual, é desejável um tamanho de chave de pelo menos 128 bits.

Com um gerador de números pseudo-aleatórios projetado corretamente, uma cifra de fluxo pode ser tão segura quanto a cifra de bloco de tamanho de chave comparável. A principal vantagem de uma cifra de fluxo é que ela é quase sempre mais rápida e utiliza muito menos código do que a cifra de bloco. O exemplo nesta seção, RC4, pode ser implementado com apenas algumas poucas linhas de código. A Tabela 6.2, usando dados de [RESC01], compara os tempos de execução do RC4 com três cifras de bloco simétricas bem conhecidas. A vantagem de uma cifra de bloco é que você pode reutilizar chaves. Porém, se dois textos claros forem criptografados com a mesma chave, usando uma cifra de fluxo, então a criptoanálise normalmente será bastante simples [DAWS96]. Se dois fluxos de texto cifrado passarem por uma operação XOR, o resultado será o XOR dos textos claros originais. Se os textos claros forem strings de texto, números de cartão de crédito ou outros fluxos de bytes com propriedades conhecidas, então a criptoanálise poderá ter sucesso.

Para aplicações que exigem criptografia/decriptografia de um fluxo de dados, como sobre um canal de comunicações ou um link de navegador/Web, uma cifra de fluxo pode ser a melhor alternativa. Para aplicações que lidam com blocos de dados, como transferência de arquivos, e-mail e banco de dados, as cifras de bloco podem ser mais apropriadas. Porém, qualquer tipo de cifra pode ser usado em praticamente qualquer aplicação.

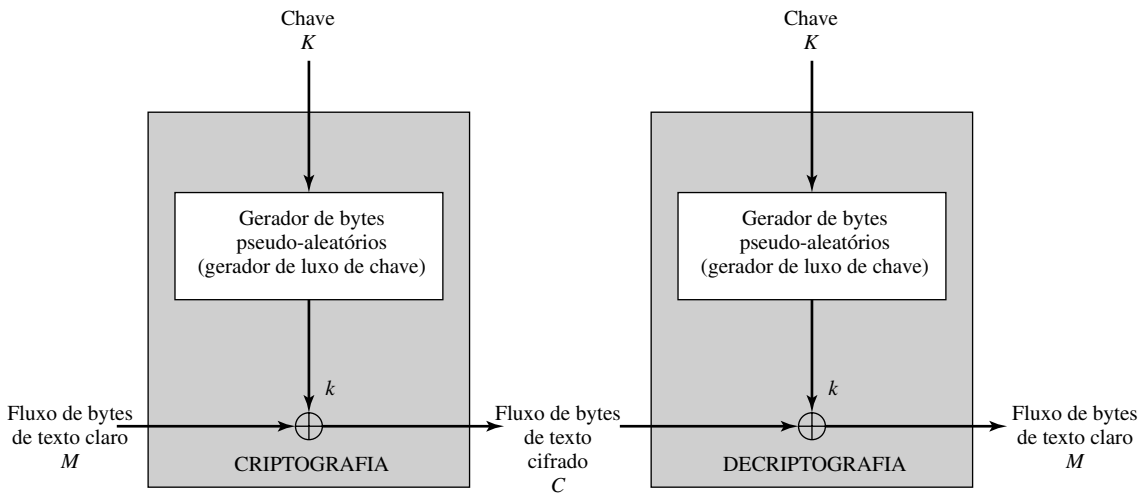


Figura 6.8 Diagrama de cifras de fluxo.

Tabela 6.2 Comparação das velocidades das cifras simétricas em um Pentium II.

Cifra	Tamanho de chave	Velocidade (Mbps)
DES	56	9
3DES	168	3
RC2	variável	0,9
RC4	variável	45

O algoritmo RC4

O RC4 é uma cifra de fluxo projetada em 1987 por Ron Rivest para a RSA Security. Ela é uma cifra de fluxo com tamanho de chave variável e operações orientadas a byte. O algoritmo é baseado no uso de uma permutação aleatória. A análise mostra que o período da cifra é muito provavelmente maior que 10^{100} [ROBS95a]. Oito a dezesseis operações de máquina são exigidas por byte de saída, e pode-se esperar que a cifra seja executada muito rapidamente em software. O RC4 é usado nos padrões SSL/TLS (Secure Sockets Layer/Transport Layer Security), definidos para a comunicação entre os navegadores Web e servidores. Ele também é usado no protocolo WEP (Wired Equivalent Privacy) e no protocolo mais recente, WPA (WiFi Protected Access), que fazem parte do padrão de LAN sem fio IEEE 802.11. O RC4 foi mantido como um segredo comercial pela RSA Security. Em setembro de 1994, entretanto, o algoritmo RC4 foi colocado anonimamente na Internet, numa lista chamada Cypherpunks.

O algoritmo RC4 é notavelmente simples e muito fácil de explicar. Uma chave de tamanho variável de 1 a 256 bytes (8 a 2048 bits) é usada para inicializar um vetor de estado S de 256 bytes, com elementos S[0], S[1], ... , S[255]. Em todos os momentos, S contém uma permutação de todos os números de 8 bits de 0 a 255. Para a criptografia e a decriptografia, um byte *k* (ver Figura 6.8) é gerado a partir de S selecionando uma das 255 entradas em um padrão sistemático. À medida que cada valor de *k* é gerado, as entradas em S são trocadas mais uma vez .

Inicialização de S Para começar, as entradas de S são definidas iguais aos valores de 0 a 255 em ordem crescente; ou seja, S[0] = 0, S[1] = 1, ... , S[255] = 255. Um vetor temporário, T, também é criado. Se o tamanho da chave K for 256 bytes, então K é transferido para T; caso contrário, para uma chave de tamanho *keylen* bytes. Então, os primeiros *keylen* elementos de T são copiados de K e, depois, K é repetido tantas vezes quanto for necessário para preencher T. Essas operações preliminares podem ser resumidas da seguinte forma:

```
/* Inicialização */
for  i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

Em seguida, usamos T para produzir a permutação inicial de S. Isso consiste em começar com S[0], prosseguir até S[255] e, para cada S[i], trocar S[i] por outro byte em S, de acordo com um esquema ditado por T[i]:

```
/* Permutação inicial de S */
j = 0;
for  i = 0 to 255 do
  j = (j + S[i] + T[i]) mod 256;
  Swap (S[i], S[j]);
```

Como a única operação sobre S é uma troca, o único efeito é uma permutação. S ainda contém todos os números de 0 até 255.

Geração de fluxo Depois que o vetor S for inicializado, a chave de entrada não será mais usada. A geração de fluxo consiste em percorrer todos os elementos de S[i] e, para cada S[i], trocar S[i] por outro byte em S, de acordo com um esquema ditado pela configuração atual de S. Depois que S[255] for alcançado, o processo continua, começando mais uma vez em S[0]:

```
/* Geração de fluxo */
i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```

Para criptografar, faça o XOR do valor *k* com o próximo byte do texto claro. Para decriptografar, faça o XOR do valor *k* com o próximo byte do texto cifrado.

A Figura 6.9 ilustra a lógica do RC4.

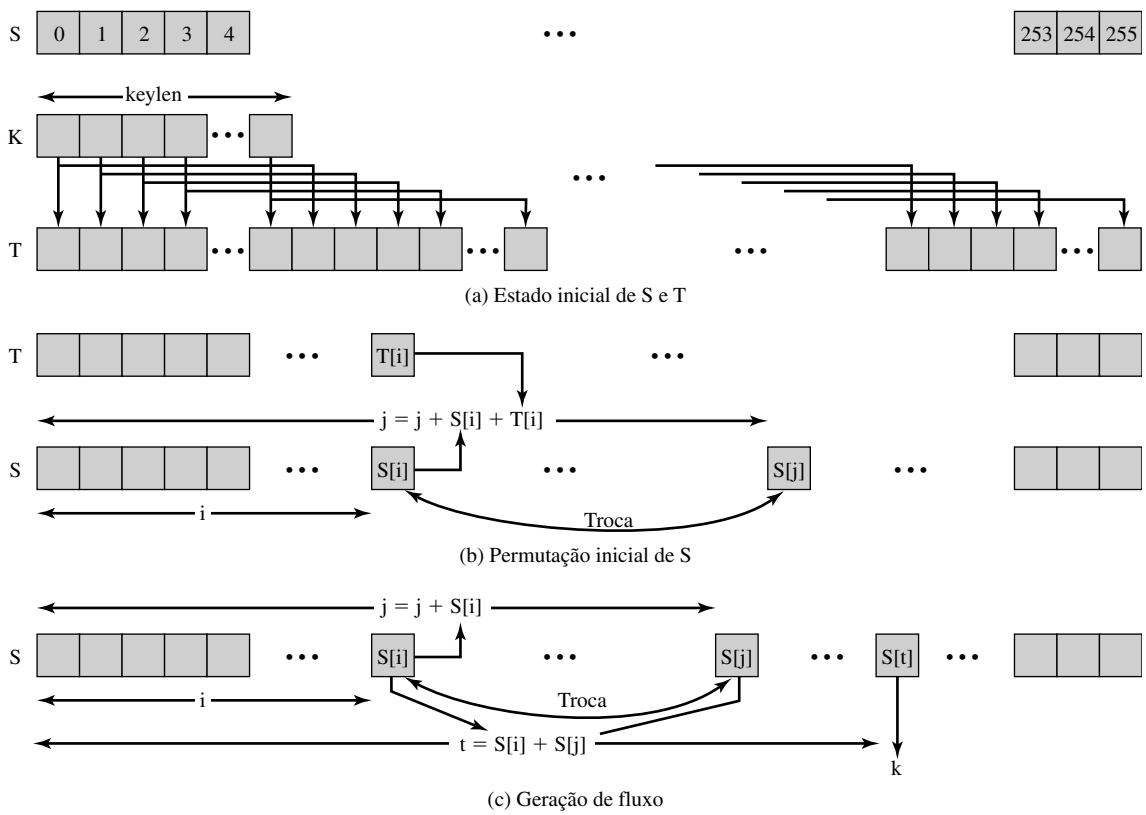


Figura 6.9 RC4.

Força do RC4 Foram publicados vários artigos analisando métodos de ataque ao RC4 (por exemplo, [KNUD98], [MIST98], [FLUH00], [MANT01]). Nenhuma dessas técnicas é prática contra o RC4 com um tamanho de chave razoável, como 128 bits. Um problema mais sério é informado em [FLUH01]. Os autores demonstram que o protocolo WEP, que pretende oferecer confidencialidade em redes locais sem fio 802.11, é vulnerável a uma técnica de ataque em particular. Na verdade, o problema não reside no próprio RC4, mas na maneira como as chaves são geradas para uso como entrada para o RC4. Esse problema específico não parece ser relevante a outras aplicações usando RC4, e pode ser remediado no WEP alterando-se o modo pelo qual as chaves são geradas. Esse problema mostra a dificuldade de se projetar um sistema seguro, que envolva tanto as funções criptográficas quanto os protocolos que as utilizam.

6.4 LEITURA E SITES WEB RECOMENDADOS

[SCHN96] fornece detalhes sobre diversas cifras de bloco simétricas, além de algumas cifras de fluxo. [ROBS95b] é um exame interessante e rico de muitos aspectos de projeto relacionados a cifras de bloco simétricas.

[KUMA97] contém uma explicação excelente e extensa dos princípios de projeto de cifra de fluxo. Outro bom tratamento, bastante matemático, é [RUEP92]. [ROBS95a] é um exame interessante e rico de muitas questões de projeto relacionadas a cifras de fluxo.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills: Aegean Park Press, 1997.
ROBS95a Robshaw, M. "Stream ciphers". RSA Laboratories Technical Report TR-701, julho de 1995. **http://www.rsasecurity.com/rsalabs**
ROBS95b Robshaw, M. "Block ciphers". RSA Laboratories Technical Report TR-601, agosto de 1995. **http://www.rsasecurity.com/rsalabs**
RUEP92 Rueppel, T. "Stream ciphers". Em [SIMM92].
SCHN96 Schneier, B. *Applied cryptography*. New York: Wiley, 1996.
SIMM92 Simmons, G., ed. *Contemporary cryptology: the science of information integrity.* Piscataway: IEEE Press, 1992.

Site Web recomendado:

- **Modos de operação da cifra de bloco:** Página do NIST com informações completas sobre os modos de operação aprovados pelo NIST.

6.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

ataque meet-in-the-middle
cifra de fluxo
DES triplo (3DES)
modo Cipher Block Chaining (CBC)
modo Cipher Feedback (CFB)
modo Counter (CTR)
modo Electronic Codebook (ECB)
modo Output Feedback (OFB)
modos de operação de cifra de bloco
RC4

Perguntas de revisão

6.1 O que é criptografia tripla?
6.2 O que é ataque meet-in-the-middle?
6.3 Quantas chaves são usadas na criptografia tripla?
6.4 Por que a parte do meio do 3DES é decriptografia em vez de criptografia?
6.5 Liste considerações de projeto importantes para uma cifra de fluxo.
6.6 Por que não é desejável reutilizar uma chave de cifra de fluxo?
6.7 Que operações primitivas são usadas no RC4?
6.8 Por que alguns modos de operação de cifra de bloco só utilizam a criptografia, enquanto outros utilizam criptografia e decriptografia?

Problemas

6.1 Você deseja construir um dispositivo de hardware para realizar a criptografia de bloco no modo de encadeamento de blocos cifrados (CBC) usando um algoritmo mais forte do que DES. 3DES é um bom candidato. A Figura 6.10 mostra duas possibilidades, ambas acompanhando a definição do CBC. Qual das duas você escolheria:

a. Para segurança?
b. Para desempenho?

6.2 Você consegue sugerir um aprimoramento de segurança para qualquer uma das opções da Figura 6.10, usando apenas três chips que implementam o DES e alguma quantidade de funções XOR? Suponha que você está limitado a duas chaves.

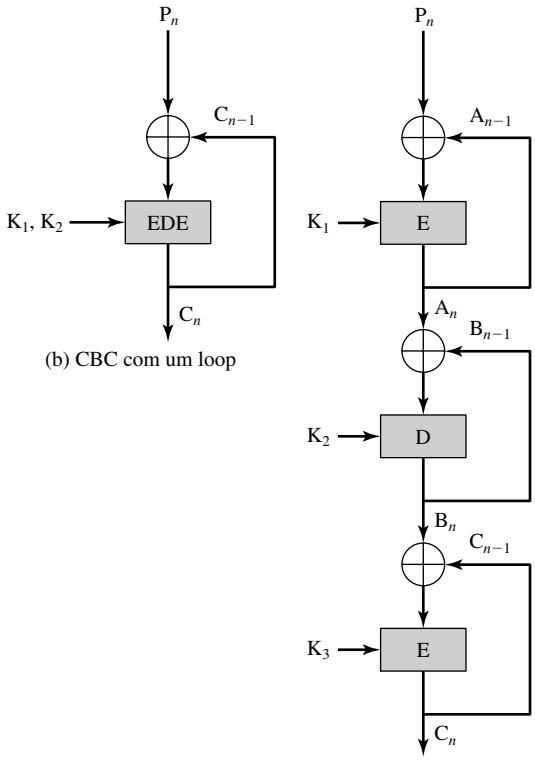


Figura 6.10 Uso de DES triplo no modo CBC.

6.3 O ataque Merkle-Hellman sobre o 3DES começa assumindo um valor de A = 0 (Figura 6.1b). Depois, para cada um dos 2^{56} valores possíveis de K_1, o texto claro P que produz A = 0 é determinado. Descreva o restante do algoritmo.

6.4 Com o modo ECB do DES, se houver um erro em um bloco do texto cifrado transmitido, somente o bloco de texto claro correspondente será afetado. Porém, no modo CBC, esse erro se propaga. Por exemplo, um erro em C_1 transmitido (Figura 6.4) obviamente adultera P_1 e P_2.
a. Algum bloco além de P_2 será afetado?
b. Suponha que haja um bit errado na versão fonte de P_1. Por quantos blocos de texto cifrado esse erro será propagado? Qual será o efeito no receptor?

6.5 Se um erro de bit ocorrer na transmissão de um caractere de texto cifrado no modo CFB de 8 bits, até que distância o erro se propagará?

6.6 Preencha o restante desta tabela:

Modo	Criptografia	Decriptografia
ECB	$C_j = E(K, P_j) \quad j = 1, \ldots, N$	$P_j = D(K, C_j) \quad j = 1, \ldots, N$
CBC	$C_1 = E(K, [P_1 \oplus IV])$	$P_1 = D(K, C_1) \oplus IV$
	$C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \ldots, N$	$P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \ldots, N$
CFB		
OFB		
CTR		

6.7 CBC-Pad é um modo de cifra de bloco usado na cifra de bloco RC5, mas poderia ser usado em qualquer cifra de bloco. CBC-Pad trata textos claros de qualquer tamanho. O texto cifrado é maior do que o texto claro no máximo o tamanho de um único bloco. A complementação é usada para garantir que a entrada do texto claro seja um múltiplo do tamanho do bloco. Considera-se que o texto claro original tenha um número inteiro de bytes. Esse texto claro é complementado no final por 1 a *bb* bytes, onde *bb* é igual ao tamanho do bloco em bytes. Os bytes de complementação são todos iguais e definidos como um byte que representa o número de bytes da complementação. Por exemplo, se houver 8 bytes de complementação, cada byte terá o padrão de bits 00001000. Por que não permitir zero byte de complementação? Ou seja, se o texto claro original for um múltiplo inteiro do tamanho do bloco, por que não evitar a complementação?

6.8 A complementação pode nem sempre ser apropriada. Por exemplo, pode-se querer armazenar os dados criptografados no mesmo buffer de memória que continha originalmente o texto claro. Nesse caso, o texto cifrado não deverá ter o mesmo tamanho do texto claro, original. Um modo para essa finalidade é o modo de roubo de texto cifrado (CTS). A Figura 6.11a mostra uma implementação desse modo.
a. Explique como ele funciona.
b. Descreva como decriptografar C_{n-1} e C_n.

6.9 A Figura 6.11b mostra uma alternativa ao CTS para produzir o texto cifrado de mesmo tamanho do texto claro quando este não for um múltiplo inteiro do tamanho do bloco.
a. Explique o algoritmo.
b. Explique por que o CTS é preferível a essa técnica ilustrada na Figura 6.11b.

6.10 Que valor de chave do RC4 deixará S inalterado durante a inicialização? Ou seja, após a permutação inicial de S, as entradas de S serão iguais aos valores de 0 até 255 em ordem crescente.

6.11 O RC4 tem um estado interno secreto, que é uma permutação de todos os valores possíveis do vetor **S** e os dois índices *i* e *j*.

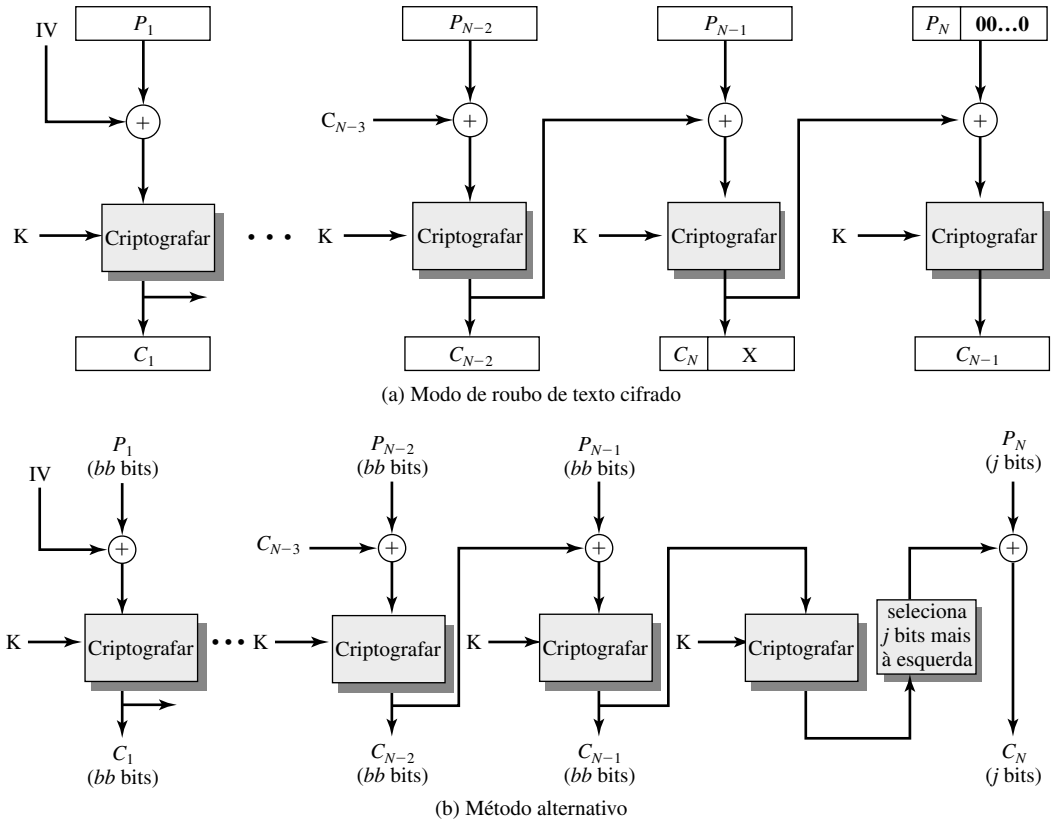


Figura 6.11 Modos de cifra de bloco para texto claro não múltiplo do tamanho do bloco.

a. Usando um esquema direto para armazenar o estado interno, quantos bits são utilizados?

b. Suponha que pensemos nisso do ponto de vista de quanta informação é representada pelo estado. Nesse caso, precisamos determinar quantos estados diferentes existem, depois tomar o log na base 2 para descobrir quantos bits de informação isso representa. Usando essa técnica, quantos bits seriam necessários para representar o estado?

6.12 Alice e Bob concordam em se comunicar privativamente por e-mail usando um esquema baseado no RC4, mas querem evitar o uso de uma nova chave secreta para cada transmissão. Alice e Bob combinam privativamente uma chave de 128 bits k. Para criptografar uma mensagem m, consistindo em uma string de bits, o seguinte procedimento é utilizado:

1. Escolha um valor de 80 bits aleatório v
2. Gere o texto cifrado $c = \text{RC4}(v \parallel k) \oplus m$
3. Envie a string de bits $(v \parallel c)$

a. Suponha que Alice use esse procedimento para enviar uma mensagem m para Bob. Descreva como Bob pode recuperar a mensagem m a partir de $(v \parallel c)$ usando k.

b. Se um adversário observar diversos valores $(v_1 \parallel c_1), (v_2 \parallel c_2), \ldots$ transmitidos entre Alice e Bob, como ele pode determinar quando o mesmo fluxo de chave foi usado para criptografar duas mensagens?

c. Aproximadamente quantas mensagens Alice pode esperar enviar antes que o mesmo fluxo de chave seja usado duas vezes? Use o resultado a partir do paradoxo do aniversário, descrito no Apêndice 11A [Equação (11.7)].

d. O que isso mostra a respeito do tempo de vida da chave k (ou seja, do número de mensagens que podem ser criptografadas usando k)?

Problemas de programação

6.13 Crie um software que possa criptografar e decriptografar no modo Cipher Block Chaining usando uma das seguintes cifras: afim módulo 256, Hill modulo 256, S-DES, DES. Teste os dados para S-DES: usando um vetor de inicialização de 1010 1010, um texto claro binário de 0000 0001 0010 0011 criptografado com uma chave binária de 01111 11101 deverá resultar um texto claro binário de 1111 0100

0000 1011. A decriptografia deverá funcionar de modo correspondente.

6.14 Crie um software que possa criptografar e decriptografar no modo Cipher Feedback de 4 bits usando uma das seguintes cifras: aditiva módulo 256, afim módulo 256, S-DES;

ou

modo Cipher Feedback de 8 bits usando uma das seguintes cifras: 2×2 Hill módulo 256. Teste os dados para S-DES: usando um vetor de inicialização binário de 1010 1011, um texto claro binário de 0001 0010 0011 0100 criptografado com uma chave binária de 01111 11101 deverá gerar um texto claro binário de 1110 1100 1111 1010. A decriptografia deverá funcionar de modo correspondente.

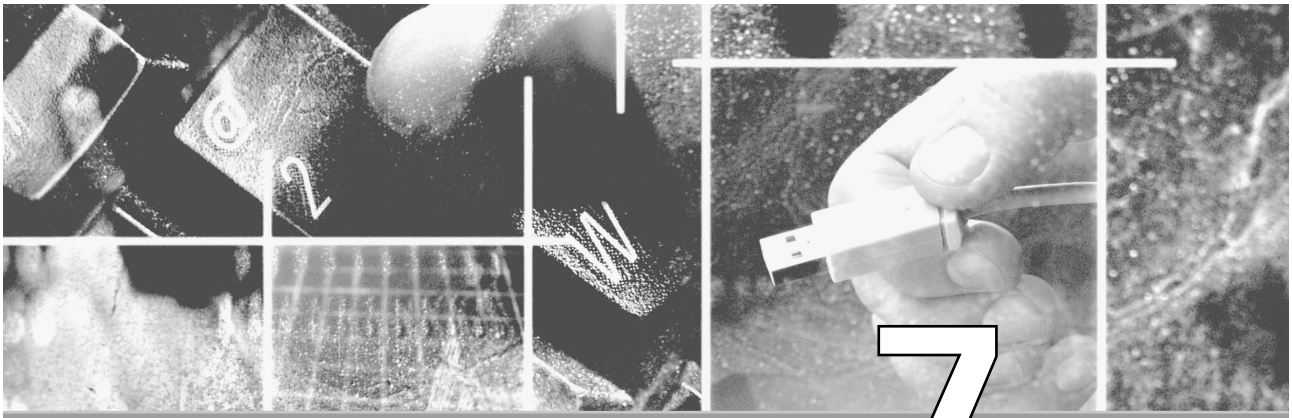
6.15 Crie um software que possa criptografar e decriptograr no modo Output Feedback de 4 bits usando uma das seguintes cifras: aditiva módulo 256, afim módulo 256, S-DES;

ou

modo Output Feedback de 8 bits usando uma das seguintes cifras: 2×2 Hill módulo 256.

6.16 Crie um software que possa criptografar e decriptograr no modo Counter usando uma das seguintes cifras: afim módulo 256, Hill módulo 256, S-DES. Teste os dados para S-DES: usando um contador começando em 0000 0000, um texto claro binário de 0000 0001 0000 0010 0000 0100 criptogragrado com uma chave binária de 01111 11101 deverá gerar um texto claro binário de 0011 1000 0100 1111 0011 0010. A decriptografia deverá funcionar de modo correspondente.

6.17 Implemente um ataque de criptoanálise diferencial sobre S-DES com três rodadas.



CAPÍTULO 7

Confidencialidade usando criptografia simétrica

7.1 **Posicionamento da função de criptografia**

7.2 **Confidencialidade do tráfego**

7.3 **Distribuição de chaves**

7.4 **Geração de números aleatórios**

7.5 **Leitura e sites Web recomendados**

7.6 **Principais termos, perguntas de revisão e problemas**

Entre as tribos da Austrália Central, todo homem, mulher e criança possui um nome secreto ou sagrado concedido por seus ancestrais após o nascimento, e que não é conhecido por ninguém além dos membros totalmente iniciados do grupo. Esse nome secreto nunca é mencionado, exceto nas ocasiões mais solenes; pronunciá-lo para homens de outro grupo seria uma séria violação do costume tribal. Quando mencionado, o nome é falado apenas em um sussurro, e somente depois de tomadas todas as precauções para que não seja ouvido por ninguém além dos membros do grupo. O nativo acredita que um estranho, conhecendo seu nome secreto, teria poder especial para causar-lhe enfermidades por meio de magia

The golden bough, Sir James George Frazer

PONTOS PRINCIPAIS

- Em um ambiente distribuído, os dispositivos de criptografia podem ser posicionados para dar suporte à criptografia de enlace ou à de ponta a ponta. Com a criptografia de enlace, cada enlace de comunicações vulnerável é equipado nas duas extremidades com um dispositivo de criptografia. Com a de ponta a ponta, o processo de criptografia é executado nos dois sistemas finais.
- Mesmo que todo o tráfego entre os usuários seja criptografado, uma análise de tráfego pode gerar informações valiosas a um oponente. Uma contramedida eficaz é o preenchimento de tráfego, que envolve o envio de bits aleatórios durante períodos em que nenhum dado criptografado esteja disponível para transmissão.
- A distribuição de chaves é a função que envia uma chave a duas partes que querem trocar dados criptografados com segurança. Algum tipo de mecanismo ou protocolo é necessário para providenciar a distribuição segura das chaves.
- A distribuição de chaves normalmente envolve o uso de chaves-mestras, que são usadas com pouca freqüência e têm longa duração, e chaves de sessão, que são geradas e distribuídas para uso temporário entre duas partes.
- Uma capacidade com aplicação para diversas funções criptográficas é a geração de números aleatórios ou pseudo-aleatórios. O requisito principal para esta capacidade é que o fluxo de números gerados seja imprevisível.

Historicamente, o foco da criptologia tem sido o uso da criptografia simétrica para oferecer confidencialidade. Somente nas últimas décadas é que outras considerações, como autenticação, integridade, assinaturas digitais e o uso da criptografia por chave pública foram incluídas na teoria e na prática da criptologia.

Antes de examinarmos alguns desses tópicos mais recentes, vamos nos concentrar neste capítulo no uso da criptografia simétrica para oferecer confidencialidade. Esse assunto continua sendo importante por si só. Além disso, o conhecimento das questões envolvidas aqui ajuda a motivar o desenvolvimento da criptografia por chave pública e esclarece as questões envolvidas em outras aplicações da criptografia, como a autenticação.

Começaremos com uma explicação do local da lógica de criptografia; a escolha principal aqui é entre o que é conhecido como criptografia de enlace e criptografia de ponta a ponta. Em seguida, examinaremos o uso da criptografia contra ataques de análise de tráfego. Depois, explicaremos o difícil problema da distribuição de chaves. Finalmente, explicaremos os princípios por trás de uma importante ferramenta para oferecer o recurso de confidencialidade: a geração de números aleatórios.

7.1 POSICIONAMENTO DA FUNÇÃO DE CRIPTOGRAFIA

Se a criptografia tiver de ser usada contra ataques à confidencialidade, precisaremos decidir o que criptografar e onde a função de criptografia deverá estar localizada. Para começar, esta seção examina os locais de ataques de segurança em potencial e, depois, examinará duas técnicas principais para o posicionamento da criptografia: enlace e ponta a ponta.

Locais em potencial para ataques de confidencialidade

Como exemplo, considere a estação de trabalho de um usuário em uma organização comercial típica. A Figura 7.1 sugere os tipos de recursos de comunicação que poderiam ser empregados por essa estação de trabalho e, portanto, oferece uma indicação dos pontos de vulnerabilidade.

Na maioria das organizações, as estações de trabalho estão ligadas a redes locais (LANs). Normalmente, o usuário pode alcançar outras estações de trabalho, hosts e servidores diretamente na LAN ou em outras LANs no mesmo prédio, interconectadas com bridges (pontes) e roteadores. Portanto, aqui está o primeiro ponto de vulnerabilidade. Nesse caso, o problema principal é a interceptação por outro funcionário. Normalmente, uma LAN é uma rede de transmissão. A transmissão de qualquer estação para qualquer outra estação é visível no meio da LAN por todas as estações. Os dados são transmitidos na forma de frames, com cada frame contendo o endereço de origem e o de destino. Um intruso pode monitorar o tráfego na LAN e captar qualquer tráfego desejado com base nos endereços de origem e de destino. Se parte ou toda a LAN for sem fio, então o potencial para interceptação é maior.

Além disso, o intruso não precisa ser necessariamente um funcionário no prédio. Se a LAN, por meio de um servidor de comunicações ou de um dos hosts na LAN, oferecer uma capacidade de conexão discada, então é possível que um intruso obtenha acesso à LAN e monitore o tráfego.

O acesso ao mundo exterior a partir da LAN quase sempre está disponível na forma de um roteador que se conecta à Internet, um banco de modems discados ou algum outro tipo de servidor de comunicações. Pelo servidor de comu-

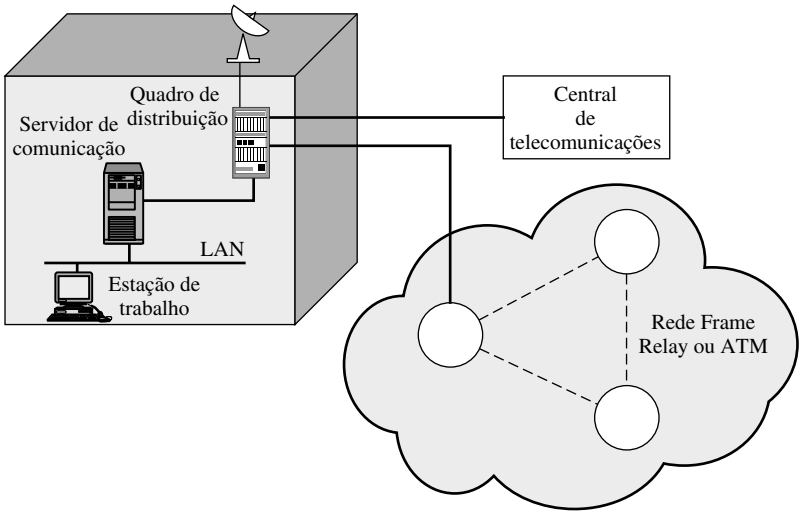


Figura 7.1 Pontos de vulnerabilidade.

nicações, existe uma linha que leva a um quadro de distribuição. O quadro de distribuição serve como painel para interconectar dados internos e linhas telefônicas, e fornecer um ponto de plataforma para comunicações externas.

O próprio quadro de distribuição é vulnerável. Se um intruso puder penetrar na cabine, ele poderá grampear cada fio para determinar quais são usados para transmissão de dados. Depois de isolar uma ou mais linhas, o intruso pode conectar um transmissor de rádio de baixa potência, e os sinais resultantes poderão ser coletados em um local próximo (por exemplo, um carro estacionado ou um prédio vizinho).

É possível que haja diferentes rotas a partir do quadro de distribuição. Uma configuração-padrão oferece acesso à central de telecomunicações da empresa telefônica local mais próxima. Os fios na central são reunidos em um cabo, que normalmente é consolidado com outros cabos no porão do prédio. A partir daí, um cabo grosso passa por via subterrânea até a central.

Além disso, o quadro de distribuição pode oferecer um enlace para uma antena de microondas, seja em uma estação terrestre para um enlace de satélite ou um enlace terrestre de microondas ponto a ponto. O enlace da antena pode ser parte de uma rede privada, ou pode ser um *bypass* local para conexão de longa distância com uma empresa.

O quadro de distribuição também pode oferecer um enlace para um nó de uma rede de comutação de pacotes. Esse enlace pode ser uma linha alugada, uma linha direta privada ou uma conexão comutada por meio de uma rede de telecomunicações pública. Dentro da rede, os dados passam por diversos nós e enlaces entre os nós, até que os dados cheguem ao nó ao qual o sistema de destino está conectado.

Um ataque pode ocorrer em qualquer um dos enlaces de comunicações. Para os ataques ativos, o atacante precisa obter controle físico de uma parte do enlace e ser capaz de inserir e captar as transmissões. Para um ataque passivo, ele simplesmente precisa ser capaz de observar as transmissões. Os enlaces de comunicações envolvidos podem ser cabo (par trançado de telefone, cabo coaxial ou fibra óptica), enlaces de microondas ou canais de satélite. Os cabos de par trançado e coaxial podem ser atacados por meio de grampos ou dispositivos indutivos, que monitoram emanações eletromagnéticas. Os grampos físicos permitem ataques ativos e passivos, enquanto os grampos indutivos são úteis em ataques passivos. Nenhum tipo de grampo é eficaz com a fibra óptica, que é uma das vantagens desse meio de transmissão. A fibra não gera emanações eletromagnéticas e, portanto, não é vulnerável a grampos indutivos. A quebra física do cabo degrada bastante a qualidade do sinal e, por isso, pode ser detectada. As transmissões por microondas e satélite podem ser interceptadas com pouco risco ao atacante. Isso é verdade especialmente em transmissões por satélite, que abrangem uma grande área geográfica. Ataques ativos a microondas e satélite também são possíveis, embora mais difíceis tecnicamente e podendo ser muito dispendiosos.

Além da vulnerabilidade em potencial dos diversos enlaces de comunicações, os diversos processadores ao longo do caminho estão sujeitos a ataques. Um ataque pode tomar a forma de tentativas de modificar o hardware ou o software, de ganhar acesso à memória do processador, ou de monitorar as emanações eletromagnéticas. Esses ataques são menos prováveis do que aqueles envolvendo enlaces de comunicações, mas, apesar disso, são uma fonte de risco.

Assim, existem diversos locais onde pode haver um ataque. Além disso, para comunicações remotas, muitos desses locais não estão sob o controle físico do usuário final. Até mesmo no caso de redes locais, em que as medidas de segurança físicas são possíveis, sempre existe a ameaça do funcionário insatisfeito.

Criptografia de enlace *versus* ponta a ponta

A técnica mais poderosa e mais comum para a proteção dos pontos de vulnerabilidade indicados na seção anterior é a criptografia. Se a criptografia for usada para conter esses ataques, então precisamos decidir o que criptografar e onde o mecanismo de criptografia deverá ser posicionado. Como a Figura 7.2 indica, existem duas opções fundamentais: criptografia de enlace e criptografia de ponta a ponta.

Técnicas básicas

Com a criptografia de enlace, cada enlace de comunicações vulnerável é equipado nas duas extremidades com um dispositivo de criptografia. Assim, todo o tráfego por todos os enlaces de comunicações é protegido. Embora esse recurso exija muitos dispositivos de criptografia em uma rede ampla, seu valor é claro. Uma de suas desvantagens é que a mensagem precisa ser decriptografada toda vez que entrar em um switch (como um switch frame relay), pois este precisa ler o endereço (número de conexão lógico) no cabeçalho do pacote, a fim de rotear o frame. Assim, a mensagem é vulnerável em cada switch. Se estivermos trabalhando com uma rede pública, o usuário não tem controle sobre a segurança dos nós.

Várias implicações da criptografia de enlace devem ser observadas. Para que essa estratégia seja eficaz, todos os enlaces em potencial no caminho da origem até o destino precisam usar a criptografia de enlace. Cada par de nós que compartilha um enlace deve compartilhar uma chave única, com uma chave diferente sendo usada em cada enlace. Assim, muitas chaves precisam ser fornecidas.

Com a criptografia de ponta a ponta, o processo é executado nos dois sistemas finais. O host ou terminal de origem criptografa os dados. Os dados no formato criptografado são, então, transmitidos sem alteração pela rede para o terminal ou host de destino. O destino compartilha uma chave com a origem e, então, é capaz de decriptografar os dados. Esse plano parece proteger a transmissão contra ataques nos enlaces ou switches da rede. Assim, a criptografia de ponta a ponta alivia o usuário final das preocupações a respeito do grau de segurança das redes e enlaces que dão suporte à comunicação. Porém, ainda existe um ponto fraco.

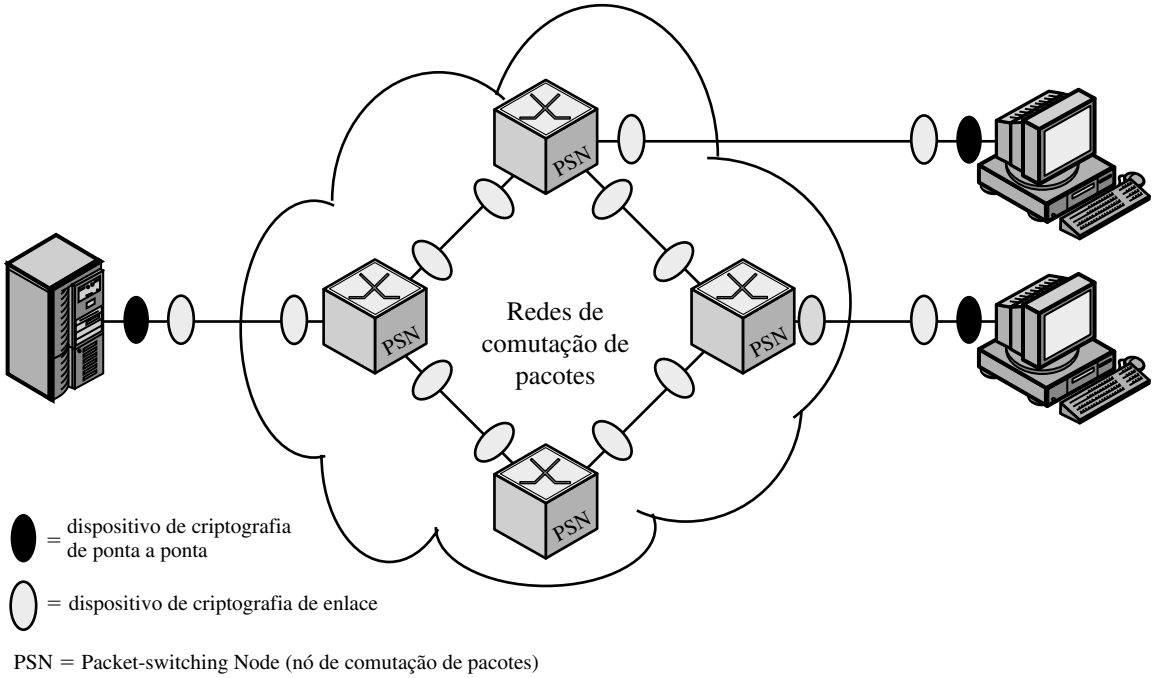


Figura 7.2 Criptografia por uma rede de comutação de pacotes.

Considere a seguinte situação. Um host se conecta a uma rede frame relay ou ATM, estabelece uma conexão lógica com outro host e está preparado para transferir dados para esse outro host, usando a criptografia de ponta a ponta. Os dados são transmitidos por essa rede na forma de pacotes que consistem em um cabeçalho e alguns dados do usuário. Que parte de cada pacote o host irá criptografar? Suponha que o host criptografe o pacote inteiro, incluindo o cabeçalho. Isso não funcionará porque, lembre-se, apenas o outro host pode realizar a decriptografia. O switch frame relay ou ATM receberá um pacote criptografado e não poderá ler o cabeçalho. Portanto, ele não poderá rotear o pacote. Segue-se que o host só pode criptografar no pacote a parte que contém os dados do usuário, e precisa deixar o cabeçalho às claras.

Assim, com a criptografia de ponta a ponta, os dados do usuário são protegidos, mas não, o padrão de tráfego, pois os cabeçalhos de pacote são transmitidos às claras. Por outro lado, a criptografia de ponta a ponta oferece um grau de autenticação. Se dois sistemas finais compartilharem uma chave de criptografia, então um destinatário terá certeza de que qualquer mensagem recebida será proveniente do emissor alegado, pois somente esse emissor compartilhará a chave apropriada. Essa autenticação não é inerente a um esquema de criptografia de enlace.

Para obter maior segurança, são necessárias a criptografia de enlace e de ponta a ponta, como vimos na Figura 7.2. Quando as duas formas de criptografia são empregadas, o host criptografa a parte dos dados do usuário contida em um pacote usando uma chave de criptografia de ponta a ponta. O pacote inteiro é então criptografado, usando uma chave de criptografia de enlace. À medida que o pacote atravessa a rede, cada switch decriptografa o pacote, usando uma chave de criptografia de enlace para ler o cabeçalho e, depois, criptografa o pacote inteiro novamente para enviá-lo pelo próximo enlace. Agora, o pacote inteiro está seguro, exceto pelo tempo em que ele está de fato na memória de um switch de pacotes, quando o cabeçalho do pacote permanece às claras.

A Tabela 7.1 resume as principais características das duas estratégias de criptografia.

Posicionamento lógico da função de criptografia de ponta a ponta Com a criptografia de enlace, a função de criptografia é realizada em um nível baixo da hierarquia de comunicações. Em termos do modelo Open Systems Interconnection (OSI), a criptografia de enlace ocorre nas camadas física ou de enlace.

Para a criptografia de ponta a ponta, várias escolhas são possíveis para o posicionamento lógico da função de criptografia. No nível prático mais baixo, a função de criptografia poderia ser realizada na camada de rede. Assim, por exemplo, a criptografia poderia ser associada ao protocolo frame relay ou ATM, de modo que a parte de dados do usuário para todos os frames ou células ATM é criptografada.

Com a criptografia da camada de rede, o número de entidades identificáveis e protegidas separadamente corresponde ao número de sistemas finais na rede. Cada sistema final pode se comprometer com uma troca criptografada

Tabela 7.1 Características da criptografia de enlace e de ponta a ponta [PFLE02]

Criptografia de enlace	Criptografia de ponta a ponta
Segurança dentro dos sistemas finais e sistemas intermediários	
Mensagem exposta no host emissor	Mensagem criptografada no host emissor
Mensagem exposta nos nós intermediários	Mensagem criptografada nos nós intermediários
Papel do usuário	
Aplicado pelo host emissor	Aplicado pelo processo emissor
Transparente ao usuário	Usuário aplica criptografia
Host mantém recurso de criptografia	Usuário precisa determinar algoritmo
Um recurso para todos os usuários	Usuários selecionam esquema de criptografia
Pode ser feito no hardware	Implementação de software
Todas ou nenhuma mensagem criptografada	Usuário escolhe criptografar ou não, para cada mensagem
Problemas de implementação	
Exige uma chave por par (host/nó intermediário) e par (nó intermediário/ nó intermediário)	Exige uma chave por par de usuários
Oferece autenticação de host	Oferece autenticação do usuário

com outro sistema final se os dois compartilharem uma chave secreta. Todos os processos e aplicações do usuário dentro de cada sistema final empregariam o mesmo esquema de criptografia com a mesma chave para alcançar um sistema final de destino em particular. Com esse arranjo, pode ser desejável passar a função de criptografia para algum tipo de processador front-end (normalmente, uma placa de comunicações no sistema final).

A Figura 7.3 mostra a função de criptografia do processador front-end (FEP). No lado do host, o FEP aceita pacotes. A parte do pacote que contém os dados do usuário é criptografada, enquanto o cabeçalho do pacote contorna o processo de criptografia[1] O pacote resultante é entregue à rede. Na direção oposta, para pacotes chegando da rede, a parte de dados do usuário é decriptografada e o pacote inteiro é entregue ao host. Se a funcionalidade da camada de transporte (por exemplo, TCP) for implementada no front-end, então o cabeçalho da camada de transporte também seria deixado às claras e a parte de dados do usuário no campo de dados do protocolo de transporte é criptografada.

A implantação dos serviços de criptografia nos protocolos de ponta a ponta, como os da camada de rede frame relay ou TCP, oferece segurança de ponta a ponta para o tráfego dentro de uma interligação de rede totalmente integrada. Porém, esse esquema não pode oferecer o serviço necessário para o tráfego que atravessa as fronteiras entre as redes, como e-mail, intercâmbio eletrônico de dados (EDI) e transferências de arquivos.

A Figura 7.4 ilustra as questões envolvidas. Nesse exemplo, um gateway de e-mail é usado para interconectar uma interligação de rede que usa uma arquitetura baseada no modelo OSI com uma que utiliza uma arquitetura baseada no modelo TCP/IP.[2] Nessa configuração, não existe protocolo de ponta a ponta abaixo da camada de aplicação. As conexões de transporte e rede de cada sistema final terminam no gateway de correio, que estabelece novas conexões de transporte e rede para o enlace com o outro sistema final. Além disso, esse cenário não está limitado ao caso de um gateway entre duas arquiteturas diferentes. Mesmo que os dois sistemas finais utilizem TCP/IP ou OSI, existem muitos casos nas configurações reais em que os gateways de correio situam-se entre redes que, de outra maneira, estariam isoladas. Assim, para aplicações como e-mail, que têm uma funcionalidade *store-and-forward*, o único local para conseguir criptografia de ponta a ponta é na camada de aplicação.

Uma desvantagem da criptografia da camada de aplicação é que o número de entidades a considerar aumenta drasticamente. Uma rede que mantém centenas de hosts deve suportar milhares de usuários e processos. Assim, muito mais chaves secretas precisam ser geradas e distribuídas.

Uma maneira interessante de ver as alternativas é observar que, à medida que subimos na hierarquia de comunicações, menos informações são criptografadas, porém a segurança aumenta. A Figura 7.5 destaca esse ponto, usando a arquitetura TCP/IP como um exemplo. Na figura, um gateway em nível de aplicação refere-se a um dispositivo store-and-forward que opera no nível da aplicação.[3]

1. Os termos *vermelho* e *preto* são utilizados com freqüência. Dados vermelhos são dados sensíveis ou confidenciais não codificados. Dados pretos são dados criptografados.
2. O Apêndice H oferece uma rápida visão geral das arquiteturas de protocolo OSI e TCP/IP.
3. Infelizmente, a maioria dos documentos sobre TCP/IP utiliza o termo *gateway* para se referir ao que normalmente é chamado de *roteador*.

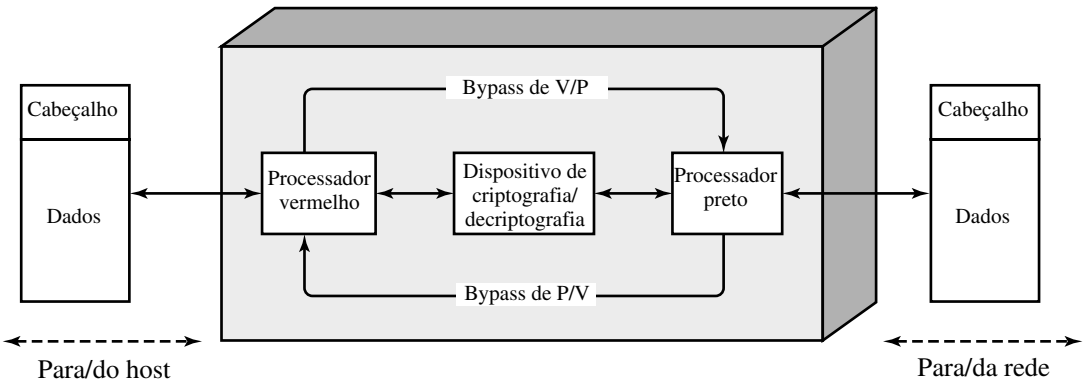


Figura 7.3 Função do processador front-end.

Com a criptografia em nível de aplicação (Figura 7.5a), somente a parte de dados do usuário de um segmento TCP é criptografada. Os cabeçalhos TCP, IP, em nível de rede, e em nível de enlace, além do trailer em nível de enlace aparecem às claras. Ao contrário, se a criptografia for realizada no nível TCP (Figura 7.5b), então, em uma única conexão de ponta a ponta, os dados do usuário e o cabeçalho TCP são criptografados. O cabeçalho IP permanece às claras, pois ele é necessário para que os roteadores roteiem o datagrama IP da origem até o destino. Porém, observe que, se uma mensagem passar por um gateway, a conexão TCP termina e uma nova conexão de transporte é aberta para o próximo hop (salto). Além disso, o gateway é tratado como um destino pelo IP subjacente. Assim, as partes criptografadas da unidade de dados são decriptografadas no gateway. Se o próximo hop for sobre uma rede TCP/IP, então os dados do usuário e o cabeçalho TCP são criptografados novamente antes da transmissão. Entretanto, no próprio gateway, a unidade de dados no buffer é mantida inteiramente às claras. Finalmente, para a criptografia em nível de enlace (Figura 7.5c), a unidade de dados inteira, exceto o cabeçalho e trailer do enlace, é criptografada em cada enlace, mas a unidade de dados inteira está às claras em cada roteador e gateway.[4]

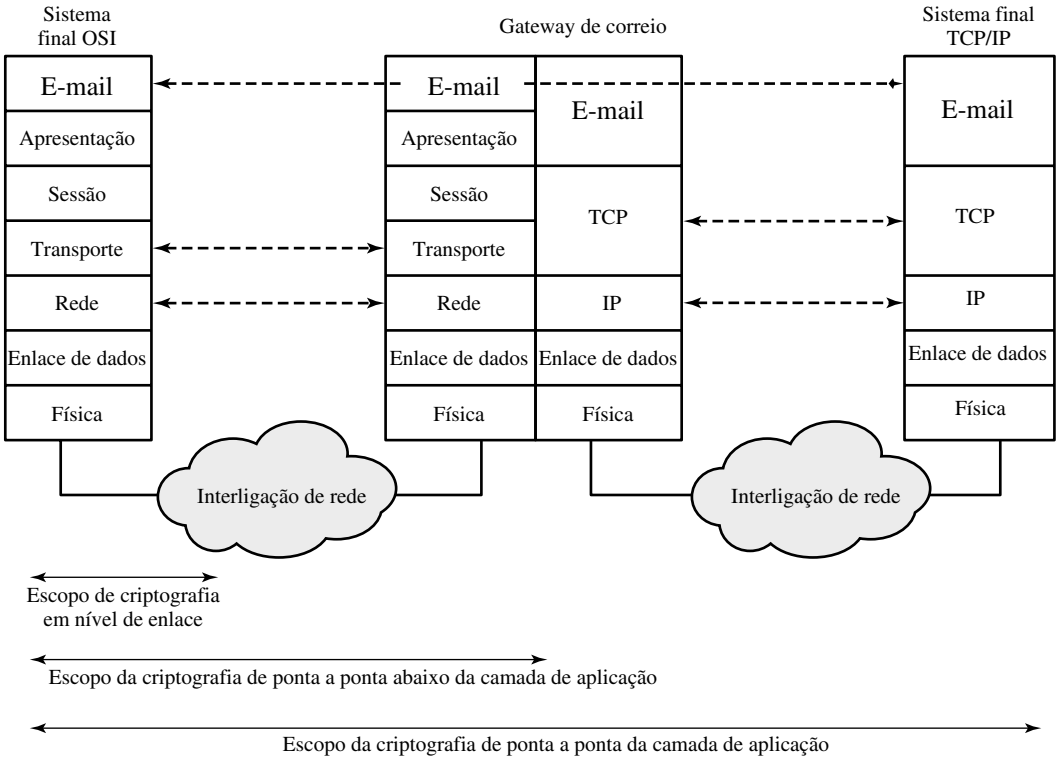
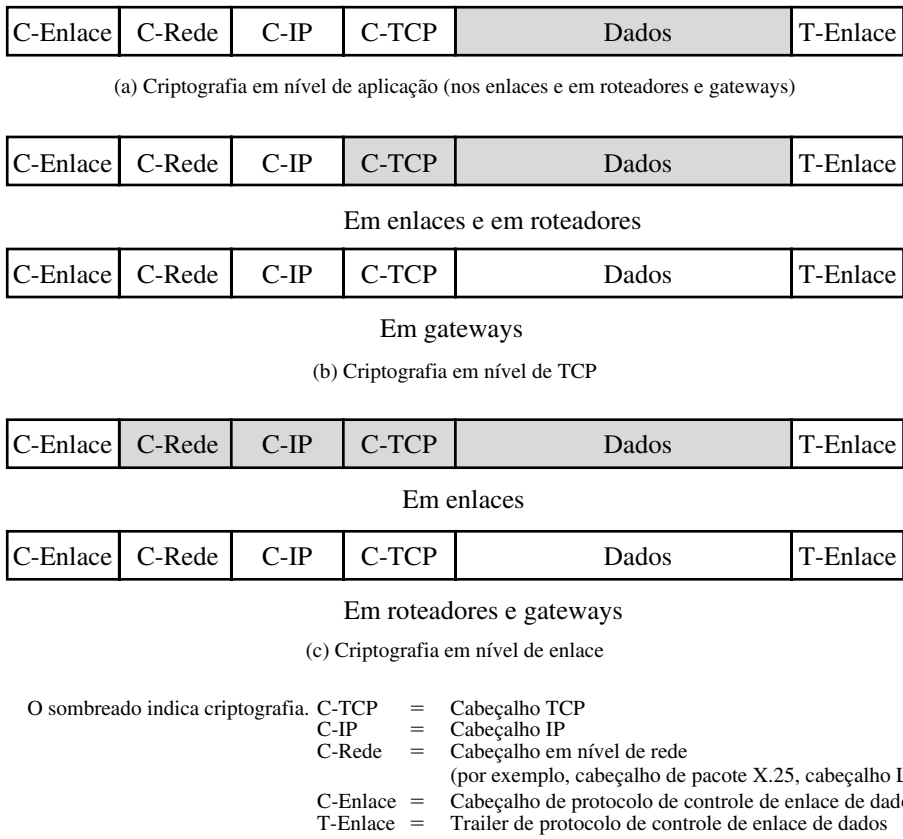


Figura 7.4 Implicações de cobertura da criptografia das comunicações *store-and-forward*.

[4]A figura realmente mostra uma alternativa a menos. Também é possível criptografar parte ou até mesmo todo o cabeçalho e trailer do enlace, exceto para os flags dos frames inicial e final.



O sombreado indica criptografia.
C-TCP = Cabeçalho TCP
C-IP = Cabeçalho IP
C-Rede = Cabeçalho em nível de rede (por exemplo, cabeçalho de pacote X.25, cabeçalho LLC)
C-Enlace = Cabeçalho de protocolo de controle de enlace de dados
T-Enlace = Trailer de protocolo de controle de enlace de dados

Figura 7.5 Relacionamento entre criptografia e níveis de protocolo.

7.2 CONFIDENCIALIDADE DO TRÁFEGO

Mencionamos no Capítulo 1 que, em alguns casos, os usuários se preocupam com a segurança a partir da análise de tráfego. O conhecimento sobre o número e o tamanho das mensagens entre os nós pode permitir que um oponente determine quem está falando com quem. Isso pode ter implicações óbvias em um conflito militar. Até mesmo em aplicações comerciais, a análise de tráfego pode gerar informações que os geradores do tráfego gostariam de ocultar. [MUFT89] relaciona os seguintes tipos de informações, que podem ser derivadas de um ataque de análise de tráfego:

- Identidades dos parceiros.
- A freqüência com que os parceiros estão se comunicando
- Padrão de mensagem, tamanho da mensagem ou quantidade de mensagens, o que sugere que informações importantes estão sendo trocadas
- Os eventos que se correlacionam com conversas especiais entre parceiros em particular

Outra preocupação relacionada ao tráfego é o uso de padrões de tráfego para criar um **canal secreto**. Um canal secreto é um meio de comunicação de um modo não planejado pelos projetistas do recurso de comunicação. Normalmente, o canal é usado para transferir informações de uma maneira que infrinja uma política de segurança. Por exemplo, um funcionário pode querer transmitir informações a alguém de fora de uma maneira que não seja detectada pela gerência, e que exija a simples interceptação da parte de quem está de fora. Os dois participantes poderiam preparar um código em que uma mensagem aparentemente legítima, menor do que um certo tamanho, representa o binário 0, enquanto uma mensagem maior representa o binário 1. Outros esquemas desse tipo são possíveis.

Técnica de criptografia de enlace

Com o uso da criptografia do enlace, os cabeçalhos da camada de rede (por exemplo, cabeçalho de frame ou célula) são criptografados, reduzindo a oportunidade de análise de tráfego. Porém, ainda é possível nessas circunstâncias que

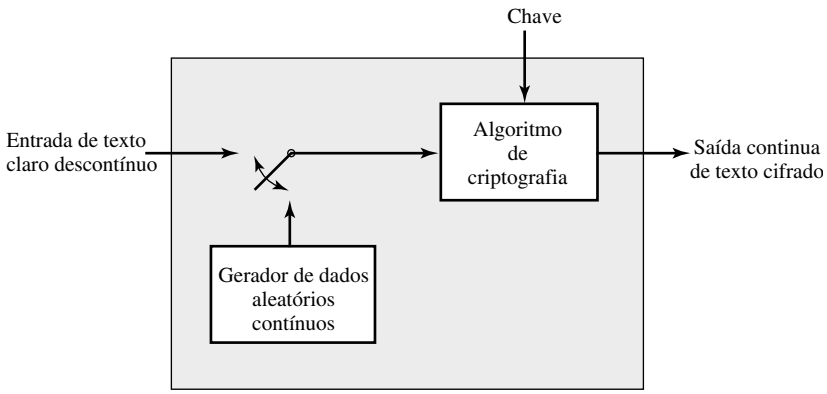


Figura 7.6 Dispositivo de criptografia com preenchimento de tráfego.

um atacante avalie a quantidade de tráfego em uma rede e observe a quantidade de tráfego entrando e saindo de cada sistema final. Uma contramedida eficaz para esse ataque é o preenchimento de tráfego, ilustrado na Figura 7.6.

O preenchimento de tráfego produz saída de texto cifrado continuamente, mesmo na ausência de texto claro. Um fluxo de dados aleatório contínuo é gerado. Quando o texto claro está disponível, ele é criptografado e transmitido. Quando a entrada de texto claro não estiver presente, os dados aleatórios são criptografados e transmitidos. Isso torna impossível para um atacante distinguir entre o fluxo de dados verdadeiro e preenchimento; portanto, é impossível deduzir a quantidade de tráfego.

Técnica de criptografia de ponta a ponta

O preenchimento do tráfego é basicamente uma função de criptografia de enlace. Se apenas a criptografia de ponta a ponta for empregada, então as medidas disponíveis ao defensor são mais limitadas. Por exemplo, se a criptografia for implementada na camada de aplicação, então um oponente pode determinar quais entidades de transporte estão ocupadas no diálogo. Se as técnicas de criptografia forem acomodadas na camada de transporte, então os endereços e padrões de tráfego da camada de rede permanecem acessíveis.

Uma técnica que poderia ser útil é preencher unidades de dados até um tamanho uniforme no nível de transporte ou de aplicação. Além disso, mensagens nulas podem ser inseridas aleatoriamente no fluxo. Essas táticas impedem que um oponente tenha conhecimento sobre a quantidade de dados trocada entre os usuários finais, e obscurece o padrão de tráfego subjacente.

7.3 DISTRIBUIÇÃO DE CHAVES

Para que a criptografia simétrica funcione, as duas partes precisam compartilhar a mesma chave, e esta precisa ser protegida contra acesso por outras partes. Além disso, mudanças freqüentes na chave normalmente são desejáveis, para limitar a quantidade de dados comprometidos se um atacante descobrir a chave. Portanto, a força de qualquer sistema criptográfico está na *técnica de distribuição de chaves*, um termo que se refere aos meios de entregar uma chave a duas partes que querem trocar dados, sem permitir que outros a vejam. Para duas partes, A e B, a distribuição de chaves pode ser feita de várias maneiras, como a seguir:

1. A pode selecionar uma chave e entregá-la fisicamente a B.
2. Um terceiro pode selecionar a chave e entregá-la fisicamente a A e B.
3. Se A e B tiverem usado uma chave prévia e recentemente, uma parte pode transmitir a nova chave criptografada à outra, usando a chave antiga.
4. Se A e B tiverem uma conexão criptografada com um terceiro C, C pode entregar uma chave a A e B pelos enlaces criptografados.

As opções 1 e 2 exigem remessa manual de uma chave. Para a criptografia de enlace, esse é um requisito razoável, pois cada dispositivo de criptografia de enlace estará trocando dados somente com seu parceiro no outro extremo do enlace. Porém, para a criptografia de ponta a ponta, a entrega manual é complicada. Em um sistema distribuído, qualquer host ou terminal pode ter de se comprometer nas trocas com muitos outros hosts e terminais com o passar do tempo. Assim, cada dispositivo precisa de uma série de chaves fornecidas dinamicamente. O problema é especialmente difícil em um sistema distribuído remoto.

A escala do problema depende do número de pares em comunicação que precisam ser suportados. Se a criptografia de ponta a ponta for feita em um nível de rede ou IP, então uma chave será necessária para cada par de hosts da rede

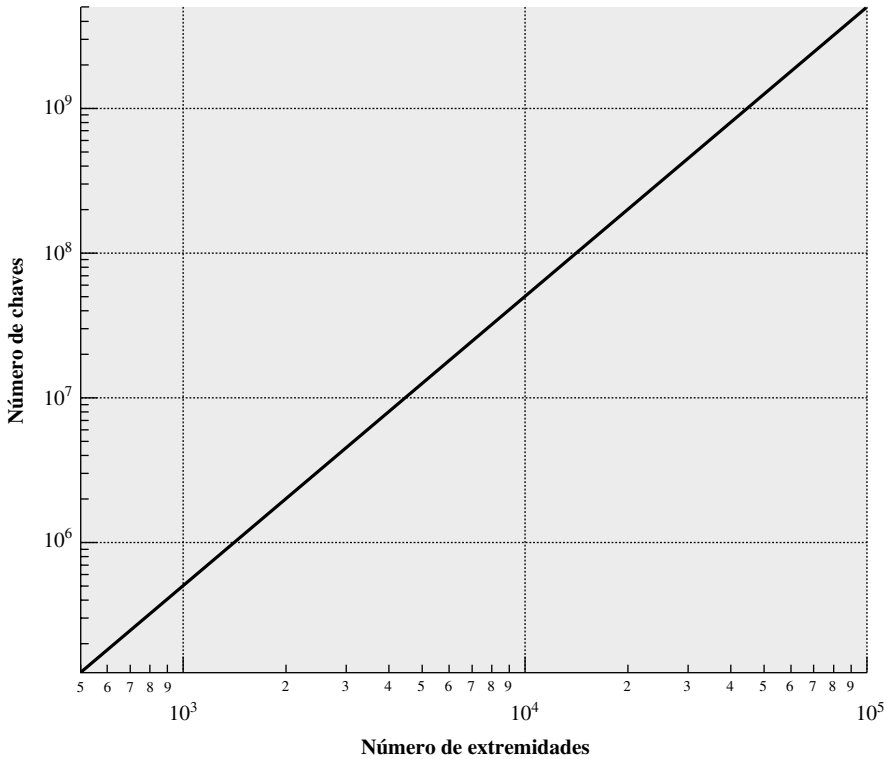


Figura 7.7 Número de chaves exigidas para dar suporte a conexões arbitrárias entre extremidades.

que queiram se comunicar. Assim, se houver N hosts, o número de chaves exigidas é $[N(N - 1)]/2$. Se a criptografia for feita no nível de aplicação, então uma chave será necessária para cada par de usuários ou processos que exijam comunicação. Assim, uma rede pode ter centenas de hosts, mas milhares de usuários e processos. A Figura 7.7 ilustra a magnitude da tarefa de distribuição de chaves para a criptografia de ponta a ponta.[5] Uma rede usando a criptografia em nível de nó com 1000 nós precisaria distribuir até meio milhão de chaves. Se essa mesma rede admitisse dez mil aplicações, então até 50 milhões de chaves poderiam ser necessárias para a criptografia em nível de aplicação.

Retornando à nossa lista, a opção 3 é uma possibilidade para criptografia de enlace ou criptografia de ponta a ponta, mas, se um atacante conseguir acesso a uma chave com sucesso, então todas as chaves subseqüentes serão reveladas. Além disso, a distribuição inicial de potencialmente milhões de chaves ainda precisaria ser feita.

Para a criptografia de ponta a ponta, uma variação da opção 4 foi amplamente adotada. Nesse esquema, um centro de distribuição de chaves é responsável por distribuir chaves a pares de usuários (hosts, processos, aplicações) conforme a necessidade. Cada usuário precisa compartilhar uma chave exclusiva com o centro de distribuição de chaves, para fins de distribuição das chaves.

O uso de um centro de distribuição de chaves é baseado no uso de uma hierarquia de chaves. No mínimo dois níveis de chaves são usados (Figura 7.8). A comunicação entre os sistemas finais é criptografada usando uma chave temporária, normalmente mencionada como **chave de sessão**. Em geral, a chave de sessão é usada durante uma conexão lógica, como uma conexão frame relay ou conexão de transporte, e depois descartada. Cada chave de sessão é obtida a partir do centro de distribuição de chaves pelas mesmas facilidades de rede usadas para a comunicação do usuário final. Por conseguinte, as chaves de sessão são transmitidas em formato criptografado, usando uma **chave-mestra** que é compartilhada pelo centro de distribuição de chaves e o sistema ou usuário final.

Para cada sistema ou usuário final, existe uma chave-mestra exclusiva, que é compartilhada com o centro de distribuição de chaves. Naturalmente, essas chaves-mestras precisam ser distribuídas de alguma maneira. Porém, a escala do problema é bastante reduzida. Se houver N entidades que queiram se comunicar em pares, então, conforme mencionamos, até $[N(N - 1)]/2$ chaves de sessão são necessárias a qualquer momento. Porém, somente N chaves mestras são exigidas, uma para cada entidade. Assim, as chaves mestras podem ser distribuídas de alguma maneira não criptográfica, por exemplo, a entrega física.

5. Observe que essa figura usa uma escala logarítmica, de modo que um gráfico linear indica crescimento exponencial. Uma revisão básica das escalas logarítmicas pode ser encontrada no documento de revisão matemática no Computer Science Student Resource Site, disponível em **WilliamStallings.com/StudentSupport.html**.

Um cenário de distribuição de chaves

O conceito de distribuição de chaves pode ser empregado de diversas maneiras. Um cenário típico é ilustrado na Figura 7.9, que é baseado em uma figura de [POPE79]. O cenário considera que cada usuário compartilha uma chave-mestra exclusiva com o Centro de Distribuição de Chaves (CDC).

Vamos supor que o usuário A queira estabelecer uma conexão lógica com B e exija uma chave de sessão de única vez para proteger os dados transmitidos pela conexão. A tem uma chave-mestra, K_a, conhecida apenas por ele mesmo e pelo CDC; de modo semelhante, B compartilha a chave-mestra K_b com o CDC. Ocorrem as seguintes etapas:

1. A emite uma solicitação ao CDC de uma chave de sessão para proteger uma conexão lógica com B. A mensagem inclui a identidade de A e B, e um identificador exclusivo, N_1, para essa transação, a qual vamos nos referir como **nonce**.[6] O nonce pode ser uma marca de data/hora, um contador ou um número aleatório; o requisito mínimo é que ele seja diferente a cada solicitação. Além disso, também precisa evitar o mascaramento, devendo ser difícil que um oponente descubra o nonce. Assim, um número aleatório é uma boa escolha para um nonce.
2. O CDC responde com uma mensagem criptografada usando K_a. Assim, A é o único que pode ler a mensagem com sucesso, e A sabe que ela foi originada no CDC. A mensagem inclui dois itens destinados a A:
   - A chave de sessão de única vez, K_s, a ser usada para a sessão
   - A mensagem de solicitação original, incluindo o nonce, para permitir que A combine essa resposta com a solicitação apropriada

   Assim, A pode verificar que sua solicitação original não foi alterada antes do recebimento pelo CDC e, devido ao nonce, que essa não é uma repetição de alguma solicitação anterior.
   Além disso, a mensagem inclui dois itens destinados a B:
   - A chave de sessão de única vez, K_s, a ser usada para a sessão
   - Um identificador de A (por exemplo, seu endereço de rede), ID_A

   Esses dois últimos itens são criptografados com K_b (a chave-mestra que o CDC compartilha com B). Eles devem ser enviados a B para estabelecer a conexão e provar a identidade de A.
3. A armazena a chave da sessão para uso na próxima sessão e encaminha a B a informação que se originou no CDC para B, a saber, $E(K_b, [K_s \| ID_A])$. Como essa informação é criptografada com K_b, ela é protegida contra interceptação. B agora conhece a chave da sessão (K_s), sabe que a outra parte é A (de ID_A) e sabe que a informação foi originada no CDC (porque foi criptografada usando K_b).

Nesse ponto, uma chave de sessão foi entregue com segurança a A e B, e as partes podem começar sua troca de mensagens protegida. Porém, duas etapas adicionais são desejáveis:

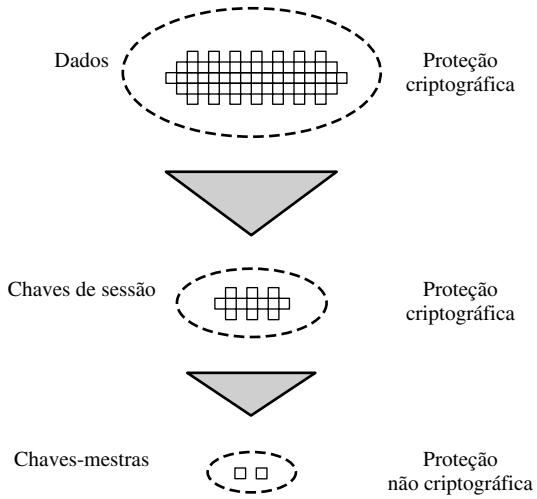


Figura 7.8 O uso de uma hierarquia de chaves.

6. As seguintes definições são úteis para se entender a finalidade do componente nonce. **Nonce:** a ocasião presente ou específica. **Palavra nonce:** uma palavra que ocorre, é inventada ou usada apenas para uma determinada ocasião. Retirado de *American heritage dictionary of the English language*, 3ª ed.

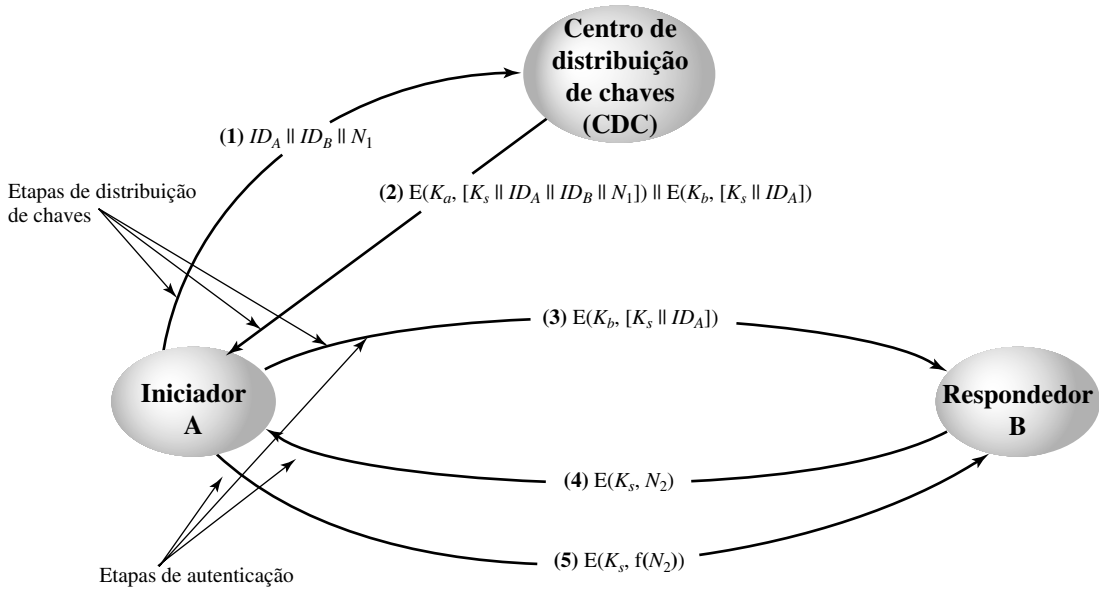


Figura 7.9 Cenário de distribuição de chaves.

4. Usando a chave de sessão recém-criada para a criptografia, B envia um nonce, N_2, para A.
5. Também usando K_s, A responde com $f(N_2)$, onde f é uma função que realiza alguma transformação em N_2 (por exemplo, somando um).

Essas etapas garantem a B que a mensagem original que ele recebeu (etapa 3) não é uma repetição.

Observe que a distribuição real da chave envolve apenas as etapas de 1 a 3, mas que as etapas 4 e 5, além da 3, realizam uma função de autenticação.

Controle hierárquico de chaves

Não é necessário limitar a função de distribuição de chave a um único CDC. Na realidade, para redes muito amplas, pode não ser prático fazer isso. Como alternativa, uma hierarquia de CDCs pode ser estabelecida. Por exemplo, pode haver CDCs locais, cada um responsável por um pequeno domínio do conjunto total de redes, como uma única LAN ou um único prédio. Para a comunicação entre entidades dentro do mesmo domínio local, o CDC local é responsável pela distribuição de chaves. Se duas entidades em domínios diferentes desejarem uma chave compartilhada, então os CDCs locais correspondentes podem se comunicar por um CDC global. Nesse caso, qualquer um dos três CDCs envolvidos pode realmente selecionar a chave. O conceito hierárquico pode ser estendido a três ou até mesmo a mais camadas, dependendo do tamanho da população de usuários e do escopo geográfico do conjunto de redes interligadas.

Um esquema hierárquico minimiza o efeito que ocorre na distribuição de chaves-mestras, pois a maioria delas são compartilhadas por um CDC local com suas entidades locais. Além disso, esse esquema limita os danos de um CDC defeituoso ou subvertido apenas à sua área local.

Tempo de vida da chave da sessão

Quanto mais freqüentemente as chaves de sessão forem trocadas, mais seguras elas serão, pois o oponente terá menos texto cifrado para trabalhar com uma chave de sessão qualquer. Por outro lado, a distribuição das chaves de sessão atrasa o início de qualquer troca e impõe uma carga sobre a capacidade da rede. Um gerente de segurança precisa tentar equilibrar essas considerações concorrentes ao determinar o tempo de vida de determinada chave de sessão.

Para protocolos orientados a conexão, uma escolha óbvia é usar a mesma chave de sessão para o período de tempo em que a conexão está aberta, usando uma nova chave de sessão para cada nova sessão. Se uma conexão lógica tiver um tempo de vida muito longo, então é prudente alterar a chave de sessão periodicamente, talvez toda vez que o número de seqüência da PDU (unidade de dados do protocolo) mudar.

Para um protocolo sem conexão, como um protocolo orientado a transação, não existe início ou término de conexão explícito. Assim, não é óbvia a freqüência em que é necessário alterar a chave da sessão. A técnica mais segura é usar uma nova chave de sessão para cada troca. Porém, isso anula um dos principais benefícios dos protocolos não orientados a conexão, que é overhead e atraso mínimos para cada transação. A melhor estratégia é usar uma determinada chave de sessão para um certo período fixo apenas por um certo número de transações.

Um esquema transparente de controle de chaves

A técnica sugerida na Figura 7.9 tem muitas variações, uma das quais é descrita nesta subseção. O esquema (Figura 7.10) é útil para oferecer criptografia de ponta a ponta em um nível de rede ou transporte, de uma maneira que seja transparente aos usuários finais. A técnica considera que a comunicação utiliza um protocolo de ponta a ponta orientado a conexão, como TCP. O elemento digno de nota dessa técnica é um módulo de segurança de sessão (SSM), que pode consistir em funcionalidade em uma camada de protocolo, que realiza criptografia de ponta a ponta e obtém chaves de sessão em favor de seu host ou terminal.

As etapas envolvidas no estabelecimento de uma conexão aparecem na figura. Quando um host deseja estabelecer uma conexão com outro host, ele transmite um pacote de solicitação de conexão (etapa 1). O SSM salva esse pacote e solicita ao CDC permissão para estabelecer a conexão (etapa 2). A comunicação entre o SSM e o CDC é criptografada usando uma chave-mestra compartilhada apenas por esse SSM e o CDC. Se o CDC aprovar a solicitação de conexão, ele gera a chave de sessão e a entrega aos dois SSMs apropriados, usando uma chave permanente exclusiva para cada SSM (etapa 3). O SSM solicitante agora pode liberar o pacote de solicitação de conexão, e uma conexão é estabelecida entre os dois sistemas finais (etapa 4). Todos os dados do usuário trocados entre os dois sistemas finais são criptografados por seus respectivos SSMs usando a chave de sessão de única vez.

A técnica de distribuição de chaves automatizada oferece a flexibilidade e as características dinâmicas necessárias para permitir que uma série de outros usuários acessem diversos hosts e que os hosts troquem dados entre si.

Controle de chaves descentralizado

O uso de um centro de distribuição de chaves impõe o requisito de que o CDC seja confiável e protegido contra subversão. Esse requisito pode ser evitado se a distribuição de chaves for totalmente descentralizada. Embora a descentralização total não seja prática para redes amplas usando apenas a criptografia simétrica, ela pode ser útil dentro de um contexto local.

A técnica descentralizada exige que cada sistema final seja capaz de se comunicar de maneira segura com todos os sistemas finais de parceiros em potencial, para fins de distribuição de chaves de sessão. Assim, pode ser preciso que haja até $[n(n-1)]/2$ chaves-mestras para uma configuração com n sistemas finais.

Uma chave de sessão pode ser estabelecida com a seguinte seqüência de etapas (Figura 7.11):

1. A emite uma solicitação a B de uma chave de sessão e inclui um nonce N_1.
2. B responde com uma mensagem criptografada usando a chave mestra compartilhada. A resposta inclui a chave de sessão selecionada por B, um identificador de B, o valor $f(N_1)$ e outro nonce, N_2.
3. Usando a nova chave de sessão, A retorna $f(N_2)$ a B.

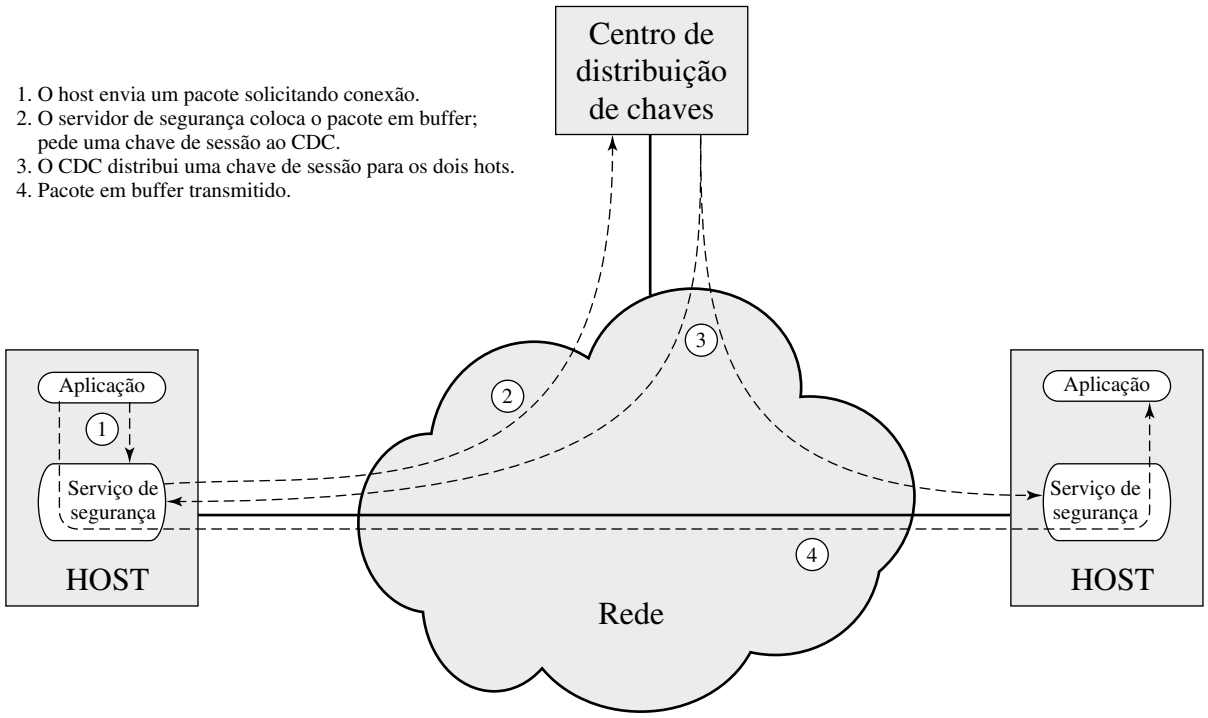


Figura 7.10 Distribuição automática de chaves para protocolo orientado a conexão.

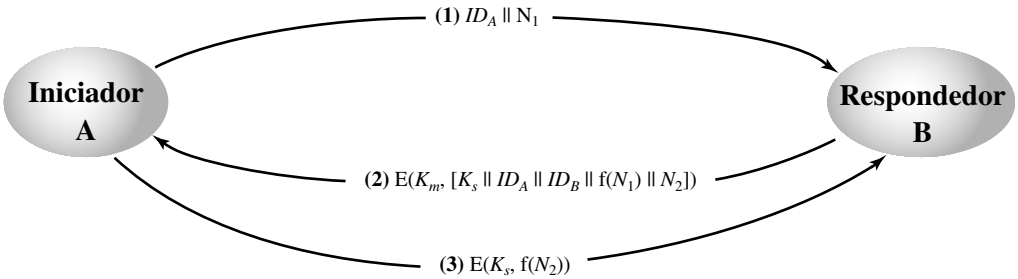


Figura 7.11 Distribuição descentralizada de chaves.

Assim, embora cada nó deva manter no máximo $(n - 1)$ chaves-mestras, tantas chaves de sessão quantas forem necessárias podem ser geradas e usadas. Como as mensagens transferidas usando a chave-mestra são curtas, a criptoanálise é difícil. Como antes, as chaves de sessão são usadas apenas por um tempo limitado para protegê-las.

Controlando o uso da chave

O conceito de uma hierarquia de chaves e o uso de técnicas automatizadas de distribuição de chaves reduz bastante o número de chaves que precisam ser gerenciadas e distribuídas manualmente. Também pode ser desejável impor algum controle sobre a maneira como a distribuição automática de chaves é utilizada. Por exemplo, além de separar chaves-mestras de chaves de sessão, podemos desejar definir diferentes tipos de chaves de sessão com base no uso, como

- Chave de criptografia de dados para comunicação geral por uma rede
- Chave de criptografia de PIN para números de identificação pessoal (PINs) usados em aplicações de transferência eletrônica de fundos e aplicações de ponto-de-venda
- Chave de criptografia de arquivo para criptografar arquivos armazenados em locais acessíveis publicamente

Para ilustrar o valor da separação de chaves por tipo, considere o risco de que uma chave-mestra seja importada como uma chave de criptografia de dados em um dispositivo. Normalmente, a chave-mestra é protegida fisicamente dentro do hardware criptográfico do centro de distribuição de chaves e dos sistemas finais. As chaves de sessão criptografadas com essa chave-mestra estão disponíveis aos programas de aplicação, assim como os dados criptografados com essas chaves de sessão. Porém, se uma chave-mestra for tratada como uma chave de sessão, pode ser possível que uma aplicação não autorizada obtenha o texto claro das chaves de sessão criptografadas com essa chave-mestra.

Assim, pode ser desejável instituir controles nos sistemas para limitar as maneiras como as chaves são usadas, com base nas características associadas a essas chaves. Um plano simples é associar uma etiqueta a cada chave ([JONE82]; ver também [DAVI89]). A técnica proposta é para uso com DES e utiliza os 8 bits extras em cada chave DES de 64 bits. Ou seja, os 8 bits que não fazem parte da chave normalmente reservados para verificação de paridade formam a etiqueta da chave. Os bits têm a seguinte interpretação:

- Um bit indica se a chave é uma chave de sessão ou uma chave-mestra.
- Um bit indica se a chave pode ser usada para criptografia.
- Um bit indica se a chave pode ser usada para decriptografia.
- Os bits restantes são reservados para uso futuro.

Como a etiqueta está embutida na chave, ela é criptografada com ela, quando essa chave é distribuída, oferecendo assim a proteção. As desvantagens desse esquema são que (1) o tamanho da etiqueta é limitado a 8 bits, limitando sua flexibilidade e funcionalidade; e (2) como a etiqueta não é transmitida de forma clara, ela só pode ser usada no ponto de decriptografia, limitando as maneiras como o uso da chave pode ser controlado.

Um esquema mais flexível, chamado vetor de controle, é descrito em [MATY91a e b]. Nesse esquema, cada chave de sessão tem um vetor de controle associado, consistindo em uma série de campos que especificam os usos e as restrições para essa chave de sessão. A extensão do vetor de controle pode variar.

O vetor de controle é acoplado criptograficamente à chave no momento de sua geração no CDC. Os processos de acoplamento e desacoplamento são ilustrados na Figura 7.12. Como primeiro passo, o vetor de controle é passado por uma função de hash, que produz um valor cujo tamanho é igual ao tamanho da chave de criptografia. As funções de hash são explicadas mais detalhadamente no Capítulo 11. Basicamente, uma função de hash compara os valores de um intervalo maior com os de um intervalo menor, com um espalhamento razoavelmente uniforme. Assim, por exemplo, se for aplicada uma função de hash nos números no intervalo de 1 a 100 para os números no intervalo de 1 a 10, aproximadamente, dez por cento dos valores de origem deverão ser relacionados a cada um dos valores de destino.

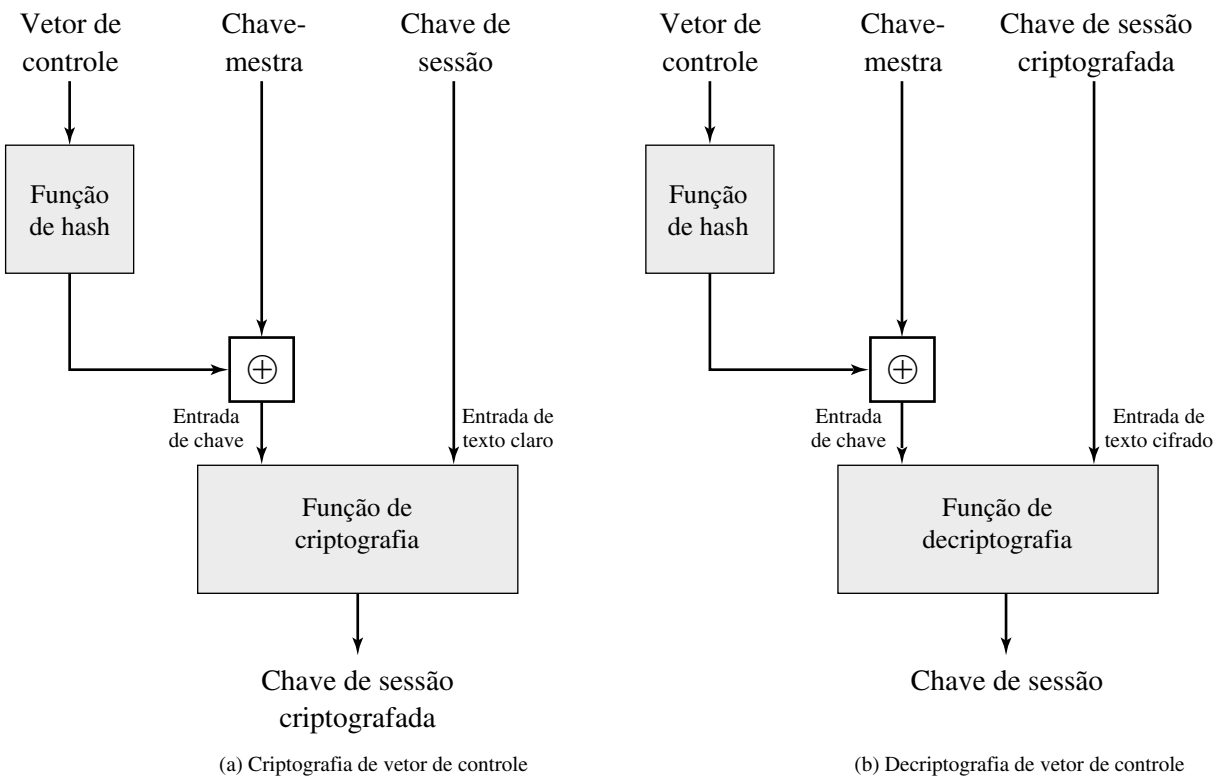


Figura 7.12 Criptografia e decriptografia do vetor de controle.

O valor de hash, então, passa por uma função XOR com a chave-mestra para produzir uma saída usada como entrada de chave para criptografar a chave de sessão. Assim,

$$\text{Valor de hash} = H = \text{h(CV)}$$
$$\text{Entrada da chave} = K_m \oplus H$$
$$\text{Texto cifrado} = \text{E}([K_m \oplus H], K_s)$$

onde K_m é a chave-mestra e K_s é a chave de sessão. A chave de sessão é recuperada em texto claro pela operação inversa:

$$\text{D}([K_m \oplus H], \text{E}([K_m \oplus H], K_s))$$

Quando uma chave de sessão é entregue a um usuário a partir do CDC, ela é acompanhada pelo vetor de controle de forma clara. A chave da sessão só pode ser recuperada usando a chave-mestra que o usuário compartilha com o CDC e o vetor de controle. Assim, a ligação entre a chave de sessão e seu vetor de controle é mantida.

O uso do vetor de controle tem duas vantagens em relação ao uso de uma etiqueta de 8 bits. Primeiro, não existe restrição quanto ao tamanho do vetor de controle, o que permite que controles arbitrariamente complexos sejam impostos sobre o uso da chave. Segundo, o vetor de controle está disponível de forma clara em todos os estágios da operação. Assim, o controle do uso da chave pode ser exercido em vários locais.

7.4 GERAÇÃO DE NÚMEROS ALEATÓRIOS

Os números aleatórios desempenham um papel importante no uso da criptografia para diversas aplicações de segurança da rede. Nesta seção, oferecemos uma rápida visão geral do uso de números aleatórios na segurança da rede e depois examinamos algumas técnicas para geração de números aleatórios.

O uso de números aleatórios

Diversos algoritmos de segurança de rede baseados na criptografia utilizam números aleatórios. Por exemplo,

- Esquemas de autenticação recíprocos, conforme ilustrado nas Figuras 7.9 e 7.11. Nesses dois cenários de distribuição de chaves, nonces são usados para o *handshaking*, para impedir ataques por repetição. O uso de números aleatórios para os nonces frustra os esforços dos oponentes para determinar ou adivinhar o nonce.
- Geração de chaves de sessão, seja por um centro de distribuição de chaves ou por um dos membros.
- Geração de chaves para o algoritmo de criptografia de chave pública RSA (descrito no Capítulo 9).

Essas aplicações fazem surgir dois requisitos distintos e não necessariamente compatíveis para uma seqüência de números aleatórios: aleatoriedade e imprevisibilidade.

Aleatoriedade Tradicionalmente, a preocupação na geração de uma seqüência de números supostamente aleatórios tem sido de que a seqüência de números seja aleatória em algum sentido estatístico bem definido. Os dois critérios a seguir são usados para validar se uma seqüência de números é aleatória:

- **Distribuição uniforme:** A distribuição dos números na seqüência deve ser uniforme, ou seja, a freqüência de ocorrência de cada um dos números deve ser aproximadamente a mesma.
- **Independência:** Nenhum valor na seqüência pode ser deduzido a partir dos outros.

Embora existam testes bem definidos para determinar se uma seqüência de números combina com uma distribuição em particular, como a distribuição uniforme, não existe um teste para 'provar' a independência. Em vez disso, diversos testes podem ser aplicados para demonstrar se uma seqüência não exibe independência. A estratégia geral é aplicar diversos desses testes até que haja certeza de que existe independência.

No contexto da nossa explicação, o uso de uma seqüência de números que parecem ser estatisticamente aleatórios normalmente ocorre no projeto de algoritmos relacionados com a criptografia. Por exemplo, um requisito fundamental do esquema de criptografia de chave pública RSA explicado no Capítulo 9 é a capacidade de gerar números primos. Em geral, é difícil determinar se determinado número grande N é primo. Uma técnica de força bruta seria dividir N por cada inteiro ímpar menor que $\sqrt{N}$. Se N estiver na ordem de, digamos, 10^{150}, uma ocorrência não rara na criptografia de chave pública, essa abordagem de força bruta estará além do alcance dos analistas humanos e seus computadores. Porém, existem diversos algoritmos eficazes que testam se um número é primo usando uma seqüência de inteiros escolhidos aleatoriamente como entrada para cálculos relativamente simples. Se a seqüência for suficientemente longa (muito menor que $\sqrt{10^{150}}$), a verificação de se um número é primo pode ser determinada quase com certeza. Esse tipo de técnica, conhecida como aleatoriedade, surge freqüentemente no projeto de algoritmos. Basicamente, se um problema for muito difícil ou demorado para ser solucionado exatamente, uma técnica mais simples e mais curta, baseada na aleatoriedade, é usada para fornecer uma resposta com qualquer nível de confiança desejado.

Imprevisibilidade Em aplicações como autenticação recíproca e geração de chaves de sessão, o requisito não é tanto se a seqüência de números é estatisticamente aleatória, mas se os membros sucessivos da seqüência são imprevisíveis. Com 'verdadeiras' seqüências aleatórias, cada número é estatisticamente independente dos outros números na seqüência e, portanto, imprevisíveis. Porém, conforme explicamos rapidamente, verdadeiros números aleatórios raramente são usados; em vez disso, seqüências de números que parecem ser aleatórios são geradas por algum algoritmo. Nesse último caso, deve-se ter cuidado para que um oponente não seja capaz de prever elementos futuros da seqüência com base nos elementos anteriores.

Geradores de números pseudo-aleatórios (PRNGs) Aplicações criptográficas utilizam técnicas algorítmicas para geração de números aleatórios. Esses algoritmos são determinísticos e, portanto, produzem seqüências de números que não são estatisticamente aleatórios. Porém, se o algoritmo for bom, as seqüências resultantes passarão por muitos testes razoáveis de aleatoriedade. Tais números são chamados de **números pseudo-aleatórios**.

Você pode estar um pouco preocupado com o conceito do uso de números gerados por um algoritmo determinístico como se todos fossem números aleatórios. Apesar do que pode ser chamado de objeções filosóficas para tal prática, isso geralmente funciona. Conforme um especialista em teoria da probabilidade explica [HAMM91]:

> Para fins práticos, somos forçados a aceitar o estranho conceito de 'relativamente aleatório', significando que, com relação ao uso proposto, não podemos ver por que não funcionarão como se fossem aleatórios (como a teoria geralmente requer). Isso é altamente subjetivo e não é muito aceito pelos puristas, mas é para isso que os estatísticos apelam regularmente quando coletam 'uma amostra aleatória' — eles esperam que quaisquer resultados que utilizem tenham aproximadamente as mesmas propriedades de uma contagem completa do espaço total da amostra que ocorre em sua teoria.

Geradores congruenciais lineares

De longe, a técnica mais usada para a geração de números pseudo-aleatórios é um algoritmo proposto inicialmente por Lehmer [LEHM51], conhecido como método de congruência linear. O algoritmo é parametrizado com quatro números, da seguinte maneira:

m	o módulo	$m > 0$
a	o multiplicador	$0 < a < m$
c	o incremento	$0 \leq c < m$
X_0	o valor inicial, ou semente	$0 \leq X_0 < m$

A seqüência de números aleatórios $\{X_n\}$ é obtida por meio da seguinte equação iterativa:

$$X_{n+1} = (aX_n + c) \bmod m$$

Se m, a, c e X_0 são inteiros, então essa técnica produzirá uma seqüência de inteiros com cada inteiro no intervalo $0 \leq X_n < m$.

A seleção de valores para a, c e m é crítica no desenvolvimento de um bom gerador de números aleatórios. Por exemplo, considere $a = c = 1$. A seqüência produzida obviamente não é satisfatória. Agora, considere os valores $a = 7, c = 0, m = 32$ e $X_0 = 1$. Isso gera a seqüência $\{7, 17, 23, 1, 7, \text{etc.}\}$, que claramente também é insatisfatória. Dos 32 valores possíveis, somente 4 são usados; assim, a seqüência é considerada como tendo período 4. Se, em vez disso, mudarmos o valor de a para 5, então a seqüência será $\{5, 25, 29, 17, 21, 9, 13, 1, 5, \text{etc.}\}$, o que aumenta o período para 8.

Queremos que m seja muito grande, de modo que haja o potencial para produzir uma longa série de números aleatórios distintos. Um critério comum é que m seja quase igual ao máximo inteiro não negativo representável para um determinado computador. Assim, normalmente é escolhido um valor m próximo ou igual a 2^{31}.

[PARK88] propõe três testes para avaliar um gerador de números aleatórios:

T_1: A função deve ser uma função de geração de período completo, ou seja, a função deve gerar todos os números entre 0 e m antes de repetir.

T_2: A seqüência gerada deve parecer aleatória. Como ela é gerada deterministicamente, a seqüência não é aleatória. Existe uma série de testes estatísticos que podem ser usados para avaliar o grau de aleatoriedade de uma seqüência.

T_3: A função deverá ser implementada eficientemente com aritmética de 32 bits.

Com valores apropriados de a, c e m, pode-se passar por esses três testes. Com relação a T_1, pode-se mostrar que, se m é primo e $c = 0$, então, para certos valores de a, o período da função de geração é $m - 1$, apenas com o valor 0 faltando. Para a aritmética de 32 bits, um valor primo conveniente de m é $2^{31} - 1$. Assim, a função de geração torna-se

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

De mais de 2 bilhões de escolhas possíveis para a, somente alguns dos multiplicadores passam por todos os três testes. Um valor desse tipo é $a = 7^5 = 16807$, que foi projetado originalmente para uso na família de computadores IBM 360 [LEWI69]. Esse gerador é amplamente utilizado e esteve sujeito a um teste mais completo do que qualquer outro PRNG. Ele é freqüentemente recomendado para o trabalho estatístico e de simulação (por exemplo, [JAIN91], [SAUE81]).

A força do algoritmo de congruência linear é que, se o multiplicador e o módulo forem escolhidos corretamente, a seqüência de números resultante será estatisticamente indistinguível de uma seqüência retirada aleatoriamente (mas sem substituição) do conjunto $1, 2, ..., m - 1$. Mas não existe nada de aleatório sobre o algoritmo, além da escolha do valor inicial X_0. Quando esse valor é escolhido, os números restantes na seqüência seguem de forma determinística. Isso possui implicações para a criptoanálise.

Se um oponente souber que o algoritmo de congruência linear está sendo usado e se os parâmetros forem conhecidos (por exemplo, $a = 7^5, c = 0, m = 2^{31} - 1$), então, quando um único número for descoberto, todos os números subseqüentes serão conhecidos. Mesmo que o oponente saiba apenas que um algoritmo de congruência linear está sendo usado, o conhecimento de uma pequena parte da seqüência é suficiente para determinar os parâmetros do algoritmo. Suponha que o oponente seja capaz de determinar os valores para X_0, X_1, X_2 e X_3. Então

$$X_1 = (aX_0 + c) \bmod m$$
$$X_2 = (aX_1 + c) \bmod m$$
$$X_3 = (aX_2 + c) \bmod m$$

Essas equações podem ser solucionadas para a, c e m.

Assim, embora seja proveitoso ser capaz de usar um bom PRNG, é desejável fazer com que a seqüência real utilizada seja não reproduzível, de modo que o conhecimento de parte da seqüência por um oponente seja insuficiente

para determinar os elementos futuros da seqüência. Esse objetivo pode ser alcançado de várias maneiras. Por exemplo, [BRIG79] sugere o uso de um clock interno do sistema para modificar o fluxo de números aleatórios. Uma forma de usar o clock seria reiniciar a seqüência depois de cada N números, usando o valor atual de clock (mod m) como nova semente. Outra forma seria simplesmente adicionar o valor atual de clock a cada número aleatório (mod m).

Números aleatórios gerados criptograficamente

Para aplicações criptográficas, faz sentido tirar proveito da lógica de criptografia disponível para produzir números aleatórios. Diversas maneiras foram usadas, e nesta subseção examinamos três exemplos representativos.

Criptografia cíclica A Figura 7.13 ilustra uma técnica sugerida em [MEYE82]. Nesse caso, o procedimento é usado para gerar chaves de sessão a partir de uma chave-mestra. Um contador com período N oferece entrada para a lógica de criptografia. Por exemplo, se chaves DES de 56 bits tiverem de ser produzidas, então um contador com período 2^{56} poderá ser usado. Após a produção de cada chave, o contador é incrementado em um. Assim, os números pseudo-aleatórios produzidos por esse esquema percorrem um período completo: cada uma das saídas $X_0, X_1, \ldots X_{N-1}$ é baseada em um valor de contador diferente e, portanto, $X_0 \neq X_1 \neq \ldots \neq X_{N-1}$. Como a chave mestra está protegida, não é computacionalmente viável deduzir qualquer uma das chaves de sessão (números aleatórios) pelo conhecimento de uma ou mais chaves de sessão anteriores.

Para fortalecer o algoritmo ainda mais, a entrada poderia ser a saída de um período completo de um PRNG, em vez de um simples contador.

Modo Output Feedback do DES O modo Output Feedback (OFB) do DES, ilustrado na Figura 6.6, pode ser usado para a geração de chaves, bem como para a criptografia de fluxo. Observe que a saída de cada estágio da operação é um valor de 64 bits, do qual os s bits mais à esquerda são retroalimentados para criptografia. Saídas sucessivas de 64 bits constituem uma seqüência de números pseudo-aleatórios com boas propriedades estatísticas. Novamente, como na técnica sugerida na subseção anterior, o uso de uma chave-mestra protegida protege as chaves de sessão geradas.

ANSI X9.17 PRNG Um dos PRNGs mais fortes (criptograficamente falando) é especificado no ANSI X9.17. Diversas aplicações empregam essa técnica, incluindo aplicações de segurança financeira e PGP (a última descrita no Capítulo 15).

A Figura 7.14 ilustra o algoritmo que utiliza o 3DES (Triple DES) para a criptografia. Os ingredientes são os seguintes:

- **Entrada:** Duas entradas pseudo-aleatórias controlam o gerador. Uma é a representação de 64 bits da data e hora atuais, que é atualizada a cada geração de número. A outra é um valor de semente de 64 bits; esta é inicializada em algum valor arbitrário e atualizada durante o processo de geração.
- **Chaves:** O gerador utiliza os três módulos de criptografia do 3DES. Todos os três utilizam algum par de chaves de 56 bits, que precisam ser mantidas secretas e são usadas apenas para geração de números pseudo-aleatórios.

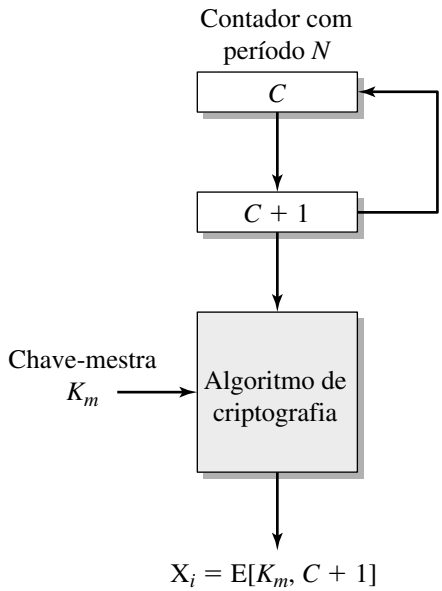


Figura 7.13 Geração de um número pseudo-aleatório a partir de um contador.

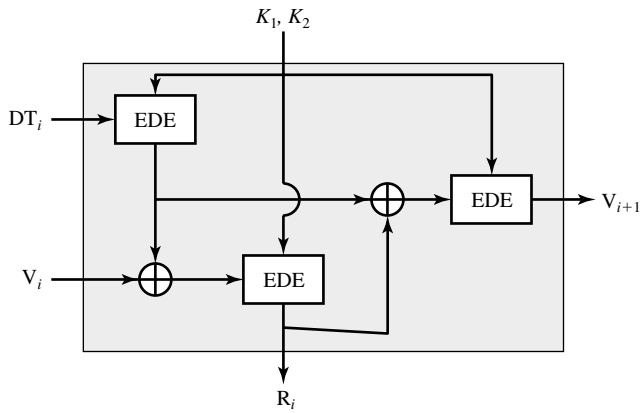


Figura 7.14 Gerador de números pseudo-aleatórios ANSI X9.17.

- **Saída:** A saída consiste em um número pseudo-aleatório de 64 bits e um valor de semente de 64 bits.

  Defina as seguintes quantidades:

  DT_i Valor de data/hora no início do *i*-ésimo estágio de geração
  V_i Valor da semente no início do *i*-ésimo estágio de geração
  R_i Número pseudo-aleatório produzido pelo *i*-ésimo estágio de geração
  K_1, K_2 Chaves DES usadas para cada estágio

Então

$$R_i = \text{EDE}([K_1, K_2], [V_i \oplus \text{EDE}([K_1, K_2], \text{DT}_i)])$$
$$V_{i+1} = \text{EDE}([K_1, K_2], [R_i \oplus \text{EDE}([K_1, K_2], \text{DT}_i)])$$

onde EDE([K_1, K_2], X) refere-se à seqüência criptografar-decriptografar-criptografar usando o 3DES com duas chaves para criptografar X.

Vários fatores contribuem para a força criptográfica desse método. A técnica envolve uma chave de 112 bits e três criptografias EDE para um total de nove criptografias DES. O esquema é controlado por duas entradas pseudo-aleatórias, o valor de data e hora e uma semente produzida pelo gerador, que é distinta do número pseudo-aleatório produzido pelo gerador. Assim, a quantidade de material que precisa ser comprometida por um oponente é esmagadora. Mesmo que um número pseudo-aleatório R_i fosse comprometido, seria impossível deduzir o V_{i+1} a partir do R_i, pois a operação EDE adicional é usada para produzir o V_{i+1}.

Gerador Blum Blum Shub

Uma técnica popular para gerar números pseudo-aleatórios seguros é conhecida como gerador Blum, Blum, Shub (BBS), que tem o nome de seus desenvolvedores [BLUM86]. Talvez ele tenha a prova pública mais forte de sua força criptográfica. O procedimento é descrito a seguir. Primeiro, escolha dois números primos grandes, p e q, ambos tendo um resto 3 quando dividido por 4. Ou seja,

$$p \equiv q \equiv 3 \pmod 4$$

Essa notação, explicada com mais detalhes no Capítulo 4, simplesmente significa que (p mod 4) = (q mod 4) = 3. Por exemplo, os números primos 7 e 11 satisfazem $7 \equiv 11 \equiv 3 \pmod 4$. Considere que $n = p \times q$. Em seguida, escolha um número aleatório s, de modo que s seja relativamente primo de n; isso é equivalente a dizer que nem p nem q são fatores de s. Então, o gerador BBS produz uma seqüência de bits B_i de acordo com o seguinte algoritmo:

```
X0 = s^2 mod n
for i = 1 to ∞
    Xi = (Xi-1)^2 mod n
    Bi = Xi mod 2
```

Assim, o bit menos significativo é obtido em cada iteração. A Tabela 7.2 mostra um exemplo da operação do BBS. Aqui, $n = 192649 = 383 \times 503$ e a semente $s = 101355$.

Tabela 7.2 Exemplo de operação de um gerador BBS[PFLE02]

i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0

i	X_i	B_i
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

O BBS é conhecido como um **gerador de bits pseudo-aleatórios criptograficamente seguro** (CSPRBG). Um CSPRBG é definido como um gerador que passa no teste *next-bit* que, por sua vez, é definido da seguinte forma [MENE97]: diz-se que um gerador de bits pseudo-aleatório passa no teste *next-bit* se não houver um algoritmo de tempo polinomial[7] que, na entrada dos primeiros k bits de uma seqüência de saída, possa prever o $(k + 1)^{\text{ésimo}}$ bit com probabilidade significativamente maior do que 1/2. Para todos os fins práticos, a seqüência é imprevisível. A segurança do BBS é baseada na dificuldade de fatorar n. Ou seja, dado n, precisamos determinar seus dois fatores primos p e q.

Geradores de números aleatórios verdadeiros

O gerador de números aleatórios verdadeiros(TRNG) utiliza uma fonte não determinística para criar aleatoriedade. A maioria opera medindo processos naturais imprevisíveis, como detectores de pulso de radiação ionizante, tubos de descarga de gás e capacitores com fuga de corrente. A Intel desenvolveu um chip disponível comercialmente que coleta amostras de ruído térmico, amplificando a voltagem medida em resistores não ativados [JUN99]. Um grupo no Bell Labs desenvolveu uma técnica que utiliza as variações do tempo de resposta de operações de leitura de um setor do disco rígido [JAKO98]. LavaRnd é um projeto de código-fonte aberto para criar números verdadeiramente aleatórios usando câmeras e hardware baratos e código-fonte aberto. O sistema utiliza um CCD saturado em uma câmera escura como uma fonte caótica para produzir a semente. O software processa o resultado e o transforma em números verdadeiramente aleatórios em diversos formatos.

Existem problemas tanto de aleatoriedade quanto de precisão desses números [BRIG79], sem falar na dificuldade de conectar um dispositivo desses a cada sistema em uma rede. Outra opção é a extração a partir de um conjunto publicado de números aleatórios de boa qualidade (por exemplo, [RAND55], [TIPP27]). Porém, esses conjuntos fornecem uma fonte muito limitada de números comparada aos requisitos em potencial de uma aplicação considerável de segurança de rede. Além disso, embora os números nesses livros realmente exibam aleatoriedade estatística, eles são previsíveis, pois um oponente que saiba que o livro está em uso pode obter uma cópia.

Propensão

Um gerador de números aleatórios verdadeiros pode produzir uma saída tendenciosa de alguma maneira, por exemplo, tendo mais uns do que zeros ou vice-versa. Diversos métodos de modificação de um fluxo de bits para reduzir ou eliminar a propensão foram desenvolvidos. Esses são conhecidos como *algoritmos antipropensão* (deskewing algorithms). Uma técnica para evitar a propensão é passar o fluxo de bits por uma função de hash, como MD5 ou SHA-1 (descrita na Parte 2). A função de hash produz uma saída de n bits a partir de uma entrada de qualquer tamanho. Para a antipropensão, blocos de m bits de entrada, com $m \geq n$, podem ser passados pela função de hash.

7.5 LEITURA E SITES WEB RECOMENDADOS

[FUMY93] é um bom estudo sobre princípios de gerenciamento de chaves.

Talvez a melhor abordagem sobre PRNGs seja encontrada em [KNUT98]. Uma alternativa ao algoritmo de congruência linear, conhecido como algoritmo de recorrência linear, é explicada com detalhes em [BRIG79]. [ZENG91] avalia diversos algoritmos PRNG para uso na geração de chaves de tamanho variável para cifras do tipo Vernam.

[7]Um algoritmo de tempo polinomial de ordem k é aquele cujo tempo de execução está vinculado por um polinômio de ordem k.

Um excelente estudo sobre PRNGs, com uma extensa bibliografia, é [RITT91]. [MENE97] também oferece uma boa explicação sobre PRNGs seguros. Outra boa abordagem, enfatizando os aspectos práticos da implementação, é a RFC 1750. Essa RFC também descreve diversas técnicas antipropensão. [KELS98] é um bom estudo sobre técnicas de PRNG seguras e ataques criptoanalíticos a elas.

BRIG79 Bright, H. e Enison, R. "Quasi-random number sequences from long-period TLP generator with remarks on application to cryptography". *Computing Surveys*, dezembro de 1979.
FUMY93 Fumy, S. e Landrock, P. "Principles of key management". *IEEE Journal on Selected Areas in Communications*, junho de 1993.
KELS98 Kelsey, J.; Schneier, B.; e Hall, C. "Cryptanalytic attacks on pseudorandom number generators". *Proceedings, Fast Software Encryption*, 1998. **http://www.schneier.com/paper-prngs.html**
KNUT98 Knuth, D. *The art of computer programming, volume 2: seminumerical algorithms*. Reading: Addison-Wesley, 1998.
MENE97 Menezes, A.; Oorshcot, P.; e Vanstone, S. *Handbook of applied cryptography*. Boca Raton: CRC, 1997.
RITT91 Ritter, T. "The efficient generation of cryptographic confusion sequences". *Cryptologia*, v. 15 n. 2, 1991. **http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM**
ZENG91 Zeng. K.; Yang, C.; Wei, D.; e Rao, T. "Pseudorandom bit generators in stream-cipher cryptography". *Computer*, fevereiro de 1991.

Sites Web recomendados:

- **Random Number Generation Technical Working Group:** Contém documentos e testes desenvolvidos pelo NIST, relacionados a PRNGs, para aplicações criptográficas. Também possui conjuntos de links úteis.
- **LavaRnd:** LavaRnd é um projeto de código-fonte aberto que utiliza uma fonte caótica para gerar números verdadeiramente aleatórios. O site também possui informações básicas sobre números aleatórios em geral.
- **A Million Random Digits:** Compilado pela RAND Corporation.

7.6 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

antipropensão
canal secreto
centro de distribuição de chaves (CDC)
chave de sessão
chave-mestra
congruência linear
criptografia de enlace
criptografia de ponta a ponta
distribuição de chaves
gerador Blum, Blum, Shub
gerador de números aleatórios verdadeiros
gerador de números pseudo-aleatório (PRNG)
nonce
preenchimento de tráfego
propensão
quadro de distribuição

Perguntas de revisão

7.1 Para uma estação de trabalho de usuário em um ambiente comercial típico, liste os locais para ataques de confidencialidade em potencial.
7.2 Qual é a diferença entre criptografia de enlace e de ponta a ponta?
7.3 Que tipos de informação poderiam ser derivados de um ataque de análise de tráfego?
7.4 O que é preenchimento de tráfego e qual é a sua finalidade?
7.5 Liste algumas maneiras como as chaves secretas podem ser distribuídas para duas partes que estão se comunicando.
7.6 Qual é a diferença entre uma chave de sessão e uma chave-mestra?
7.7 O que é um nonce?
7.8 O que é um centro de distribuição de chaves?
7.9 Qual é a diferença entre aleatoriedade estatística e imprevisibilidade?

Problemas

7.1 Os sistemas de e-mail diferem na maneira como os vários destinatários são tratados. Em alguns sistemas, o gerenciador de correio de origem faz todas as cópias necessárias, e estas são enviadas independentemente. Uma técnica alternativa é determinar primeiro a rota para cada destino. Depois, uma única

mensagem é enviada em uma parte comum da rota, e as cópias são feitas apenas quando as rotas divergirem; esse processo é conhecido como *mail bagging*.

a. Deixando de lado as considerações de segurança, explique as vantagens e desvantagens relativas dos dois métodos.
b. Explique os requisitos e as implicações de segurança dos dois métodos.

7.2 A Seção 7.2 descreve o uso do tamanho da mensagem como um meio para a construção de um canal secreto. Descreva três esquemas adicionais para usar os padrões de tráfego para a construção de um canal secreto.

7.3 Um fornecedor de rede local oferece um recurso de distribuição de chaves, conforme ilustrado na Figura 7.15.

a. Descreva o esquema.
b. Compare esse esquema com o da Figura 7.9. Quais são os prós e os contras?

7.4 "Estamos sob grande pressão, Holmes". O detetive Lestrade parecia nervoso. "Descobrimos que cópias de documentos confidenciais do governo estão armazenadas em computadores de uma embaixada estrangeira em Londres. Normalmente, esses documentos estão apenas em formato eletrônico em alguns poucos computadores selecionados do governo, que satisfazem os requisitos de segurança mais severos. Porém, às vezes, eles precisam ser enviados pela rede que conecta todos os computadores do governo. Mas todas as mensagens nessa rede são criptografadas usando um algoritmo altamente secreto, certificado por nossos melhores especialistas em criptografia. Nem mesmo a NSA e a KGB conseguem quebrá-lo. E agora esses documentos apareceram nas mãos de diplomatas de um país pequeno e insignificante. E não temos idéia de como isso pode ter acontecido."

"Mas vocês têm alguma suspeita de quem fez isso, não?", perguntou Holmes.

"Sim, fizemos uma investigação de rotina. Existe um homem que possui acesso a um dos computadores do governo e tem contatos freqüentes com diplomatas da embaixada. Mas o computador ao qual ele tem acesso não é um dos confiáveis, onde esses documentos normalmente são armazenados. Ele é o suspeito, mas não temos idéia de como ele pode ter obtido cópias dos documentos. Mesmo que pudesse obter uma cópia de um documento criptografado, não poderia decriptografá-lo."

"Hmm, por favor, descreva o protocolo de comunicação usado na rede." Holmes abriu seus olhos, provando que tinha acompanhado a fala de Lestrade com uma atenção que contrastava com sua aparência sonolenta.

"Bem, o protocolo é o seguinte. Cada nó N da rede recebe uma chave secreta K_n. Essa chave é usada para proteger a comunicação entre o nó e um servidor confiável. Ou seja, todas as chaves são armazenadas também no servidor. O usuário A, desejando enviar uma mensagem secreta M ao usuário B, inicia o seguinte protocolo:

1. A gera um número aleatório R e envia ao servidor seu nome A, destino B e $E(K_a, R)$.
2. O servidor responde enviando $E(K_b, R)$ para A.
3. A envia $E(R, M)$ junto com $E(K_b, R)$ para B.
4. B conhece K_b, e por isso decriptografa $E(K_b, R)$ para obter R e subseqüentemente usará R para decriptografar $E(R, M)$ e obter M.

Você vê que uma chave aleatória é gerada toda vez que uma mensagem precisa ser enviada. Admito que o homem poderia interceptar mensagens enviadas entre os nós confiáveis altamente secretos, mas não sei como ele poderia decriptografá-las."

"Bem, acho que você tem seu homem, Lestrade. O protocolo não é seguro porque o servidor não autentica os usuários que lhe enviam uma solici-

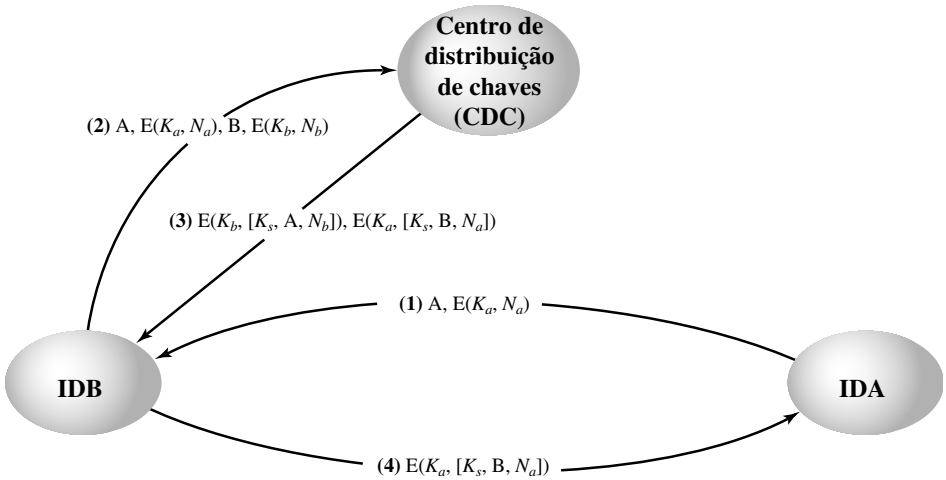


Figura 7.15 Figura para o Problema 7.3.

tação. Aparentemente, os projetistas do protocolo acreditaram que enviar $E(K_x, R)$ autenticaria implicitamente o usuário X como emissor, pois somente X (e o servidor) conhece K_x. Mas você sabe que $E(K_x, R)$ pode ser interceptado e mais tarde replicado. Quando você entender onde está o furo, poderá obter provas suficientes monitorando o uso que o homem fez do computador ao qual ele tem acesso. Provavelmente, ele trabalha da seguinte maneira: depois de interceptar $E(K_a, R)$ e $E(R, M)$ (ver etapas 1 e 2 do protocolo), o homem, que vamos chamar de Z, continuará fingindo ser A e ...

Termine a sentença para Holmes.

7.5 Se tomarmos um algoritmo congruencial linear com um componente aditivo de 0:

$$X_{n+1} = (aX_n) \bmod m$$

então, podemos mostrar que, se m é primo, e se determinado valor de a produz o período máximo de $m - 1$, então a^k também produzirá o período máximo, desde que k seja relativamente primo a $m - 1$ e que $m - 1$ não seja divisível por k. Demonstre isso usando $X_0 = 1$ e $m = 31$, e produzindo as seqüências para $a = 3, 3^2, 3^3$ e 3^4.

7.6 **a.** Qual é o período máximo que pode ser obtido do seguinte gerador?

$$X_{n+1} = (aX_n) \bmod 2^4$$

b. Qual deverá ser o valor de a?

c. Que restrições são exigidas na semente?

7.7 Você pode questionar por que o módulo $m = 2^{31} - 1$ foi escolhido para o método de congruência linear em vez de simplesmente 2^{31}, pois esse último número pode ser representado sem bits adicionais e a operação mod deve ser mais fácil de realizar. Em geral, o módulo $2^k - 1$ é preferível a 2^k. Por que isso acontece?

7.8 Com o algoritmo congruencial linear, uma escolha de parâmetros que oferece um período completo não necessariamente oferece uma boa aleatoriedade. Por exemplo, considere os dois geradores a seguir:

$$X_{n+1} = (6X_n) \bmod 13$$
$$X_{n+1} = (7X_n) \bmod 13$$

Escreva as duas seqüências para mostrar que ambos são de período completo. Qual deles lhe parece ser mais aleatório?

7.9 Em qualquer uso de números pseudo-aleatórios, seja para criptografia, simulação ou projeto estatístico, é perigoso confiar cegamente no gerador de números aleatórios disponível na biblioteca de sistema do seu computador. [PARK88] descobriu que muitos livros-texto contemporâneos e pacotes de programação utilizam algoritmos com falhas para a geração de números pseudo-aleatórios. Este exercício lhe permitirá testar seu sistema.

O teste é baseado em um teorema atribuído a Ernesto Cesaro (veja [KNUT98] para uma prova), que afirma o seguinte: dados dois inteiros escolhidos aleatoriamente, x e y, a probabilidade de que $\text{mdc}(x, y) = 1$ é $6/\pi^2$. Use esse teorema em um programa para determinar estatisticamente o valor de π. O programa principal deverá chamar três subprogramas: o gerador de números aleatórios pela biblioteca do sistema para gerar os inteiros aleatórios; um subprograma para calcular o máximo divisor comum de dois inteiros usando o algoritmo de Euclides; e um subprograma que calcula raízes quadradas. Se esses dois últimos programas não estiverem disponíveis, você terá de escrevê-los também. O programa principal deverá percorrer uma grande quantidade de números aleatórios para dar uma estimativa da probabilidade supracitada. A partir disso, é muito simples achar sua estimativa de π.

Se o resultado for próximo de 3,14, parabéns! Se não, então o resultado provavelmente é baixo, normalmente um valor em torno de 2,7. Por que esse resultado inferior seria obtido?

7.10 Suponha que você tenha um gerador de bits aleatórios verdadeiros, onde cada bit no fluxo gerado tem a mesma probabilidade de ser um 0 ou um 1 quanto qualquer outro bit no fluxo e que os bits não estejam correlacionados; ou seja, os bits são gerados a partir da distribuição independente e identicamente distribuída (i.i.d.). Porém, o fluxo de bits é tendencioso. A probabilidade de um 1 é $0{,}5 + \partial$ e a probabilidade de um 0 é $0{,}5 - \partial$, onde $0 < \partial < 0{,}5$. Um algoritmo antipropensão simples é o seguinte: examine o fluxo de bits como uma seqüência de pares não sobrepostos. Descarte todos os pares 00 e 11. Substitua cada par 01 por 0 e cada par 10 por 1.

a. Qual é a probabilidade de ocorrência de cada par na seqüência original?

b. Qual é a probabilidade de ocorrência de 0 e 1 na seqüência modificada?

c. Qual é o número esperado de bits de entrada para produzir x bits de saída?

d. Suponha que o algoritmo utilize pares de bits sucessivos sobrepostos em vez de pares de bits sucessivos não sobrepostos. Ou seja, o primeiro bit de saída é baseado nos bits de entrada 1 e 2, o segundo bit de saída é baseado nos bits de entrada 2 e 3, e assim por diante. O que você pode dizer sobre o fluxo de bits de saída?

7.11 Outra técnica antipropensão é considerar o fluxo de bits como uma seqüência de grupos não sobrepostos de n bits cada e a saída a paridade de cada grupo. Ou seja, se um grupo contém um número ímpar de uns, a saída será 1; caso contrário, a saída será 0.

a. Expresse essa operação em termos de uma função booleana básica.

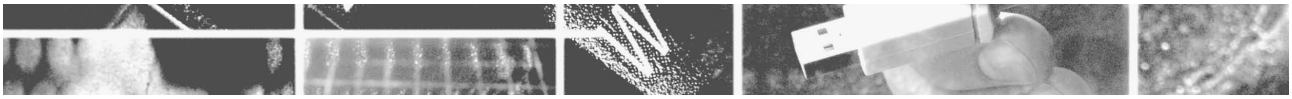
b. Considere, como no problema anterior, que a probabilidade de um 1 é de $0{,}5 + \partial$. Se cada

grupo consiste em 2 bits, qual é a probabilidade de uma saída de 1?

c. Se cada grupo consiste em 4 bits, qual é a probabilidade de uma saída de 1?

d. Generalize o resultado para encontrar a probabilidade de uma saída de 1 para grupos de entrada de n bits.

7.12 Suponha que alguém sugira a seguinte forma de confirmar que vocês dois estão de posse da mesma chave secreta. Você cria uma string de bits aleatórios do tamanho da chave, efetua um OU-exclusivo (*XOR*) dessa string com a chave e envia o resultado pelo canal. Seu parceiro efetua um OU-exclusivo do bloco de entrada com a chave (que deverá ser a mesma que a sua) e envia o resultado de volta. Você verifica e, se o que receber for sua string aleatória original, então terá verificado que seu parceiro tem a mesma chave secreta, embora nenhum de vocês tenha sequer transmitido a chave. Existe alguma falha nesse esquema?



PARTE DOIS

Criptografia de chave pública e funções de hash

Por motivos práticos, é desejável usar diferentes chaves de criptografia e decriptografia em um criptossistema. Esses sistemas assimétricos permitem que a chave de criptografia se torne disponível a qualquer um, enquanto garantem que somente as pessoas que mantêm a chave de decriptografia poderão decifrar a informação.

Computers at risk: safe computing in the information age,
National Research Council, 1991

Depois da criptografia simétrica, a outra forma importante de criptografia é a de chave pública, que revolucionou a segurança das comunicações. Uma área da criptografia relacionada é a das funções criptográficas de hash. As funções de hash são usadas em conjunto com as cifras simétricas para assinaturas digitais. Além disso, as funções de hash são usadas para autenticação de mensagens. As cifras simétricas também são usadas para gerenciamento de chave. Todas essas áreas são explicadas na Parte Dois.

CONTEÚDO DA PARTE DOIS

Capítulo 8: Introdução à teoria dos números
A maior parte dos esquemas de chave pública baseia-se na Teoria dos Números. Embora o leitor possa aceitar os resultados teóricos dos números sem questionar, é importante ter um conhecimento básico dos conceitos da Teoria dos Números. O Capítulo 8 contém uma visão geral e diversos exemplos para esclarecer os conceitos.

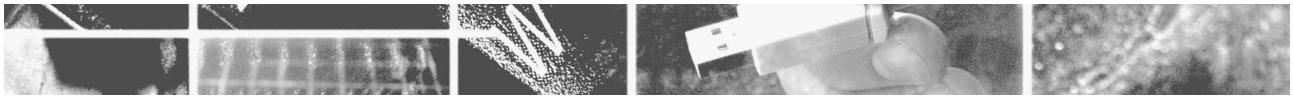
Capítulo 9: Criptografia de chave pública e RSA
O Capítulo 9 apresenta a criptografia de chave pública e se concentra em seu uso para obter confidencialidade. Este capítulo também examina a cifra de chave pública mais utilizada, o algoritmo Rivest-Shamir-Adleman (RSA).

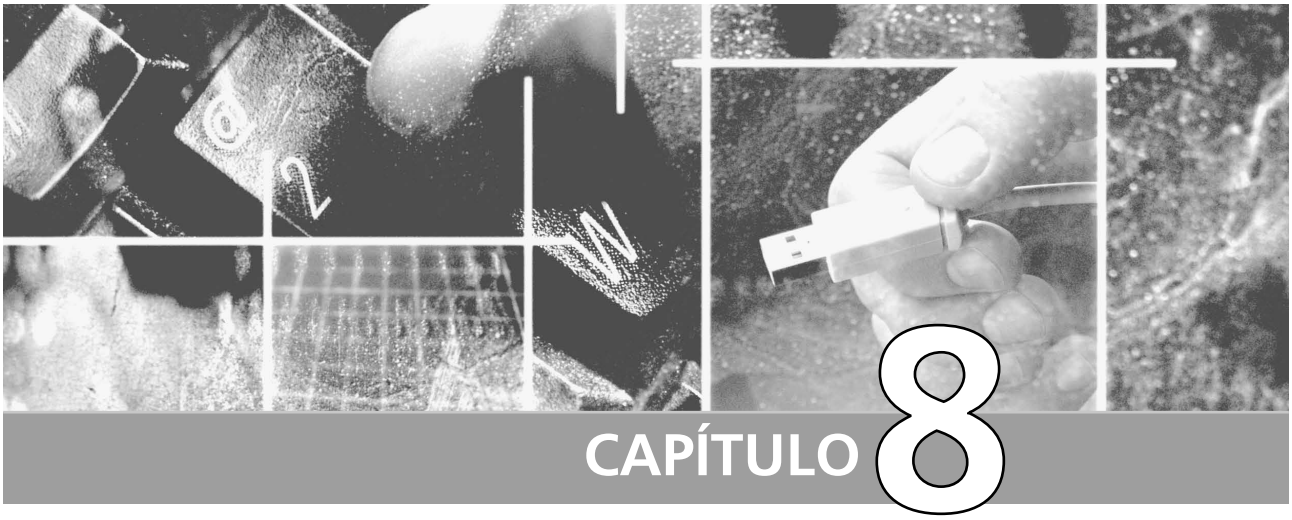
Capítulo 10: Gerenciamento de chave — outros criptossistemas de chave pública
O Capítulo 10 retoma a questão do gerenciamento de chave à luz das capacidades das cifras simétricas. O capítulo também aborda a técnica de troca de chaves Diffie-Hellman e examina uma abordagem de chave pública mais recente, baseada em curvas elípticas.

Capítulo 11: Autenticação de mensagem e funções de hash
Também importante para a confidencialidade como uma medida de segurança é a autenticação. No mínimo, a autenticação garante que uma mensagem provém da origem alegada. Além disso, a autenticação pode incluir proteção contra modificação, atraso, repetição e reordenação. O Capítulo 11 começa com uma análise dos requisitos para autenticação e depois oferece uma apresentação sistemática das técnicas para autenticação. Um elemento-chave dos esquemas de autenticação é o uso de um autenticador, normalmente um código de autenticação de mensagem (MAC — Message Authentication Code) ou uma função de hash. Examinamos as considerações de projeto desses dois tipos de algoritmos e analisamos diversos exemplos.

Capítulo 12: Algoritmos de hash e MAC
O Capítulo 12 estende a explicação do capítulo anterior e aborda duas das funções criptográficas de hash mais importantes (SHA e Whirlpool) e dois dos MACs mais importantes (HMAC e CMAC).

Capítulo 13: Assinaturas digitais e protocolos de autenticação
Um tipo importante de autenticação é a assinatura digital. O Capítulo 13 examina as técnicas usadas para criar assinaturas digitais e examina um padrão importante, o Digital Signature Standard (DSS).
As diversas técnicas de autenticação baseadas em assinaturas digitais são os elementos da montagem dos algoritmos de autenticação. O projeto desses algoritmos envolve a análise de ataques sutis, que podem derrotar muitos protocolos aparentemente seguros. Essa questão é analisada no Capítulo 14.





CAPÍTULO 8

Introdução à teoria dos números

8.1 **Números primos**

8.2 **Teoremas de Fermat e Euler**

8.3 **Testando se o número é primo**

8.4 **O teorema chinês do resto**

8.5 **Logaritmos discretos**

8.6 **Leitura e sites Web recomendados**

8.7 **Principais termos, questões para revisão e problemas**

O diabo disse a Daniel Webster: "Dê-me uma tarefa que eu não possa executar, e eu lhe darei qualquer coisa no mundo que você pedir".

Daniel Webster: "Muito justo. Prove que, para n maior que 2, a equação $a^n + b^n = c^n$ não possui solução trivial no conjunto dos inteiros".

Eles concordaram com um período de três dias para o trabalho, e o diabo desapareceu.

Ao final dos três dias, o diabo se apresentou, desfigurado, mordendo seus lábios. Daniel Webster lhe disse: "Bem, como você se saiu na minha tarefa? Conseguiu provar o teorema?"

"Não... não, eu não consegui provar".

"Então eu posso ter o que quiser? Dinheiro? A presidência?"

"O quê? Oh, é claro... Mas, escute, se pudéssemos simplesmente provar os dois lemas a seguir..."

The mathematical Magpie, Clifton Fadiman

PONTOS PRINCIPAIS

- Um número primo é um inteiro que só pode ser dividido sem resto por valores positivos e negativos de si mesmo e por 1. Os números primos desempenham um papel fundamental na Teoria dos Números e na criptografia.
- Dois teoremas que desempenham papéis importantes na criptografia de chave pública são o teorema de Fermat e o teorema de Euler.
- Um requisito importante em diversos algoritmos criptográficos é a capacidade de escolher um número primo grande. Uma área de pesquisa contínua é o desenvolvimento de algoritmos eficientes para determinar se um inteiro grande, escolhido aleatoriamente, é um número primo.
- Logaritmos discretos são fundamentais para diversos algoritmos de chave pública. Logaritmos discretos são semelhantes aos logaritmos comuns, mas operam sobre a aritmética modular.

Diversos conceitos da Teoria dos Números são essenciais no projeto dos algoritmos criptográficos de chave pública. Este capítulo oferece uma visão geral dos conceitos mencionados em outros capítulos. O leitor familiarizado com esses tópicos pode seguramente ignorar este capítulo.

Assim como o Capítulo 4, este capítulo inclui diversos exemplos, cada um dos quais destacados em uma caixa sombreada.

8.1 NÚMEROS PRIMOS[1]

Uma questão fundamental da Teoria dos Números é o estudo dos números primos. Na realidade, livros inteiros foram escritos sobre o assunto (por exemplo, [CRAN01], [RIBE96]). Nesta seção oferecemos uma visão geral relevante às questões deste livro.

Um inteiro $p > 1$ é um número primo se e somente se seus únicos divisores[2] forem ± 1 e $\pm p$. Os números primos desempenham um papel importante na Teoria dos Números e nas técnicas explicadas neste capítulo. A Tabela 8.1 mostra os primos menores que 2000. Observe o modo como os primos são distribuídos. Em particular, observe a quantidade de primos em cada intervalo de cem números.

Qualquer inteiro $a > 1$ pode ser fatorado de uma forma exclusiva como

$$a = p_1^{a_1} p_2^{a_2} \ldots p_t^{a_t} \tag{8.1}$$

onde $p_1 < p_2 < \cdots < p_t$ são números primos e onde cada a_i é um inteiro positivo. Isso é conhecido como teorema fundamental da aritmética; uma prova poderá ser encontrada em qualquer texto sobre Teoria dos Números.

$$\begin{aligned} 91 &= 7 \times 13 \\ 3600 &= 2^4 \times 3^2 \times 5^2 \\ 11011 &= 7 \times 11^2 \times 13 \end{aligned}$$

É útil, para o texto a seguir, expressar isso de outra maneira. Se P é o conjunto de todos os números primos, então qualquer inteiro positivo a pode ser escrito exclusivamente da seguinte forma:

$$a = \prod_{p \in \mathrm{P}} p^{a_p} \quad \text{onde cada } a_p \geq 0$$

O lado direito é o produto sobre todos os números primos possíveis p; para qualquer valor em particular de a, a maioria dos expoentes a_p será 0.

O valor de qualquer inteiro positivo dado pode ser especificado simplesmente listando-se todos os expoentes diferentes de zero na formulação a seguir.

O inteiro 12 é representado por $\{a_2 = 2, a_3 = 1\}$.
O inteiro 18 é representado por $\{a_2 = 1, a_3 = 2\}$.
O inteiro 91 é representado por $\{a_7 = 1, a_{13} = 1\}$.

A multiplicação de dois números é equivalente à adição dos expoentes correspondentes. Dados $a = \prod_{p \in \mathrm{P}} p^{a_p}$, $b = \prod_{p \in \mathrm{P}} p^{b_p}$, defina $k = ab$. Sabemos que o inteiro k pode ser expresso como o produto de potências de primos: $k = \prod_{p \in \mathrm{P}} p^{k_p}$. Segue-se que $k_p = a_p + b_p$ para todo $p \in P$.

1. Nesta seção, a menos que observado de outra forma, lidamos apenas com inteiros não negativos. O uso de inteiros negativos não introduziria diferenças essenciais.
2. Lembre-se, conforme explicado no Capítulo 4, de que o inteiro a é considerado um divisor do inteiro b se não houver resto na divisão. De modo equivalente, dizemos que a divide b.

Tabela 8.1 Primos menores que 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

$$k = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$$
$$k_2 = 2 + 1 = 3; k_3 = 1 + 2 = 3$$
$$216 = 2^3 \times 3^3 = 8 \times 27$$

Isso significa, em termos dos fatores primos de a e b, dizer que a divide b? Qualquer inteiro na forma p^n pode ser dividido apenas por um inteiro que seja de potência menor ou igual ao mesmo número primo, p^j com $j \leq n$. Assim, podemos dizer o seguinte:

Dado $a = \prod_{p \in \mathbf{P}} p^{a_p}$, $b = \prod_{p \in \mathbf{P}} p^{b_p}$. Se $a|b$, então $a_p \leq b_p$ para todo p.

$$a = 12; b = 36; 12|36$$
$$12 = 2^2 \times 3; 36 = 2^2 \times 3^2$$
$$a_2 = 2 = b_2$$
$$a_3 = 1 \leq 2 = b_3$$

Assim, a desigualdade $a_p \leq b_p$ é satisfeita para todos os números primos.

É fácil determinar o máximo divisor comum[3] de dois inteiros positivos se expressarmos cada inteiro como o produto dos primos.

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\text{mdc}(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

O seguinte relacionamento sempre é verdadeiro:

$$\text{Se} \quad k = \text{mdc}(a, b) \qquad \text{então} \quad k_p = \min(a_p, b_p) \quad \text{para todo } p$$

Não é fácil determinar os fatores primos de um número grande, de modo que o relacionamento anterior não leva diretamente a um método prático de calcular o máximo divisor comum.

8.2 TEOREMAS DE FERMAT E EULER

Dois teoremas que desempenham papéis importantes na criptografia de chave pública são o teorema de Fermat e o teorema de Euler.

Teorema de Fermat[4]

O teorema de Fermat afirma o seguinte: Se p é primo e a é um inteiro positivo não divisível por p, então

$$a^{p-1} \equiv 1 (\text{mod } p) \tag{8.2}$$

Prova: Considere o conjunto de inteiros positivos menores que p: $\{1, 2, \ldots, p - 1\}$ e multiplique cada elemento por a, módulo p, para obter o conjunto $X = \{a \bmod p, 2a \bmod p, \ldots (p - 1)a \bmod p\}$. Nenhum dos elementos de X é igual a zero porque p não divide a. Além do mais, dois inteiros em X não são iguais. Para verificar isso, considere que $ja \equiv ka (\text{mod } p)$ onde $1 \leq j < k \leq p - 1$. Como a é relativamente primo[5] de p, podemos eliminar a dos dois lados da equação [ver Equação (4.3)] resultando em: $j \equiv k (\text{mod } p)$. Essa última igualdade é impossível, porque j e k são ambos

3. Lembre-se, conforme o Capítulo 4, de que o máximo divisor comum dos inteiros a e b, expresso como mdc(a,b), é um inteiro c que divide a e b sem resto e que qualquer divisor de a e b é um divisor de c.
4. Este às vezes é conhecido como pequeno teorema de Fermat.
5. Lembre-se, conforme o Capítulo 4, de que dois números são relativamente primos se não tiverem fatores primos em comum, ou seja, seu único divisor comum é 1. Isso equivale a dizer que dois números são relativamente primos se seu máximo divisor comum for 1.

inteiros positivos menores que p. Portanto, sabemos que os $(p - 1)$ elementos de X são todos inteiros positivos, sem que dois elementos sejam iguais. Podemos concluir que X consiste no conjunto de inteiros $\{1, 2, \ldots, p - 1\}$ em alguma ordem. Multiplicar os números nos dois conjuntos e apanhar o resultado mod p gera

$$a \times 2a \times \ldots \times (p - 1) \equiv [(1 \times 2 \times \ldots \times (p - 1)](\text{mod } p)$$
$$a^{p-1}(p - 1)! \equiv (p - 1)!(\text{mod } p)$$

Podemos cancelar o termo $(p - 1)!$ porque ele é relativamente primo de p [ver Equação (4.3)]. Isso gera a Equação (8.2).

$$\begin{aligned}
&a = 7, p = 19\\
&7^2 = 49 \equiv 11(\text{mod } 19)\\
&7^4 \equiv 121 \equiv 7(\text{mod } 19)\\
&7^8 \equiv 49 \equiv 11(\text{mod } 19)\\
&7^{16} \equiv 121 \equiv 7(\text{mod } 19)\\
&a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1(\text{mod } 19)
\end{aligned}$$

Uma forma alternativa do teorema de Fermat também é útil: Se p é primo e a é um inteiro positivo, então

$$a^p \equiv a(\text{mod } p) \tag{8.3}$$

Observe que a primeira forma do teorema [Equação (8.2)] requer que a seja relativamente primo de p, mas esta última forma não.

$$\begin{aligned}
&p = 5, a = 3 \qquad a^p = 3^5 = 243 \equiv 3(\text{mod } 5) = a(\text{mod } p)\\
&p = 5, a = 10 \qquad a^p = 10^5 = 100000 \equiv 10(\text{mod } 5) = 0(\text{mod } 5) = a(\text{mod } p)
\end{aligned}$$

Função tociente de Euler

Antes de apresentar o teorema de Euler, precisamos apresentar uma quantidade importante na Teoria dos Números, chamada de função tociente de Euler, escrita como $\phi(n)$, definida como o número de inteiros positivos menores que n e relativamente primos de n. Por convenção, $\phi(1) = 1$

Determine $\phi(37)$ e $\phi(35)$.
Como 37 é primo, todos os inteiros positivos de 1 até 36 são relativamente primos de 37. Assim, $\phi(37) = 36$.
Para determinar $\phi(35)$, listamos todos os inteiros positivos menores que 35 que são relativamente primos dele:

$$1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18,$$
$$19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34.$$

Existem 24 números na lista, de modo que $\phi(35) = 24$.

A Tabela 8.2 lista os 30 primeiros valores de $\phi(n)$. O valor $\phi(1)$ não tem significado, mas é definido para que tenha o valor 1.

Deverá ficar claro que, para um número primo p,

$$\phi(p) = p - 1$$

Agora, suponha que tenhamos dois números primos p e q, com $p \neq q$. Então, podemos mostrar que, para $n = pq$,

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p - 1) \times (q - 1)$$

Tabela 8.2 Alguns valores da função tociente de Euler $\phi(n)$

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

Para verificar que $\phi(n) = \phi(p) \times \phi(q)$, considere que o conjunto de inteiros positivos menores que n é o conjunto $\{1, ..., (pq - 1)\}$. Os inteiros nesse conjunto que não são relativamente primos de n são o conjunto $\{p, 2p, ..., (q - 1)p\}$ e o conjunto $\{q, 2q, \ldots, (p - 1)q\}$. Por conseguinte,

$$\begin{aligned}\phi(n) &= (pq - 1) - [(q - 1) + (p - 1)] \\ &= pq - (p + q) + 1 \\ &= (p - 1) \times (q - 1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

$\phi(21) = \phi(3) \times \phi(7) = (3 - 1) \times (7 - 1) = 2 \times 6 = 12$
onde os 12 inteiros são $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

Teorema de Euler

O teorema de Euler afirma que, para cada a e n que sejam relativamente primos:

$$a^{\phi(n)} \equiv 1 (\text{mod } n) \tag{8.4}$$

$a = 3; n = 10; \phi(10) = 4$	$a^{\phi(n)} = 3^4 = 81 \equiv 1(\text{mod } 10) = 1(\text{mod } n)$
$a = 2; n = 11; \phi(11) = 10$	$a^{\phi(n)} = 2^{10} = 1024 \equiv 1(\text{mod } 11) = 1(\text{mod } n)$

Prova: A Equação (8.4) é verdadeira se n for primo, pois nesse caso $\phi(n) = (n - 1)$ e o teorema de Fermat prevalece. Porém, ela também prevalece para qualquer inteiro n. Lembre-se de que $\phi(n)$ é o número de inteiros positivos menores que n que são relativamente primos de n. Considere o conjunto desses inteiros, rotulados da seguinte forma:

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$$

Ou seja, cada elemento x_i de R é um inteiro positivo exclusivo, menor que n com $\text{mdc}(x_i, n) = 1$. Agora, multiplique cada elemento por a, módulo n:

$$S = \{(ax_1 \text{ mod } n), (ax_2 \text{ mod } n), \ldots, (ax_{\phi(n)} \text{ mod } n)\}$$

O conjunto S é uma permutação de R, pela seguinte linha de raciocínio:

1. Como a é relativamente primo de n e x_i é relativamente primo de n, ax_i também deverá ser relativamente primo de n. Assim, todos os membros de S são inteiros menores que n e relativamente primos de n.
2. Existem duplicatas em S. Consulte a Equação (4.3). Se $ax_i \text{ mod } n = ax_j \text{ mod } n$, então $x_i = x_j$.

Portanto,

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$
$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$
$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$
$$a^{\phi(n)} \equiv 1 \pmod n$$

Essa é a mesma linha de raciocínio aplicada à prova do teorema de Fermat. Como acontece com o teorema de Fermat, uma forma alternativa do teorema também é útil:

$$a^{\phi(n)+1} \equiv a \pmod n \tag{8.5}$$

Novamente, de modo semelhante ao caso do teorema de Fermat, a primeira forma do teorema de Euler [Equação (8.4)] requer que a seja relativamente primo de n, mas essa forma não.

8.3 TESTANDO SE O NÚMERO É PRIMO

Para muitos algoritmos criptográficos, é necessário selecionar um ou mais números primos muito grandes aleatoriamente. Assim, encaramos a tarefa de determinar se certo número grande é primo. Não existe um meio simples e eficiente de realizar essa tarefa.

Nesta seção, apresentamos um algoritmo atraente e bastante utilizado. Você pode ficar surpreso ao descobrir que esse algoritmo gera um número que não é necessariamente primo. Porém, o algoritmo pode gerar um número que quase certamente é primo. Isso será explicado nesta seção. Também fazemos referência a um algoritmo determinístico para encontrar primos. A seção termina com uma explicação a respeito da distribuição dos primos.

Algoritmo de Miller-Rabin[6]

O algoritmo devido a Miller e Rabin [MILL75, RABI80] normalmente é usado para testar se um número grande é primo. Antes de explicar o algoritmo, precisamos de alguma base. Primeiro, qualquer inteiro positivo ímpar $n \geq 3$ pode ser expresso da seguinte forma:

$$n - 1 = 2^k q \qquad \text{com } k > 0, q \text{ odd}$$

Para verificar isso, observe que $(n - 1)$ é um inteiro par. Depois, divida $(n - 1)$ por 2 até que o resultado seja um número ímpar q, para gerar um total de k divisões. Se n for expresso como um número binário, então o resultado será alcançado deslocando-se o número para a direita até que o dígito mais à direita seja 1, gerando um total de k deslocamentos. Agora desenvolveremos duas propriedades dos números primos, das quais precisaremos.

Duas propriedades dos números primos A **primeira propriedade** é expressa da seguinte maneira: se p é primo e a é um inteiro positivo menor que p, então $a^2 \bmod p = 1$ se e somente se $a \bmod p = 1$ ou $a \bmod p = -1 \bmod p = p - 1$. Pelas regras da aritmética modular, $(a \bmod p)\ (a \bmod p) = a^2 \bmod p$. Assim, se $a \bmod p = 1$ ou $a \bmod p = -1$, então $a^2 \bmod p = 1$. Reciprocamente, se $a^2 \bmod p = 1$, então $(a \bmod p)^2 = 1$, que é verdadeiro somente para $a \bmod p = 1$ ou $a \bmod p = -1$.

A **segunda propriedade** é expressa da seguinte maneira: Considere que p seja um número primo maior que 2. Podemos, então, escrever $p - 1 = 2^k q$, com $k > 0$, q ímpar. Considere que a seja qualquer inteiro no intervalo $1 < a < p - 1$. Então, uma das duas condições a seguir é verdadeira:

1. a^q é congruente a 1 módulo p. Ou seja, $a^q \bmod p = 1$ ou, de forma equivalente, $a^q \equiv 1 \pmod p$.
2. Um dos números $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ é congruente a -1 módulo p. Ou seja, existe algum número j no intervalo $(1 \leq j \leq k)$ tal que $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$, ou, de forma equivalente, $a^{2^{j-1}q} \equiv -1 \pmod p$.

Prova: O teorema de Fermat [Equação (8.2)] afirma que $a^{n-1} \equiv 1 \pmod n$ se n for primo. Temos que $p - 1 = 2^k q$. Assim, sabemos que $a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$. Assim, se examinarmos a seqüência de números

6. Também conhecido na literatura como algoritmo de Rabin-Miller, ou teste de Rabin-Miller, ou teste de Miller-Rabin.

$$a^q \bmod p, a^{2q} \bmod p, a^{4q} \bmod p, \ldots, a^{2^{k-1}q} \bmod p, a^{2^k q} \bmod p \quad \textbf{(8.6)}$$

saberemos que o último número na lista tem o valor 1. Além disso, cada número na lista é o quadrado do número anterior. Portanto, uma das seguintes possibilidades precisa ser verdadeira:

1. O primeiro número na lista, e, portanto, todos os números subseqüentes na lista, é igual a 1.
2. Algum número na lista não é igual a 1, mas seu quadrado mod p é igual a 1. Em virtude da primeira propriedade dos números primos, definida acima, sabemos que o único número que satisfaz essa condição é $p - 1$. Assim, nesse caso, a lista contém um elemento igual a $p - 1$.

Isso completa a prova.

Detalhes do algoritmo Essas considerações levam à conclusão de que, se n é primo, então ou o primeiro elemento da lista de resíduos, ou restos, $(a^q, a^{2q}, ..., a^{2^{k-1}q}, a^{2^k q})$ módulo n é igual a 1, ou algum elemento na lista é igual a $(n - 1)$; caso contrário, n é composto, ou seja, não é primo. Por outro lado, se a condição for atendida, isso não necessariamente significa que n é primo. Por exemplo, se $n = 2047 = 23 \times 89$, então $n - 1 = 2 \times 1023$. Calculando, $2^{1023} \bmod 2047 = 1$, de modo que 2047 atende a condição, mas não é primo.

Podemos usar a propriedade anterior para criar um teste que verifica se o número é primo. O procedimento TESTE usa um candidato inteiro n como entrada e retorna o resultado **composto** se n definitivamente não for primo, e o resultado **inconclusivo** se n pode ou não ser primo.

```
   TEST (n)
1. Encontre inteiros k, q, com k > 0, q ímpar, de modo que (n - 1 = 2^k q);
2. Selecione um inteiro aleatório a, 1 < a < n - 1;
3. Se a^q mod n = 1 então return("inconclusivo");
4. para j = 0 até k - 1 faça
5.     se a^(2^j q) mod n ≡ n - 1 então return("inconclusivo");
6. return("composto");
```

Vamos aplicar o teste ao número primo $n = 29$. Temos $(n - 1) = 28 = 2^2(7) = 2^k q$. Primeiro, vamos testar $a = 10$. Calculamos $10^7 \bmod 29 = 17$, que não é 1 nem 28, de modo que continuamos o teste. O próximo cálculo descobre que $(10^7)^2 \bmod 29 = 28$, e o teste retorna **inconclusivo**, ou seja, 29 pode ser primo. Vamos testar novamente com $a = 2$. Temos os seguintes cálculos: $2^7 \bmod 29 = 12$; $2^{14} \bmod 29 = 28$; e o teste novamente retorna **inconclusivo**. Se realizarmos o teste para todos os inteiros a no intervalo de 1 a 28, obteremos o mesmo resultado inconclusivo, que é compatível com n sendo um número primo.

Agora, vamos testar com o número composto $n = 13 \times 17 = 221$. Então, $(n - 1) = 220 = 2^2(55) = 2^k q$. Vamos testar $a = 5$. Então, temos $5^{55} \bmod 221 = 112$, que não é 1 nem 220; $(5^{55})^2 \bmod 221 = 168$. Como usamos todos os valores de j (ou seja, $j = 0$ e $j = 1$) na linha 4 do algoritmo TESTE, o teste retorna **composto**, indicando que 221 é definitivamente um número composto. Mas suponha que tivéssemos selecionado $a = 21$. Então, teríamos $21^{55} \bmod 221 = 200$; $(21^{55})^2 \bmod 221 = 220$; e o teste retornaria **inconclusivo**, indicando que 221 pode ser primo. De fato, dos 218 inteiros de 2 até 219, quatro desses retornarão um resultado inconclusivo, a saber, 21, 47, 174 e 200.

Uso repetido do algoritmo de Miller-Rabin Como podemos usar o algoritmo de Miller-Rabin para determinar com um alto grau de confiança se um inteiro é ou não primo? Pode ser mostrado [KNUT98] que, dado um número ímpar n que não é primo e um inteiro escolhido aleatoriamente, a com $1 < a < n - 1$, a probabilidade de que TESTE retorne **inconclusivo** (ou seja, que deixe de detectar que n não é primo) é menor que 1/4. Assim, se t diferentes valores de a forem escolhidos, a probabilidade de que todos eles passem no TESTE (retornem neste tipo **inconclusivo**) para n é menor que $(1/4)^t$. Por exemplo, para $t = 10$, a probabilidade de que um número não-primo passe em todos os dez testes é menor que 10^{-6}. Assim, para um valor suficientemente grande de t, podemos confiar que n é primo se o teste de Miller sempre retornar **inconclusivo**.

Isso nos dá uma base para determinar se um inteiro ímpar n é primo com um razoável grau de confiança. O procedimento é o seguinte: chame repetidamente TESTE(n) usando valores escolhidos aleatoriamente para a. Se, em qualquer ponto, TESTE retornar **composto**, então n será determinado como sendo não-primo. Se o TESTE continuar a retornar **inconclusivo** por t testes, para um valor suficientemente grande de t, consideramos que n é primo.

Um algoritmo determinístico para números primos

Antes de 2002 não havia um método conhecido para provar com eficiência se números muito grandes são primos. Todos os algoritmos em uso, incluindo o mais popular (Miller-Rabin), produziam um resultado probabilístico. Em 2002, Agrawal, Kayal e Saxena [AGRA02] desenvolveram um algoritmo determinístico relativamente simples, que determina com eficiência se determinado número grande é primo. O algoritmo, conhecido como algoritmo AKS, não parece ser tão eficiente quanto o algoritmo de Miller-Rabin. Até agora ele não suplantou essa técnica probabilística mais antiga [BORN03].

Distribuição de números primos

Vale a pena observar quantos números provavelmente serão rejeitados antes que um número primo seja encontrado usando o teste de Miller-Rabin, ou qualquer outro teste de números primos. Um resultado da Teoria dos Números, conhecido como teorema do número primo, afirma que os primos próximos de n são espaçados em média a cada ($\ln n$) inteiros. Assim, na média, seria preciso testar algo na ordem de $\ln(n)$ inteiros antes que um número primo fosse encontrado. Como todos os inteiros pares podem ser imediatamente rejeitados, o valor correto é $0{,}5 \ln(n)$. Por exemplo, se um primo na ordem de grandeza de 2^{200} fosse procurado, então seriam necessárias cerca de $0{,}5 \ln(2^{200}) = 69$ tentativas para encontrar um primo. Porém, esse valor é apenas uma média. Em alguns lugares ao longo da linha de números, os primos são mais próximos, e em outros lugares existem lacunas maiores.

> Os dois inteiros ímpares consecutivos 1.000.000.000.061 e 1.000.000.000.063 são ambos primos. Por outro lado, $1001! + 2, 1001! + 3, \ldots, 1001! + 1000, 1001! + 1001$ é uma seqüência de 1000 inteiros compostos consecutivos.

8.4 O TEOREMA CHINÊS DO RESTO

Um dos mais úteis resultados da teoria dos números é o teorema chinês do resto (CRT — Chinese Remainder Theorem).[7] Basicamente, o TCR diz que é possível reconstruir inteiros em um certo intervalo a partir de seus resíduos módulo um conjunto de pares de módulos relativamente primos.

> Os 10 inteiros em Z_{10}, que são os inteiros de 0 a 9, podem ser reconstruídos a partir de seus dois resíduos módulo 2 e 5 (os fatores relativamente primos de 10). Digamos que os resíduos conhecidos de um dígito decimal x sejam $r_2 = 0$ e $r_5 = 3$, ou seja, $x \bmod 2 = 0$ e $x \bmod 5 = 3$. Portanto, x é um inteiro par em Z_{10} cujo resto, na divisão por 5, é 3. A única solução é $x = 8$.

O TCR pode ser declarado de várias maneiras. Apresentamos aqui uma formulação que é mais útil do ponto de vista deste texto. Uma formulação alternativa é explorada no Problema 8.17. Considere

$$M = \prod_{i=1}^{k} m_i$$

onde o m_i são pares de números relativamente primos, ou seja, $\text{mdc}(m_i, m_j) = 1$ para $1 \le i, j \le k$, e $i \ne j$. Podemos representar qualquer inteiro A em Z_M por uma tupla de k cujos elementos estão em Z_{M_i}, usando a seguinte correspondência:

$$A \leftrightarrow (a_1, a_2, \ldots, a_k) \tag{8.7}$$

onde $A \in Z_M$, $a_i \in Z_{M_i}$, e $a_i = A \bmod m_i$ para $1 \le i \le k$. O TCR faz duas asserções.

1. O mapeamento da Equação (8.7) é uma correspondência biunívoca (chamada de **bijeção**) entre Z_M e o produto cartesiano $Z_{M_i} \times Z_{M_2} \times \ldots \times Z_{M_k}$. Ou seja, para cada inteiro A tal que $0 \le A < M$, existe uma tupla de k elementos exclusiva $(a_1, a_2, \ldots, a_k)$ com $0 \le a_i < m_i$ que o representa, e para cada tupla de k *elementos* $(a_1, a_2, \ldots, a_k)$ existe um inteiro exclusivo A em Z_M.

7. O TCR tem esse nome porque acredita-se que tenha sido descoberto pelo matemático chinês Sun-Tsu, por volta do ano 100 DC.

2. As operações realizadas sobre os elementos de Z_M podem ser equivalentemente realizadas sobre as tuplas de k correspondentes, realizando-se a operação independentemente, em cada posição de coordenada no sistema apropriado.

Vamos demonstrar a **primeira asserção**. A transformação de A para $(a_1, a_2, ..., a_k)$ obviamente é exclusiva; ou seja, cada a_i é calculado exclusivamente como $a_i = A \bmod m_i$. O cálculo de A a partir de $(a_1, a_2, ..., a_k)$ pode ser feito da seguinte forma. Considere $M_i = M/m_i$ para $1 \leq i \leq k$. Observe que $M_i = m_1 \times m_2 \times ... \times m_{i-1} \times m_{i+1} \times ... \times m_k$, de modo que $M_i \equiv 0 (\bmod m_j)$ para todo $j \neq i$. Então, considere

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{para } 1 \leq i \leq k \tag{8.8}$$

Pela definição de M_i, ele é relativamente primo de m_i e, portanto, possui inverso multiplicativo exclusivo mod m_i. Assim, a Equação (8.8) é bem definida e produz um valor exclusivo c_i. Podemos agora calcular:

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) (\bmod M) \tag{8.9}$$

Para mostrar que o valor de A produzido pela Equação (8.9) está correto, temos de mostrar que $a_i = A \bmod m_i$, para $1 \leq i \leq k$. Observe que $c_j \equiv M_j \equiv 0 (\bmod m_i)$ se $j \neq i$ e que $c_i \equiv 1 (\bmod m_i)$. Segue-se que $a_i = A \bmod m_i$.

A **segunda asserção** do TCR, referente a operações aritméticas, vem das regras da aritmética modular. Ou seja, a segunda asserção pode ser declarada da seguinte forma: Se

$$A \longleftrightarrow (a_1, a_2, \ldots, a_k)$$
$$B \longleftrightarrow (b_1, b_2, \ldots, b_k)$$

então

$$(A + B) \bmod M \longleftrightarrow ((a_1 + b_1) \bmod m_1, \ldots, (a_k + b_k) \bmod m_k)$$
$$(A - B) \bmod M \longleftrightarrow ((a_1 - b_1) \bmod m_1, \ldots, (a_k - b_k) \bmod m_k)$$
$$(A \times B) \bmod M \longleftrightarrow ((a_1 \times b_1) \bmod m_1, \ldots, (a_k \times b_k) \bmod m_k)$$

Um dos recursos úteis do teorema chinês do resto é que ele oferece um modo de manipular números (potencialmente muito grandes) mod M em termos de tuplas de números menores. Isso pode ser útil quando M tem 150 dígitos ou mais. Porém, observe que é preciso saber de antemão a fatoração de M.

Para representar 973 mod 1813 como um par de números mod 37 e 49, defina[8]

$$m_1 = 37$$
$$m_2 = 49$$
$$M = 1813$$
$$A = 973$$

Também temos $M_1 = 49$ e $M_2 = 37$. Usando o algoritmo euclidiano estendido, calculamos $M_1^{-1} = 34 \bmod m_1$ e $M_2^{-1} = 4 \bmod m_2$. (Observe que só precisamos calcular cada M_i e cada M_i^{-1} uma vez.) Apanhando resíduos módulo 37 e 49, nossa representação de 973 é (11, 42), pois 973 mod 37 = 11 e 973 mod 49 = 42.

Agora, suponha que queiramos somar 678 a 973. O que fazemos com (11, 42)? Primeiro, calculamos (678) ↔ (678 mod 37, 678 mod 49) = (12,41). Depois, somamos as tuplas para cada elemento e reduzimos (11 + 12 mod 37, 42 + 41 mod 49) = (23, 34). Para verificar se isso tem o efeito correto, calculamos

$$\begin{aligned}(23, 34) \longleftrightarrow\ & a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \bmod M \\ &= [(23)(49)(34) + (34)(37)(4)] \bmod 1813 \\ &= 43350 \bmod 1813 \\ &= 1651\end{aligned}$$

8. Este exemplo foi fornecido pelo professor Ken Calvert, da Georgia Tech.

e verificamos se é igual a (973 + 678) mod 1813 = 1651. Lembre-se de que, na derivação acima, M_1^{-1} é o inverso multiplicativo de M_1 módulo m_1, e M_2^{-1} é o inverso multiplicativo de M_2 módulo m_2.

Suponha que queiramos multiplicar 1651 (mod 1813) por 73. Multiplicamos (23, 34) por 73 e reduzimos para obter (23 × 73 mod 37, 34 × 73 mod 49) = (14, 32). Pode ser facilmente verificado que

$$\begin{aligned}(14, 32) &\longleftrightarrow [(14)(49)(34) + (32)(37)(4)] \bmod 1813\\ &= 865\\ &= 1651 \times 73 \bmod 1813\end{aligned}$$

8.5 LOGARITMOS DISCRETOS

Logaritmos discretos são fundamentais para diversos algoritmos de chave pública, incluindo a troca de chave Diffie-Hellman e o algoritmo de assinatura digital (DSA). Esta seção oferece uma rápida visão geral dos logaritmos discretos. Para o leitor interessado, mais detalhes desse assunto podem ser encontrados em [ORE67] e [LEVE90].

As potências de um inteiro, módulo *n*

Lembre-se, pelo teorema de Euler [Equação (8.4)], de que, para cada a e n que são relativamente primos:

$$a^{\phi(n)} \equiv 1 (\bmod n)$$

onde $\phi(n)$, a função tociente de Euler, é o número de inteiros positivos menores que n e relativamente primos de n. Agora, considere a expressão mais geral:

$$a^m \equiv 1 (\bmod n) \qquad \textbf{(8.10)}$$

Se a e n são relativamente primos, então existe pelo menos um inteiro m que satisfaz a Equação (8.10), a saber, $m = \phi(n)$. O expoente menos positivo m para o qual a Equação (8.10) é verdadeira, pode ser expresso de várias maneiras:

- a ordem de a (mod n)
- o expoente ao qual a pertence a (mod n)
- a extensão do período gerado por a

Para ver esse último ponto, considere as potências de 7, módulo 19:

$$\begin{aligned}7^1 &\equiv & 7 (\bmod 19)\\ 7^2 &= 49 = 2 \times 19 + 11 \equiv & 11 (\bmod 19)\\ 7^3 &= 343 = 18 \times 19 + 1 \equiv & 1 (\bmod 19)\\ 7^4 &= 2401 = 126 \times 19 + 7 \equiv & 7 (\bmod 19)\\ 7^5 &= 16807 = 884 \times 19 + 11 \equiv & 11 (\bmod 19)\end{aligned}$$

Não há sentido em continuar, pois a seqüência é repetitiva. Isso pode ser provado observando-se que $7^3 \equiv 1 (\bmod 19)$ e, portanto, $7^{3+j} \equiv 7^3 7^j \equiv 7^j (\bmod 19)$, e por isso duas potências quaisquer de 7 cujos expoentes diferem em 3 (ou um múltiplo de 3) são congruentes entre si (mod 19). Em outras palavras, a seqüência é periódica, e a extensão do período é o menor expoente positivo m tal que $7^m \equiv 1 (\bmod 19)$.

A Tabela 8.3 mostra todas as potências de a, módulo 19, para todo positivo $a < 19$. A extensão da seqüência para cada valor de base é indicada pelo sombreado. Observe o seguinte:

1. Todas as seqüências terminam em 1. Isso é coerente com o raciocínio dos parágrafos anteriores.
2. A extensão de uma seqüência divide $\phi(19) = 18$, ou seja, um número inteiro de seqüências ocorre em cada linha da tabela.

3. Algumas das seqüências têm extensão 18. Nesse caso, diz-se que o inteiro de base a gera (por potências) o conjunto de inteiros diferentes de zero módulo 19. Cada um desses inteiros é chamado de raiz primitiva do módulo 19.

Geralmente, podemos dizer que o expoente mais alto possível ao qual um número pode pertencer (mod n) é $\phi(n)$. Se um número for dessa ordem, ele é considerado uma **raiz primitiva** de n. A importância dessa noção é que, se a for uma raiz primitiva de n, então suas potências

$$a, a^2, \ldots, a^{\phi(n)}$$

são distintas (mod n) e são todas relativamente primas de n. Em particular, para um número primo p, se a é uma raiz primitiva de p, então

$$a, a^2, \ldots, a^{p-1}$$

são distintos (mod p). Para o número primo 19, suas raízes primitivas são 2, 3, 10, 13, 14 e 15.

Nem todos os inteiros possuem raízes primitivas. De fato, os únicos inteiros com raízes primitivas são aqueles no formato 2, 4, p^α e $2p^\alpha$ onde p é qualquer primo ímpar e α é um inteiro positivo. A prova não é simples, mas pode ser encontrada em muitos livros sobre a Teoria dos Números, incluindo [ORE76].

Logaritmos para aritmética modular

Com números reais positivos comuns, a função de logaritmo é o inverso da exponenciação. Existe uma função semelhante para a aritmética modular.

Vamos rever rapidamente as propriedades dos logaritmos comuns. O logaritmo de um número é definido como sendo a potência à qual alguma base positiva (exceto 1) precisa ser elevada para igualar o número, ou seja, para a base x e para um valor y:

$$y = x^{\log_x(y)}$$

As propriedades dos logaritmos incluem as seguintes:

$$\log_x(1) = 0$$
$$\log_x(x) = 1$$
$$\log_x(yz) = \log_x(y) + \log_x(z) \quad \textbf{(8.11)}$$
$$\log_x(y^r) = r \times \log_x(y) \quad \textbf{(8.12)}$$

Tabela 8.3 Potências de inteiros, módulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Considere uma raiz primitiva a para algum número primo p (o argumento pode ser desenvolvido para não-primos também). Então, sabemos que as potências de a partir de 1 até $(p - 1)$ produzem cada inteiro de 1 até $(p - 1)$ exatamente uma vez. Também sabemos que qualquer inteiro b satisfaz

$$b \equiv r(\text{mod } p) \text{ para algum } r, \text{ onde } 0 \leq r \leq (p - 1)$$

pela definição da aritmética modular. Segue-se que, para qualquer inteiro b e uma raiz primitiva a de número primo p, podemos encontrar um expoente exclusivo i tal que

$$b \equiv a^i(\text{mod } p) \text{ onde } 0 \leq i \leq (p - 1)$$

Esse expoente i é conhecido como o logaritmo discreto do número b para a base $a(\text{mod } p)$. Indicamos esse valor como $\text{dlog}_{a,p}(b)$.[9]

Observe o seguinte:

$$\text{dlog}_{a,p}(1) = 0, \text{ pois } a^0 \text{ mod } p = 1 \text{ mod } p = 1 \quad \textbf{(8.13)}$$

$$\text{dlog}_{a,p}(a) = 1, \text{ pois } a^1 \text{ mod } p = a \quad \textbf{(8.14)}$$

Aqui está um exemplo usando um módulo não-primo, $n = 9$. Aqui, $\phi(n) = 6$, e $a = 2$ é uma raiz primitiva. Calculamos as diversas potências de a e encontramos

$$\begin{array}{ll} 2^0 = 1 & 2^4 \equiv 7(\text{mod } 9) \\ 2^1 = 2 & 2^5 \equiv 5(\text{mod } 9) \\ 2^2 = 4 & 2^6 \equiv 1(\text{mod } 9) \\ 2^3 = 8 & \end{array}$$

Isso nos dá a seguinte tabela de números com determinados logaritmos discretos (mod 9) para a raiz $a = 2$:

Logaritmo	0	1	2	3	4	5
Número	1	2	4	8	7	5

Para facilitar a obtenção dos logaritmos discretos de determinado número, rearrumamos a tabela:

Número	1	2	4	5	7	8
Logaritmo	0	1	2	5	4	3

Agora, considere

$$x = a^{\text{dlog}_{a,p}(x)} \text{ mod } p \qquad y = a^{\text{dlog}_{a,p}(y)} \text{ mod } p$$
$$xy = a^{\text{dlog}_{a,p}(xy)} \text{ mod } p$$

Usando as regras da multiplicação modular,

$$\begin{aligned} xy \text{ mod } p &= [(x \text{ mod } p)(y \text{ mod } p)] \text{ mod } p \\ a^{\text{dlog}_{a,p}(xy)} \text{ mod } p &= \left[\left(a^{\text{dlog}_{a,p}(x)} \text{ mod } p\right)\left(a^{\text{dlog}_{a,p}(y)} \text{ mod } p\right)\right] \text{ mod } p \\ &= \left(a^{\text{dlog}_{a,p}(x) + d\text{log}_{a,p}(y)}\right) \text{ mod } p \end{aligned}$$

Mas agora considere o teorema de Euler, que afirma que, para cada a e n que são relativamente primos:

$$a^{\phi(n)} \equiv 1(\text{mod } n)$$

9. Muitos textos se referem ao logaritmo discreto como o *índice*. Geralmente, não existe uma notação convencionada para esse conceito, muito menos um nome convencionado.

Qualquer inteiro positivo z pode ser expresso na forma $z = q + k\phi(n)$, com $0 \leq q < \phi(n)$. Portanto, pelo teorema de Euler,

$$a^z \equiv a^q (\text{mod } n) \qquad \text{se } z \equiv q \bmod \phi(n)$$

Aplicando isso à igualdade anterior, temos

$$\text{dlog}_{a,p}(xy) \equiv [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)] \ (\text{mod } \phi(p))$$

e generalizando,

$$\text{dlog}_{a,p}(y^r) \equiv [r \times \text{dlog}_{a,p}(y)] \ (\text{mod } \phi(p))$$

Isso demonstra a analogia entre logaritmos verdadeiros e logaritmos discretos.

Lembre-se de que os logaritmos discretos exclusivos mod m em alguma base a só existem se a for uma raiz primitiva de m.

A Tabela 8.4, que é derivada diretamente da Tabela 8.3, mostra os conjuntos de logaritmos discretos que podem ser definidos para o módulo 19.

Cálculo de logaritmos discretos

Considere a equação

$$y = g^x \bmod p$$

Dados g, x e p, é muito simples calcular y. No pior dos casos, temos de realizar x multiplicações repetidas, e existem algoritmos para se conseguir maior eficiência (ver Capítulo 9).

Porém, dados y, g e p, em geral, é muito difícil calcular x (considere o logaritmo discreto). A dificuldade parece estar na mesma ordem de grandeza daquela da fatoração de números primos, exigida para o RSA. No momento em que este livro foi escrito, o algoritmo mais rápido conhecido para realizar logaritmos discretos módulo um número primo estava na ordem de [BETH91]:

$$e^{\left((\ln p)^{1/3}(\ln(\ln p))^{2/3}\right)}$$

que não é viável para grandes números primos.

Tabela 8.4 Tabelas de logaritmos discretos, módulo 19

(a) Logaritmos discretos com a base 2, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Logaritmos discretos com a base 3, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Logaritmos discretos com a base 10, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Logaritmos discretos com a base 13, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Logaritmos discretos com a base 14, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Logaritmos discretos com a base 15, módulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

8.6 LEITURA E SITES WEB RECOMENDADOS

Existem muitos textos básicos a respeito de Teoria de Números, que oferecem muito mais detalhes do que a maioria dos leitores deste livro deseja. Uma rápida introdução elementar, embora útil, é [ORE67]. Para o leitor interessado em uma abordagem mais profunda, dois livros-texto excelentes sobre o assunto são [KUMA98] e [ROSE05]. [LEVE90] também é um abordagem fácil e detalhada. Todos esses livros incluem problemas com soluções, aumentando seu valor como auto-estudo.

Para os leitores que desejam realmente despender tempo, talvez a melhor maneira de obter um conhecimento sólido dos fundamentos da Teoria dos Números seja trabalhar com [BURN97], que consiste unicamente em uma série de exercícios com soluções que acompanham o aluno passo a passo pelos conceitos da Teoria dos Números; fazer todos os exercícios é equivalente a concluir um curso de formação sobre Teoria dos Números.

BURN97 Burn, R. *A pathway to number theory*. Cambridge: Cambridge University Press, 1997.
KUMA98 Kumanduri, R. e Romero, C. *Number theory with computer applications*. Upper Saddle River: Prentice Hall, 1998.
LEVE90 Leveque, W. *Elementary theory of numbers*. New York: Dover, 1990.
ORE67 Ore, O. *Invitation to number theory*. Washington: The Mathematical Association of America, 1967.
ROSE05 Rosen, K. *Elementary number theory and its applications*. Reading: Addison-Wesley, 2000.

Site Web recomendado:

- **The Prime Pages:** Pesquisa, registros e recursos para números primos.

8.7 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

função tociente de Euler
índice
logaritmo discreto
número composto
número primo
ordem
raiz primitiva
teorema chinês do resto
teorema de Euler
teorema de Fermat

Perguntas de revisão

8.1 O que é um número primo?

8.2 Qual é o significado da expressão *a divide b*?

8.3 O que é a função tociente de Euler?

8.4 O teste de Miller-Rabin pode determinar se um número não é primo, mas não pode determinar se um número é primo. Como esse algoritmo pode ser usado para testar se o número é primo?

8.5 O que é uma raiz primitiva de um número?

8.6 Qual é a diferença entre um índice e um logaritmo discreto?

Problemas

8.1 A finalidade deste problema é determinar quantos números primos existem. Suponha que haja um total de n números primos, e os listemos em ordem: $p_1 = 2 < p_2 = 3 < p_3 = 5 < \cdots < p_n$.

a. Defina $X = 1 + p_1p_2 ... p_n$. Ou seja, X é igual a um mais o produto de todos os primos. Podemos encontrar o número primo p_m que divide X?

b. O que você pode dizer sobre m?

c. Deduza que o número total de primos não pode ser finito.

d. Mostre que $p_{n+1} \leq 1 + p_1p_2 \cdots p_n$.

8.2 A finalidade deste problema é demonstrar que a probabilidade de que dois números aleatórios sejam relativamente primos é de cerca de 0,6.

a. Considere $P = \Pr[\text{mdc}(a, b) = 1]$. Mostre que $\Pr[\text{mdc}(a, b) = d] = P/d^2$. *Dica*: Considere a quantidade igualdade $\text{mdc}\left(\frac{a}{d}, \frac{b}{d}\right)$.

b. A soma do resultado da parte (a) sobre todos os valores possíveis de d é 1. Ou seja: $\sum_{d \geq 1} \Pr[\gcd(a, b) = d] = 1$. Use essa igualdade para determinar o valor de P. *Dica:* Use a igualdade $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$.

8.3 Por que $\text{mdc}(n, n + 1) = 1$ para dois inteiros consecutivos n e $n + 1$?

8.4 Usando o teorema de Fermat, encontre $3^{201} \bmod 11$.

8.5 Use o Teorema de Fermat para encontrar um número a entre 0 e 72 com a congruente a 9794 módulo 73.

8.6 Use o Teorema de Fermat para encontrar um número x entre 0 e 28 com x^{85} congruente a 6 módulo 29. (Você não precisará usar qualquer pesquisa por força bruta.)

8.7 Use o Teorema de Euler para encontrar um número a entre 0 e 9 tal que a seja congruente a 7^{1000} módulo 10. (Observe que isso é o mesmo que o último dígito da expansão decimal de 7^{1000}).

8.8 Use o Teorema de Euler para encontrar um número x entre 0 e 28 com x^{85} congruente a 6 módulo 35. (Você não precisará usar qualquer pesquisa por força bruta.)

8.9 Observe, na Tabela 8.2, que $\phi(n)$ é par para $n > 2$. Isso é verdade para todo $n > 2$. Forneça um argumento conciso para explicar por que isso acontece.

8.10 Prove o seguinte: Se p é primo, então $\phi(p^i) = p^i - p^{i-1}$. *Dica*: que números têm um fator em comum com p^i?

8.11 Podemos mostrar (consulte qualquer livro sobre a Teoria de Números) que, se $\text{mdc}(m, n) = 1$, então $\phi(mn) = \phi(m)\phi(n)$. Usando essa propriedade e a propriedade desenvolvida no problema anterior, e a propriedade de que $\phi(p) = p - 1$ para p primo, é simples determinar o valor de $\phi(n)$ para qualquer n. Determine o seguinte:

a. $\phi(41)$ **b.** $\phi(27)$
c. $\phi(231)$ **d.** $\phi(440)$

8.12 Também pode ser mostrado que, para qualquer inteiro positivo a, $\phi(a)$ é dado por:

$$\phi(a) = \prod_{i=1}^{t}\left[p_i^{a_i-1}(p_i - 1)\right]$$

onde a é dado pela Equação (8.1), a saber: $a = p_1^{a_1}p_2^{a_2}\ldots p_t^{a_t}$. Demonstre esse resultado.

8.13 Considere a função: $f(n)$ = número de elementos no conjunto $\{a{:}0 \le a < n \text{ e } \text{mdc}(a, n) = 1\}$. O que é essa função?

8.14 Embora os antigos matemáticos chineses tivessem feito um bom trabalho apresentando seu teorema do resto, eles nem sempre acertavam. Eles tinham um teste de números primos. O teste dizia que n é primo se e somente se n dividir $(2^n - 2)$.

a. Dê um exemplo que satisfaça a condição usando um primo ímpar.
b. A condição é obviamente verdadeira para $n = 2$. Prove que a condição é verdadeira se n for um primo ímpar (provando a condição **se**).
c. Dê um exemplo de um n ímpar que não seja primo e que não satisfaça a condição. Você pode fazer isso com números não primos até um valor muito grande. Isso confundiu os matemáticos chineses, que pensaram que, se a condição fosse verdadeira, então n seria primo.
d. Infelizmente, os antigos chineses nunca experimentaram $n = 341$, que não é primo ($341 = 11 \times 31$) e ainda 341 divide $2^{341} - 2$ sem resto. Demonstre que $2^{341} \equiv 2 \pmod{341}$ (invalidando a condição **somente se**). *Dica*: não é necessário calcular 2^{341}; em vez disso, use as congruências.

8.15 Mostre que, se n é um inteiro composto ímpar, então o teste de Miller-Rabin retornará `inconclusivo` para a = 1 e $a = (n - 1)$.

8.16 Se n é composto e passa no teste de Miller-Rabin para a base a, então n é chamado de *pseudoprimo forte na base a*. Mostre que 2047 é um pseudoprimo na base 2.

8.17 Uma formulação comum do teorema chinês do resto (TCR) é a seguinte: Considere que $m_1, ..., m_k$ sejam inteiros relativamente primos par a par, para $1 \le i, j \le k$, e $i \ne j$. Defina M como sendo o produto de todos os m_i's. Considere que $a_1, ..., a_k$ sejam inteiros. Então, o conjunto de congruências:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

tem uma solução exclusiva módulo M. Mostre que o teorema declarado dessa forma é verdadeiro.

8.18 O exemplo usado por Sun-Tsu para ilustrar o TCR foi

$$x \equiv 2 \pmod 3; \quad x \equiv 3 \pmod 5; \quad x \equiv 2 \pmod 7$$

Resolva x.

8.19 Seis professores iniciam cursos na segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira e sábado, respectivamente, e anunciam suas intenções ministrar palestras em intervalos de 2, 3, 4, 1, 6 e 5 dias, respectivamente. As diretrizes da universidade proíbem palestras aos domingos (de modo que uma palestra no domingo deverá ser omitida). Quando todos os seis professores serão obrigados pela primeira vez a omitir uma palestra? *Dica*: use o TCR.

8.20 Encontre todas as raízes primitivas de 25.

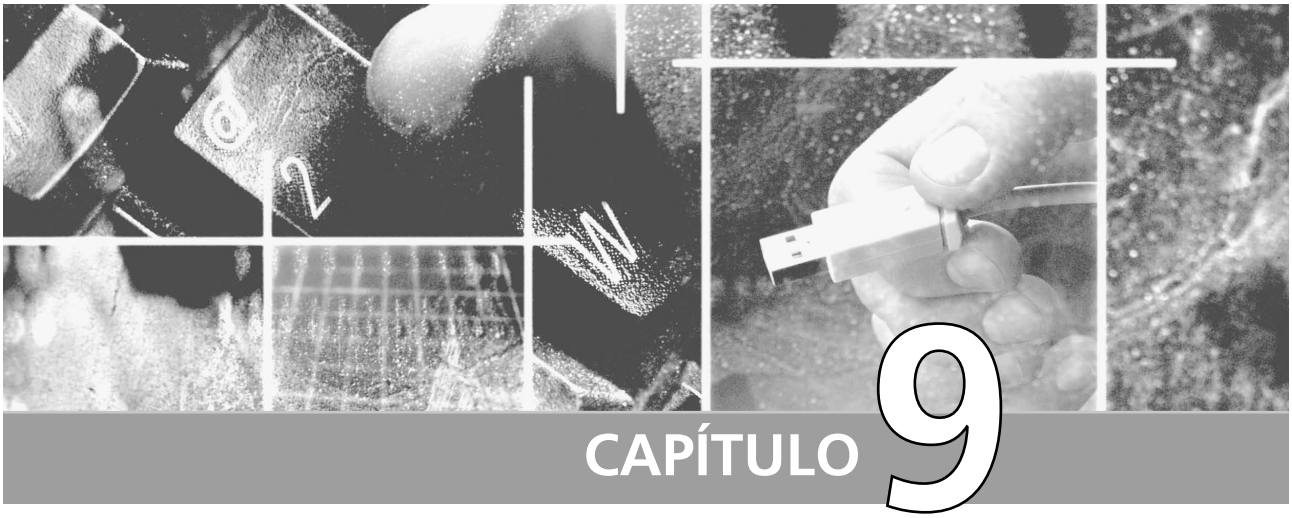
8.21 Dado 2 como uma raiz primitiva de 29, construa uma tabela de logaritmos discretos e use-a para solucionar as seguintes congruências:

a. $17x^2 \equiv 10 \pmod{29}$
b. $x^2 - 4x - 16 \equiv 0 \pmod{29}$
c. $x^7 \equiv 17 \pmod{29}$

Problemas de programação

8.22 Escreva um programa de computador que implemente a exponenciação rápida (quadrados sucessivos) módulo n.

8.23 Escreva um programa de computador que implemente o algoritmo de Miller-Rabin para um n especificado pelo usuário. O programa deverá permitir que o usuário faça duas escolhas: (1) especifique uma possível testemunha a para testar o uso do procedimento Witness, ou (2) especifique um número s de testemunhas aleatórias para o teste de Miller-Rabin verificar.



CAPÍTULO 9

Criptografia de chave pública e RSA

9.1 **Princípios de criptossistemas de chave pública**

9.2 **O algoritmo RSA**

9.3 **Leitura e sites Web recomendados**

9.4 **Principais termos, perguntas de revisão e problemas**

Apêndice 9A Prova do algoritmo RSA

Apêndice 9B A complexidade dos algoritmos

Cada egípcio recebia dois nomes, que eram conhecidos respectivamente como o nome verdadeiro e o nome bom, ou o nome grande e o nome pequeno; e enquanto o nome bom ou pequeno era público, o nome verdadeiro ou grande era cuidadosamente ocultado.

—*The golden bough,* Sir James George Frazer

PONTOS PRINCIPAIS

- A criptografia assimétrica é uma forma de criptossistema em que a criptografia e a decriptografia são realizadas usando diferentes chaves — uma chave pública e uma chave privada. Ela também é conhecida como criptografia de chave pública.
- A criptografia assimétrica transforma o texto claro em texto cifrado usando uma de duas chaves e um algoritmo de criptografia. Usando a outra chave associada e um algoritmo de decriptografia, o texto claro é recuperado a partir do texto cifrado.
- A criptografia assimétrica pode ser usada para confidencialidade, autenticação ou ambos.
- O criptossistema de chave pública mais utilizado é o RSA. A dificuldade de atacar o RSA está na dificuldade de encontrar os fatores primos de um número composto.

O desenvolvimento da criptografia de chave pública é a maior e talvez a única verdadeira revolução na história da criptografia. Desde o seu início até os tempos modernos, praticamente todos os sistemas criptográficos têm sido baseados nas ferramentas elementares da substituição e permutação. Depois de milênios de trabalho com algoritmos que basicamente poderiam ser calculados à mão, um grande avanço na criptografia simétrica ocorreu com o desenvolvimento da máquina de criptografia/decriptografia de rotor. O rotor eletromecânico permitiu o desenvolvimento de sistemas de cifra incrivelmente complexos. Com a disponibilidade dos computadores, sistemas ainda mais complexos foram criados, sendo que o mais importante foi o Lucifer na IBM, o qual culminou com o Data Encryption Standard (DES). Mas tanto as máquinas de rotor quanto o DES, embora representando avanços significativos, ainda contavam com ferramentas básicas de substituição e permutação.

A criptografia de chave pública oferece uma mudança radical em relação a tudo o que havia sido feito. Por um lado, os algoritmos de chave pública são baseados em funções matemáticas, em vez de na substituição e permutação. Mais importante, a criptografia de chave pública é assimétrica, envolvendo o uso de duas chaves separadas, diferentemente da criptografia simétrica, que usa apenas uma chave. O uso de duas chaves tem profundas conseqüências nas áreas de confidencialidade, distribuição de chaves e autenticação, conforme veremos.

Antes de prosseguirmos, temos de mencionar várias concepções erradas comuns com relação à criptografia de chave pública. Uma delas é que a criptografia de chave pública é mais segura contra criptoanálise do que a criptografia simétrica. Na verdade, a segurança de qualquer esquema de criptografia depende do tamanho da chave e do trabalho computacional envolvido para quebrar uma cifra. Não há nada, em princípio, sobre a criptografia simétrica ou de chave pública que torne uma superior à outra, do ponto de vista de resistência à criptoanálise.

Um segundo erro de conceito é que a criptografia de chave pública é uma técnica de uso geral que tornou a criptografia simétrica obsoleta. Ao contrário, devido ao overhead computacional dos esquemas de criptografia de chave pública atuais, parece improvável que a criptografia simétrica seja abandonada. Conforme um dos inventores da criptografia de chave pública disse em [DIFF88], "a restrição da criptografia de chave pública a aplicações de gerenciamento de chaves e de assinatura é aceita quase universalmente".

Finalmente, há uma sensação de que a distribuição de chaves é trivial quando se usa a criptografia de chave pública, em comparação com o tratamento um tanto desajeitado dos centros de distribuição de chaves para a criptografia simétrica. Na verdade, é preciso haver alguma forma de protocolo, geralmente envolvendo um agente central, e os procedimentos envolvidos não são mais simples nem mais eficientes do que os exigidos para a criptografia simétrica (por exemplo, veja a análise em [NEED78]).

Este capítulo e o seguinte oferecem uma visão geral da criptografia de chave pública. Primeiro, examinaremos sua estrutura conceitual. É interessante que o conceito dessa técnica tenha sido desenvolvido e publicado antes que fosse demonstrado ser prático adotá-la. Em seguida, examinaremos o algoritmo RSA, que é o mais importante algoritmo de criptografia/decriptografia viável para a criptografia de chave pública. Outros tópicos serão explorados no Capítulo 10.

Grande parte da teoria dos criptossistemas de chave pública baseia-se na Teoria dos Números. Se alguém estiver preparado para aceitar os resultados fornecidos neste capítulo, um conhecimento da Teoria dos Números não é estritamente necessário. Porém, para apreciar por completo os algoritmos de chave pública, algum conhecimento da Teoria dos Números é exigido. O Capítulo 8 oferece a base necessária sobre Teoria dos Números.

9.1 PRINCÍPIOS DE CRIPTOSSISTEMAS DE CHAVE PÚBLICA

O conceito de criptografia de chave pública evoluiu de uma tentativa de atacar dois dos problemas mais difíceis associados à criptografia simétrica. O primeiro problema é o da distribuição de chaves, que foi examinada com detalhes no Capítulo 7.

Como vimos, a distribuição de chaves sob a criptografia simétrica requer (1) que dois comunicantes já compartilhem uma chave de alguma forma distribuída a eles; ou (2) o uso de um centro de distribuição de chaves. Whitfield Diffie, um dos inventores da criptografia de chave pública (juntamente com Martin Hellman, ambos da Stanford University, na época), descobriu que esse segundo requisito anulava a própria essência da criptografia: a capacidade de manter sigilo total sobre sua própria comunicação. Conforme foi dito por Diffie [DIFF88]: "afinal, qual é a vantagem de desenvolver criptossistemas impenetráveis, se seus usuários forem forçados a compartilhar suas chaves com um CDC que pode estar sujeito a roubo ou suborno?"

O segundo problema sobre o qual Diffie ponderou, e que não estava aparentemente relacionado com o primeiro, foi o das 'assinaturas digitais'. Se o uso da criptografia tivesse de se tornar comum, não apenas nas situações militares, mas para fins comerciais e particulares, então as mensagens e documentos eletrônicos precisariam do equivalente das assinaturas usadas nos documentos em papel. Ou seja, poderia ser criado um método para garantir, de modo que todas as partes ficassem convencidas, que uma mensagem digital foi enviada por determinada pessoa? Este é um requisito um pouco mais amplo que o da autenticação, e suas características e ramificações serão exploradas no Capítulo 13.

Diffie e Hellman obtiveram um avanço incrível em 1976 [DIFF76a, b], ao retornarem quatro mil anos e apresentarem um método que resolvia os dois problemas e que era radicalmente diferente de todas as técnicas anteriores de criptografia.[1]

Na próxima subseção, examinamos a estrutura geral para a criptografia de chave pública. Depois, examinamos os requisitos para o algoritmo de criptografia/decriptografia que estão no âmago do esquema.

Criptossistemas de chave pública

Os algoritmos assimétricos contam com uma chave para criptografia e uma chave diferente, porém relacionada, para a decriptografia. Esses algoritmos têm a seguinte característica importante:

- É computacionalmente inviável determinar a chave de decriptografia dado apenas o conhecimento do algoritmo de criptografia e da chave de criptografia.

Além disso, alguns algoritmos, como o RSA, também exibem a seguinte característica:

- Qualquer uma das duas chaves relacionadas pode ser usada para criptografia, com a outra usada para a decriptografia.

Um esquema de criptografia de chave pública possui seis ingredientes (Figura 9.1a; compare com a Figura 2.1):

- **Texto claro:** essa é a mensagem ou dados legíveis que são alimentados no algoritmo como entrada.
- **Algoritmo de criptografia:** o algoritmo de criptografia realiza várias transformações no texto claro.
- **Chaves pública e privada:** esse é um par de chaves que foi selecionado de modo que, se uma for usada para criptografia, a outra será usada para decriptografia. As transformações exatas realizadas pelo algoritmo dependem da chave pública ou privada que é fornecida como entrada.
- **Texto cifrado:** essa é a mensagem codificada produzida como saída. Ela depende do texto claro e da chave. Para uma determinada mensagem, duas chaves diferentes produzirão dois textos cifrados diferentes.
- **Algoritmo de decriptografia:** esse algoritmo aceita o texto cifrado e a chave correspondente e produz o texto claro original.

As etapas essenciais são as seguintes:

1. Cada usuário gera um par de chaves a ser usado para a criptografia e a decriptografia das mensagens.
2. Cada usuário coloca uma das duas chaves em um registro público ou outro arquivo acessível. Essa é a chave pública. A outra chave permanece privada. Como a Figura 9.1a sugere, cada usuário mantém um conjunto de chaves públicas obtidas de outros usuários.
3. Se Bob deseja enviar uma mensagem confidencial para Alice, Bob criptografa a mensagem usando a chave pública de Alice.
4. Quando Alice recebe a mensagem, ela a decriptografa usando sua chave privada. Nenhum outro destinatário pode decriptografar a mensagem, pois somente Alice conhece a sua chave privada.

Com essa técnica, todos os participantes têm acesso às chaves púbicas, as chaves privadas são geradas localmente por cada participante e, portanto, nunca precisam ser distribuídas. Desde que a chave privada de um usuário permaneça protegida e secreta, a comunicação que chega está protegida. A qualquer momento, um sistema pode alterar sua chave privada e publicar a chave pública correspondente para substituir sua antiga chave pública.

A Tabela 9.1 resume alguns aspectos importantes da criptografia simétrica e da de chave pública. Para diferenciar as duas, vamos nos referir à chave usada na criptografia simétrica como uma **chave secreta.** As duas chaves usadas para a criptografia assimétrica são denominadas **chave pública** e **chave privada**.[2] Invariavelmente, a chave privada é mantida secreta, mas é conhecida como chave privada, em vez de chave secreta, para evitar confusão com a criptografia simétrica.

---

1. Diffie e Hellman apresentaram *publicamente* os conceitos da criptografia de chave pública em 1976. Porém, esse não é o verdadeiro início. O almirante Bobby Inman, enquanto diretor da National Security Agency (NSA), afirmou que a criptografia de chave pública tinha sido criada na NSA em meados da década de 1960 [SIMM93]. A primeira apresentação *documentada* desses conceitos aconteceu em 1970, pelo Communications-Electronics Security Group, o equivalente britânico da NSA, em um relatório confidencial de James Ellis [ELLI70]. Ellis referia-se à técnica como criptografia não secreta (nonsecret encryption), e descreve a criação em [ELLI99].

2. A notação a seguir é usada consistentemente em todo o livro. Uma chave secreta é representada por K_{mm}, onde m é algum modificador; por exemplo, K_a é uma chave secreta possuída pelo usuário A. Uma chave pública é representada por PU_a, para o usuário A, e a chave privada correspondente é PR_a. A criptografia do texto claro X pode ser realizada com uma chave secreta, uma chave pública ou uma chave privada, indicada por E(Ka,X), E(PUa, X), E(PRa, X), respectivamente. De modo semelhante, a decriptografia do texto cifrado C pode ser realizada com uma chave secreta, uma chave pública ou uma chave privada, indicada por D(Ka,X), D(PUa, X), D(PRa, X), respectivamente.

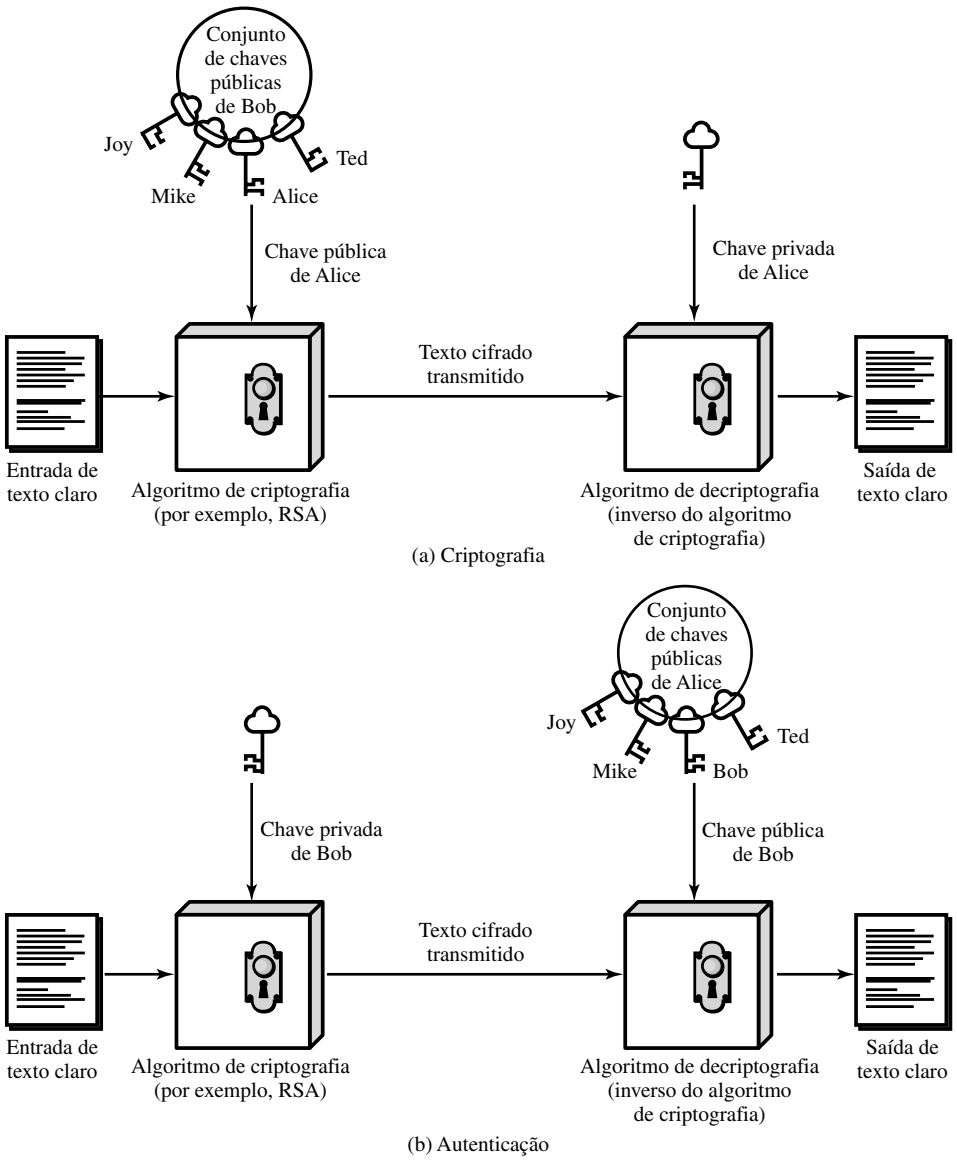


Figura 9.1 Criptografia de chave pública.

Tabela 9.1 Criptografia convencional e de chave pública

Criptografia convencional	Criptografia de chave pública
Necessário para funcionar: 1. O mesmo algoritmo com a mesma chave é usado para criptografia e decriptografia. 2. O emissor e o receptor precisam compartilhar o algoritmo e a chave. **Necessário para a segurança:** 1. A chave precisa permanecer secreta. 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais amostras do texto cifrado precisam ser insuficientes para determinar a chave.	**Necessário para funcionar:** 1. Um algoritmo é usado para criptografia e decriptografia com um par de chaves, uma para criptografia e outra para decriptografia. 2. O emissor e o receptor precisam ter uma das chaves do par casado de chaves (não a mesma chave). **Necessário para a segurança:** 1. Uma das duas chaves precisa permanecer secreta. 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais uma das chaves mais amostras do texto cifrado precisam ser insuficientes para determinar a outra chave.

Vamos examinar mais de perto os elementos essenciais de um esquema de criptografia de chave pública, usando a Figura 9.2 (compare com a Figura 2.2). Existe alguma origem A que produz uma mensagem em texto claro, $X = [X_1, X_2, \ldots, X_M]$. Os M elementos de X são letras em algum alfabeto finito. A mensagem é destinada a B. B gera um par de chaves: uma chave pública, PU_b, e uma chave privada, PR_b. PR_b é conhecida apenas de B, enquanto PU_b está disponível publicamente e, portanto, acessível por A.

Com a mensagem X e a chave de criptografia PU_b como entrada, A forma o texto cifrado $Y = [Y_1, Y_2, \ldots, Y_N]$:

$$Y = \mathrm{E}(PU_b, X)$$

O receptor a quem se destina a mensagem, de posse da chave privada correspondente, é capaz de inverter a transformação.

$$X = \mathrm{D}(PR_b, Y)$$

Um adversário, observando Y e tendo acesso a PU_b, mas não tendo acesso a PR_b ou X, precisa tentar recuperar X e/ou PR_b. Considera-se que o adversário tem conhecimento dos algoritmos de criptografia (E) e decriptografia (D). Se o adversário estiver interessado apenas nessa mensagem em particular, então o foco do esforço será recuperar X, gerando uma estimativa do texto claro $\hat{X}$. Freqüentemente, porém, o adversário também está interessado em poder ler mensagens futuras, quando tentará recuperar PR_b gerando um $\hat{PR}_b$ estimado.

Já mencionamos que qualquer uma das duas chaves relacionadas pode ser usada para criptografia, com a outra sendo usada para decriptografia. Isso permite a implementação de um esquema criptográfico diferente. Enquanto o esquema ilustrado na Figura 9.2 oferece confidencialidade, as Figuras 9.1b e 9.3 mostram o uso da criptografia de chave pública para oferecer autenticação:

$$Y = \mathrm{E}(PR_a, X)$$
$$X = \mathrm{D}(PU_a, Y)$$

Nesse caso, A prepara uma mensagem para B e a criptografa usando a chave privada de A antes de transmiti-la. B pode decriptografar a mensagem usando a chave pública de A. Como a mensagem foi criptografada usando a chave privada de A, somente A poderia ter preparado a mensagem. Portanto, toda a mensagem criptografada serve como uma *assinatura digital*. Além disso, é impossível alterar a mensagem sem acesso à chave privada de A, de modo que a mensagem é autenticada tanto em termos da origem quanto da integridade dos dados.

No esquema anterior, a mensagem inteira é criptografada, o que, embora valide tanto o autor quanto o conteúdo, requer uma grande quantidade de armazenamento. Cada documento precisa ser mantido em texto claro para ser usado para fins práticos. Uma cópia também precisa ser armazenada em texto cifrado, para que a origem e o conteúdo possam ser verificados em caso de uma disputa. Uma forma mais eficiente de conseguir os mesmos resultados é criptografar um pequeno bloco de bits que é uma função do documento. Esse bloco, chamado autenticador, precisa ter uma propriedade que torne inviável alterar o documento sem alterar o autenticador. Se o autenticador for criptografado com a chave privada do emissor, ele servirá como uma assinatura que verifica origem, conteúdo e seqüência. O Capítulo 13 examina essa técnica com detalhes.

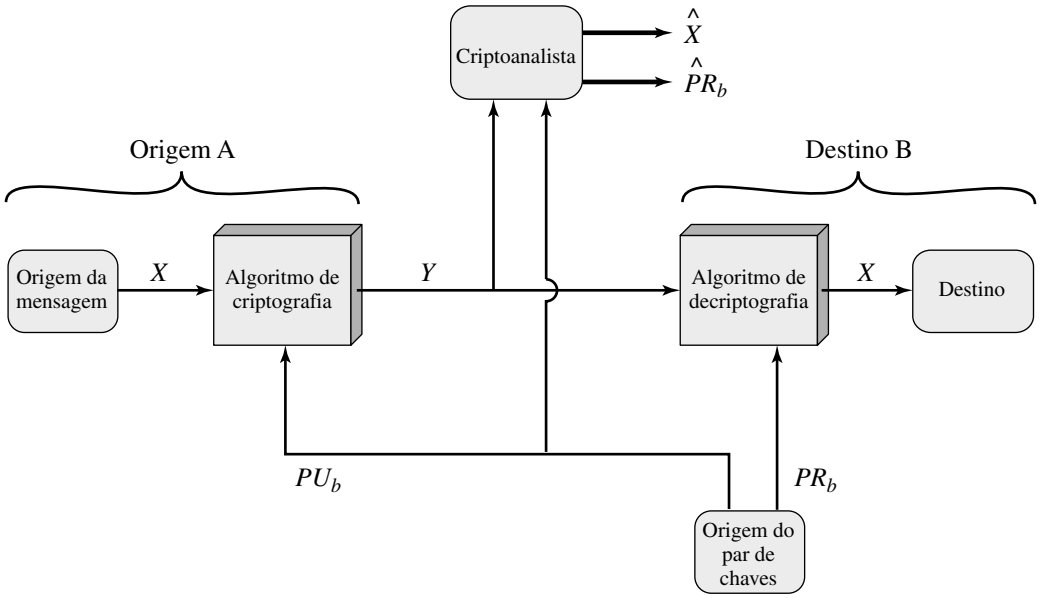


Figura 9.2 Criptossistema de chave pública: sigilo.

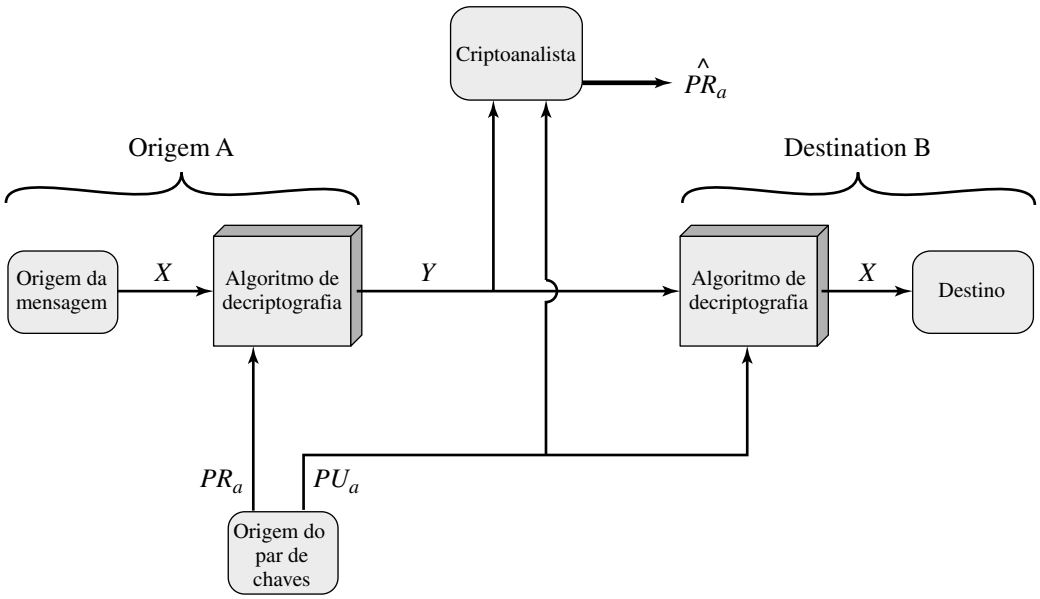


Figura 9.3 Criptossistema de chave pública: autenticação.

É importante enfatizar que o processo de criptografia representado nas Figuras 9.1b e 9.3 não oferece confidencialidade. Ou seja, a mensagem sendo enviada está segura contra alteração, mas não contra interceptação. Isso é óbvio no caso de uma assinatura baseada em uma parte da mensagem, pois o restante da mensagem é transmitido às claras. Mesmo no caso de criptografia completa, como mostrado na Figura 9.3, não existe proteção quanto à confidencialidade, pois qualquer observador pode decriptografar a mensagem usando a chave pública do emissor.

Porém, é possível oferecer a função de autenticação e confidencialidade com um uso duplo do esquema de chave pública (Figura 9.4):

$$Z = \mathrm{E}(PU_b, \mathrm{E}(PR_a, X))$$
$$X = \mathrm{D}(PU_a, \mathrm{D}(PR_b, Z))$$

Nesse caso, começamos como antes, criptografando uma mensagem, usando a chave privada do emissor. Isso fornece a assinatura digital. Em seguida, criptografamos novamente, usando a chave pública do receptor. O texto cifrado final só pode ser decriptografado pelo receptor pretendido, o qual é o único a possuir a chave privada correspondente. Assim, a confidencialidade é fornecida. A desvantagem dessa técnica é que o algoritmo de chave pública, que é complexo, precisa ser usado quatro vezes, em vez de duas, em cada comunicação.

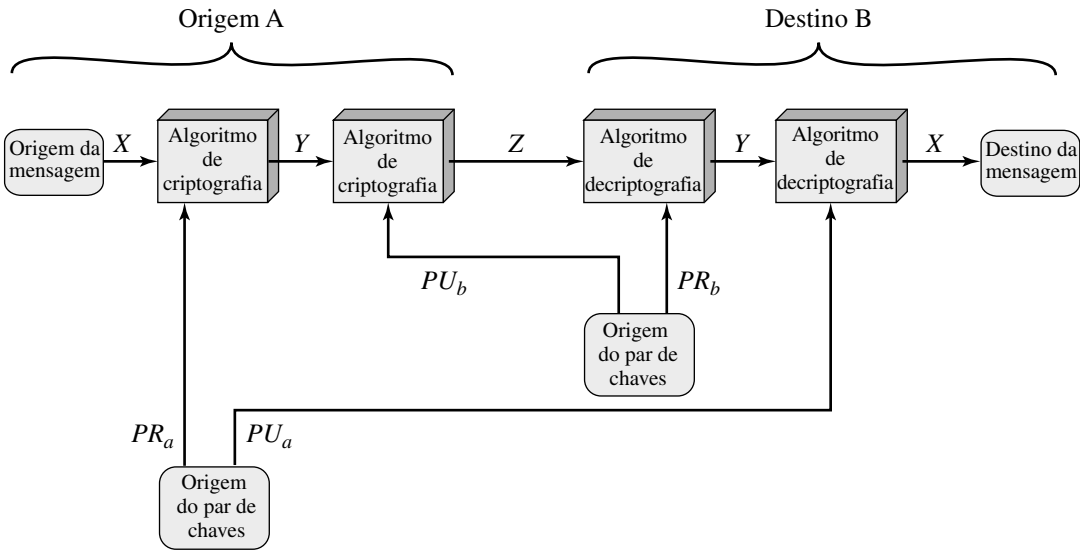


Figura 9.4 Criptossistema de chave pública: autenticação e sigilo.

Aplicações para criptossistemas de chave pública

Antes de prosseguir, precisamos esclarecer um aspecto dos criptossistemas de chave pública que pode causar confusão. Os sistemas de chave pública são caracterizados pelo uso de um algoritmo criptográfico com duas chaves, uma mantida privada e uma disponível publicamente. Dependendo da aplicação, o emissor utiliza a chave privada do emissor ou a chave pública do receptor, ou ambas, para realizar algum tipo de função criptográfica. Em termos gerais, podemos classificar o uso dos criptossistemas de chave pública em três categorias:

- **Criptografia/decriptografia:** o emissor criptografa uma mensagem com a chave pública do destinatário.
- **Assinatura digital:** o emissor 'assina' uma mensagem com sua chave privada. A assinatura é feita por um algoritmo criptográfico aplicado à mensagem ou a um pequeno bloco de dados que é uma função da mensagem.
- **Troca de chave:** dois lados cooperam para trocar uma chave de sessão. Várias técnicas diferentes são possíveis, envolvendo a(s) chave(s) privada(s) de uma ou de ambas as partes.

Alguns algoritmos são adequados para as três aplicações, enquanto outros só podem ser usados para uma ou duas dessas aplicações. A Tabela 9.2 indica as aplicações possíveis dos algoritmos explicados neste livro.

Requisitos para criptografia de chave pública

O criptossistema ilustrado nas figuras de 9.2 a 9.4 depende de um algoritmo criptográfico baseado em duas chaves relacionadas. Diffie e Hellman postularam esse sistema sem demonstrar que esses algoritmos existem. Porém, eles estabeleceram as condições a que esses algoritmos precisam atender [DIFF76b]:

1. Ser computacionalmente fácil para uma parte B gerar um par (chave pública PU_b, chave privada PR_b).
2. Ser computacionalmente fácil para um emissor A, conhecendo a chave pública e a mensagem a ser criptografada, M, gerar o texto cifrado correspondente:

$$C = \mathrm{E}(PU_b, M)$$

3. Ser computacionalmente fácil para o receptor B decriptografar o texto cifrado resultante usando a chave privada para recuperar a mensagem original:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. Ser computacionalmente inviável para um adversário, conhecendo a chave pública, PU_b, determinar a chave privada PR_b.
5. Ser computacionalmente inviável para um adversário, conhecendo a chave pública, PU_b,e um texto cifrado, C, recuperar a mensagem original, M.

Podemos incluir um sexto requisito que, embora útil, não é necessário para todas as aplicações de chave pública:

6. As duas chaves poderem ser aplicadas em qualquer ordem:

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

Esses são requisitos formidáveis, conforme evidenciado pelo fato de que somente alguns algoritmos (RSA, criptografia de chave elíptica, Diffie-Hellman, DSS) receberam ampla aceitação nas várias décadas desde que o conceito de criptografia de chave pública foi proposto.

Antes de detalhar por que os requisitos são tão formidáveis, vamos primeiro reformulá-los. Os requisitos resumem-se à necessidade de uma função unidirecional com segredo (*trap-door one-way function*). Uma função unidirecional[3] é aquela que mapeia um domínio em um intervalo de modo que todo valor da função tem um inverso único, com a condição de que o cálculo da função seja fácil, ao passo que o cálculo do inverso seja inviável.

$$Y = \mathrm{f}(X) \qquad \text{fácil}$$
$$X = \mathrm{f}^{-1}(Y) \qquad \text{inviável}$$

Geralmente, *fácil* significa um problema que pode ser resolvido em tempo polinomial como uma função do tamanho da entrada. Assim, se o tamanho da entrada for n bits, então o tempo para calcular a função será proporcional a n^a, onde a é uma constante fixa. Diz-se que esses algoritmos pertencem à classe **P**. O termo *inviável* é um con-

3. Não confunda com uma função unidirecional de hash, que toma um campo de dados arbitrariamente grande como seu argumento e o mapeia em uma saída de tamanho fixo. Tais funções são usadas para autenticação (ver Capítulo 11).

Tabela 9.2 Aplicações para criptossistemas de chave pública

Algoritmo	Criptografia/decriptografia	Assinatura digital	Troca de chave
RSA	Sim	Sim	Sim
Curva elíptica	Sim	Sim	Sim
Diffie-Hellman	Não	Não	Sim
DSS	Não	Sim	Não

ceito muito mais vago. Em geral, podemos dizer que um problema é inviável se o esforço para solucioná-lo aumentar mais rapidamente do que o tempo polinomial como uma função do tamanho da entrada. Por exemplo, se o tamanho da entrada for n bits e o tempo para calcular a função for proporcional a 2^n, o problema será considerado inviável. Infelizmente, é difícil determinar se um algoritmo em particular apresenta essa complexidade. Além disso, noções tradicionais de complexidade computacional focalizam a complexidade no pior caso e no caso médio de um algoritmo. Essas medidas são inadequadas para a criptografia, o que exige que seja inviável reverter uma função para praticamente todas as entradas, e não para o pior caso ou ainda para o caso médio. Uma breve apresentação de alguns desses conceitos é fornecida no Apêndice 9B.

Agora passamos para a definição de uma função unidirecional com segredo, que é fácil de calcular em uma direção e inviável de calcular na outra, a menos que certa informação adicional seja conhecida. Com a informação adicional, o inverso pode ser calculado em tempo polinomial. Podemos resumir da seguinte forma: uma função unidirecional com segredo é uma família de funções reversíveis f_k, tal que

$$Y = f_k(X) \qquad \text{fácil, se } k \text{ e } X \text{ forem conhecidos}$$
$$X = f_k^{-1}(Y) \qquad \text{fácil, se } k \text{ e } Y \text{ forem conhecidos}$$
$$X = f_k^{-1}(Y) \qquad \text{inviável, se } Y \text{ for conhecido, mas } k \text{ não for conhecido}$$

Assim, o desenvolvimento de um esquema de chave pública prático depende da descoberta de uma função unidirecional com segredo adequada.

Criptoanálise de chave pública

Assim como a criptografia simétrica, um esquema de criptografia de chave pública é vulnerável a um ataque de força bruta. A contramedida é a mesma: usar chaves grandes. Porém, existe um compromisso a ser considerado. Os sistemas de chave pública dependem do uso de algum tipo de função matemática inversível. A complexidade do cálculo dessas funções pode não aumentar linearmente com o número de bits na chave, mas sim mais rapidamente do que isso. Assim, o tamanho da chave precisa ser grande o suficiente para tornar o ataque de força bruta impraticável, mas pequeno o suficiente para que a criptografia e a decriptografia sejam viáveis. Na prática, os tamanhos de chave que têm sido propostos tornam o ataque por força bruta impraticável, mas resultam em velocidades de criptografia/decriptografia muito lentas para o uso geral. Em vez disso, conforme já dissemos, a criptografia de chave pública atualmente está confinada a aplicações de gerenciamento de chaves e de assinatura.

Outra forma de ataque é encontrar alguma maneira de calcular a chave privada dada a chave pública. Até o momento, não foi provado matematicamente que essa forma de ataque é inviável para determinado algoritmo de chave pública. Assim, qualquer algoritmo, incluindo o bastante utilizado algoritmo RSA, é suspeito. A história da criptoanálise mostra que um problema que parece insolúvel de um ponto de vista pode ter uma solução se for visto de uma maneira inteiramente diferente.

Finalmente, existe uma forma de ataque peculiar aos sistemas de chave pública que é, basicamente, um ataque de mensagem provável. Suponha, que uma mensagem tivesse de ser enviada contendo exclusivamente uma chave DES de 56 bits. Um adversário poderia criptografar todas as chaves DES possíveis de 56 bits usando a chave pública e descobrir a chave criptografada, combinando com o texto cifrado transmitido. Assim, não importa o tamanho de chave do esquema de chave pública, o ataque é reduzido a um ataque por força bruta em uma chave de 56 bits. Esse ataque pode ser impedido acrescentando-se alguns bits aleatórios a essas mensagens simples.

9.2 O ALGORITMO RSA

O artigo pioneiro de Diffie e Hellman [DIFF76b] apresentou uma nova técnica para criptografia e, na realidade, desafiou os criptologistas a encontrarem um algoritmo criptográfico que atendesse aos requisitos para os sistemas de chave pública. Uma das primeiras respostas ao desafio foi desenvolvida em 1977 por Ron Rivest, Adi Shamir e Len

Adleman, no MIT, e publicada em 1978 [RIVE78].[4] Desde então, o esquema Rivest-Shamir-Adleman (RSA) tem reinado soberano como a técnica de uso geral mais aceita e implementada para a criptografia de chave pública.

O esquema RSA é uma cifra de bloco em que o texto claro e o texto cifrado são inteiros entre 0 e $n - 1$, para algum n. Um tamanho típico para n é 1.024 bits, ou 309 dígitos decimais. Ou seja, n é menor que 2^{1024}. Examinaremos o RSA nesta seção com alguns detalhes, começando com uma explicação do algoritmo. Depois, examinaremos algumas das implicações computacionais e criptoanalíticas do RSA.

Descrição do algoritmo

O esquema desenvolvido por Rivest, Shamir e Adleman utiliza uma expressão com exponenciais. O texto claro é criptografado em blocos, com cada bloco tendo um valor binário menor que algum número n; ou seja, o tamanho do bloco precisa ser menor ou igual a $\log_2(n)$. Na prática, o tamanho do bloco é de i bits, onde $2^i < n \leq 2^{i+1}$. A criptografia e a decriptografia têm a seguinte forma, para algum bloco de texto claro M e bloco de texto cifrado C:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Tanto o emissor quanto o receptor precisam conhecer o valor de n. O emissor conhece o valor de e, e somente o receptor conhece o valor de d. Assim, esse é um algoritmo de criptografia de chave pública com uma chave pública $PU = \{e, n\}$ e uma chave privada $PR = \{d, n\}$. Para que esse algoritmo seja satisfatório para a criptografia de chave pública, os seguintes requisitos precisam ser atendidos:

1. Ser possível encontrar valores de e, d, n tais que $M^{ed} \bmod n = M$ para todo $M < n$.
2. Ser relativamente fácil calcular $M^e \bmod n$ e $C^d \bmod n$ para todos os valores de $M < n$.
3. Ser inviável determinar d dados e e n.

Por enquanto, focalizamos o primeiro requisito e consideremos as outras questões mais adiante. Precisamos encontrar um relacionamento na forma

$$M^{ed} \bmod n = M$$

O relacionamento a seguir se mantém se e e d forem inversos multiplicativos módulo $\phi(n)$, onde $\phi(n)$ é a função tociente de Euler. O Capítulo 8 mostrou que, para p, q primos, $\phi(pq) = (p - 1)(q - 1)$. O relacionamento entre e e d pode ser expresso como

$$ed \bmod \phi(n) = 1 \tag{9.1}$$

Isso é equivalente a dizer

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

Ou seja, e e d são inversos multiplicativos mod $\phi(n)$. Observe que, de acordo com as regras da aritmética modular, isso é verdadeiro somente se d (e, portanto, e) for relativamente primo a $\phi(n)$. De modo equivalente, $\text{mdc}(\phi(n), d) = 1$. Veja no Apêndice 9A uma prova de que a Equação (9.1) satisfaz o requisito para o RSA.

Agora, estamos prontos para formular o esquema RSA. Os ingredientes são os seguintes:

p, q, dois números primos	(privados, escolhidos)
$n = pq$	(público, calculado)
e, com $\text{mdc}(\phi(n), e) = 1; 1 < e < \phi(n)$	(público, escolhido)
$d \equiv e^{-1} (\bmod \phi(n))$	(privado calculado)

A chave privada consiste em $\{d, n\}$ e a chave pública consiste em $\{e, n\}$. Suponha que o usuário A tenha publicado sua chave pública e que o usuário B queira enviar a mensagem M para A. Então, B calcula $C = M^e \bmod n$ e transmite C. Ao receber esse texto cifrado, o usuário A decriptografa calculando $M = C^d \bmod n$.

A Figura 9.5 resume o algoritmo RSA. Um exemplo, obtido de [SING99], aparece na Figura 9.6. Para esse exemplo, as chaves foram geradas da seguinte forma:

4. Aparentemente, o primeiro sistema de chave pública funcional para criptografia/decriptografia foi proposto por Clifford Cocks, da CESG britânica em 1973 [COCK73]; o método de Cocks é praticamente idêntico ao RSA.

1. Selecione dois números primos, $p = 17$ e $q = 11$.
2. Calcule $n = pq = 17 \times 11 = 187$.
3. Calcule $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Selecione e tal que e seja relativamente primo a $\phi(n) = 160$ e menor que $\phi(n)$; escolhemos $e = 7$.
5. Determine d tal que $de \equiv 1 \pmod{160}$ e $d < 160$. O valor correto é $d = 23$, pois $23 \times 7 = 161 = 1 \times 160 + 1$; d pode ser calculado usando o algoritmo estendido de Euclides (Capítulo 4).

As chaves resultantes são a chave pública $PU = \{7,187\}$ e a chave privada $PR = \{23,187\}$. O exemplo mostra o uso dessas chaves para uma entrada de texto claro $M = 88$. Para a criptografia, temos de calcular $C = 88^7 \bmod 187$. Explorando as propriedades da aritmética modular, podemos fazer isso da seguinte forma:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

Geração de chaves

Selecione p, q	p e q são primos, $p \neq q$
Calcule $n = p \times q$	
Calcule $\phi(n) = (p - 1)(q - 1)$	
Selecione o inteiro e	mdc $(\phi(n), e) = 1$; $1 < e < \phi(n)$
Calcule d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chave pública	$PU = \{e, n\}$
Chave privada	$PR = \{d, n\}$

Criptografia

Texto claro:	$M < n$
Texto cifrado:	$C = M^e \bmod n$

Decriptografia

Texto cifrado:	C
Texto claro:	$M = C^d \bmod n$

Figura 9.5 O algoritmo RSA.

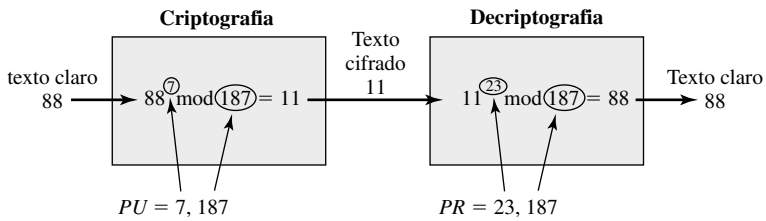


Figura 9.6 Exemplo de algoritmo RSA.

Para a decriptografia, calculamos $M = 11^{23} \bmod 187$:

$$\begin{aligned}
11^{23} \bmod 187 &= [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times \\
&\qquad (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187 \\
11^1 \bmod 187 &= 11 \\
11^2 \bmod 187 &= 121 \\
11^4 \bmod 187 &= 14{,}641 \bmod 187 = 55 \\
11^8 \bmod 187 &= 214{,}358{,}881 \bmod 187 = 33 \\
11^{23} \bmod 187 &= (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88
\end{aligned}$$

Aspectos computacionais

Agora, passemos à questão da complexidade da computação exigida para usar o RSA. Existem na realidade duas questões a considerar: criptografia/decriptografia e geração de chaves. Vamos examinar primeiro o processo de criptografia e decriptografia, para depois considerarmos o de geração de chaves.

Exponenciação na aritmética modular Tanto a criptografia quanto a decriptografia no RSA envolvem elevar um inteiro a uma potência inteira, mod n. Se a exponenciação fosse feita sobre os inteiros e depois reduzida módulo n, os valores intermediários seriam gigantescos. Felizmente, como mostra o exemplo anterior, podemos utilizar uma propriedade da aritmética modular:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Assim, podemos reduzir resultados intermediários módulo n. Isso torna o cálculo prático.

Outra consideração é a eficiência da exponenciação, pois com RSA estamos lidando com expoentes potencialmente grandes. Para ver como a eficiência poderia ser aumentada, considere que queremos calcular x^{16}. Uma técnica direta requer 15 multiplicações:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

Porém, podemos conseguir o mesmo resultado final com apenas quatro multiplicações se apanharmos repetidamente o quadrado de cada resultado parcial, formando sucessivamente x^2, x^4, x^8, x^{16}. Como outro exemplo, suponha que queiramos calcular $x^{11} \bmod n$ para alguns inteiros x e n. Observe que $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. Nesse caso, calculamos $x \bmod n$, $x^2 \bmod n$, $x^4 \bmod n$, e $x^8 \bmod n$, e depois calculamos $[(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$.

Generalizando, suponha que queiramos encontrar o valor a^b, com a e b inteiros positivos. Se expressarmos b como um número binário $b_k b_{k-1} \ldots b_0$, então temos

$$b = \sum_{b_i \neq 0} 2^i$$

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  bi = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

Nota: O inteiro b é expresso como um número binário $b_k b_{k-1} \ldots b_0$

Figura 9.7 Algoritmo para calcular a^b mod n.

Portanto,

$$a^b = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^b \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i \neq 0} \left[a^{(2^i)} \bmod n\right]\right) \bmod n$$

Então, podemos desenvolver o algoritmo[5] para calcular $a^b \bmod n$, mostrado na Figura 9.7. A Tabela 9.3 mostra um exemplo da execução desse algoritmo. Observe que a variável c não é necessária; ela é incluída para fins explicativos. O valor final de c é o valor do expoente.

Operação eficiente usando a chave pública Para agilizar a operação do algoritmo RSA usando a chave pública, normalmente é feita uma escolha específica de e. A escolha mais comum é 65537 ($2^{16} + 1$); duas outras escolhas bastante utilizadas são 3 e 17. Cada uma dessas escolhas tem apenas dois bits 1 e, por isso, o número de multiplicações exigidas para realizar a exponenciação é reduzido.

Porém, com uma chave pública muito pequena, como $e = 3$, o RSA torna-se vulnerável a um ataque simples. Suponha que tenhamos três usuários RSA diferentes, todos usando o valor $e = 3$, mas que tenham valores exclusivos de n, a saber, n_1, n_2, n_3. Se o usuário A enviar a mesma mensagem criptografada M aos três usuários, então os três textos cifrados serão $C_1 = M^3 \bmod n_1$; $C_2 = M^3 \bmod n_2$; $C_3 = M^3 \bmod n_3$. É provável que n_1, n_2 e n_3 sejam relativamente primos par a par. Portanto, pode-se usar o teorema chinês do resto (CRT— *chinese remainder theorem*) para calcular $M^3 \bmod (n_1 n_2 n_3)$. Pelas regras do algoritmo RSA, M é menor que cada um dos n_i; portanto, $M^3 < n_1 n_2 n_3$. Por conseguinte, o atacante só precisa calcular a raiz cúbica de M^3. Esse ataque pode ser impedido somando-se uma string de bits pseudo-aleatórios exclusivos como preenchimento a cada instância de M a ser criptografada. Essa técnica é explicada mais adiante.

O leitor pode ter observado que a definição do algoritmo RSA (Figura 9.5) requer que, durante a geração da chave, o usuário selecione um valor de e que seja relativamente primo a $\phi(n)$. Assim, por exemplo, se um usuário tiver pré-selecionado $e = 65537$ e depois gerado os primos p e q, pode acontecer que $\text{mdc}(\phi(n), e) \neq 1$. Assim, o usuário terá de rejeitar qualquer valor de p ou q que não seja congruente a 1 (mod 65537).

Operação eficiente usando a chave privada De maneira semelhante, não podemos escolher um valor pequeno e constante de d para operação eficiente. Um valor pequeno de d é vulnerável a um ataque por força bruta e a outras formas de criptoanálise [WIEN90]. Porém, existe uma forma de agilizar a computação usando o CRT. Queremos calcular o valor $M = C^d \bmod n$. Vamos definir os seguintes resultados intermediários:

$$V_p = C^d \bmod p \qquad V_q = C^d \bmod q$$

Seguindo o CRT, Equação (8.8), defina as quantidades:

$$X_p = q \times (q^{-1} \bmod p) \qquad X_q = p \times (p^{-1} \bmod q)$$

O CRT então mostra, usando a Equação (8.9), que

$$M = (V_p X_p + V_q X_q) \bmod n$$

Além disso, podemos simplificar o cálculo de V_p e V_q usando o teorema de Fermat, que declara que $a^{p-1} \equiv 1 (\bmod p)$ se p e a forem relativamente primos. Alguma reflexão deverá convencê-lo de que o seguinte é válido:

$$V_p = C^d \bmod p = C^{d \bmod (p-1)} \bmod p \qquad V_q = C^d \bmod q = C^{d \bmod (q-1)} \bmod q$$

Tabela 9.3 Resultado do algoritmo da exponenciação modular rápida para $a^b \bmod n$, onde $a = 7$, $b = 560 = 1000110000$, $n = 561$

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

5. O algoritmo tem uma longa história; esta expressão de pseudocódigo em particular é de [CORM01].

As quantidades $d \bmod (p - 1)$ e $d \bmod (q - 1)$ podem ser pré-calculadas. O resultado final é que o cálculo é aproximadamente quatro vezes mais rápido do que a avaliação de $M = C^d \bmod n$ diretamente [BONE02].

Geração de chaves Antes da aplicação do criptossistema de chave pública, cada participante precisa gerar um par de chaves. Isso envolve as seguintes tarefas:

- Determinar dois números primos, p e q
- Selecionar e ou d e calcular o outro

Primeiramente, considere a seleção de p e q. Como o valor de $n = pq$ será conhecido por qualquer adversário em potencial, para impedir a descoberta de p e q por métodos exaustivos, esses primos precisam ser escolhidos a partir de um conjunto suficientemente grande (ou seja, p e q precisam ser números grandes). Por outro lado, o método usado para encontrar números primos grandes precisa ser razoavelmente eficiente.

No momento, não existem técnicas úteis para gerar números primos de qualquer tamanho, de modo que é preciso algum outro meio de resolver o problema. O procedimento geralmente usado é a escolha aleatória de um número ímpar da ordem de grandeza desejada, testando, para saber se esse número é primo. Se não for, escolha números aleatórios sucessivos até que um primo seja encontrado.

Diversos testes de primalidade têm sido desenvolvidos (por exemplo, veja uma descrição de diversos desses testes em [KNUT98]). Quase invariavelmente, os testes são probabilísticos, ou seja, simplesmente determinam que um inteiro provavelmente é primo. Apesar dessa falta de certeza, esses testes podem ser executados de modo a tornar a probabilidade tão próxima de 1,0 quanto for desejado. Como exemplo, um dos algoritmos mais eficientes e conhecidos, o algoritmo Miller-Rabin, é descrito no Capítulo 8. Com esse algoritmo, e com a maioria dos algoritmos, o procedimento para testar se determinado inteiro n é primo consiste em realizar algum cálculo que envolva n e um inteiro escolhido aleatoriamente a. Se n falhar no teste, então n não é primo. Se n passar no teste, então n pode ser primo ou não. Se n passar em muitos desses testes com muitos valores escolhidos aleatoriamente para a, então podemos acreditar que n, realmente, é primo.

Resumindo, o procedimento para escolher um número primo é o seguinte.

1. Escolha aleatoriamente um inteiro ímpar n (por exemplo, usando um gerador de números pseudo-aleatórios).
2. Escolha aleatoriamente um inteiro $a < n$.
3. Realize o teste de primalidade probabilístico, como Miller-Rabin, usando a como um parâmetro. Se n falhar no teste, rejeite o valor n e volte para a etapa 1.
4. Se n tiver passado em um número suficiente de testes, aceite n; caso contrário, volte para a etapa 2.

Esse é um procedimento um tanto tedioso. Porém, lembre-se de que esse processo é realizado com relativamente pouca freqüência: somente quando um novo par (PU, PR) é necessário.

Vale a pena observar quantos números provavelmente serão rejeitados antes que um número primo seja encontrado. Um resultado da Teoria dos Números, conhecido como teorema do número primo, afirma que os primos próximos de N são espaçados na média a cada $(\ln N)$ inteiros. Assim, na média, haveria necessidade de testar algo na ordem de $\ln(N)$ inteiros antes que um primo fosse encontrado. Na realidade, como todos os inteiros pares podem ser imediatamente rejeitados, o valor correto é $\ln(N)/2$. Por exemplo, se um primo com a ordem de grandeza de 2^{200} fosse procurado, então cerca de $\ln(2^{200})/2 = 70$ tentativas seriam necessárias para encontrar um primo.

Tendo determinado os números primos p e q, o processo de geração de chaves é completado selecionando-se um valor de e e calculando-se d ou, como alternativa, selecionando-se um valor de d e calculando-se e. Considerando o primeiro, então precisamos selecionar um e tal que $\text{mdc}(\phi(n), e) = 1$ e depois calcular $d \equiv e^{-1} (\bmod\ \phi(n))$. Felizmente, existe um algoritmo simples que, ao mesmo tempo, calculará o máximo divisor comum de dois inteiros e, se o mdc for 1, determinará o inverso de um dos inteiros módulo o outro. Esse algoritmo, conhecido como algoritmo euclidiano estendido, é explicado no Capítulo 8. Assim, o procedimento é gerar uma série de números aleatórios, testando cada um em relação a $\phi(n)$ até que seja encontrado um número relativamente primo de $\phi(n)$. Novamente, podemos fazer a pergunta: quantos números aleatórios precisamos testar para encontrar um número utilizável, ou seja, um número relativamente primo a $\phi(n)$? Pode ser demonstrado facilmente que a probabilidade de que dois números aleatórios sejam relativamente primos é de cerca de 0,6; assim, muito poucos testes seriam necessários para encontrar um inteiro adequado (ver Problema 8.2).

A segurança do RSA

As quatro abordagens possíveis para atacar o algoritmo RSA são as seguintes:

- **Força bruta:** isso envolve tentar todas as chaves privadas possíveis.
- **Ataques matemáticos:** existem várias técnicas, todas equivalentes em esforço a fatorar o produto de dois primos.

- **Ataques de temporização (*timing attack*):** dependem do tempo de execução do algoritmo de decriptografia.
- **Ataques de texto cifrado escolhido:** esse tipo de ataque explora as propriedades do algoritmo RSA.

A defesa contra a técnica de força bruta é a mesma para o RSA e para outros criptossistemas, ou seja, usar um tamanho de chave grande. Assim, quanto maior o número de bits em d, melhor. Porém, devido à complexidade dos cálculos envolvidos, tanto na geração de chaves quanto na criptografia/decriptografia, quanto maior o tamanho da chave, mais lento o sistema será.

Nesta subseção, oferecemos uma visão geral dos ataques matemáticos e de temporização.

O problema da fatoração Podemos identificar três técnicas para atacar o RSA matematicamente:

- Fatorar n em seus dois fatores primos. Isso permite o cálculo de $\phi(n) = (p - 1) \times (q - 1)$, que, por sua vez, permite determinar $d \equiv e^{-1} \pmod{\phi(n)}$.
- Determinar $\phi(n)$ diretamente, sem determinar p e q primeiro. Novamente, isso permite a determinação de $d \equiv e^{-1} \pmod{\phi(n)}$.
- Determinar d diretamente, sem determinar $\phi(n)$ primeiro.

A maioria das discussões sobre a criptoanálise do RSA tem focalizado a tarefa de fatorar n em seus dois fatores primos. Determinar $\phi(n)$ dado n é equivalente a fatorar n [RIBE96]. Com os algoritmos atualmente conhecidos, determinar d dados e e n parece ser pelo menos tão demorado quanto o problema de fatoração [KALI95]. Logo, podemos usar o desempenho da fatoração como um padrão de referência em relação ao qual avaliaremos a segurança do RSA.

Para um n grande, com fatores primos grandes, fatorar é um problema difícil, mas não tão difícil como costumava ser. Um exemplo marcante disso é o seguinte. Em 1977, os três inventores do RSA desafiaram os leitores da *Scientific American* a decodificarem uma cifra que eles imprimiram na coluna 'Mathematical games' (jogos matemáticos) de Martin Gardner [GARD77]. Eles ofereceram uma recompensa de US$ 100 para o retorno de uma sentença em texto claro, um evento que, conforme previam, não poderia acontecer por cerca de 40 quadrilhões de anos. Em abril de 1994, um grupo trabalhando pela Internet reivindicou o prêmio depois de oito meses de trabalho [LEUT94]. Esse desafio usava uma chave pública com tamanho (comprimento de n) de 129 dígitos decimais, ou cerca de 428 bits. Nesse meio tempo, assim como haviam feito para o DES, o RSA Laboratories lançou desafios para a cifra RSA com tamanhos de chave de 100, 110, 120 dígitos e assim por diante. O desafio mais recente a ser vencido é o desafio RSA-200, com um tamanho de chave de 200 dígitos decimais, ou cerca de 663 bits. A Tabela 9.4 mostra os resultados até agora. O nível de esforço é medido em MIPS-anos: um processador de um milhão de instruções por segundo rodando por um ano, que é cerca de 3×10^{13} instruções executadas. Um Pentium de 1 GHz é uma máquina com cerca de 250 MIPS.

Um fato marcante na Tabela 9.4 refere-se ao método utilizado. Até meados da década de 1990, os ataques de fatoração eram feitos usando-se uma técnica conhecida como crivo quadrático. O ataque no RSA-130 usou um algoritmo mais novo, o crivo de corpo numérico generalizado (GNFS — *generalized number field sieve*), e foi capaz de fatorar um número maior do que o RSA-129 com apenas vinte por cento do esforço de computação.

A ameaça a tamanhos de chave maiores é dupla: o aumento contínuo no poder de computação e o refinamento contínuo dos algoritmos de fatoração. Temos visto que a passagem para um algoritmo diferente resultou em um drástico ganho de velocidade. Podemos esperar outras melhorias no GNFS, e o uso de um algoritmo ainda melhor também é uma possibilidade. De fato, um algoritmo relacionado, o crivo de corpo numérico especial (SNFS — *special*

Tabela 9.4 Progresso na fatoração

Número de dígitos decimais	Número aproximado de bits	Data em que foi alcançado	MIPS-anos	Algoritmo
100	332	abril de 1991	7	Crivo quadrático
110	365	abril de 1992	75	Crivo quadrático
120	398	junho de 1993	830	Crivo quadrático
129	428	abril de 1994	5000	Crivo quadrático
130	431	abril de 1996	1000	Crivo de corpo numérico generalizado
140	465	fevereiro de 1999	2000	Crivo de corpo numérico generalizado
155	512	agosto de 1999	8000	Crivo de corpo numérico generalizado
160	530	abril de 2003	—	Crivo de malha (Lattice sieve)
174	576	dezembro de 2003	—	Crivo de malha (Lattice sieve)
200	663	maio de 2005	—	Crivo de malha (Lattice sieve)

number field sieve), pode fatorar números de modo especializado, consideravelmente mais rápido do que o crivo de corpo numérico generalizado. A Figura 9.8 compara o desempenho dos dois algoritmos. É razoável esperar um avanço que permita um desempenho de fatoração geral aproximadamente com o mesmo tempo do SNFS, ou ainda melhor [ODLY95]. Assim, precisamos ter o cuidado na escolha de um tamanho de chave para o RSA. Em um futuro próximo, um tamanho de chave na faixa de 1.024 a 2.048 bits parece ser razoável.

Além de especificar o tamanho de n, várias outras restrições foram sugeridas pelos pesquisadores. Para evitar valores de n que possam ser fatorados mais facilmente, os inventores do algoritmo sugerem as seguintes restrições a p e q:

1. p e q devem diferir em tamanho por apenas alguns dígitos. Assim, para uma chave de 1.024 bits (309 dígitos decimais), tanto p quanto q deverão estar na ordem de grandeza de 10^{75} a 10^{100}.
2. Tanto $(p - 1)$ quanto $(q - 1)$ deveriam conter um fator primo grande.
3. O mdc$(p - 1, q - 1)$ deve ser pequeno.

Além disso, tem sido demonstrado que, se $e < n$ e $d < n^{1/4}$, então d pode ser determinado com facilidade [WIEN90].

Ataques de temporização (timing attacks) Se alguém precisa de outra lição sobre a dificuldade de avaliar a segurança de um algoritmo criptográfico, o surgimento dos ataques de temporização oferece uma atordoante. Paul Kocher, um consultor criptográfico, demonstrou que um intruso pode determinar uma chave privada acompanhando o tempo que um computador leva para decifrar as mensagens [KOCH96, KALI96b]. Os ataques de temporização não se aplicam apenas ao RSA, mas também a outros sistemas de criptografia de chave pública. Este ataque é alarmante por dois motivos: ele vem de uma direção completamente inesperada, e é um ataque apenas para texto cifrado.

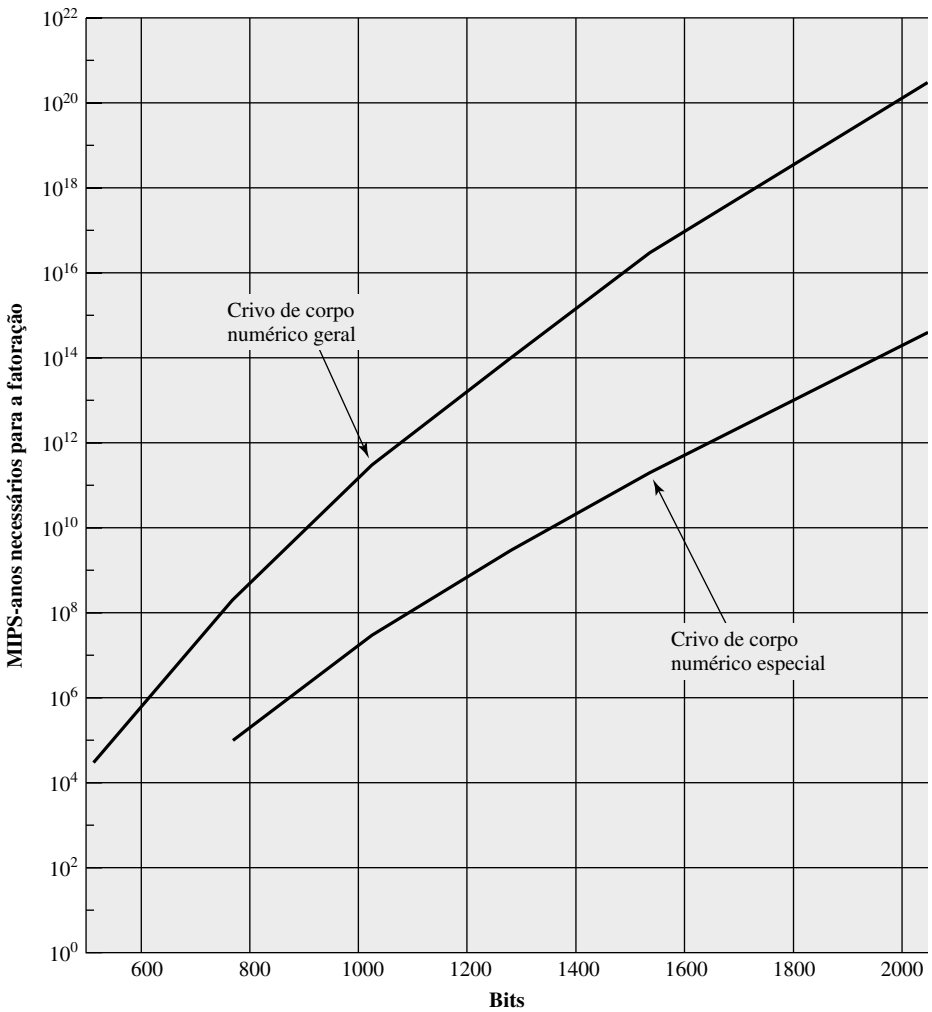


Figura 9.8 MIPS-anos necessários para a fatoração.

Um ataque de temporização é semelhante a um assaltante tentando adivinhar a combinação de um cofre pela observação de quanto tempo é gasto para alguém girar o mostrador de um número para outro. Podemos explicar o ataque usando o algoritmo de exponenciação modular da Figura 9.7, mas o ataque pode ser adaptado para funcionar com qualquer implementação que não seja executada em um tempo fixo. Nesse algoritmo, a exponenciação modular é realizada bit a bit, com uma multiplicação modular realizada a cada iteração e uma multiplicação modular adicional realizada para cada bit 1.

Conforme Kocher indica em seu artigo, o ataque é mais simples de entender em um caso extremo. Suponha que o sistema-alvo use uma função de multiplicação modular que é muito rápida em quase todos os casos, mas em alguns poucos casos leva muito mais tempo do que a exponenciação modular média completa. O ataque prossegue bit a bit, começando com o bit mais à esquerda, b_k. Suponha que os primeiros j bits sejam conhecidos (para obter o expoente inteiro, comece com $j = 0$ e repita o ataque até que o expoente inteiro seja conhecido). Para determinado texto cifrado, o atacante pode completar as primeiras j iterações do loop **for**. A operação da etapa seguinte depende do bit desconhecido do expoente. Se o bit estiver em 1, $d \leftarrow (d \times a) \bmod n$ será executado. Para alguns valores de a e d, a multiplicação modular será extremamente lenta, e o atacante saberá quais são eles. Portanto, se o tempo observado para executar o algoritmo de decriptografia sempre for lento quando essa iteração em particular for lenta com um bit 1, então esse bit será considerado como sendo 1. Se diversos tempos de execução observados para o algoritmo inteiro forem rápidos, então esse bit será considerado como sendo 0.

Na prática, as implementações da exponenciação modular não têm variações de tempo tão extremas, em que o tempo de execução de uma única iteração possa exceder o tempo de execução médio do algoritmo inteiro. Apesar disso, existe variação suficiente para tornar esse ataque prático. Para obter detalhes, consulte [KOCH96].

Embora o ataque de temporização seja uma ameaça séria, existem contramedidas simples que podem ser usadas, incluindo as seguintes:

- **Tempo de exponenciação constante:** garanta que todas as exponenciações levem o mesmo tempo antes de retornar um resultado. Essa é uma modificação simples, porém diminui o desempenho.
- **Atraso aleatório:** um desempenho melhor pode ser alcançado incluindo-se um atraso aleatório no algoritmo de exponenciação para confundir o ataque de temporização. Kocher aponta que, se os defensores não incluírem ruído suficiente, os atacantes ainda poderiam ter sucesso coletando medições adicionais para compensar os retardos aleatórios.
- **Ocultamento (*blinding*):** multiplique o texto cifrado por um número aleatório antes de realizar a exponenciação. Esse processo impede que o atacante saiba quais bits do texto cifrado estão sendo processados dentro do computador e, portanto, impede a análise bit a bit, essencial para o ataque de temporização.

A RSA Data Security incorpora um recurso de ocultamento em alguns de seus produtos. A operação da chave privada $M = C^d \bmod n$ é implementada da seguinte forma:

1. Gere um número aleatório secreto r entre 0 e $n - 1$.
2. Calcule $C' = C(r^e) \bmod n$, onde e é o expoente público.
3. Calcule $M' = (C')^d \bmod n$, com a implementação RSA normal.
4. Calcule $M = M'r^{-1} \bmod n$. Nessa equação, r^{-1} é o inverso multiplicativo de $r \bmod n$; veja no Capítulo 8 uma explicação desse conceito. Pode-se demonstrar que esse é o resultado correto observando-se que $r^{ed} \bmod n = r \bmod n$.

A RSA Data Security estima um comprometimento do desempenho de dois a dez por cento devido ao ocultamento.

Ataque de texto cifrado escolhido e preenchimento da criptografia assimétrica ideal O algoritmo RSA básico é vulnerável a um ataque de texto cifrado escolhido (CCA - *chosen ciphertext attack*). O CCA é definido como um ataque em que o adversário escolhe diversos textos cifrados e então recebe os textos claros correspondentes, decriptografados com a chave privada do destino. Assim, o adversário pode selecionar um texto claro, criptografá-lo com a chave pública do destino e depois ser capaz de obter o texto claro de volta, decriptografado com a chave privada. Claramente, isso não oferece ao adversário qualquer informação nova. Em vez disso, o adversário explora as propriedades do RSA e seleciona blocos de dados que, quando processados usando a chave privada do destino, geram informações necessárias para a criptoanálise.

Um exemplo simples de um CCA contra o RSA tira proveito da seguinte propriedade do RSA:

$$\mathrm{E}(PU, M_1) \times \mathrm{E}(PU, M_2) = \mathrm{E}(PU, [M_1 \times M_2]) \qquad \textbf{(9.2)}$$

Podemos decriptografar $C = M^e \bmod n$ usando um CCA da seguinte maneira:

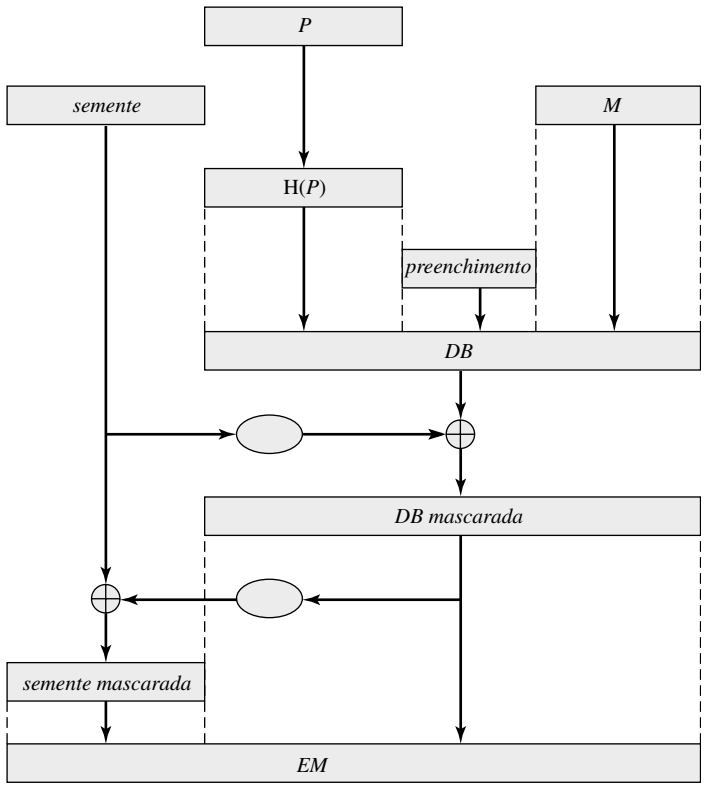
1. Calcule $X = (C \times 2^e) \bmod n$.
2. Envie X como um texto cifrado escolhido e receba de volta $Y = X^d \bmod n$.

Mas agora observe o seguinte:

$$\begin{aligned} X &= (C \bmod n) \times (2^e \bmod n) \\ &= (M^e \bmod n) \times (2^e \bmod n) \\ &= (2M)^e \bmod n \end{aligned}$$

Portanto, $Y = (2M) \bmod n$. A partir disso, podemos deduzir M. Para contornar esse ataque simples, criptossistemas práticos baseados no RSA preenchem aleatoriamente o texto claro antes da criptografia. Isso torna o texto cifrado aleatório, de modo que a Equação (9.2) não é mais verdadeira. Porém, CCAs mais sofisticados são possíveis e um preenchimento simples com um valor aleatório tem se mostrado insuficiente para fornecer a segurança desejada. Para evitar esses ataques, a RSA Security Inc., um fornecedor líder em RSA e antigo mantenedor da patente do RSA, recomenda modificar o texto claro usando um procedimento conhecido como preenchimento ideal de criptografia assimétrica (OAEP — *optimal asymmetric encryption padding*). Uma explicação completa das ameaças e do OAEP estão fora do nosso escopo; veja uma introdução em [POIN02] e uma análise completa em [BELL94a]. Aqui, simplesmente resumimos o procedimento OAEP.

A Figura 9.9 representa a criptografia OAEP. Como um primeiro passo, a mensagem M a ser criptografada é preenchida. Um conjunto de parâmetros opcionais P passa por uma função de hash H.[6] A saída é, então, preenchida com zeros, para se obter o tamanho desejado no bloco completo de dados (DB). Em seguida, uma semente



P = parâmetros de codificação
M = mensagem a ser codificada
H = função de hash
DB = bloco de dados
MGF = função de geração de máscara
EM = mensagem codificada

Figura 9.9 Criptografia usando preenchimento ideal de criptografia assimétrica (OAEP).

6. Uma função de hash relaciona um bloco de dados ou mensagem de tamanho variável a um valor de comprimento fixo, chamado código de hash. As funções de hash são explicadas com detalhes nos Capítulos 11 e 12.

aleatória é gerada e passa por outra função de hash, chamada função de geração de máscara (MGF — *mask generating function*). O valor de hash resultante passa por um OU-EXCLUSIVO bit a bit com o DB para produzir um DB mascarado. O DB mascarado, por sua vez, é passado pela MGF para formar um hash que passa por um OU-EXCLUSIVO com a semente, para produzir a semente mascarada. A concatenação da semente mascarada e do DB mascarado forma a mensagem codificada (EM — *encoded message*). Observe que a EM inclui a mensagem preenchida, mascarada pela semente, e a semente, mascarada pelo DB mascarado. A EM é então criptografada usando RSA.

9.3 LEITURA E SITES WEB RECOMENDADOS

Os tratamentos recomendados da criptografia relacionados no Capítulo 3 abrangem a criptografia de chave pública e também simétrica.

[DIFF88] descreve com detalhes as diversas tentativas de criar criptoalgoritmos seguros de duas chaves e a evolução gradual de uma série de protocolos baseados neles. [CORM01] oferece um resumo conciso, porém completo e legível de todos os algoritmos relevantes à verificação, cálculo e criptoanálise do RSA. [BONE99] explica diversos ataques de criptoanálise ao RSA. Uma explicação mais recente é [SHAM03].

BONE99 Boneh, D. "Twenty years of attacks on the RSA cryptosystem". *Notices of the American Mathematical Society*, fevereiro de 1999.
CORM01 Cormen, T.; Leiserson, C.; Rivest, R.; e Stein, C. *Introduction to algorithms*. Cambridge, MA: MIT Press, 2001.
DIFF88 Diffie, W. "The first ten years of public-key cryptography." *Proceedings of the IEEE*, maio de 1988. Reimpresso em [SIMM92].
SHAM03 Shamir, A., e Tromer, E. "On the cost of factoring RSA-1024." *CryptoBytes*, verão de 2003. **http://www.rsasecurity.com/rsalabs**

Site Web recomendado:

- **RSA Laboratories:** extenso conjunto de material técnico sobre RSA e outros tópicos relacionados a criptografia.

9.4 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

assinatura digital
ataque de temporização
ataque de texto cifrado escolhido (CCA)
chave privada
chave pública
complexidade do tempo
criptografia de chave pública
criptossistemas de chave pública
função unidirecional
função unidirecional com segredo
preenchimento ideal de criptografia assimétrica (OAEP)
RSA
troca de chave

Perguntas de revisão

9.1 Quais são os principais elementos de um criptossistema de chave pública?
9.2 Quais são os papéis da chave pública e da chave privada?
9.3 Quais são as três categorias gerais de aplicações dos criptossistemas de chave pública?
9.4 Que requisitos os criptossistemas de chave pública precisam cumprir para serem um algoritmo seguro?
9.5 O que é uma função unidirecional?
9.6 O que é uma função unidirecional com segredo?
9.7 Descreva, em termos gerais, um procedimento eficiente para escolher um número primo.

Problemas

9.1 Antes da descoberta de quaisquer esquemas de chave pública específicos, como RSA, uma prova de existência foi desenvolvida, cuja finalidade era demonstrar que a criptografia de chave pública era possível em teoria. Considere as funções $f_1(x_1) = z_1$; $f_2(x_2, y_2) = z_2$; $f_3(x_3, y_3) = z_3$, em que todos os valores são inteiros com $1 \leq x_i, y_i, z_i \leq N$. A função f_1 pode ser representada por um vetor M1 de tamanho N, em que a *k-ésima* entrada é o valor de $f_1(k)$. De modo semelhante, f_2 e f_3 podem ser representados pelas matrizes $N \times N$ M2 e M3. A intenção é representar o processo de criptografia/decriptografia por pesquisas em tabela para as tabelas com valores de N muito grandes. Essas tabelas seriam impraticavelmente extensas, mas, a princípio, poderiam ser construídas. O esquema funciona da seguinte forma: construa M1 com uma permutação aleatória de todos os inteiros entre 1 e N; isto é, cada inteiro aparece exatamente uma vez em M1. Construa M2 de modo que cada linha contenha uma permutação aleatória dos N primeiros inteiros. Finalmente, preencha M3 para satisfazer a seguinte condição:

$f_3(f_2(f_1(k), p), k) = p$ para todo k,

p com $1 \leq k, p \leq N$

Em palavras,

1. M1 toma uma entrada k e produz uma saída x.
2. M2 toma as entradas x e p, dando a saída z.
3. M3 toma as entradas z e k e produz p.

As três tabelas, uma vez construídas, se tornam públicas.

a. Deve ser claro que é possível construir M3 para satisfazer a condição anterior. Como exemplo, preencha M3 para o caso simples abaixo:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

Convenção: o i-ésimo elemento de M1 corresponde a $k = i$. A i-ésima linha de M2 corresponde a $x = i$; a j-ésima coluna de M2 corresponde a $p = j$. A i-ésima linha de M3 corresponde a $z = i$; a j-ésima coluna de M3 corresponde a $k = j$.

b. Descreva o uso desse conjunto de tabelas para realizar criptografia e decriptografia entre dois usuários.

c. Mostre que esse é um esquema seguro.

9.2 Realize a criptografia e decriptografia usando o algoritmo RSA, como na Figura 9.6, para o seguinte:

a. $p = 3$; $q = 11$, $e = 7$; $M = 5$
b. $p = 5$; $q = 11$, $e = 3$; $M = 9$
c. $p = 7$; $q = 11$, $e = 17$; $M = 8$
d. $p = 11$; $q = 13$, $e = 11$; $M = 7$
e. $p = 17$; $q = 31$, $e = 7$; $M = 2$. *Dica:* a decriptografia não é tão difícil quanto você pensa; use algum artifício.

9.3 Em um sistema de chave pública usando RSA, você intercepta o texto cifrado $C = 10$ enviado a um usuário cuja chave pública é $e = 5$, $n = 35$. Qual é o texto claro M?

9.4 Em um sistema RSA, a chave pública de determinado usuário é $e = 31$, $n = 3599$. Qual é a chave privada desse usuário? *Dica:* primeiro, use a tentativa e erro para determinar p e q; então, use o algoritmo estendido de Euclides para encontrar o inverso multiplicativo de 31 módulo $\phi(n)$.

9.5 No uso do algoritmo RSA, se um pequeno número de codificações repetidas retornam o texto claro, qual é a causa provável?

9.6 Suponha que tenhamos um conjunto de blocos codificados com o algoritmo RSA e não tenhamos a chave privada. Suponha que $n = pq$, e seja a chave pública. Suponha também que alguém nos diga que saiba que um dos blocos de texto claro tem um fator comum com n. Isso nos ajuda de alguma maneira?

9.7 No esquema de criptografia de chave pública RSA, cada usuário tem uma chave pública, e, e uma chave privada, d. Suponha que Bob revele acidentalmente sua chave privada. Em vez de gerar um novo módulo, ele decide gerar uma nova chave pública e uma nova chave privada. Isso é seguro?

9.8 Suponha que Bob use o criptossistema RSA com um módulo muito grande n para o qual a fatoração não possa ser encontrada em um período de tempo razoável. Suponha que Alice envie uma mensagem a Bob representando cada caractere alfabético como um inteiro entre 0 e 25 ($A \rightarrow 0, ..., Z \rightarrow 25$), e depois codifique cada número separadamente, usando RSA com e e n grandes. Esse método é seguro? Se não, descreva o ataque mais eficiente contra esse método de criptografia.

9.9 Usando uma planilha (como o Excel), ou uma calculadora, realize as operações descritas a seguir. Documente os resultados de todas as multiplicações modulares intermediárias. Determine um número de multiplicações modulares para cada transformação importante (como criptografia, decriptografia, teste de primalidade etc.).

a. Teste todos os números ímpares no intervalo de 233 a 241 quanto à primalidade usando o teste de Miller-Rabin com base 2.
b. Criptografe o bloco de mensagem $M = 2$ usando RSA com os seguintes parâmetros: $e = 23$ e $n = 233 \times 241$.
c. Calcule uma chave privada (d, p, q) correspondente à chave pública indicada acima (e, n).
d. Realize a decriptografia do texto cifrado obtido usando dois métodos diferentes:
   1. sem usar o teorema chinês do resto.
   2. usando o teorema chinês do resto.

9.10 Suponha que você gere uma mensagem autenticada e criptografada aplicando primeiro a transformação RSA determinada por sua chave privada e, depois, cifrando a mensagem por meio da chave pública do destinatário (observe que você NÃO usa a função de hash antes da primeira transformação). Esse esquema funcionará corretamente, ou seja, permitirá reconstruir a mensagem original no lado do destinatário, para todas as relações possíveis entre o módulo n_S do emissor e o módulo n_R do destinatário ($n_S > n_R$, $n_S < n_R$, $n_S = n_R$)? Explique sua resposta. Caso sua resposta seja 'não', como você corrigiria esse esquema?

9.11 "Quero lhe dizer, Holmes", Dr. Watson estava com voz de entusiasmo, "que suas atividades recentes em segurança de rede aumentaram meu interesse por criptografia. E foi somente ontem que descobri um modo de tornar prática a criptografia *one-time pad*."

"Verdade?" o rosto de Holmes perdeu sua aparência sonolenta.

"Sim, Holmes. A idéia é muito simples. Para determinada função unidirecional F, eu gero uma longa seqüência pseudo-aleatória de elementos aplicando F a alguma seqüência padrão de argumentos. Considera-se que o criptoanalista conheça F e a natureza geral da seqüência, que pode ser tão simples quanto S, S + 1, S + 2, ..., mas não o segredo S. E devido à natureza unidirecional de F, ninguém é capaz de extrair S dada F(S + i) para algum i, de modo que, mesmo obtendo de alguma forma um certo segmento da seqüência, ele não poderá determinar o restante."

"Receio, Watson, que sua proposta não seja infalível e precise, pelo menos, que algumas condições adicionais sejam satisfeitas por F. Vamos considerar, por exemplo, a função de criptografia RSA, que é $F(M) = M^K \bmod N$, K secreto. Acredita-se que essa função seja unidirecional, mas eu não recomendaria seu uso, por exemplo, na seqüência $M = 2, 3, 4, 5, 6, ...$"

"Mas por que, Holmes?" Dr. Watson aparentemente não entendeu. "Por que você acha que a seqüência resultante $2^K \bmod N$, $3^K \bmod N$, $4^K \bmod N$, ... não é apropriada para a criptografia one-time pad se K for mantido secreto?"

"Porque ela é, pelo menos parcialmente, previsível, meu caro Watson, mesmo que K seja mantido secreto. Você disse que o criptoanalista conhece F e a natureza geral da seqüência. Agora, vamos supor que ele obtenha de alguma forma um segmento curto da seqüência de saída. Nos círculos de criptologia, essa suposição geralmente é considerada viável. E para essa seqüência de saída, apenas o conhecimento dos dois primeiros elementos permitirá que ele preveja muitos dos próximos elementos da seqüência, ainda que não todos eles, e por isso essa seqüência não pode ser considerada criptograficamente forte. E com o conhecimento de um segmento maior, ele poderia prever ainda mais dos próximos elementos da seqüência. Veja, conhecendo a natureza geral da seqüência e seus dois primeiros elementos $2^K \bmod N$ e $3^K \bmod N$, você pode facilmente calcular seus elementos seguintes."

Mostre como isso pode ser feito.

9.12 Mostre como o RSA pode ser representado pelas matrizes M1, M2 e M3 do Problema 9.1.

9.13 Considere o seguinte esquema:

1. Escolha um número ímpar, E.
2. Escolha dois números primos, P e Q, onde $(P - 1)(Q - 1) - 1$ seja igualmente divisível por E.
3. Multiplique P e Q para obter N.
4. Calcule $D = \dfrac{(P - 1)(Q - 1)(E - 1) + 1}{E}$.

Esse esquema é equivalente ao RSA? Mostre por que sim ou por que não.

9.14 Considere o seguinte esquema pelo qual B criptografa uma mensagem para A.

1. A escolhe dois primos grandes P e Q que também são relativamente primos a $(P - 1)$ e $(Q - 1)$.
2. A envia $N = PQ$ como sua chave pública.
3. A calcula P' e Q' tal que $PP' \equiv 1 \pmod{Q - 1}$ e $QQ' \equiv 1 \pmod{P - 1}$.
4. B criptografa a mensagem M como $C = M^N \bmod N$.
5. A encontra M solucionando $M \equiv C^P \pmod{Q}$ e $M \equiv C^{Q'} \pmod{P}$.

a. Explique como funciona esse esquema.
b. Como ele difere do RSA?
c. Existe alguma vantagem em particular do RSA em comparação a esse esquema?

d. Mostre como esse esquema pode ser representado pelas matrizes M1, M2 e M3 do Problema 9.1.

9.15 "Esse é um caso muito interessante, Watson", disse Holmes. "O jovem ama uma garota e ela o ama também. Porém, o pai dela é um sujeito estranho, que insiste que seu futuro genro tem de projetar um protocolo simples e seguro para um criptossistema de chave pública apropriado, a ser usado na rede de computadores de sua empresa. O jovem apareceu com o seguinte protocolo para comunicação entre duas partes, por exemplo, o usuário A querendo enviar a mensagem M ao usuário B: as mensagens trocadas estão no formato (nome do emissor, texto, nome do receptor)."

1. A envia a B o seguinte bloco: $(A, \mathrm{E}(PU_b, [M, A]), B)$.
2. B confirma o recebimento enviando a A o seguinte bloco: $(B, \mathrm{E}(PU_a, [M, B]), A)$.

"Você pode ver que o protocolo é realmente simples. Mas o pai da garota afirma que o jovem não satisfez sua exigência de um protocolo simples, pois a proposta contém uma certa redundância e pode ser simplificada ainda mais da seguinte forma:"

1. A envia a B o bloco: $(A, \mathrm{E}(PU_b, M), B)$.
2. B confirma o recebimento enviando a A o bloco: $(B, \mathrm{E}(PU_a, M), A)$.

"Com base nisso, o pai da garota recusa-se a permitir que sua filha case com o jovem, tornando ambos infelizes. O jovem acabou de vir aqui para me pedir ajuda."

"Humm, não vejo como você possa ajudá-lo". Watson estava visivelmente infeliz com a idéia de que o simpático jovem tivesse de perder seu amor.

"Bem, acho que eu poderia ajudar. Você sabe, Watson, que a redundância às vezes é boa para garantir a segurança do protocolo. Assim, a simplificação que o pai da garota propôs pode tornar o novo protocolo vulnerável a um ataque ao qual o protocolo original seria capaz de resistir", refletiu Holmes. "Sim, é isso, Watson. Veja, tudo de que um adversário precisa é ser um dos usuários da rede e ser capaz de interceptar mensagens trocadas entre A e B. Sendo um usuário da rede, ele tem sua própria chave de criptografia pública e é capaz de enviar suas próprias mensagens para A ou para B, e receber a deles. Com a ajuda do protocolo simplificado, ele poderia então obter a mensagem M que o usuário A enviou anteriormente para B usando o seguinte procedimento:"

Complete a descrição.

9.16 Use o algoritmo de exponenciação rápida da Figura 9.7 para determinar 5^{596} mod 1234. Mostre as etapas envolvidas no cálculo.

9.17 Aqui está outra implementação do algoritmo de exponenciação rápida. Demonstre que ele é equivalente ao da Figura 9.7.

1. $\mathrm{f} \leftarrow 1$; $\mathrm{T} \leftarrow \mathrm{a}$; $\mathrm{E} \leftarrow \mathrm{b}$
2. **if** odd(e) **then** $\mathrm{f} \leftarrow \mathrm{f} \times \mathrm{T}$
3. $\mathrm{E} \leftarrow \lfloor \mathrm{E}/2 \rfloor$
4. $\mathrm{T} \leftarrow \mathrm{T} \times \mathrm{T}$
5. **if** $\mathrm{E} > 0$ **then goto** 2
6. **output** f

9.18 O problema ilustra uma aplicação simples do ataque de texto cifrado escolhido. Bob intercepta um texto cifrado C destinado a Alice e criptografado com a chave pública de Alice e. Bob deseja obter a mensagem original $M = C^d \bmod n$. Bob escolhe um valor aleatório r menor que n e calcula

$$Z = r^e \bmod n$$
$$X = ZC \bmod n$$
$$t = r^{-1} \bmod n$$

Em seguida, Bob pede que Alice autentique (assine) X com a chave privada dela (como na Figura 9.3), desse modo, decriptografando X. Alice retorna $Y = X^d \bmod n$. Mostre como Bob pode usar a informação agora disponível para determinar M.

9.19 Mostre a operação de decodificação OAEP, usada para a decriptografia, que corresponde à operação de codificação da Figura 9.9.

9.20 Aprimore o algoritmo P1 do Apêndice 9B.

a. Desenvolva um algoritmo que exija $2n$ multiplicações e $n + 1$ adições. *Dica:* $x^{i+1} = x^i \times x$.

b. Desenvolva um algoritmo que exija apenas $n + 1$ multiplicações e $n + 1$ adições. *Dica:* $\mathrm{P}(x) = a_0 + x \times q(x)$, onde $q(x)$ é um polinômio de grau $(n - 1)$.

Nota: os problemas restantes referem-se ao algoritmo de chave pública de mochila (*knapsack public-key algorithm*).

9.21 Que itens estão na mochila da Figura F.1?

9.22 Realize a criptografia e decriptografia usando o algoritmo de mochila para os seguintes:

a. $\mathbf{a}' = (1, 3, 5, 10)$; $w = 7$; $m = 20$; $\mathbf{x} = 1101$

b. $\mathbf{a}' = (1, 3, 5, 11, 23, 46, 136, 263))$; $w = 203$; $m = 491$; $\mathbf{x} = 11101000$

c. $\mathbf{a}' = (2, 3, 6, 12, 25)$; $w = 46$; $m = 53$; $\mathbf{x} = 11101$

d. $\mathbf{a}' = (15, 92, 108, 279, 563, 1172, 2243, 4468)$; $w = 2393$; $m = 9291$; $\mathbf{x} = 10110001$

9.23 Por que é um requisito que $m > \sum_{1=1}^{n} a_i'$?

APÊNDICE 9A PROVA DO ALGORITMO RSA

Os elementos básicos do algoritmo RSA podem ser resumidos da seguinte maneira. Dados dois números primos p e q, com $n = pq$ e um bloco de mensagem $M < n$, dois inteiros e e d são escolhidos de modo que

$$M^{ed} \bmod n = M$$

Afirmamos na Seção 9.2 que o relacionamento anterior é verdadeiro se e e d forem inversos multiplicativos módulo $\phi(n)$, onde $\phi(n)$ é a função tociente de Euler. Mostramos no Capítulo 8 que, para p, q primos, $\phi(pq) = (p - 1)(q - 1)$. O relacionamento entre e e d pode ser expresso como

$$ed \bmod \phi(n) = 1$$

Outra maneira de declarar isso é que existe um inteiro k tal que $ed = k\phi(n) + 1$. Assim, temos que mostrar que

$$M^{k\phi(n)+1} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = M \qquad \textbf{(9.3)}$$

Resultados básicos

Antes de provar a Equação (9.3), resumimos alguns resultados básicos. No Capítulo 4, mostramos que uma propriedade da aritmética modular é a seguinte:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

A partir disso, deverá ser fácil ver que, se tivermos $x \bmod n = 1$, então $x^2 \bmod n = 1$ e para qualquer inteiro y, teremos $x^y \bmod n = 1$. De modo semelhante, se tivermos $x \bmod n = 0$, para qualquer inteiro y, teremos $x^y \bmod n = 0$.

Outra propriedade da aritmética modular é

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

O outro resultado de que precisamos é o teorema de Euler, que foi desenvolvido no Capítulo 8. Se os inteiros a e n forem relativamente primos, então $a^{\phi(n)} \bmod n = 1$.

Prova

Primeiro, mostramos que $M^{k(p-1)(q-1)+1} \bmod p = M \bmod p$. Existem dois casos a considerar.

Caso 1: M e p não são relativamente primos; ou seja, p divide M. Nesse caso, $M \bmod p = 0$ e, portanto, $M^{k(p-1)(q-1)+1} \bmod p = 0$. Assim, $M^{k(p-1)(q-1)+1} \bmod p = M \bmod p$.

Caso 2: Se M e p são relativamente primos, pelo teorema de Euler, $M^{\phi(p)} \bmod p = 1$. Prosseguimos da seguinte forma:

$$\begin{aligned}
M^{k(p-1)(q-1)+1} \bmod p &= [(M)M^{k(p-1)(q-1)}] \bmod p \\
&= [(M)(M^{(p-1)})^{k(q-1)}] \bmod p \\
&= [(M)(M^{\phi(p)})^{k(q-1)}] \bmod p \\
&= (M \bmod p) \times [(M^{\phi(p)}) \bmod p]^{k(q-1)} \\
&= (M \bmod p) \times (1)^{k(q-1)} \qquad \text{(pelo teorema de Euler)} \\
&= M \bmod p
\end{aligned}$$

Agora, observamos que

$$[M^{k(p-1)(q-1)+1} - M] \bmod p = [M^{k(p-1)(q-1)+1} \bmod p] - [M \bmod p] = 0$$

Assim, p divide $[M^{k(p-1)(q-1)+1} - M]$. Pelo mesmo raciocínio, podemos mostrar que q divide $[M^{k(p-1)(q-1)+1} - M]$. Como p e q são primos distintos, é preciso haver um inteiro r que satisfaça

$$[M^{k(p-1)(q-1)+1} - M] = (pq)r = nr$$

Portanto, n divide $[M^{k(p-1)(q-1)+1} - M]$, e por isso $M^{k\phi(n)+1} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = M$.

APÊNDICE 9B A COMPLEXIDADE DOS ALGORITMOS

A questão central na avaliação da resistência de um algoritmo de criptografia à criptoanálise é a quantidade de tempo que determinado tipo de ataque levará. Normalmente, não se pode ter certeza de que alguém encontrou o algoritmo de ataque mais eficiente. O máximo que alguém pode dizer é que, para determinado algoritmo, o nível de esforço para um ataque é de uma determinada ordem de grandeza. Pode-se, então, comparar essa ordem de grandeza com a velocidade dos processadores atuais ou previstos, para determinar o nível de segurança de determinado algoritmo.

Uma medida comum da eficiência de um algoritmo é a sua complexidade de tempo. Definimos a **complexidade de tempo** de um algoritmo como sendo $f(n)$ se, para todo n e todas as entradas de tamanho n, a execução do algoritmo levar no máximo $f(n)$ passos. Assim, para determinado tamanho de entrada e determinada velocidade de processador, a complexidade de tempo é um limite máximo no tempo de execução.

Existem várias ambigüidades aqui. Primeiramente, a definição de um passo não é precisa. Um passo pode ser uma única operação de uma máquina de Turing, uma única instrução de máquina de um processador, uma única instrução de máquina em linguagem de alto nível etc. Porém, essas várias definições de passo devem estar todas relacionadas por constantes multiplicativas simples. Para valores muito grandes de n, essas constantes não são importantes. O que é importante é quão rapidamente o tempo de execução relativo está crescendo. Por exemplo, se estivermos preocupados em usar chaves de 50 dígitos ($n = 10^{50}$) ou de 100 dígitos ($n = 10^{100}$) para RSA, não é necessário (ou, na realidade, possível) saber exatamente quanto tempo levaria para quebrar cada tamanho de chave. Em vez disso, estamos interessados em valores aproximados para o nível de esforço e em saber quanto esforço relativo extra é exigido para o maior tamanho de chave.

Uma segunda questão é que, de modo geral, não podemos apontar uma fórmula exata para $f(n)$. Só podemos aproximá-la. Novamente, porém, estamos interessados principalmente na taxa de mudança de $f(n)$ quando n se torna muito grande.

Existe uma notação matemática padrão, conhecida como notação O ('O grande' ou '*big-O*'), para caracterizar a complexidade de tempo dos algoritmos, que é útil neste contexto. A definição é a seguinte: $f(n) = O(g)n))$ se e somente se houver dois números a e M tais que

$$|f(n)| \leq a \times |g(n)|, \qquad n \geq M \tag{9.4}$$

Um exemplo ajuda a esclarecer o uso dessa notação. Suponha que queiramos avaliar um polinômio geral na forma

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

O algoritmo pouco refinado a seguir vem de [POHL81]:

```
algorithm P1;
   n, i, j: integer; x, polyval: real;
   a, S: array [0..100] of real;
   begin
      read(x, n);
      for i := 0 upto n do
      begin
         S[i] := 1; read(a[i]);
         for j := 1 upto i do S[i] := x × S[i];
         S[i] := a[i] × S[i]
      end;
      polyval := 0;
      for i := 0 upto n do polyval := polyval + S[i];
      write ('value at', x, 'is', polyval)
   end.
```

Nesse algoritmo, cada subexpressão é avaliada separadamente. Cada $S[i]$ requer $(i + 1)$ multiplicações: i multiplicações para calcular $S[i]$ e uma para multiplicar por $a[i]$. Calcular todos os n termos exige

$$\sum_{i=0}^{n}(i + 1) = \frac{(n + 2)(n + 1)}{2}$$

multiplicações. Existem também $(n + 1)$ adições, que podemos ignorar em relação ao número muito maior de multiplicações. Assim, a complexidade de tempo desse algoritmo é $f(n) = (n + 2)(n + 1)/2$. Agora, mostramos que $f(n) = O(n^2)$. A partir da definição da Equação (9.4), queremos mostrar que, para $a = 1$ e $M = 4$, a relação se mantém para $g(n) = n^2$. Fazemos isso por indução em n. A relação é mantida para $n = 4$ porque $(4 + 2)(4 + 1)/2 = 15 < 4^2 = 16$. Agora, suponha que ela seja válida para todos os valores de n até k [ou seja, $(k + 2)(k + 1);2 < k^2$]. Então, com $n = k + 1$,

$$\begin{aligned}\frac{(n+2)(n+1)}{2} &= \frac{(k+3)(k+2)}{2}\\ &= \frac{(k+2)(k+1)}{2} + k + 2\\ &\leq k^2 + k + 2\\ &\leq k^2 + 2k + 1 = (k+1)^2 = n^2\end{aligned}$$

Portanto, o resultado é verdadeiro para $n = k + 1$.

Em geral, a notação O utiliza o termo que cresce mais rápido. Por exemplo,

1. $O[ax^7 + 3x^3 + \sin(x)] = O(ax^7) = O(x^7)$
2. $O(e^n + an^{10}) = O(e^n)$
3. $O(n! + n^{50}) = O(n!)$

Existe muito mais na notação O, com ramificações fascinantes. Para o leitor interessado, dois dos melhores relatos estão em [GRAH94] e [KNUT97].

Um algoritmo com uma entrada de tamanho n é considerado

- **Linear:** se o tempo de execução for $O(n)$
- **Polinomial:** se o tempo de execução for $O(n^t)$ para alguma constante t
- **Exponencial:** se o tempo de execução for $O(t^{h(n)})$ para alguma constante t e polinômio $h(n)$

Geralmente, um problema que pode ser solucionado no tempo polinomial é considerado viável, enquanto qualquer coisa pior do que o tempo polinomial, especialmente o tempo exponencial, é considerada inviável. Mas você precisa ter cuidado com esses termos. Primeiro, se o tamanho da entrada for suficientemente pequeno, até mesmo algoritmos muito complexos se tornam viáveis. Suponha, por exemplo, que você tenha um sistema que possa executar 10^{12} operações por unidade de tempo. A Tabela 9.5 mostra o tamanho da entrada que pode ser tratada em uma unidade de tempo para algoritmos de complexidades variadas. Para algoritmos de tempo exponencial ou fatorial, somente entradas muito pequenas podem ser acomodadas.

A segunda coisa com que se deve ter cuidado é o modo como a entrada é caracterizada. Por exemplo, a complexidade da criptoanálise de um algoritmo de criptografia pode ser caracterizada igualmente bem em termos do número de chaves possíveis ou do tamanho da chave. Para o Advanced Encryption Standard (AES), por exemplo, o número de chaves possíveis é 2^{128}, e o tamanho da chave é de 128 bits. Se considerarmos que uma única criptografia é um 'passo' e o número de chaves possíveis é $N = 2^n$, então a complexidade de tempo do algoritmo é linear em termos do número de chaves $[O(N)]$, mas exponencial em termos do tamanho da chave $[O(2^n)]$.

Tabela 9.5 Nível de esforço para vários níveis de complexidade

Complexidade	Tamanho	Operações
$\log_2 n$	$2^{10^{12}} = 10^{3\times10^{11}}$	10^{12}
N	10^{12}	10^{12}
n^2	10^6	10^{12}
n^6	10^2	10^{12}
2^n	39	10^{12}
$n!$	15	10^{12}

CAPÍTULO 10

Gerenciamento de chaves; outros criptossistemas de chave pública

10.1 **Gerenciamento de chaves**

10.2 **Acordo de chaves Diffie-Hellman**

10.3 **Aritmética de curva elíptica**

10.4 **Criptografia com curvas elípticas**

10.5 **Leitura e sites Web recomendados**

10.6 **Principais termos, perguntas de revisão e problemas**

Nenhum singalês, seja homem ou mulher, se aventuraria a sair de casa sem um punhado de chaves em sua mão, pois sem esse talismã, ele teme que algum demônio possa tirar proveito do seu estado fraco para entrar em seu corpo.

The golden bough, Sir James George Frazer

PONTOS PRINCIPAIS

- Os esquemas de criptografia de chave pública são seguros apenas se a autenticação da chave pública for garantida. Um esquema de certificação de chave pública oferece a segurança necessária.
- Um algoritmo simples de chave pública é o acordo de chaves Diffie-Hellman. Esse protocolo permite que dois usuários definam uma chave secreta usando um esquema de chave pública baseado em logaritmos discretos. O protocolo é seguro apenas se a autenticação dos dois participantes puder ser estabelecida.
- A aritmética de curva elíptica pode ser usada para desenvolver uma série de esquemas de criptografia de curva elíptica (ECC), incluindo acordo de chaves, criptografia e assinatura digital.
- Para fins de ECC, a aritmética de curva elíptica envolve o uso de uma equação de curva elíptica definida sobre um corpo finito. Os coeficientes e variáveis na equação são elementos de um corpo finito. Esquemas usando $\mathbf{Z}_p$ e GF(2^m) têm sido desenvolvidos.

Este capítulo continua nossa visão geral da criptografia de chave pública. Nele, examinaremos a distribuição e o gerenciamento de chaves para sistemas de chave pública, incluindo uma explicação sobre o acordo de chaves Diffie-Hellman. Finalmente, oferecemos uma introdução à criptografia com curvas elípticas.

10.1 GERENCIAMENTO DE CHAVES

No Capítulo 7, examinamos o problema da distribuição das chaves secretas. Um dos principais papéis da criptografia de chave pública tem sido resolver o problema da distribuição de chaves. Na realidade, existem dois aspectos distintos para o uso da criptografia de chave pública com relação a isso:

- A distribuição das chaves públicas
- O uso da criptografia de chave pública para distribuir as chaves secretas

Vamos examinar cada uma dessas áreas por sua vez.

Distribuição de chaves públicas

Várias técnicas têm sido propostas para a distribuição de chaves públicas. Praticamente todas essas propostas podem ser agrupadas nos seguintes esquemas gerais:

- Anúncio público
- Diretório disponível publicamente
- Autoridade de chave pública
- Certificados de chave pública

Anúncio público de chaves públicas Assim, o argumento da criptografia de chave pública é que a chave pública é pública. Portanto, se houver algum algoritmo de chave pública amplamente aceito, como o RSA, qualquer participante pode enviar sua chave pública a qualquer outro participante ou transmitir a chave por broadcast à comunidade em geral (Figura 10.1). Por exemplo, devido à crescente popularidade do PGP (Pretty Good Privacy, explicado no Capítulo 15), que utiliza o RSA, muitos usuários do PGP adotaram a prática de anexar sua chave pública às mensagens que enviam a fóruns públicos, como newgroups da USENET e listas de correspondência da Internet.

Embora essa técnica seja conveniente, ela possui um ponto fraco importante. Qualquer um pode falsificar esse anúncio público. Ou seja, algum usuário pode fingir ser o usuário A e enviar uma chave pública para outro participante ou transmitir essa chave pública por broadcast. Até que o usuário A descubra a falsificação e alerte outros participantes, o falsificador terá sido capaz de ler todas as mensagens cifradas enviadas para A, e pode usar as chaves falsificadas para autenticação (ver Figura 9.3).

Figura 10.1 Distribuição não controlada de chaves públicas.

Diretório disponível publicamente Um maior grau de segurança pode ser alcançado mantendo-se um diretório dinâmico disponível publicamente com chaves públicas. A manutenção e a distribuição do diretório público teria de ser de responsabilidade de alguma entidade ou organização confiável (Figura 10.2). Esse esquema incluiria os seguintes elementos:

1. A autoridade mantém um diretório com uma entrada {nome, chave pública} para cada participante.
2. Cada participante registra uma chave pública com a autoridade de diretório. O registro teria de ser feito pessoalmente ou por algum modo de comunicação autenticada segura.
3. Um participante pode substituir a chave existente por uma nova a qualquer momento, seja pelo desejo de substituir uma chave pública que já foi usada para uma grande quantidade de dados, seja porque a chave privada correspondente foi comprometida de alguma maneira.
4. Os participantes também podem acessar o diretório eletronicamente. Para essa finalidade, é obrigatório que haja uma comunicação segura e autenticada da autoridade para o participante.

Esse esquema é claramente mais seguro do que os anúncios públicos individuais, mas ainda tem vulnerabilidades. Se um adversário conseguir obter ou calcular a chave privada da autoridade do diretório, este poderá passar chaves públicas com autorização forjada e, mais tarde, fazer-se passar por qualquer participante e captar mensagens enviadas a qualquer outro participante. Outra maneira de atingir o mesmo fim é se o adversário mexer nos registros mantidos pela autoridade.

Autoridade de chave pública A segurança mais forte para distribuição de chaves públicas pode ser obtida oferecendo-se um controle mais rigoroso sobre a distribuição de chaves públicas pelo diretório. Um cenário típico é ilustrado na Figura 10.3, que é baseada em uma figura de [POPE79]. Como antes, o cenário considera que uma autoridade central mantém um diretório dinâmico de chaves públicas de todos os participantes. Além disso, cada participante conhece com segurança uma chave pública para a autoridade, apenas com a autoridade conhecendo a chave privada correspondente. As seguintes etapas (combinadas por número com a Figura 10.3) ocorrem:

1. A envia uma mensagem com marca de data/hora à autoridade de chave pública, contendo uma solicitação para a chave pública atual de B.
2. A autoridade responde com uma mensagem que é cifrada usando-se a chave privada da autoridade, PR_{auth}. Assim, A é capaz de decifrar a mensagem usando a chave pública da autoridade. Portanto, A tem garantias de que a mensagem partiu da autoridade. A mensagem inclui o seguinte:
   - A chave pública de B, PU_b, que A pode usar para cifrar mensagens destinadas a B
   - A solicitação original, para permitir que A corresponda essa resposta com a solicitação anterior correspondente e verifique se a solicitação original não foi alterada antes do recebimento pela autoridade
   - A marca de data/hora original, para que A possa determinar que essa não é uma mensagem antiga da autoridade, contendo uma chave diferente da chave pública atual de B
3. A armazena a chave pública de B e também a utiliza para cifrar uma mensagem para B, contendo um identificador de A (ID_A) e um nonce (N_1), que é usado para identificar essa transmissão exclusivamente.

4, 5. B obtém a chave pública de A com a autoridade da mesma maneira que A obteve a chave pública de B.

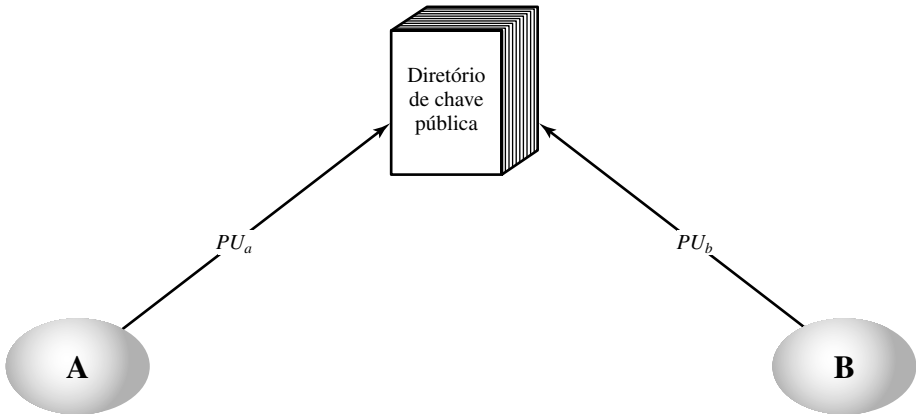


Figura 10.2 Publicação de chave pública.

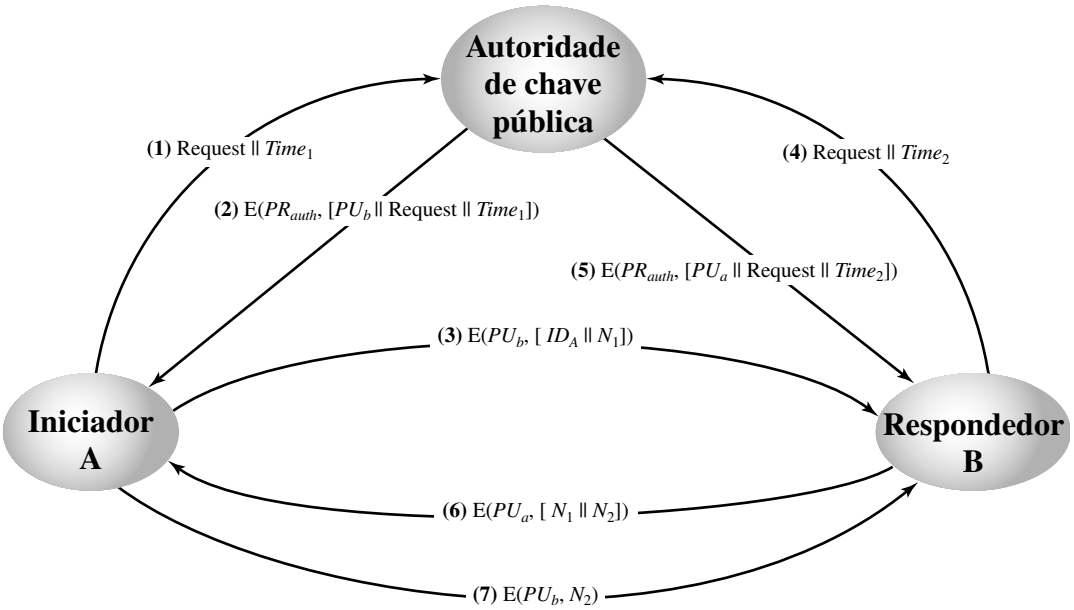


Figura 10.3 Cenário de distribuição de chave pública.

Nesse ponto, as chaves públicas foram entregues com segurança a A e B, e eles podem iniciar seu acordo protegido. Porém, duas etapas adicionais são desejáveis:

6. B envia uma mensagem a A cifrada com PU_a e contendo o nonce (N_1) de A, além de um novo nonce gerado por B (N_2). Como somente B poderia ter decriptogafado a mensagem (3), a presença de N_1 na mensagem (6) garante a A que o interlocutor é B.
7. A retorna N_2 cifrado, usando a chave pública de B, para garantir a B que seu interlocutor é A.

Assim, é necessário um total de sete mensagens. Porém, as quatro mensagens iniciais só precisam ser usadas com muito pouca freqüência, pois tanto A quanto B podem salvar a chave pública um do outro para uso futuro, uma técnica conhecida como *caching*. Periodicamente, um usuário deverá solicitar cópias recentes das chaves públicas de seus correspondentes para garantir a atualidade destas.

Certificados de chave pública

O cenário da Figura 10.3 é atraente, embora tenha algumas desvantagens. A autoridade de chave pública poderia ser um gargalo no sistema, pois um usuário precisa apelar para a autoridade para obter uma chave pública para cada outro usuário com quem queira se comunicar. Como antes, o diretório de nomes e chaves públicas, mantido pela autoridade, é vulnerável a violação.

Uma técnica alternativa, sugerida inicialmente por Kohnfelder [KOHN78], é usar certificados que possam ser usados pelos participantes para negociar chaves sem contatar uma autoridade de chave pública, de um modo tão confiável quanto se as chaves fossem obtidas diretamente de uma autoridade de chave pública. Basicamente, um certificado consiste em uma chave pública mais um identificador do proprietário da chave, com o bloco inteiro assinado por um terceiro confiável. Normalmente, o terceiro é uma autoridade certificadora, como uma agência do governo ou uma instituição financeira, na qual a comunidade de usuários confia. Um usuário pode apresentar sua chave pública à autoridade de uma maneira segura, e obter um certificado. O usuário pode, então, publicar o certificado. Qualquer um que precise da chave pública desse usuário pode obter o certificado e verificar se ele é válido por meio de uma assinatura confiável anexada. Um participante também pode levar sua própria informação para outro transmitindo seu certificado. Outros participantes podem verificar se o certificado foi criado pela autoridade. Podemos colocar os seguintes requisitos nesse esquema:

1. Qualquer participante pode ler um certificado para determinar o nome e a chave pública do proprietário do certificado.
2. Qualquer participante pode verificar se o certificado foi originado da autoridade certificadora, e não é uma falsificação.
3. Somente a autoridade certificadora pode criar e atualizar certificados.

Esses requisitos são satisfeitos pela proposta original em [KOHN78]. Denning [DENN83] acrescentou o seguinte requisito adicional:

4. Qualquer participante pode verificar se o certificado está atualizado.

Um esquema de certificado é ilustrado na Figura 10.4. Cada participante se candidata à autoridade certificadora, fornecendo uma chave pública e solicitando um certificado. A aplicação precisa ser feita pessoalmente ou por alguma forma de comunicação autenticada segura. Para o participante A, a autoridade oferece um certificado na forma

$$C_A = \mathrm{E}(PR_{auth}, [T \| ID_A \| PU_a])$$

onde PR_{auth} é a chave privada usada pela autoridade e T é uma marca de data/hora de tempo. A pode, então, passar esse certificado adiante para qualquer outro participante, que lê e verifica o certificado da seguinte forma:

$$\mathrm{D}(PU_{auth}, C_A) = \mathrm{D}(PU_{auth}, \mathrm{E}(PR_{auth}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$$

O destinatário usa a chave pública da autoridade, PU_{auth}, para decriptografar o certificado. Como o certificado só pode ser lido por meio da chave pública da autoridade, isso confirma que o certificado é proveniente da autoridade certificadora. Os elementos ID_A e PU_a oferecem ao destinatário o nome e a chave pública do mantenedor do certificado. A marca de data/hora T verifica se o certificado está atualizado. A marca de data/hora T opõe-se ao seguinte cenário. A chave privada de A é descoberta por um adversário. A gera um novo par de chaves privada/pública e solicita da autoridade certificadora um novo certificado. Nesse meio tempo, o adversário replica o certificado antigo para B. Se B, então, tentar criptografar mensagens usando a antiga chave pública comprometida, o adversário poderá ler essas mensagens.

Nesse contexto, o comprometimento de uma chave privada é comparável à perda de um cartão de crédito. O proprietário cancela o número do cartão de crédito, mas está em risco até que todos os comunicantes estejam cientes de que o cartão antigo está obsoleto. Assim, a marca de data/hora funciona como uma data de expiração. Se um certificado for suficientemente antigo, ele será considerado expirado.

Um esquema foi aceito universalmente para formatar certificados de chave pública: o padrão X.509. Certificados X.509 são usados na maioria das aplicações de segurança de rede, incluindo segurança de IP, Secure Sockets Layer (SSL), Secure Electronic Transactions (SET) e S/MIME, todos explicados na Parte 2. X.509 é examinado com detalhes no Capítulo 14.

Distribuição de chaves secretas com criptografia de chave pública

Quando as chaves públicas são distribuídas ou se tornam acessíveis, é possível haver comunicação segura, o que impede a interceptação (Figura 9.2), a violação (Figura 9.3) ou ambos (Figura 9.4). Porém, poucos usuários desejarão fazer uso exclusivo da criptografia de chave pública para a comunicação, devido à velocidade de dados relativamente baixa que pode ser alcançada. Por conseguinte, a criptografia de chave pública providencia a distribuição de chaves secretas a serem usadas para a criptografia convencional.

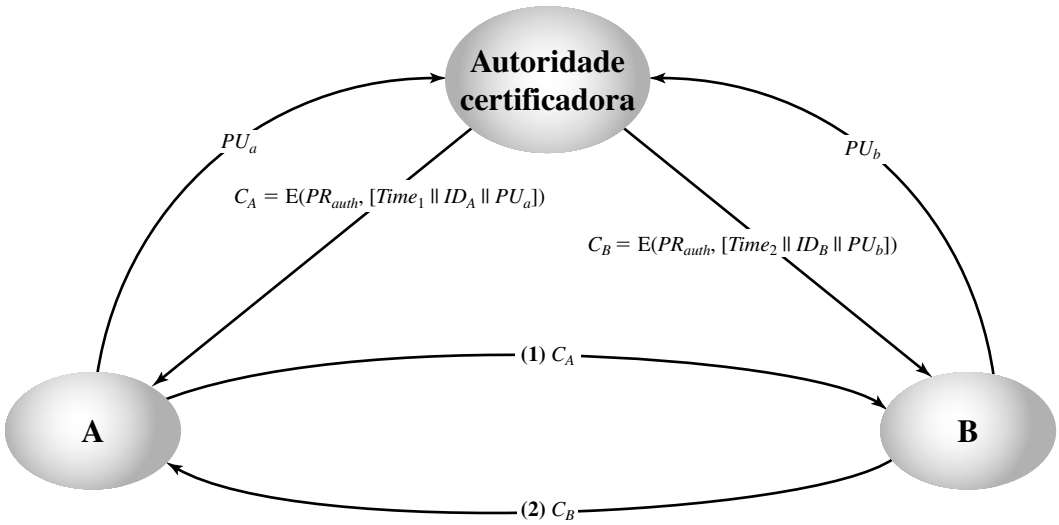


Figura 10.4 Troca de certificados de chave pública.

Distribuição simples de chave secreta Um esquema extremamente simples foi apresentado por Merkle [MERK79], conforme ilustramos na Figura 10.5. Se A deseja se comunicar com B, o seguinte procedimento é empregado:

1. A gera um par de chaves pública/privada $\{PU_a, PR_a\}$ e transmite uma mensagem para B consistindo em PU_a e um identificador de A, ID_A.
2. B gera uma chave secreta, K_s, e a transmite para A, cifrada com a chave pública de A.
3. A calcula $D(PR_a, E(PU_a, K_s))$ para recuperar a chave secreta. Como somente A pode decriptografar a mensagem, somente A e B saberão a identidade de K_s.
4. A descarta PU_a e PR_a e B descarta PU_a.

A e B agora podem se comunicar seguramente usando criptografia convencional e a chave de sessão. Ao término da comunicação, tanto A quanto B descartam K_s. Apesar de sua simplicidade, esse é um protocolo atraente. Não existem chaves antes do início da comunicação nem depois do término da comunicação. Assim, o risco de comprometer as chaves é mínimo. Ao mesmo tempo, a comunicação é segura contra interceptação.

O protocolo representado na Figura 10.5 é inseguro contra um adversário que pode interceptar mensagens e depois replicar a mensagem interceptada ou substituir outra mensagem (ver Figura 1.4c). Esse ataque é conhecido como ataque *man-in-the-middle* [RIVE84]. Nesse caso, se um adversário E tem controle sobre o canal de comunicação utilizado, então E pode comprometer a comunicação da seguinte forma, sem ser detectado:

1. A gera um par de chaves pública/privada $\{PU_a, PR_a\}$ e transmite uma mensagem destinada a B consistindo em PU_a e um identificador de A, ID_A.
2. E intercepta a mensagem, cria seu próprio par de chaves pública/privada $\{PU_e, PR_e\}$ e transmite $PU_e \| ID_A$ para B.
3. B gera uma chave secreta, K_s, e transmite $E(PU_e, K_s)$.
4. E intercepta a mensagem e descobre K_s calculando $D(PU_e, E(PU_e, K_s))$.
5. E transmite $E(PU_a, K_s)$ para A.

O resultado é que tanto A quanto B conhecem K_s e não sabem que K_s também foi revelado a E. A e B agora podem trocar mensagens usando K_s. E não interfere mais ativamente com o canal de comunicações, mas simplesmente intercepta. Conhecendo K_s, E pode decriptografar todas as mensagens, sem A nem B estarem cientes do problema. Assim, esse protocolo simples é útil apenas em um ambiente onde a única ameaça é a interceptação.

Distribuição de chave secreta com confidencialidade e autenticação A Figura 10.6, baseada em uma técnica sugerida em [NEED78], oferece proteção contra ataques ativos e passivos. Começamos em um ponto em que se supõe que A e B negociaram chaves públicas por um dos esquemas descritos antes nesta seção. Depois ocorrem as seguintes etapas:

1. A usa a chave pública de B para criptograr uma mensagem para B contendo um identificador de A (ID_A) e um nonce (N_1), que é usado para identificar essa transação de forma exclusiva.
2. B envia uma mensagem para A cifrada com PU_a e contendo o nonce (N_1) de A, além de um novo nonce, gerado por B (N_2). Como somente B poderia ter decodificado a mensagem (1), a presença de N_1 na mensagem (2) garante a A que o correspondente é B.
3. A retorna N_2, cifrado usando a chave pública de B, para garantir a B que seu correspondente é A.
4. A seleciona uma chave secreta K_s e envia $M = E(PU_b, E(PR_a, K_s))$ para B. A criptografia dessa mensagem com a chave pública de B garante que somente B poderá lê-la; a criptografia com a chave privada de A garante que somente A poderia tê-la enviado.
5. B calcula $D(PU_a, D(PR_b, M))$ para recuperar a chave secreta.

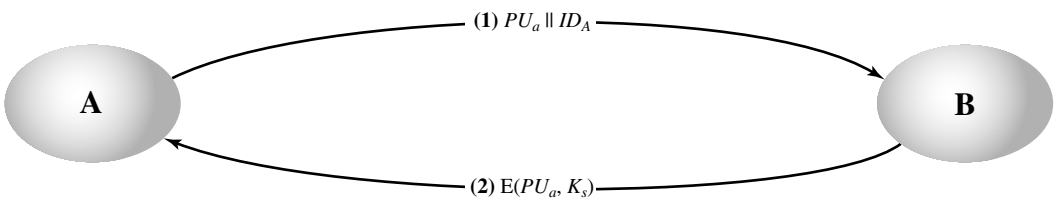


Figura 10.5 Uso simples da criptografia de chave pública para estabelecer uma chave de sessão.

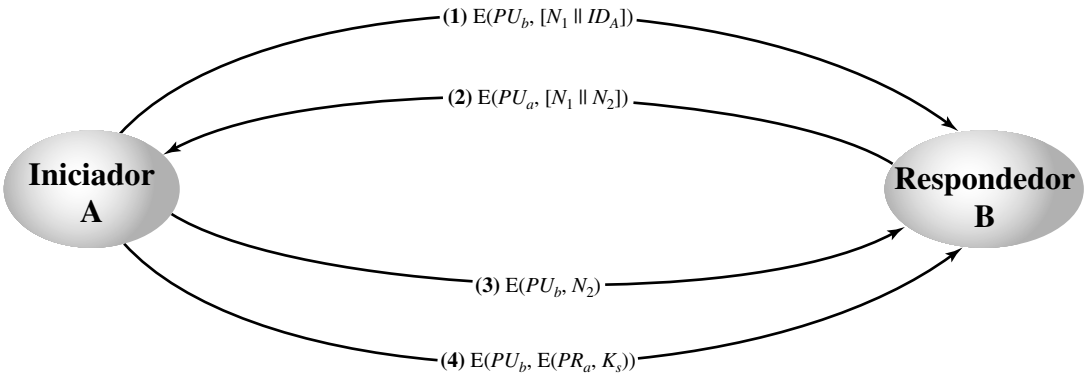


Figura 10.6 Distribuição de chaves secretas por chave pública.

Observe que as três primeiras etapas desse esquema são iguais à últimas três etapas da Figura 10.3. O resultado é que esse esquema garante tanto confidencialidade quanto autenticação no acordo de uma chave secreta.

Um esquema híbrido Outra forma de usar a criptografia de chave pública para distribuir chaves secretas é uma técnica híbrida, em uso nos mainframes IBM [LE93]. Esse esquema retém o uso de um centro de distribuição de chaves (CDC) que compartilha uma chave mestra secreta com cada usuário e distribui chaves de sessão secretas, cifradas com a chave mestra. O esquema de chave-pública é usado para distribuir as chaves-mestras. O seguinte raciocínio é fornecido para usar essa técnica de três níveis:

- **Desempenho:** Existem muitas aplicações, especialmente aplicações orientadas a transação, em que as chaves de sessão mudam com freqüência. A distribuição das chaves de sessão por criptografia de chave pública pode diminuir o desempenho geral do sistema, em razão de carga computacional relativamente alta da criptografia e decriptografia com chaves assimétricas. Com uma hierarquia de três níveis, a criptografia de chave pública é usada apenas ocasionalmente para atualizar a chave-mestra entre um usuário e o CDC.
- **Compatibilidade:** O esquema híbrido é facilmente coberto em um esquema de CDC existente, com o mínimo de interrupção ou mudanças de software.

O acréscimo de uma camada de chave pública oferece um meio seguro e eficiente de distribuir chaves-mestras. Essa é uma vantagem em uma configuração em que um único CDC atende a um conjunto de usuários bastante distribuído.

10.2 ACORDO DE CHAVES DIFFIE-HELLMAN

O primeiro algoritmo de chave pública publicado apareceu no artigo inicial de Diffie e Hellman que definiu a criptografia de chave pública [DIFF76b] e geralmente é chamado de acordo de chaves Diffie-Hellman.[1] Diversos produtos comerciais empregam essa técnica de acordo de chaves.

A finalidade do algoritmo é permitir que dois usuários negociem uma chave com segurança, a qual pode então ser usada para a subseqüente criptografia das mensagens. O próprio algoritmo é limitado ao acordo de valores secretos.

O algoritmo Diffie-Hellman depende, para sua segurança, da dificuldade de se calcular logaritmos discretos. Resumindo, podemos definir o logaritmo discreto da seguinte maneira. Primeiro, definimos uma raiz primitiva de um número primo p como aquela cujas potências módulo p geram todos os inteiros de 1 até $p - 1$. Ou seja, se a é uma raiz primitiva do número primo p, então os números

$$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$$

são distintos e consistem nos inteiros de 1 até $p - 1$ em alguma permutação.

Para qualquer inteiro b e uma raiz primitiva a do número primo p, podemos encontrar um expoente exclusivo i tal que

$$b \equiv a^i \pmod p \qquad \text{onde } 0 \leq i \leq (p - 1)$$

1. Williamson, da CESG britânica, publicou um esquema idêntico alguns meses antes em um documento confidencial [WILL76] e diz tê-lo descoberto muitos anos antes disso; veja uma explicação em [ELLI99].

O expoente i é referenciado como o logaritmo discreto de **b** na base a, mod p. Expressamos esse valor como $\text{dlog}_{a,p}(b)$. Veja, no Capítulo 8, uma explicação detalhada sobre logaritmos discretos.

O algoritmo

A Figura 10.7 resume o algoritmo de acordo de chaves Diffie-Hellman. Para esse esquema, existem dois números conhecidos publicamente: um número primo q e um inteiro α que é uma raiz primitiva de q. Suponha que os usuários A e B queiram negociar uma chave. O usuário A seleciona um inteiro aleatório $X_A < q$ e calcula $Y_A = \alpha^{X_A} \bmod q$. Da mesma maneira, o usuário B seleciona independentemente um inteiro aleatório $X_B < q$ e calcula $Y_B = \alpha^{X_B} \bmod q$. Cada lado mantém o valor X privado e torna o valor Y disponível publicamente ao outro lado. O usuário A calcula a chave como $K = (Y_B)^{X_A} \bmod q$ e o usuário B calcula a chave como $K = (Y_A)^{X_B} \bmod q$. Esses dois cálculos produzem resultados idênticos:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{pelas regras da aritmética modular} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

O resultado é que os dois lados negociaram um valor secreto. Além disso, como X_A e X_B são privados, um adversário só tem os seguintes ingredientes para trabalhar: q, α, Y_A e Y_B. Assim, o adversário é forçado a calcular um logaritmo discreto para determinar a chave. Por exemplo, para determinar a chave privada do usuário B, um adversário precisa calcular

$$X_B = \text{dlog}_{\alpha,q}(Y_B)$$

Elementos públicos globais

q	número primo
α	$\alpha < q$ e α uma raiz primitiva de q

Geração de chave do usuário A

Selecionar X_A privada	$X_A < q$
Calcular Y_A pública	$Y_A = \alpha^{X_A} \bmod q$

Geração de chave do usuário B

Selecionar X_B privada	$X_B < q$
Calcular Y_B pública	$Y_B = \alpha^{X_B} \bmod q$

Cálculo da chave secreta pelo usuário A

$K = (Y_B)^{X_A} \bmod q$

Cálculo da chave secreta pelo usuário B

$K = (Y_A)^{X_B} \bmod q$

Figura 10.7 O algoritmo de acordo de chaves Diffie-Hellman.

O adversário pode, então, calcular a chave K da mesma maneira que o usuário B a calcula.

A segurança do acordo de chaves Diffie-Hellman está no fato de que, embora seja relativamente fácil calcular exponenciais módulo um primo, é muito difícil calcular logaritmos discretos. Para números primos grandes, essa última tarefa é considerada intratável.

Aqui está um exemplo. O acordo de chaves baseia-se no uso do número primo $q = 353$ e uma raiz primitiva de 353, neste caso, $\alpha = 3$. A e B selecionam chaves secretas $X_A = 97$ e $X_B = 233$, respectivamente. Cada um calcula sua chave pública:

$$\text{A calcula } Y_A = 3^{97} \bmod 353 = 40.$$
$$\text{B calcula } Y_B = 3^{233} \bmod 353 = 248.$$

Depois que eles negociarem chaves públicas, cada um poderá calcular a chave secreta comum:

$$\text{A calcula } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$
$$\text{B calcula } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

Consideramos que um agressor teria à sua disposição as seguintes informações:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

Nesse exemplo simples, seria possível, pela força bruta, determinar a chave secreta 160. Em particular, um agressor E pode determinar a chave comum descobrindo uma solução para a equação $3^a \bmod 353 = 40$ ou para a equação $3^b \bmod 353 = 248$. A técnica de força bruta é calcular potências de 3 módulo 353, parando quando o resultado for igual a 40 ou 248. A resposta desejada é alcançada com o valor de expoente 97, que oferece $3^{97} \bmod 353 = 40$.

Com números maiores, o problema se torna impraticável.

Protocolos de acordo de chaves

A Figura 10.8 mostra um protocolo simples que utiliza o cálculo de Diffie-Hellman. Suponha que o usuário A queira estabelecer uma conexão com o usuário B e use uma chave secreta para cifrar mensagens nessa conexão. O usuário A pode gerar uma chave secreta de única vez X_A, calcular Y_A e enviar isso para o usuário B. O usuário B poder responder gerando um valor privado X_B, calculando Y_B, e enviando Y_B ao usuário A. Os dois usuários agora podem calcular a chave. Os valores públicos necessários q e α teriam de ser conhecidos antes da hora. Como alternativa, o usuário A poderia escolher valores para q e α e incluí-los na primeira mensagem.

Como exemplo de outro uso do algoritmo Diffie-Hellman, suponha que um grupo de usuários (por exemplo, todos os usuários em uma LAN) gere um valor privado de longa duração (para o usuário i) e calcule um valor público Y_i. Esses valores públicos, junto com os valores públicos globais para q e α, são armazenados em algum diretório central. A qualquer momento, o usuário j pode acessar o valor público do usuário i, calcular uma chave secreta e usá-la para enviar uma mensagem cifrada ao usuário A. Se o diretório central for confiável, então essa forma de comunicação oferece confidencialidade e um certo grau de autenticação. Como somente i e j podem determinar a chave, nenhum outro usuário pode ler a mensagem (confidencialidade). O destinatário i sabe que somente o usuário j poderia ter criado uma mensagem usando essa chave (autenticação). Porém, a técnica não protege contra ataques de repetição.

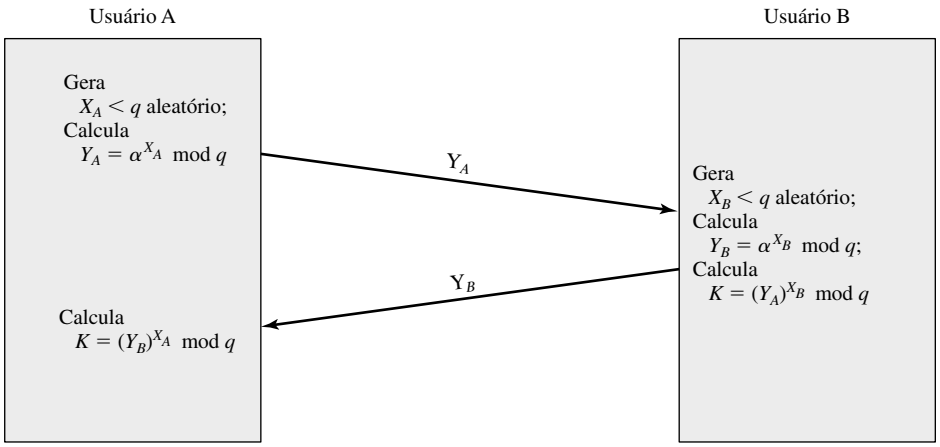


Figura 10.8 Acordo de chaves Diffie-Hellman.

Ataque *man-in-the-middle*

O protocolo representado na Figura 10.8 não é seguro contra um ataque *man-in-the-middle* (homem no meio). Suponha que Alice e Bob queiram negociar chaves, e Darth seja o adversário. O ataque acontece da seguinte forma:

1. Darth se prepara para o ataque gerando duas chaves privadas aleatórias XD_1 e XD_2 e, depois, calculando as chaves públicas correspondentes Y_{D1} e Y_{D2}.
2. Alice transmite Y_A para Bob.
3. Darth intercepta Y_A e transmite X_{D1} para Bob. Darth também calcula $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob recebe X_{D1} e calcula $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob transmite Y_B para Alice.
6. Darth intercepta Y_B e transmite Y_{D2} para Alice. Darth calcula $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. Alice recebe Y_{D2} e calcula $K2 = K2 = (Y_{D2})^{X_A} \bmod q$.

Nesse ponto, Bob e Alice acham que compartilham uma chave secreta, mas Bob e Darth compartilham a chave secreta $K1$ e Alice e Darth compartilham a chave secreta $K2$. Toda a comunicação futura entre Bob e Alice estará comprometida da seguinte maneira:

1. Alice envia uma mensagem cifrada M: $E(K2, M)$.
2. Darth intercepta a mensagem cifrada e a decriptografa, para recuperar M.
3. Darth envia a Bob $E(K1, M)$ ou $E(K1, M')$, onde M' é qualquer mensagem. No primeiro caso, Darth simplesmente quer interceptar a comunicação sem alterá-la. No segundo caso, Darth quer modificar a mensagem que está sendo enviada para Bob.

O protocolo de acordo de chaves é vulnerável a tal ataque porque não autentica os participantes. Essa vulnerabilidade pode ser contornada com o uso de assinaturas digitais e certificados de chave pública; esses assuntos são explorados nos capítulos 13 e 14.

10.3 ARITMÉTICA DE CURVA ELÍPTICA

A maior parte dos produtos que utilizam a criptografia de chave pública para criptografia e assinaturas digitais utiliza RSA. Conforme já vimos, o tamanho da chave para o uso seguro do RSA tem aumentado nos últimos anos, e isso gerou uma carga de processamento maior sobre as aplicações usando RSA. Esse peso tem ramificações, especialmente para sites de comércio eletrônico, que realizam grandes quantidades de transações seguras. Recentemente, um sistema concorrente começou a desafiar o RSA: criptografia de curva elíptica (ECC — Elliptic Curve Cryptography). O ECC já está aparecendo em esforços de padronização, como o IEEE P1363 Standard for Public-Key Cryptography.

O atrativo principal do ECC, em comparação com o RSA, é que ele parece oferecer igual segurança com um tamanho de chave muito menor, reduzindo assim o overhead do processamento. Por outro lado, embora a teoria do ECC já exista há algum tempo, só recentemente esses produtos começaram a aparecer, e tem havido interesse criptoanalítico contínuo em encontrar algum ponto fraco. Por conseguinte, o nível de confiança no ECC ainda não é tão alto quando no RSA.

O ECC é fundamentalmente mais difícil de explicar do que o RSA ou Diffie-Hellman, e uma descrição matemática completa está fora do escopo deste livro. Esta seção e a próxima oferecem alguma base sobre curvas elípticas e ECC. Começaremos com uma rápida revisão do conceito de grupo abeliano. Em seguida, examinaremos o conceito de curvas elípticas, definido sobre números reais. Isso será acompanhado por uma visão das curvas elípticas definidas sobre corpos finitos. Finalmente, podemos examinar as cifras de curva elíptica.

O leitor poderá rever o material sobre corpos finitos no Capítulo 4, antes de prosseguir.

Grupos abelianos

Lembre-se, do Capítulo 4, de que um **grupo** abeliano G, às vezes indicado por $\{G, \bullet\}$, é um conjunto de elementos com uma operação binária, indicada por $\bullet$, que associa a cada par ordenado (a, b) de elementos em G um elemento $(a \bullet b)$ em G, tal que os seguintes axiomas sejam obedecidos:[2]

(A1) Fechamento:	Se a e b pertencem a G, então $a \bullet b$ também está em G.
(A2) Associativo:	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ para todo a, b, c em G.

2. O operador $\bullet$ é genérico e pode se referir a adição, multiplicação ou alguma outra operação matemática.

(A3) Elemento identidade: Existe um elemento e em G tal que $a \bullet e = e \bullet a = a$ para todo a em G.

(A4) Elemento inverso: Para cada a em G existe um elemento a' em G tal que $a \bullet a' = a' \bullet a = e$.

(A5) Commutativo: $a \bullet b = b \bullet a$ para todo a, b em G.

Diversas cifras de chave pública baseiam-se no uso de um grupo abeliano. Por exemplo, o acordo de chaves Diffie-Hellman envolve multiplicar pares de inteiros diferentes de zero módulo um número primo q. As chaves são geradas pela exponenciação pelo grupo, com a exponenciação definida como multiplicação repetida. Por exemplo, $a^k \bmod q = (\underbrace{a \times a \times \ldots \times a}_{k \text{ vezes}}) \bmod q$. Para atacar o Diffie-Hellman, o agressor precisa determinar k dados a e a^k; esse é o problema do log discreto.

Para a criptografia de curva elíptica, uma operação sobre curvas elípticas, chamada adição, é usada. A multiplicação é definida pela adição repetida. Por exemplo, $a \times k = (\underbrace{a + a + \ldots + a}_{k \text{ vezes}})$, onde a adição é realizada sobre uma curva elíptica. A criptoanálise envolve determinar k dados a e $(a \times k)$.

Uma curva elíptica é definida por uma equação nas duas variáveis, com coeficientes. Para a criptografia, as variáveis e os coeficientes são restritos a elementos em um corpo finito, o que resulta na definição de um grupo abeliano finito. Antes de examinarmos isso, primeiramente veremos as curvas elípticas em que as variáveis e os coeficientes são números reais. Esse caso talvez seja mais fácil de visualizar.

Curvas elípticas sobre números reais

As curvas elípticas não são elipses. Elas têm esse nome porque são descritas por equações cúbicas, semelhantes às usadas para calcular a circunferência de uma elipse. Em geral, as equações cúbicas para curvas elípticas têm a forma

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

onde a, b, c, d e e são números reais e x e y assumem valores nos números reais.[3] Para a nossa finalidade, é suficiente limitarmos a equações na forma

$$y^2 = x^3 + ax + b \qquad \textbf{(10.1)}$$

Essas equações são consideradas cúbicas, ou de grau 3, pois o expoente mais alto que elas contêm é um 3. Também incluído na definição de uma curva elíptica está um único elemento indicado por *0* e chamado de *ponto na infinito* ou *ponto zero*, que explicaremos mais tarde. Para traçar essa curva, precisamos calcular

$$y = \sqrt{x^3 + ax + b}$$

Para determinados valores de a e b, o gráfico consiste em dois valores de y (um positivo e um negativo) para cada valor de x. Assim, cada curva é simétrica em relação a $y = 0$. A Figura 10.9 mostra dois exemplos de curvas elípticas. Como se pode ver, a fórmula às vezes produz curvas de aparência estranha.

Agora, considere o conjunto de pontos $E(a, b)$ consistindo em todos os pontos (x, y) que satisfazem a Equação (10.1) juntamente com o elemento O. Usar um valor diferente do par (a, b) resulta em um conjunto diferente $E(a, b)$. Usando essa terminologia, as duas curvas da Figura 10.9 representam os conjuntos $E(-1,0)$ e $E(1,1)$, respectivamente.

Descrição geométrica da adição Podemos mostrar que um grupo pode ser definido com base no conjunto $E(a, b)$ para valores específicos de a e b na Equação (10.1), desde que a condição a seguir seja atendida:

$$4a^3 + 27b^2 \neq 0 \qquad \textbf{(10.2)}$$

Para definir o grupo, temos de definir uma operação, chamada adição e indicada por $+$, para o conjunto $E(a, b)$, onde a e b satisfazem a Equação (10.2). Em termos geométricos, as regras para a adição podem ser indicadas da seguinte forma: se três pontos em uma curva elíptica se encontram em uma linha reta, sua soma é O. Por essa definição, podemos definir as regras da adição sobre uma curva elíptica:

1. O funciona como a identidade aditiva. Assim, $O = -O$; para qualquer ponto P na curva elíptica, $P + O = P$. A seguir, consideramos $P \neq O$ e $Q \neq O$.

3. Observe que x e y são variáveis verdadeiras, que assumem valores. Isso é contrário à nossa explicação dos anéis e corpos polinomiais, do Capítulo 4, na qual x era tratada como um indeterminado.

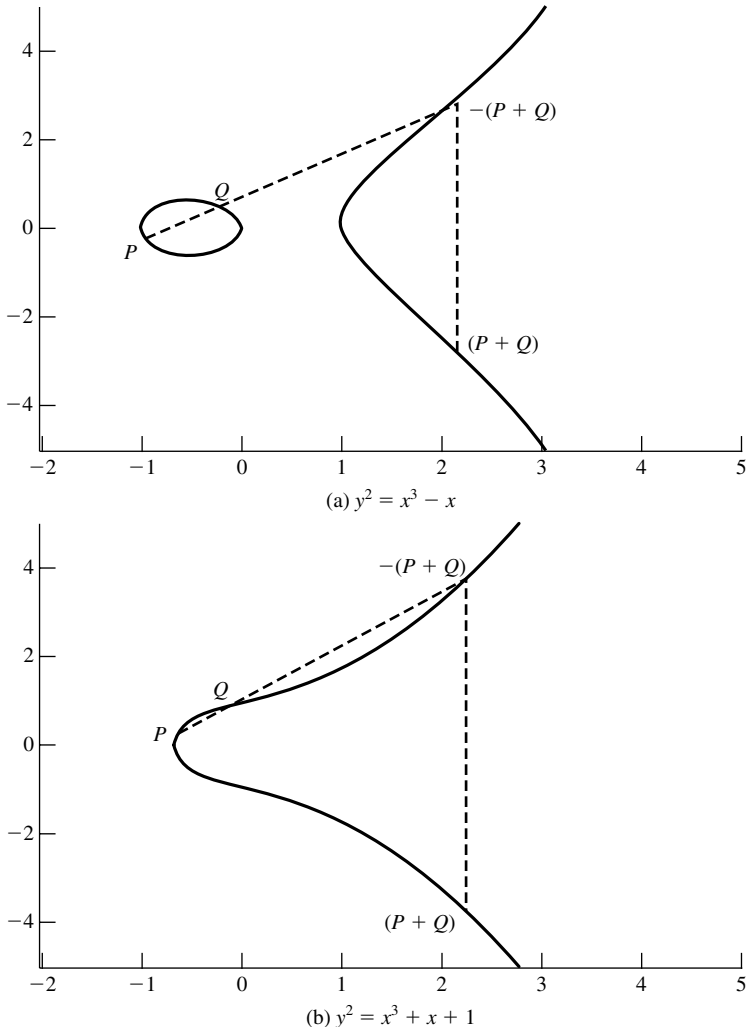


Figura 10.9 Exemplo de curvas elípticas.

2. O negativo de um ponto P é o ponto com a mesma coordenada x, sem o negativo da coordenada y; ou seja, se $P = (x, y)$, então $-P = (x, -y)$. Observe que esses dois pontos podem ser juntados por uma linha vertical. Observe que $P + (-P) = P - P = O$.
3. Para somar dois pontos P e Q com coordenadas x diferentes, desenhe uma linha reta entre eles e encontre o terceiro ponto de interseção R. Pode-se ver facilmente que existe um único ponto R que é o ponto de interseção (a menos que a linha seja tangente à curva em P ou Q, quando consideramos $R = P$ ou $R = Q$, respectivamente). Para formar uma estrutura de grupo, precisamos definir a adição sobre três pontos da seguinte forma: $P + Q = -R$. Ou seja, definimos $P + Q$ como sendo a imagem-espelho (com relação ao eixo x) do terceiro ponto da interseção. A Figura 10.9 ilustra essa construção.
4. A interpretação geométrica do item anterior também se aplica a dois pontos, P e $-P$, com a mesma coordenada x. Os pontos são reunidos por uma linha vertical, que também pode ser vista como a interseção da curva no ponto infinito. Portanto, temos $P + (-P) = O$, coerente com o item (2).
5. Para dobrar um ponto Q, desenhe a linha tangente e encontre o outro ponto da interseção S. Então, $Q + Q = 2Q = -S$.

Com a lista de regras apresentada, pode-se mostrar que o conjunto $E(a, b)$ é um grupo abeliano.

Descrição algébrica da adição

Nesta subseção, apresentamos alguns resultados que permitem o cálculo de adições sobre curvas elípticas.[4] Para dois pontos distintos $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ que não são negativos um do outro, a inclinação da linha l que os junta

4 Para ver derivações desses resultados, consulte [KOBL94] ou outros métodos matemáticos de curvas elípticas.

é $\Delta = (y_Q - y_P)/(x_Q - x_P)$. Existe exatamente um outro ponto onde l cruza a curva elíptica, e esse é o negativo da soma de P e Q. Após alguma manipulação algébrica, podemos expressar a soma $R = P + Q$ da seguinte forma:

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

Também precisamos ser capazes de incluir um ponto em si mesmo: $P + P = 2P = R$. Quando $y_P \neq 0$, as expressões são

$$\begin{aligned} x_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \\ y_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \end{aligned} \tag{10.4}$$

Curvas elípticas sobre Z_p

A criptografia de curva elíptica utiliza curvas elípticas em que as variáveis e coeficientes são todos restritos a elementos de um corpo finito. Duas famílias de curvas elípticas são usadas nas aplicações criptográficas: curvas primas sobre $\mathbf{Z}_p$ e curvas binárias sobre GF(2^m). Para uma curva prima sobre $\mathbf{Z}_p$, usamos uma equação cúbica em que todas as variáveis e coeficientes assumem valores no conjunto de inteiros de 0 até $p - 1$ e em que os cálculos são realizados módulo p. Para uma curva binária definida sobre GF(2^m), todas as variáveis e coeficientes assumem valores em GF(2^n) e nos cálculos são realizados sobre GF(2^n). [FERN99] aponta que as curvas primas são melhores para aplicações de software, pois as operações com manipulação estendida de bits, necessária para curvas binárias, não são exigidas; e que as curvas binárias são melhores para aplicações de hardware, onde são necessárias muito menos portas lógicas para criar um criptossistema poderoso e rápido. Examinaremos essas duas famílias nesta seção e na seguinte.

Não existe uma interpretação geométrica óbvia da aritmética de curva elíptica sobre corpos finitos. A interpretação algébrica usada para a aritmética de curva elíptica sobre números reais é prontamente transportada, e essa é a técnica que consideramos.

Para curvas elípticas sobre $\mathbf{Z}_p$, assim como em números reais, nos limitamos a equações na forma da Equação (10.1), mas nesse caso com coeficientes e variáveis limitados a $\mathbf{Z}_p$:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \tag{10.5}$$

Por exemplo, a Equação (10.5) é satisfeita para $a = 1, b = 1, x = 9, y = 7, p = 23$:

$$\begin{aligned} 7^2 \bmod 23 &= (9^3 + 9 + 1) \bmod 23 \\ 49 \bmod 23 &= 739 \bmod 23 \\ 3 &= 3 \end{aligned}$$

Agora, considere o conjunto $E_p(a, b)$ consistindo em todos os pares de inteiros (x, y) que satisfazem a Equação (10.5), juntamente com um ponto no infinito O. Os coeficientes a e b e as variáveis x e y são todos elementos de $\mathbf{Z}_p$.

Por exemplo, considere $p = 23$ e considere a curva elíptica $y^2 = x^3 + x + 1$. Nesse caso, $a = b = 1$. Observe que essa equação é a mesma da Figura 10.9b. A figura mostra uma curva contínua com todos os pontos reais que satisfazem a equação. Para o conjunto $E_{23}(1, 1)$, só estamos interessados nos inteiros não negativos no quadrante de $(0, 0)$ até $(p - 1, p - 1)$ que satisfaçam a equação mod p. A Tabela 10.1 relaciona os pontos (diferentes de O) que fazem parte de $E_{23}(1, 1)$. A Figura 10.10 desenha os pontos de $E_{23}(1, 1)$; observe que os pontos, com uma exceção, são simétricos em torno de $y = 11{,}5$.

Tabela 10.1 Pontos na curva elíptica $E_{23}(1, 1)$

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

Pode-se mostrar que um grupo abeliano finito pode ser definido com base no conjunto $E_p(a, b)$, desde que $(x^3 + ax + b) \bmod p$ não tenha fatores repetidos. Isso é equivalente à condição

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \qquad \textbf{(10.6)}$$

Observe que a Equação (10.6) tem a mesma forma da Equação (10.2).

As regras para adição sobre $E_p(a, b)$ correspondem à técnica algébrica descrita para as curvas elípticas definidas sobre números reais. Para todos os pontos $P, Q \in E_p(a, b)$:

1. $P + O = P$.
2. Se $P - (x_P, y_P)$, então $P + (x_P, -y_P) = O$. O ponto $(x_P, -y_P)$ é o negativo de P, indicado como $-P$. Por exemplo, em $E_{23}(1,1)$, para $P = (13, 7)$, temos $-P = (13, -7)$. Mas $-7 \bmod 23 = 16$. Portanto, $-P = (13, 16)$, que também está em $E_{23}(1, 1)$.
3. Se $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ com $P \neq -Q$, então $R = P + Q = (x_R, y_R)$ é determinado pelas seguintes regras:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

onde

$$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{se } P \neq Q \\ \left(\dfrac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{se } P = Q \end{cases}$$

4. A multiplicação é definida como adição repetida; por exemplo, $4P = P + P + P + P$.

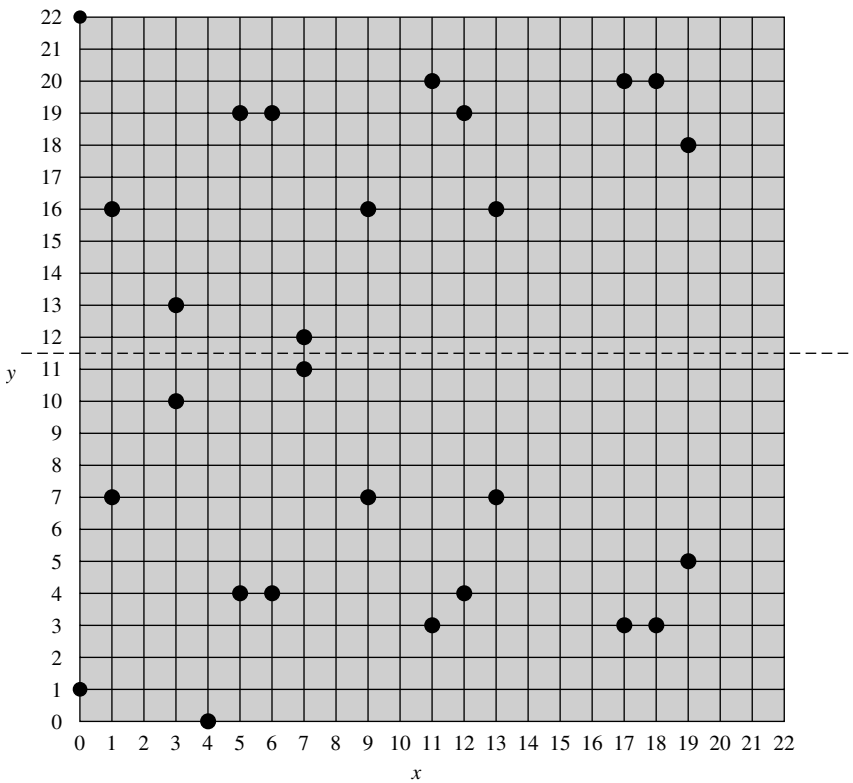


Figura 10.10 A curva elíptica $E_{23}(1, 1)$.

Por exemplo, considere $P = (3, 10)$ e $Q = (9, 7)$ em $E_{23}(1, 1)$. Então

$$\lambda = \left(\frac{7 - 10}{9 - 3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$
$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

Assim, $P + Q = (17, 20)$. Para encontrar $2P$,

$$\lambda = \left(\frac{3(3^2) + 1}{2 \times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

A última etapa na equação anterior consiste em calcular o inverso multiplicativo de 4 em Z_{23}. Isso pode ser feito usando-se o algoritmo euclidiano estendido, definido na Seção 4.4. Para confirmar, observe que $(6 \times 4) \bmod 23 = 24 \bmod 23 = 1$.

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$
$$y_R = (6(3 - 7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

e $2P = (7, 12)$.

Para determinar a segurança de várias cifras de curva elíptica, é interessante saber o número de pontos em um grupo abeliano finito definido sobre uma curva elíptica. No caso do grupo finito $E_p(a, b)$, o número de pontos N é limitado por

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$$

Observe que o número de pontos em $E_p(a, b)$ é aproximadamente igual ao número de elementos em Z_p, a saber, p elementos.

Curvas elípticas sobre GF(2^m)

Lembre-se, do Capítulo 4, de que um corpo finito GF(2^m) consiste em 2^m elementos, juntamente com operações de adição e multiplicação que podem ser definidas sobre polinômios. Para curvas elípticas sobre GF(2^m), usamos uma equação cúbica em que as variáveis e coeficientes assumem valores em GF(2^m) para algum número m, e na qual os cálculos são realizados usando-se as regras da aritmética em GF(2^m).

Acontece que a forma da equação cúbica apropriada para aplicações criptográficas para curvas elípticas é um pouco diferente para GF(2^m) e para Z_p. A forma é

$$y^2 + xy = x^3 + ax^2 + b \qquad \textbf{(10.7)}$$

onde é entendido que as variáveis x e y e os coeficientes a e b são elementos de GF(2^m) e que os cálculos são realizados em GF(2^m).

Agora, considere o conjunto $E_{2^m}(a,b)$ consistindo em todos os pares de inteiros (x,y) que satisfazem a Equação (10.7), juntamente com um ponto no infinito O.

Por exemplo, vamos usar o corpo finito GF(2^4) com o polinômio irredutível $f(x) = x^4 + x + 1$. Isso produz um gerador que satisfaz $f(g) = 0$, com um valor de $g^4 = g + 1$, ou, em binário, 0010. Podemos desenvolver as potências de g da seguinte forma:

$g^0 = 0001$	$g^4 = 0011$	$g^8 = 0101$	$g^{12} = 1111$
$g^1 = 0010$	$g^5 = 0110$	$g^9 = 1010$	$g^{13} = 1101$
$g^2 = 0100$	$g^6 = 1100$	$g^{10} = 0111$	$g^{14} = 1001$
$g^3 = 1000$	$g^7 = 1011$	$g^{11} = 1110$	$g^{15} = 0001$

Por exemplo, $g^5 = (g^4)(g) = g^2 + g = 0110$.

Agora considere a curva elíptica $y^2 + xy = x^3 + g^4x^2 + 1$. Nesse caso, $a = g^4$ e $b = g^0 = 1$. Um ponto que satisfaz essa equação é (g^4, g^3):

$$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$$
$$g^6 + g^8 = g^{15} + g^{14} + 1$$
$$1100 + 0101 = 0001 + 1001 + 0001$$
$$1001 = 1001$$

Tabela 10.2 Pontos na curva elíptica $E_{2^4}(g^4, 1)$

$(0, 1)$	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	(g^6, g^8)	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12}, 0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})

A Tabela 10.2 relaciona os pontos (diferentes de O) que fazem parte de $E_{2^4}(g^4, 1)$. A Figura 10.11 desenha os pontos de $E_{2^4}(g^4, 1)$.

Pode-se mostrar que um grupo abeliano finito pode ser definido com base no conjunto $E_{2^m}(a, b)$, desde que $b \neq 0$. As regras para a adição podem ser declaradas da seguinte forma. Para todos os pontos $P, Q \in E_{2^m}(a, b)$:

1. $P + O = P$.
2. Se $P = (x_P, y_P)$, então $P + (x_P, x_P + y_P) = O$. O ponto $(x_P, x_P + y_P)$ é o negativo de P, indicado como $-P$.
3. Se $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ com $P \neq -Q$ e $P \neq Q$, então $R = P + Q = (x_R, y_R)$ é determinado pelas seguintes regras

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$
$$y_R = \lambda(x_P + x_R) + x_R + y_P$$

onde

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. Se $P = (x_P, y_P)$, então $R = 2P = (x_R, y_R)$ é determinado pelas seguintes regras:

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = x_P^2 + (\lambda + 1)x_R$$

onde

$$\lambda = x_P + \frac{y_P}{x_P}$$

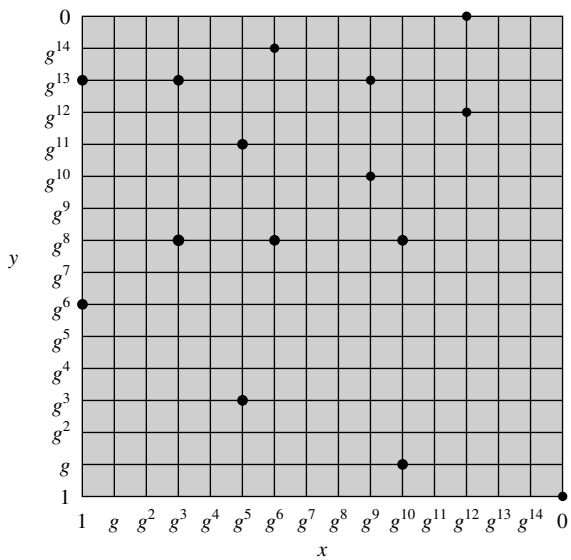


Figura 10.11 A curva elíptica $E_{2^4}(g^4, 1)$

10.4 CRIPTOGRAFIA DE CURVA ELÍPTICA

A operação de adição no ECC é equivalente à multiplicação modular no RSA, e a adição múltipla é o equivalente da exponenciação modular. Para formar um criptossistema usando curvas elípticas, precisamos encontrar um 'problema difícil' como fatorar o produto de dois primos ou calcular o logaritmo discreto.

Considere a equação $Q = kP$, onde $Q, P \in E_p(a, b)$ e $k < p$. É relativamente fácil calcular Q a partir de k e P, mas é computacionalmente difícil determinar k a partir de Q e P. Isso é chamado de problema de logaritmo discreto para curvas elípticas.

Damos um exemplo tirado do site Web da Certicom (www.certicom.com). Considere o grupo $E_{23}(9, 17)$. Esse é o grupo definido pela equação $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$. Qual é o logaritmo discreto k de $Q = (4, 5)$ à base $P = (16, 5)$? O método da força bruta é calcular múltiplos de P até que Q seja encontrado. Assim,

$$P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10);$$
$$6P = (7, 3); 7P = (8, 7); 8P = (12, 17); 9P = (4, 5).$$

Como $9P = (4, 5) = Q$, o logaritmo discreto $Q = (4, 5)$ à base $P = (16, 5)$ é $k = 9$. Em uma aplicação real, k seria muito grande, tornando uma técnica de força bruta inviável.

No restante desta seção mostraremos duas técnicas para o ECC que lhe darão uma idéia da técnica.

Análogo elíptico do acordo de chaves Diffie-Hellman

O acordo de chaves usando curvas elípticas pode ser feito da seguinte maneira. Primeiro, selecione um inteiro grande q, que seja um número primo p ou um inteiro da forma 2^m e parâmetros de curva elíptica a e b para a Equação (10.5) ou a Equação (10.7). Isso define o grupo elíptico de pontos $E_q(a, b)$. Em seguida, escolha um ponto base $G = (x_1, y_1)$ em $E_q(a, b)$ cuja ordem seja um valor muito grande n. A **ordem** n de um ponto G em uma curva elíptica é o menor inteiro positivo n tal que $nG = O$. $E_q(a, b)$ e G são os parâmetros do criptossistema, conhecidos por todos os participantes.

Elementos públicos globais

$E_q(a, b)$	curva elíptica com parâmetros a, b e q, onde q é um primo ou um inteiro na forma 2^m
G	ponto na curva elíptica cuja ordem é o valor grande n

Geração de chave do usuário A

Selecione privada n_A	$n_A < n$
Calcule pública P_A	$P_A = n_A \times G$

Geração de chave do usuário B

Selecione privada n_B	$n_B < n$
Calcule pública P_B	$P_B = n_B \times G$

Cálculo da chave secreta pelo usuário A

$K = n_A \times P_B$

Cálculo da chave secreta pelo usuário B

$K = n_B \times P_A$

Figura 10.12 Acordo de chaves Diffie-Hellman elíptico.

Um acordo de chaves entre os usuários A e B pode ser realizado da seguinte maneira (Figura 10.12):

1. A seleciona um inteiro n_A menor que n. Essa é a chave privada de A. A, então, calcula a chave pública $P_A = n_A \times G$; a chave pública é um ponto em $E_q(a, b)$.
2. Do mesmo modo, B seleciona uma chave privada n_B e calcula a chave pública P_B.
3. A gera a chave secreta $K = n_A \times P_B$. B gera a chave secreta $K = n_B \times P_A$.

Os dois cálculos na etapa 3 produzem o mesmo resultado, porque

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

Para quebrar esse esquema, um agressor teria de ser capaz de calcular k a partir de G e kG, o que é considerado difícil.

Como exemplo,[5] use $p = 211$; $E_p(0, -4)$, que é equivalente à curva $y^2 = x^3 - 4$; e $G = (2, 2)$. Pode-se verificar que $240 \times G = O$. A chave privada de A é $n_A = 121$, de modo que a chave pública de A é $P_A = 121 \times (2, 2) = (115,48)$. A chave privada de B é $n_B = 203$, de modo que a chave pública de B é $P_B = 203 \times (2, 2) = (130, 203)$. A chave secreta compartilhada é $121 \times (130, 203) = 203 \times (115, 48) = (161, 69)$.

Observe que a chave secreta é um par de números. Se essa chave tiver de ser usada como uma chave de sessão para a criptografia convencional, então um único número precisará ser gerado. Poderíamos simplesmente usar as coordenadas x ou alguma função simples da coordenada x.

Criptografia/decriptografia com curvas elípticas

Várias técnicas de criptografia/decriptografia usando curvas elípticas foram analisadas na literatura. Nesta subseção, examinamos talvez a mais simples. A primeira tarefa nesse sistema é codificar a mensagem de texto claro m a ser enviada como um ponto $x-y$ P_m. Esse é o ponto P_m que será critografado como um texto cifrado e mais tarde decriptografado. Observe que não podemos simplesmente codificar a mensagem como a coordenada x ou y de um ponto, pois nem todas as coordenadas estão em $E_q(a, b)$; por exemplo, consulte a Tabela 10.1. Novamente, existem várias técnicas para essa codificação, que não explicaremos aqui, mas basta dizer que existem técnicas relativamente simples que podem ser usadas.

Assim como no sistema de acordo de chaves, um sistema de criptografia/decriptografia requer um ponto G e um grupo elíptico $E_q(a, b)$ como parâmetros. Cada usuário A seleciona uma chave privada n_A e gera uma chave pública $P_A = n_A \times G$.

Para cifrar e enviar uma mensagem P_m a B, A escolhe um inteiro positivo aleatório k e produz o texto cifrado C_m consistindo no par de pontos:

$$C_m = \{kG, P_m + kP_B\}$$

Observe que A usou a chave pública de B, P_B. Para decriptografar o texto cifrado, B multiplica o primeiro ponto do par pela chave secreta de B e subtrai o resultado do segundo ponto:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

A mascarou a mensagem P_m somando kP_B a ela. Ninguém além de A conhece o valor de k, de modo que embora P_B seja uma chave pública, ninguém pode remover a máscara KP_B. Porém, A também inclui uma 'dica', que é suficiente para remover a máscara se alguém souber a chave privada n_B. Para um agressor recuperar a mensagem, ele teria de calcular k a partir de G e kG, o que é considerado difícil.

Como exemplo do processo de criptografia (retirado de [KOBL94]), considere $p = 751$; $E_p(-1,188)$, que é equivalente à curva $y^2 = x^3 - x + 188$; e $G = (0,376)$. Suponha que A queira enviar uma mensagem a B que está codificada no ponto elíptico $P_m = (562, 201)$ e que A selecione o número aleatório $k = 386$. A chave pública de B é $P_B = (201, 5)$. Temos $386(0, 376) = (676, 558)$ e $(562, 201) + 386(201, 5) = (385, 328)$. Assim, A envia o texto cifrado $\{(676, 558),(385, 328)\}$.

Segurança da criptografia com curvas elípticas

A segurança do ECC depende da dificuldade de se determinar k dados kP e P. Isso é conhecido como problema do logaritmo elíptico. A técnica mais rápida conhecida para calcular o logaritmo elíptico é conhecida como método Pollard rho. A Tabela 10.3 compara diversos algoritmos mostrando tamanhos de chave comparáveis em termos do esforço computacional para a criptoanálise. Como podemos ver, um tamanho de chave consideravelmente menor pode ser usado para o ECC em comparação com o RSA. Além disso, para tamanhos de chave iguais, o esforço computacional exigido para ECC e RSA é comparável [JURI97]. Assim, existe uma vantagem computacional para o uso do ECC com um tamanho de chave mais curto do que um RSA comparavelmente seguro.

5. Fornecido por Edição Schaefer, da Universidade de Santa Clara.

Tabela 10.3 Tamanhos de chave comparáveis em termos de esforço computacional de criptoanálise

Esquema simétrico (tamanho da chave em bits)	Esquema baseado em ECC (tamanho de *n* em bits)	RSA/DSA (módulo tamanho em bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Fonte: Certicom

10.5 LEITURA E SITES WEB RECOMENDADOS

Um texto bastante acessível sobre criptografia com curvas elípticas é [ROSI99]; a ênfase está na implementação do software. Outro livro acessível, porém rígido, é [HANK04]. Dois outros bons textos, ambos contendo matemática rígida, são [BLAK99] e [ENGE99]. Também há boas descrições, porém concisas, em [KUMA98], [STIN02] e [KOBL94]. Dois textos de estudo interessantes são [FERN99] e [JURI97].

BLAK99 Blake, I.; Seroussi, G.; e Smart, N. *Elliptic curves in cryptography*. Cambridge: Cambridge University Press, 1999.
ENGE99 Enge, A. *Elliptic curves and their applications to cryptography*. Norwell, MA: Kluwer Academic Publishers, 1999.
FERN99 Fernandes, A. "Elliptic curve cryptography." *Dr. Dobb's Journal*, dezembro de 1999.
HANK04 Hankerson, D.; Menezes, A.; e Vanstone, S. *Guide to Elliptic Curve Cryptography*. New York: Springer, 2004.
JURI97 Jurisic, A., e Menezes, A. "Elliptic curves and cryptography." *Dr. Dobb's Journal*, abril de 1997.
KOBL94 Koblitz, N. *A course in number theory and cryptography*. New York: Springer-Verlag, 1994.
KUMA98 Kumanduri, R., e Romero, C. *Number theory with computer applications*. Upper Saddle River: Prentice Hall, 1998.
ROSI99 Rosing, M. *Implementing elliptic curve cryptography*. Greenwich: Manning Publications, 1999.
STIN02 Stinson, D. *Cryptography: theory and practice*. Boca Raton: CRC Press, 2002.

Site Web recomendado:

- **Certicom:** Extensa coleção de material técnico a respeito de criptografia de curva elíptica e outros tópicos sobre criptografia.

10.6 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

acordo de chaves Diffie-Hellman
aritmética de curva elíptica
ataque *man-the-middle*
corpo finito
certificado de chave pública
criptografia com curvas elípticas
curva binária
curva elíptica
curva prima
diretório de chave pública
distribuição de chaves
equação cúbica
gerenciamento de chaves
grupo abeliano
logaritmo discreto
ponto zero
raiz primitiva

Perguntas de revisão

10.1 Quais são dois usos diferentes da criptografia de chave pública relacionados à distribuição de chaves?

10.2 Relacione quatro categorias gerais de esquemas para a distribuição de chaves públicas.

10.3 Quais são os ingredientes essenciais de um diretório de chave pública?

10.4 O que é um certificado de chave pública?

10.5 Quais são os requisitos para o uso de um esquema de certificado de chave pública?

10.6 Explique rapidamente o acordo de chaves Diffie-Hellman.

10.7 O que é uma curva elíptica?

10.8 O que é o ponto zero de uma curva elíptica?

10.9 Qual é a soma de três pontos em uma curva elíptica que se encontra em uma linha reta?

Problemas

10.1 Os usuários A e B utilizam a técnica de acordo de chaves Diffie-Hellman com um primo comum e uma raiz primitiva $\alpha = 7$.

a. Se o usuário A tem chave privada $X_A = 5$, qual é a chave pública de A, Y_A?

b. Se o usuário B tem chave privada $X_B = 12$, qual é a chave pública de B, Y_B?

c. Qual é a chave secreta compartilhada?

10.2 Considere um esquema Diffie-Hellman com um primo comum $q = 11$ e uma raiz primitiva $\alpha = 2$.

a. Mostre que 2 é uma raiz primitiva de 11.

b. Se o usuário A possui chave pública $Y_A = 9$, qual é a chave privada de A, X_A?

c. Se o usuário B tem chave pública $Y_B = 3$, qual é a chave secreta compartilhada K, compartilhada com A?

10.3 No protocolo Diffie-Hellman, cada participante seleciona um número secreto x e envia ao outro participante $\alpha^x \bmod q$ para algum número público α. O que aconteceria se, em vez disso, os participantes enviassem uns aos outros $x\alpha$ para algum número público α? Dê pelo menos um método que Alice e Bob poderiam usar para combinar uma chave. Eva pode quebrar seu sistema sem descobrir os números secretos? Eva pode descobrir os números secretos?

10.4 Esse problema ilustra a questão de que o protocolo Diffie-Hellman não é seguro sem a etapa em que o módulo é calculado; ou seja, o 'problema do logaritmo indiscreto' não é um problema difícil! Você é Eva, e capturou Alice e Bob e os aprisionou. Você ouve às escondidas o seguinte diálogo:

Bob: Oh, não vamos nos preocupar com o primo no protocolo Diffie-Hellman, isso tornará as coisas mais fáceis.

Alice: Tudo bem, mas ainda precisamos de uma base α para exponenciar. Que tal $g = 3$?

Bob: Está certo, então meu resultado é 27.

Alice: E o meu é 243.

Qual é o X_B secreto de Bob e o X_A secreto de Alice? Qual é sua chave combinada secreta? (Não se esqueça de mostrar seu trabalho.)

10.5 A Seção 10.2 descreve um ataque *man-in-the-middle* sobre o protocolo de acordo de chaves Diffie-Hellman, em que o adversário gera dois pares de chaves pública-privada para o ataque. O mesmo ataque poderia ser realizado com um par? Explique.

10.6 Em 1985, T. ElGamal anunciou um esquema de chave pública baseado em logaritmos discretos, bastante relacionados com a técnica Diffie-Hellman. Assim como Diffie-Hellman, os elementos globais do esquema de ElGamal são um número primo q e α, uma raiz primitiva de q. Um usuário A seleciona uma chave privada X_A e calcula uma chave pública Y_A como em Diffie-Hellman. O usuário A criptografa um texto claro $M < q$ destinado ao usuário B, da seguinte forma:

1. Escolha um inteiro aleatório k tal que $1 \leq k \leq q - 1$.
2. Calcule $K = (Y_B)^k \bmod q$.
3. Criptografe M como o par de inteiros (C_1, C_2), onde

$$C_1 = \alpha^k \bmod q \quad C_2 = KM \bmod q$$

Escolha um inteiro aleatório k tal que:

1. $1 \leq k \leq q - 1$.
2. Calcule $K = (Y_B)^k \bmod q$.

Mostre que o sistema funciona; ou seja, mostre que o processo de decriptografia recupera o texto claro.

10.7 Considere um esquema de ElGamal com um primo comum $q = 71$ e uma raiz primitiva $\alpha = 7$.

a. Se B tem chave pública $Y_B = 3$ e A escolheu um inteiro aleatório $k = 2$, qual é o texto cifrado de $M = 30$?

b. Se A agora escolher um valor diferente de k, de modo que a codificação de $M = 30$ seja $C = (59, C_2)$, qual será o inteiro C_2?

10.8 A regra (5) para realizar a aritmética em curvas elípticas sobre números reais afirma que, para dobrar um ponto Q, deve-se desenhar a linha tangente e encontrar o outro ponto de interseção S. Então, $Q + Q = 2Q = -S$. Se a linha tangente não for vertical, haverá exatamente um ponto de interseção. Porém, suponha que a linha tangente seja vertical. Nesse caso, qual é o valor $2Q$? Qual é o valor $3Q$?

10.9 Demonstre que as duas curvas elípticas da Figura 10.9 satisfazem as leis de grupo sobre os números reais.

10.10 (4, 7) é um ponto na curva elíptica $y^2 = x^3 - 5x + 5$ sobre números reais?

10.11 Na curva elíptica sobre os números reais $y^2 = x^3 - 36x$, considere $P = (-3{,}5, 9{,}5)$ e $Q = (-2{,}5, 8{,}5)$. Encontre $P + Q$ e $2P$.

10.12 A equação de curva elíptica $y^2 = x^3 + 10x + 5$ define um grupo sobre Z_{17}?

10.13 Considere a curva elíptica $E_{11}(1, 6)$: ou seja, a curva é definida por $y^2 = x^3 + x + 6$ com um módulo de $p = 11$. Determine todos os pontos em $E_{11}(1, 6)$. *Dica:* Comece calculando o lado direito da equação para todos os valores de x.

10.14 Quais são os negativos dos seguintes pontos de curva elíptica sobre Z_{17}? P = (5, 8); Q = (3, 0); R = (0, 6).

10.15 Para $E_{11}(1,6)$, considere o ponto $G = (2, 7)$. Calcule os múltiplos de G de $2G$ até $13G$.

10.16 Este problema realiza criptografia/decriptografia de curva elíptica usando o esquema esboçado na Seção 10.4. Os parâmetros do criptossistema são $E_{11}(1, 6)$ e $G = (2, 7)$. A chave secreta de B é $n_B = 7$.

a. Calcule a chave pública P_B de B.

b. A deseja cifrar a mensagem $P_m = (10, 9)$ e escolhe o valor aleatório $k = 3$. Determine o texto cifrado C_m.

c. Mostre o cálculo pelo qual B recupera P_m de C_m.

10.17 A seguir, vemos uma primeira tentativa de um esquema de assinatura de curva elíptica. Temos uma curva elíptica global, o primo p, e o gerador G. Alice escolhe uma chave de assinatura privada X_A e forma a chave de verificação pública $Y_A = X_A G$. Para assinar uma mensagem M:

- Alice seleciona um valor k.
- Alice envia para Bob M,k e a assinatura $S = M - kX_A G$.
- Bob verifica se $M = S + kY_A$

a. Mostre que esse esquema funciona, ou seja, que o processo de verificação produz uma igualdade se a assinatura for válida.

b. Mostre que o esquema é inaceitável descrevendo uma técnica simples para falsificar a assinatura de um usuário em uma mensagem qualquer.

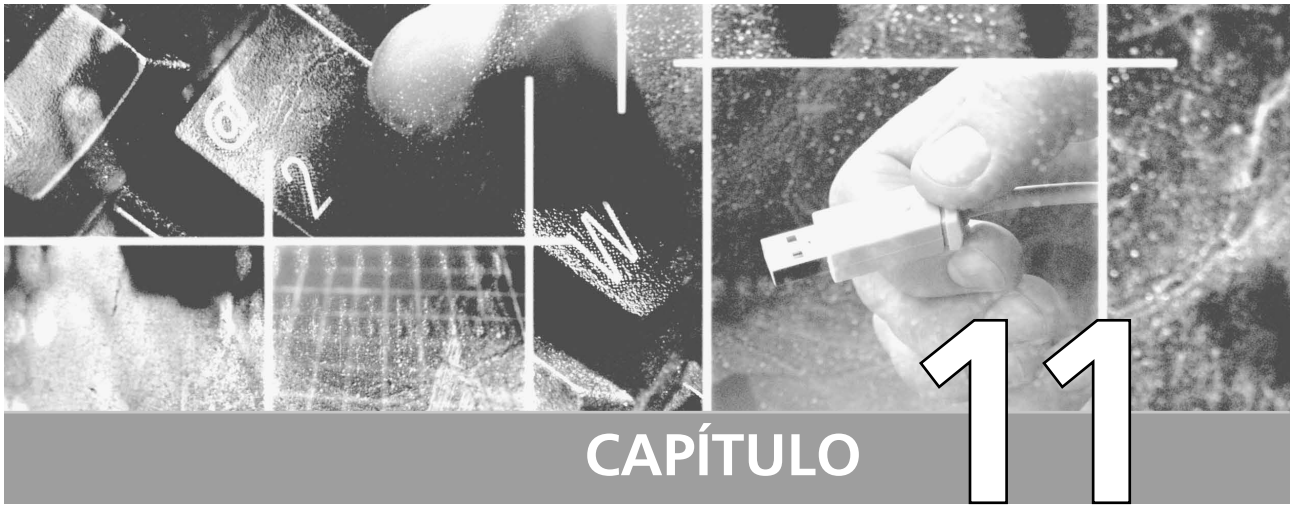
10.18 Aqui está uma versão melhorada do esquema dado no problema anterior. Como antes, temos uma curva elíptica global, o primo p, e o 'gerador' G. Alice seleciona uma chave de assinatura privada X_A e forma a chave de verificação pública $Y_A = X_A G$. Para assinar uma mensagem M,

- Bob seleciona um valor k.
- Bob envia a Alice $C_1 = kG$.
- Alice envia para Bob M e a assinatura $S = M - X_A C_1$.
- Bob verifica se $M = S + kY_A$

a. Mostre que esse esquema funciona, ou seja, que o processo de verificação produz uma igualdade se a assinatura for válida.

b. Mostre que forjar uma mensagem nesse esquema é tão difícil quanto quebrar a criptografia de curva elíptica (ElGamal). (Ou encontre um modo mais fácil de forjar uma mensagem.)

c. Esse esquema tem uma rodada extra em comparação a outros criptossistemas e esquemas de assinatura que vimos. Quais são as desvantagens disso?



Autenticação de mensagem e funções de hash

11.1 **Requisitos de autenticação**

11.2 **Funções de autenticação**

11.3 **Códigos de autenticação de mensagem (MACs – Message Authentication Codes)**

11.4 **Funções de hash**

11.5 **Segurança das funções de hash e MACs**

11.6 **Leitura recomendada**

11.7 **Principais termos, perguntas de revisão e problemas**

Apêndice 11A Base matemática do ataque do aniversário

No domingo, ele apanhou a mensagem de dentro da coluna e acrescentou o nome de Peter Moran aos dois nomes já impressos no código 'Brontossauro'. A mensagem agora era: "Leviathan para Dragon: Martin Hillman, Trevor Allan, Peter Moran: observem e sigam de perto." Qual foi a vantagem disso, John não sabia. Ele se sentiu melhor, achando que, enfim, atacou Peter Moran em vez de esperar passivamente sem qualquer retaliação. Além disso, qual é a vantagem de estar de posse da chave do código se ele nunca tirar proveito disso?

Talking to strange men, Ruth Rendell

PONTOS PRINCIPAIS

- A autenticação de mensagens é um mecanismo ou serviço usado para verificar a integridade de uma mensagem. A autenticação garante que os dados recebidos sejam exatamente iguais aos enviados (ou seja, não contêm modificação, inserção, exclusão ou repetição) e que a identidade afirmada pelo emissor é válida.
- A criptografia simétrica oferece autenticação entre os que compartilham a chave secreta. A criptografia de uma mensagem pela chave privada de um emissor também oferece uma forma de autenticação.
- As duas técnicas criptográficas mais comuns de autenticação de mensagens são um código de autenticação de mensagens (*MAC — Message Authentication Code*) e uma função de hash segura.
- Um MAC é um algoritmo que requer o uso de uma chave secreta. Um MAC apanha uma mensagem de comprimento e uma chave secreta como entrada e produz um código de autenticação. Um destinatário de posse da chave secreta pode gerar um código de autenticação para verificar a integridade da mensagem.
- Uma função de hash mapeia uma mensagem de tamanho variável em um valor de hash de tamanho fixo, ou um resumo de mensagem *(message digest)*. Para autenticação da mensagem, uma função de hash segura precisa ser combinada de alguma forma com uma chave secreta.

Talvez a área mais confusa da segurança de rede seja a da autenticação de mensagens e o tema relacionado a assinaturas digitais. Os ataques e contramedidas tornam-se tão complicados que os profissionais dessa área começam a lembrar os astrônomos de antigamente, que montavam epiciclos em cima de epiciclos em uma tentativa de lidar com todas as contingências. Felizmente, parece que os projetistas de protocolos criptográficos atuais, diferentemente daqueles astrônomos há muito esquecidos, estão trabalhando a partir de um modelo fundamentalmente sadio.

Seria impossível, em algo menor que o tamanho de um livro, esgotar todas as funções e protocolos criptográficos que foram propostos ou implementados para autenticação de mensagens e assinaturas digitais. Em vez disso, a finalidade deste capítulo e dos dois seguintes é oferecer uma ampla visão geral do assunto, e desenvolver um meio sistemático de descrever as diversas técnicas.

Este capítulo começa com uma introdução aos requisitos para autenticação e assinatura digital, e os tipos de ataques que serão combatidos. Em seguida são estudadas as abordagens básicas, incluindo a cada vez mais importante área das funções de hash seguras. Funções de hash específicas são examinadas no Capítulo 12.

11.1 REQUISITOS DE AUTENTICAÇÃO

No contexto das comunicações por uma rede, os seguintes ataques podem ser identificados:

1. **Divulgação:** Liberação do conteúdo da mensagem a qualquer pessoa ou processo que não possui a chave criptográfica apropriada.
2. **Análise de tráfego:** Descoberta do padrão de tráfego entre as partes. Em uma aplicação orientada a conexão, a freqüência e a duração das conexões podem ser determinadas. Em um ambiente orientado a conexão ou também em um ambiente sem conexão, o número e o comprimento das mensagens entre as partes podem ser determinados.
3. **Mascaramento (*masquerade*):** Inserção de mensagens na rede a partir de uma origem fraudulenta. Isso inclui a criação de mensagens por um oponente, que fingem ter vindo de uma entidade autorizada. Também estão incluídas as confirmações fraudulentas de recebimento ou não-recebimento de mensagens por alguém que não seja o destinatário da mensagem.
4. **Modificação de conteúdo:** Mudanças no conteúdo de uma mensagem, incluindo inserção, exclusão, transposição e modificação.
5. **Modificação de seqüência:** Qualquer modificação em uma seqüência de mensagens entre as partes, incluindo inserção, exclusão e reordenação.
6. **Modificação de tempo:** Atraso ou repetição de mensagens. Em uma aplicação orientada a conexão, uma sessão inteira ou uma seqüência de mensagens pode ser uma repetição de alguma sessão anterior válida, ou mensagens individuais na seqüência podem ser adiadas ou repetidas. Em uma aplicação sem conexão, uma mensagem individual (por exemplo, datagrama) pode ser adiada ou repetida.
7. **Retratação da origem:** Negação de transmissão de mensagem pela origem.
8. **Retratação do destino:** Negação do recebimento da mensagem pelo destino.

Medidas para lidar com os dois primeiros ataques estão no âmbito da confidencialidade da mensagem, e são tratadas na Parte 1. As medidas para lidar com os itens de 3 a 6 na lista anterior geralmente são consideradas como autenticação de mensagem. Os mecanismos para tratar especificamente do item 7 estão sob o título de assinaturas digitais. Em geral, uma técnica de assinatura digital também agirá contra alguns ou todos os ataques listados sob os itens de 3 a 6. O tratamento do item 8 pode exigir uma combinação do uso de assinaturas digitais e um protocolo projetado para impedir esse ataque.

Resumindo, a autenticação de mensagens é um procedimento para verificar se as mensagens recebidas provêm da origem afirmada e se não foram alteradas. A autenticação da mensagem pode ainda verificar a seqüência e o instante correto. Uma assinatura digital é uma técnica de autenticação que também inclui medidas para impedir a retratação pela origem.

11.2 FUNÇÕES DE AUTENTICAÇÃO

Qualquer mecanismo de autenticação de mensagens ou de assinatura digital possui dois níveis de funcionalidade. No nível mais baixo, é preciso haver algum tipo de função que produza um autenticador: um valor a ser usado para autenticar uma mensagem. Essa função de baixo nível é então usada como uma primitiva em um protocolo de autenticação de nível mais alto, que permite a um receptor verificar a autenticidade de uma mensagem.

Esta seção trata dos tipos de funções que podem ser usadas para produzir um autenticador. Elas podem ser agrupadas em três classes, da seguinte forma:

- **Criptografia de mensagens:** O texto cifrado da mensagem inteira serve como seu autenticador
- **Código de autenticação de mensagens (MAC — *Message Authentication Code*):** Uma função da mensagem e de uma chave secreta que produz um valor de tamanho fixo, que serve como autenticador
- **Função de hash:** Uma função que relaciona uma mensagem de qualquer tamanho a um valor de hash de tamanho fixo, que serve como autenticador

Agora, examinaremos rapidamente cada um desses tópicos; MACs e funções de hash serão examinados com mais detalhes nas Seções 11.3 e 11.4.

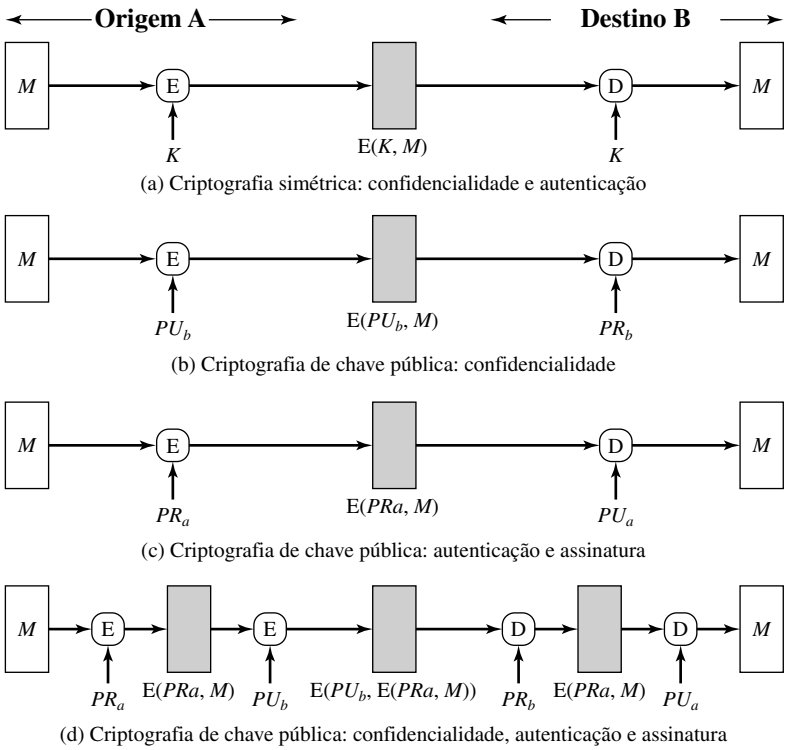
Criptografia de mensagens

A criptografia de mensagens por si só pode oferecer uma medida de autenticação. A análise difere para esquemas de criptografia de chave simétrica e pública.

Criptografia simétrica Considere o uso direto da criptografia simétrica (Figura 11.1a). Uma mensagem M transmitida da origem A para o destino B é criptografada usando uma chave secreta K compartilhada por A e B. Se nenhuma outra parte conhecer a chave, então a confidencialidade é obtida: nenhuma outra parte pode recuperar o texto claro da mensagem.

Além disso, podemos dizer que B tem garantias de que a mensagem foi gerada por A. Por quê? A mensagem deve ter vindo de A porque A é a única outra parte que possui K e, portanto, a única outra parte com a informação necessária para construir o texto cifrado que pode ser decriptografado com K. Além disso, se M for recuperado, B saberá que nenhum dos bits de M foi alterado, pois um oponente que não conhece K não saberia como alterar os bits no texto cifrado para produzir as mudanças desejadas no texto claro.

Assim, podemos dizer que a criptografia simétrica oferece autenticação, bem como confidencialidade. Porém, essa afirmação absoluta precisa ser qualificada. Considere exatamente o que está acontecendo em B. Dada uma função de decriptografia D e uma chave secreta K, o destino aceitará *qualquer* entrada X e produzirá saída $Y = D(K, X)$. Se X for o texto cifrado de uma mensagem legítima M produzida pela função de criptografia correspondente, então Y será alguma mensagem de texto claro M. Caso contrário, Y provavelmente será uma seqüência de bits sem significado. Podem ser necessários alguns meios automatizados para determinar em B se Y é texto claro legítimo e, por conseqüência, proveniente de A.



(a) Criptografia simétrica: confidencialidade e autenticação

(b) Criptografia de chave pública: confidencialidade

(c) Criptografia de chave pública: autenticação e assinatura

(d) Criptografia de chave pública: confidencialidade, autenticação e assinatura

Figura 11.1 Usos básicos da criptografia de mensagens.

As implicações da linha de raciocínio do parágrafo anterior são profundas, do ponto de vista da autenticação. Suponha que a mensagem M possa ser qualquer padrão de bits arbitrário. Nesse caso, não há um meio de determinar automaticamente, no destino, se uma mensagem recebida é o texto cifrado de uma mensagem legítima. Essa conclusão é indiscutível: se M pode ser qualquer padrão de bits, então, independente do valor de X, o valor $Y = D(K, X)$ é *algum* padrão de bits e, portanto, precisa ser aceito como texto claro autêntico.

Assim, em geral, exigimos que somente um pequeno subconjunto de todos os padrões de bits possíveis sejam considerados texto claro legítimo. Nesse caso, provavelmente qualquer texto cifrado forjado não produzirá texto claro legítimo. Por exemplo, suponha que somente um padrão de bits em 10^6 seja texto claro legítimo. Então, a probabilidade de que qualquer padrão de bits escolhido aleatoriamente, tratado como texto cifrado, produza uma mensagem de texto claro legítimo é de apenas 10^{-6}.

Para diversas aplicações e esquemas de criptografia, as condições desejadas prevalecem naturalmente. Por exemplo, suponha que estejamos transmitindo mensagens em idioma inglês usando uma cifra de César com um deslocamento de um ($K = 1$). A envia o seguinte texto cifrado legítimo:

```
nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz
```

B o decriptografa para produzir o seguinte texto claro:

```
mareseatoatsanddoeseatoatsandlittlelambseativy
```

Uma análise de freqüência simples confirma que essa mensagem tem o perfil do inglês comum. Por outro lado, se um oponente gerar a seguinte seqüência aleatória de letras:

```
zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq
```

isso será decriptografado como:

```
ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp
```

que não se encaixa no perfil do inglês comum.

Pode ser difícil determinar *automaticamente* se o texto cifrado que chega é decriptografado para um texto claro inteligível. Se o texto claro for, digamos, um arquivo de objeto binário ou raios-X digitalizados, pode ser difícil determinar um texto claro corretamente formado e, portanto, autêntico. Assim, um oponente poderia conseguir certo grau de perturbação simplesmente emitindo mensagens com conteúdo aleatório, alegando serem provenientes de um usuário legítimo.

Uma solução para esse problema é forçar o texto claro a ter alguma estrutura que seja facilmente reconhecida, mas que não possa ser repetida sem lançar mão da função de criptografia. Poderíamos, por exemplo, anexar um código de detecção de erros, também conhecido como seqüência de verificação de quadro (FCS — *Frame Check Sequence*) ou soma de verificação (*checksum*), a cada mensagem antes da criptografia, conforme ilustramos na Figura 11.2a. A prepara uma mensagem de texto claro M e depois a fornece como entrada para uma função F que produz uma FCS. A FCS é anexada a M e o bloco inteiro é então criptografado. No destino, B decriptografa o bloco recebido e trata os resultados como uma mensagem com uma FCS anexada. B aplica a mesma função F para tentar reproduzir a FCS. Se a FCS calculada for igual à FCS recebida, então a mensagem será considerada autêntica. É improvável que uma seqüência aleatória de bits qualquer apresente o relacionamento desejado.

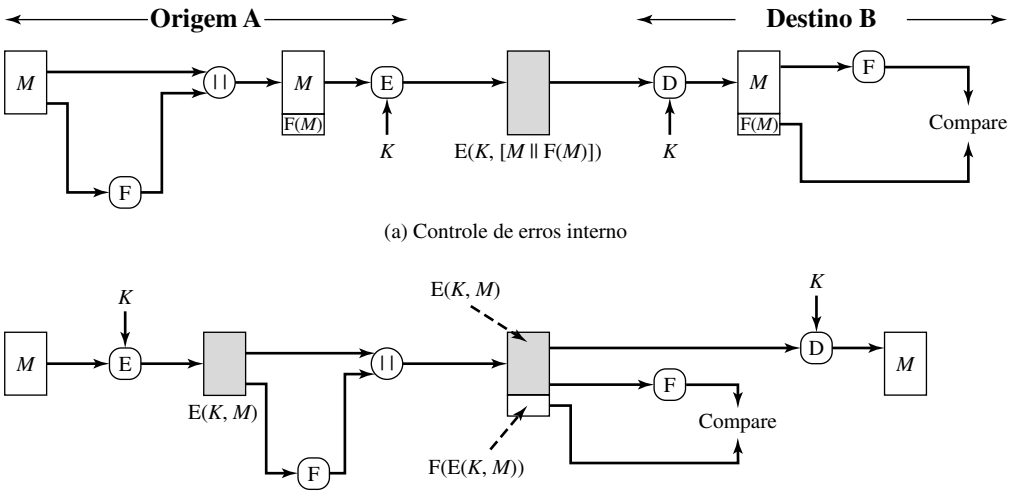


Figura 11.2 Controle de erros interno e externo.

Observe que a ordem em que as funções de FCS e criptografia são realizadas é crítica. A seqüência ilustrada na Figura 11.2a é citada em [DIFF79] como controle de erros interno, que os autores diferenciam do controle de erro externo (Figura 11.2b). Com o controle de erros interno, a autenticação é fornecida porque um oponente teria dificuldade em gerar texto cifrado que, quando decriptografado, tivesse bits de controle de erro válidos. Se, em vez disso, a FCS for o código externo, um oponente poderá construir mensagens com códigos de controle de erro válidos. Embora o oponente não possa saber como será o texto claro decriptografado, ele ainda pode criar perturbações.

Um código de controle de erros é apenas um exemplo; na verdade, qualquer tipo de estruturação acrescentada à mensagem transmitida serve para fortalecer a capacidade de autenticação. Essa estrutura é fornecida pelo uso de uma arquitetura de comunicações que consiste de protocolos em camadas. Como exemplo, considere a estrutura das mensagens transmitidas usando a arquitetura de protocolo TCP/IP. A Figura 11.3 mostra o formato de um segmento TCP, ilustrando o cabeçalho TCP. Agora, suponha que cada par de hosts compartilhe uma única chave secreta, de modo que todas as trocas entre um par de hosts usem a mesma chave, independentemente da aplicação. Então, poderíamos simplesmente criptografar todo o datagrama, exceto o cabeçalho IP (ver Figura 7.5). Novamente, se um oponente substituísse algum padrão de bits pelo segmento TCP criptografado, o texto claro resultante não incluiria um cabeçalho significativo. Nesse caso, o cabeçalho inclui não apenas uma soma de verificação (que abrange o cabeçalho), mas também outras informações úteis, como o número de seqüência. Como segmentos TCP sucessivos em determinada conexão são numerados seqüencialmente, a criptografia garante que um oponente não atrase, tire da ordem ou exclua quaisquer segmentos.

Criptografia de chave pública O uso direto da criptografia de chave pública (Figura 11.1b) oferece confidencialidade, mas não autenticação. A origem (A) utiliza a chave pública PU_b do destino (B) para criptografar M. Como somente B tem a chave privada correspondente PR_b, somente B pode decriptografar a mensagem. Esse esquema não oferece autenticação, pois qualquer oponente também pode usar a chave pública de B para criptografar uma mensagem, afirmando ser A.

Para oferecer autenticação, A utiliza sua chave privada para criptografar a mensagem, e B usa a chave pública de A para decriptografá-la (Figura 11.1c). Isso oferece autenticação usando o mesmo tipo de raciocínio do caso da criptografia simétrica: a mensagem precisa ter vindo de A, pois A é a única parte que possui PR_{aa} e, portanto, a única parte com a informação necessária para construir um texto cifrado que pode ser decriptografado com PU_a. Novamente, o mesmo raciocínio de antes se aplica: é preciso haver alguma estrutura interna no texto claro para que o receptor possa distinguir entre o texto claro bem formado e os bits aleatórios.

Supondo que exista tal estrutura, então o esquema da Figura 11.1c realmente oferece autenticação. Ele também fornece o que é conhecido como *assinatura digital*.[1] Somente A poderia ter construído o texto cifrado, pois somente A possui PR_a. Nem sequer B, o destinatário, poderia ter construído o texto cifrado. Portanto, se B estiver de posse do texto cifrado, B terá meios para provar que a mensagem deve ter vindo de A. Na verdade, A 'assinou' a mensagem ao usar sua chave privada para criptografá-la. Observe que esse esquema não oferece confidencialidade. Qualquer um de posse da chave pública de A pode decriptografar o texto cifrado.

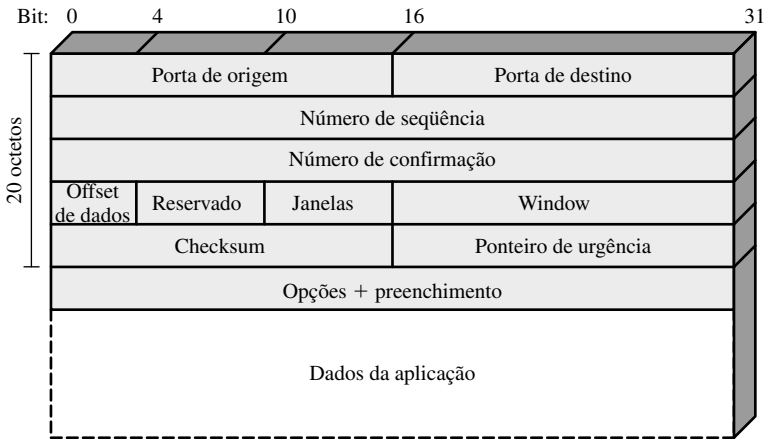


Figura 11.3 Segmento TCP.

1. Esse não é o modo como as assinaturas digitais são construídas, conforme veremos, mas o princípio é o mesmo.

Para oferecer confidencialidade e autenticação, A pode criptografar *M* primeiro usando sua chave privada, o que resulta na assinatura digital e, depois usando a chave pública de B, o que fornece a confidencialidade (Figura 11.1d). A desvantagem dessa abordagem é que o algoritmo de chave pública, que é complexo, precisa ser usado quatro vezes, em vez de duas, em cada comunicação.

A Tabela 11.1 resume as conseqüências de confidencialidade e autenticação dessas diversas técnicas de criptografia de mensagem.

Código de autenticação de mensagem

Uma técnica de autenticação alternativa envolve o uso de uma chave secreta para gerar um pequeno bloco de dados de tamanho fixo, conhecido como soma de verificação criptográfica ou MAC, que é anexado à mensagem. Essa técnica assume que duas partes na comunicação, digamos, A e B, compartilham a chave secreta *K*. Quando A tem uma mensagem para enviar a B, ele calcula o MAC como uma função da mensagem e da chave: MAC = C(*K*,*M*), onde

M = mensagem de entrada
C = função MAC
K = chave secreta compartilhada
MAC = código de autenticação de mensagem *(Message Authentication Code)*

A mensagem mais o MAC são transmitidos ao destinatário desejado. O destinatário realiza o mesmo cálculo sobre a mensagem recebida, usando a mesma chave secreta, para gerar um novo MAC. O MAC recebido é comparado com

Tabela 11.1 Conseqüências de confidencialidade e autenticação da criptografia da mensagem (ver Figura 11.1)

A → B: E(K, M)
- Oferece confidencialidade
  — somente A e B compartilham K
- Oferece um grau de autenticação
  — Só poderia vir de A
  — Não foi alterada em trânsito
  — Requer alguma formatação/redundância
- Não oferece assinatura
  — Receptor poderia forjar mensagem
  — Emissor poderia negar mensagem

(a) Criptografia simétrica

A → B: E(PU_b, M)
- Oferece confidencialidade
  — Somente B tem PR_b para decriptografar
- Não oferece autenticação
  — Qualquer parte poderia usar PU_b para criptografar a mensagem e afirmar ser A

(b) Criptografia de chave pública (simétrica): confidencialidade

A → B: E(PR_a, M)
- Oferece autenticação e assinatura
— Somente A tem PR_a para criptografar
— Não foi alterada em trânsito
— Requer alguma formatação/redundância
— Qualquer parte pode usar PU_a para verificar a assinatura

(c) Criptografia de chave pública: autenticação e assinatura

A → B: E(PU_b, E(PR_a, M))
- Oferece confidencialidade por causa de PU_b
- Oferece autenticação e assinatura por causa de PR_a

(d) Criptografia de chave pública: confidencialidade, autenticação e assinatura

o MAC calculado (Figura 11.4a). Se consideramos que somente o receptor e o emissor conhecem a identidade da chave secreta, e se o MAC recebido for igual ao MAC calculado, então:

1. O receptor tem garantias de que a mensagem não foi alterada. Se um atacante alterar a mensagem, mas não alterar o MAC, então o cálculo do MAC pelo receptor será diferente do MAC recebido. Como se considera que o atacante não conhece a chave secreta, ele não poderá alterar o MAC para corresponder às alterações na mensagem.
2. O receptor tem garantias de que a mensagem é do emissor declarado. Como ninguém mais sabe a chave secreta, ninguém mais poderia preparar uma mensagem com um MAC apropriado.
3. Se a mensagem inclui um número de seqüência (como o que é usado com HDLC, X.25 e TCP), então o receptor pode ter certeza da seqüência apropriada, pois um atacante não poderá alterar o número de seqüência.

Uma função MAC é semelhante à criptografia. Uma diferença é que o algoritmo MAC não precisa ser reversível, como precisaria ser para a decriptografia. Em geral, a função MAC é uma função muitos-para-um. O domínio da função consiste de mensagens de comprimento arbitrário, enquanto a imagem consiste de todos os MACs possíveis e todas as chaves possíveis. Se um MAC de n bits for usado, então existem 2^n MACs possíveis, enquanto existem N mensagens possíveis com $N >> 2^n$. Além disso, com uma chave de k bits, existem 2^k chaves possíveis.

Por exemplo, suponha que estejamos usando mensagens de 100 bits e um MAC de 10 bits. Então, existe um total de 2^{100} mensagens diferentes, mas somente 2^{10} MACs diferentes. Assim, em média, cada valor de MAC é gerado por um total de $2^{100}/2^{10} = 2^{90}$ mensagens diferentes. Se uma chave de 5 bits for usada, então existem $2^5 = 32$ mapeamentos diferentes do conjunto de mensagens para o conjunto de valores MAC.

Acontece que, devido às propriedades matemáticas da função de autenticação, ela é menos vulnerável a ser quebrada do que a criptografia.

O processo representado na Figura 11.4a oferece autenticação, mas não confidencialidade, pois a mensagem como um todo é transmitida às claras. A confidencialidade pode ser obtida realizando-se a criptografia da mensagem depois (Figura 11.4b) ou antes (Figura 11.4c) do algoritmo MAC. Nos dois casos, duas chaves distintas são necessárias, cada qual compartilhada pelo emissor e o receptor. No primeiro caso, o MAC é calculado tendo a mensagem como entrada, e depois é concatenado à mensagem. O bloco resultante é, então, criptografado. No segundo caso, a mensagem é criptografada antes. O MAC é então calculado a partir do texto cifrado resultante e é concatenado ao texto cifrado para formar o bloco transmitido. Normalmente, é preferível ligar a autenticação diretamente ao texto claro, de modo que o método da Figura 11.4b é utilizado.

Como a criptografia simétrica oferece a autenticação e como ela é bastante usada com produtos prontamente disponíveis, por que não simplesmente utilizá-la em vez de um código de autenticação de mensagem separado? [DAVI89] sugere três situações em que um código de autenticação de mensagens é usado:

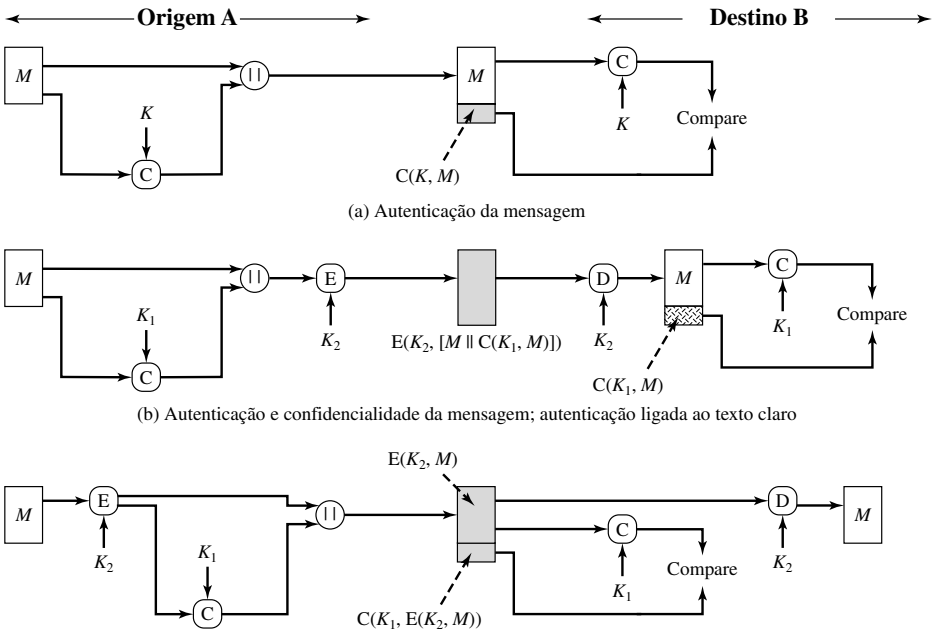


Figura 11.4 Usos básicos do código de autenticação de mensagens (MAC).

1. Existem várias aplicações em que a mesma mensagem é transmitida por broadcast a diversos destinos. Alguns exemplos são a notificação aos usuários de que a rede agora está indisponível ou um sinal de alarme em um centro de controle militar. É mais barato e mais confiável ter apenas um destino responsável por monitorar a autenticidade. Assim, a mensagem precisa ser enviada em texto claro com um código de autenticação de mensagem associado. O sistema responsável tem uma chave secreta e realiza autenticação. Se houver uma violação, os outros sistemas de destino são alertados por um alarme geral.
2. Outro cenário possível é uma troca em que um lado está muito ocupado e não dispõe de tempo para decriptografar todas as mensagens que chegam. A autenticação é executada em uma base seletiva, com as mensagens sendo escolhidas aleatoriamente para verificação.
3. A autenticação de um programa de computador em texto claro é um serviço atraente. O programa de computador pode ser executado sem ter de decriptografá-lo todas as vezes, o que seria um desperdício de recursos do processador. Porém, se um código de autenticação de mensagem estivesse anexado ao programa, ele poderia ser verificado sempre que fosse exigida uma garantia de integridade do programa.

Três outras razões podem ser acrescentadas, como segue:

4. Para algumas aplicações, pode não ser preciso manter as mensagens secretas, mas é importante autenticá-las. Um exemplo é o Simple Network Management Protocol Version 3 (SNMPv3), que separa as funções de confidencialidade e de autenticação. Para essa aplicação, normalmente é importante que um sistema gerenciado autentique as mensagens SNMP que chegam, particularmente se uma mensagem tiver um comando para trocar parâmetros no sistema gerenciado. Por outro lado, pode não ser necessário ocultar o tráfego SNMP.
5. A separação das funções de autenticação e confidencialidade proporciona flexibilidade no que diz respeito à arquitetura. Por exemplo, pode-se querer realizar a autenticação no nível da aplicação, mas oferecer confidencialidade em um nível mais baixo, como a camada de transporte.
6. Um usuário pode querer prolongar o período de proteção além do tempo de recebimento e ainda permitir o processamento do conteúdo da mensagem. Com a criptografia da mensagem, a proteção é perdida quando a mensagem é decriptografada, de modo que a mensagem é protegida contra modificações fraudulentas apenas em trânsito, mas não dentro do sistema de destino.

Finalmente, observe que o MAC não oferece uma assinatura digital, pois emissor e receptor compartilham a mesma chave.

A Tabela 11.2 resume as implicações de confidencialidade e autenticação das técnicas ilustradas na Figura 11.4.

Tabela 11.2 Usos básicos do código de autenticação de mensagens C (ver Figura 11.4)

$A \rightarrow B: M \| C(K, M)$
- Oferece autenticação
  — Somente A e B compartilham K

(a) Autenticação da mensagem

$A \rightarrow B: E(K_2, [M \| C(K, M)])$
- Oferece autenticação
  — Somente A e B compartilham K_1
- Oferece confidencialidade
  — Somente A e B compartilham K_2

(b) Autenticação e confidencialidade da mensagem: autenticação ligada ao texto claro

$A \rightarrow B: E(K_2, M) \| C(K_1, E(K_2, M))$
- Oferece autenticação
  — Usando K_1
- Oferece confidencialidade
  — Usando K_2

(c) Autenticação e confidencialidade da mensagem: autenticação ligada ao texto cifrado

Função de hash

Uma variação no código de autenticação de mensagens é a função de hash unidirecional. Assim como o código de autenticação de mensagens, uma função de hash aceita uma mensagem de comprimento variável M como entrada e produz uma saída de comprimento fixo, conhecida como **código de hash** H(M). Diferentemente de um MAC, um código de hash não usa uma chave, sendo uma função apenas da mensagem de entrada. O código de hash também é conhecido como **síntese de mensagem** ou **valor de hash**. O código de hash é uma função de todos os bits da mensagem e oferece uma capacidade de detecção de erros: uma mudança em qualquer bit ou bits na mensagem resulta em uma mudança no código de hash.

A Figura 11.5 ilustra diversas maneiras como um código de hash pode ser usado para oferecer autenticação de mensagens, como segue:

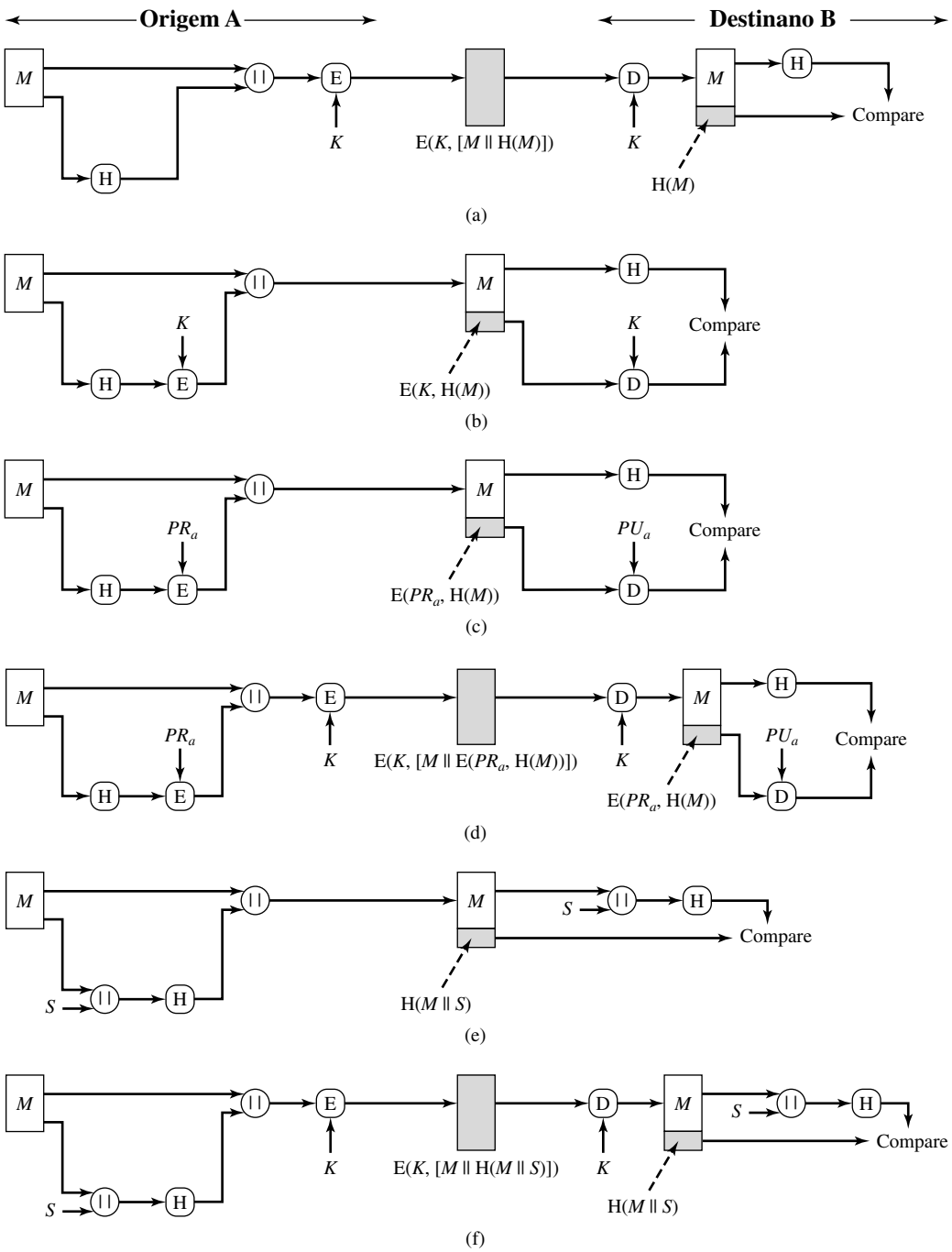


Figura 11.5 Usos básicos da função de hash.

a. A mensagem, mais o código de hash concatenado, é criptografada usando a criptografia simétrica. Essa maneira é estruturalmente idêntica à estratégia de controle de erros interno, mostrada na Figura 11.2a. A linha de raciocínio é a mesma: como somente A e B compartilham a chave secreta, a mensagem precisa ter vindo de A e não ter sido alterada. O código de hash oferece a estrutura ou a redundância exigida para conseguir a autenticação. Como a criptografia é aplicada à mensagem inteira e ao código de hash, a confidencialidade também é obtida.

b. Somente o código de hash é criptografado, usando a criptografia simétrica. Isso reduz a carga de processamento para aquelas aplicações que não exigem confidencialidade. Observe que a combinação de hashing e criptografia resulta em uma função geral que, de fato, é um MAC (Figura 11.4a). Ou seja, $E(K, H(M))$ é uma função da mensagem de comprimento variável M e uma chave secreta K, e produz uma saída de comprimento fixo que é segura contra um oponente que não conhece a chave secreta.

c. Somente o código de hash é criptografado, usando a criptografia de chave pública e a chave privada do emissor. Assim como em (b), isso oferece autenticação. Também fornece uma assinatura digital, pois somente o emissor poderia ter produzido o código de hash criptografado. Na verdade, essa é a essência da técnica de assinatura digital.

d. Se forem desejadas confidencialidade e assinatura digital, então a mensagem mais o código de hash criptografado com a chave privada podem ser criptografados usando a chave secreta simétrica. Essa é uma técnica comum.

e. É possível usar uma função de hash, mas não a criptografia para autenticação da mensagem. A técnica considera que as duas partes que se comunicam compartilham um valor secreto S comum. A calcula o valor de hash sobre a concatenação de M e S, e anexa o valor de hash resultante a M. Como B possui S, ele pode recalcular o valor do hash para verificação. Como o valor secreto propriamente dito não é enviado, um oponente não pode modificar uma mensagem interceptada e não pode gerar uma mensagem falsa.

f. A confidencialidade pode ser acrescentada à técnica de (e), criptografando a mensagem inteira mais o código de hash.

Quando a confidencialidade não é exigida, os métodos (b) e (c) têm uma vantagem sobre aqueles que criptografam a mensagem inteira, pois é necessária menos computação. Apesar disso, tem havido interesse crescente em técnicas que evitam a criptografia (Figura 11.5e). Vários motivos para esse interesse são apontados em [TSUD92]:

- O software de criptografia é relativamente lento. Mesmo que a quantidade de dados a ser criptografada por mensagem seja pequena, pode haver um fluxo constante de mensagens para dentro e para fora de um sistema.

Tabela 11.3 Usos básicos da função de hash H (ver Figura 11.5).

$A \rightarrow B$: $E(K, [M \| H(M)])$ • Oferece confidencialidade Somente A e B compartilham K • Oferece autenticação — $H(M)$ é protegido criptograficamente	$A \rightarrow B$: $E(K, [M \| E(PR_a, H(M))])$ • Oferece autenticação e assinatura digital • Oferece confidencialidade — Somente A e B compartilham K
(a) Criptografia de mensagem mais código de hash	(d) Criptografia do resultado de (c) — chave secreta compartilhada
$A \rightarrow B$: $M \| E(K, H(M))$ • Oferece autenticação — $H(M)$ é criptograficamente protegido	$A \rightarrow B$: $M \| H(M \| S)$ • Oferece autenticação — Somente A e B compartilham S
(b) Criptografia do código de hash — chave secreta compartilhada	(e) Cálculo do código de hash da mensagem mais o valor secreto
$A \rightarrow B$: $M \| E(PR_a, H(M))$ • Oferece autenticação e assinatura digital — $H(M)$ é criptograficamente protegido — Somente A poderia criar $E(PR_a, H(M))$	$A \rightarrow B$: $E(K, [M \| H(M \| S])$ • Oferece autenticação — Somente A e B compartilham S • Oferece confidencialidade — Somente A e B compartilham K
(c) Criptografia do código de hash — chave privada do emissor	(f) Encrypt result of (e) Criptografia do resultado de (e)

- Os custos de hardware de criptografia não são insignificantes. Implementações de chip de baixo custo do DES estão disponíveis, mas o custo aumenta se todos os nós em uma rede tiverem de ter essa capacidade.
- O hardware de criptografia é otimizado para grandes tamanhos de dados. Para pequenos blocos de dados, uma grande proporção do tempo é gasta no overhead de inicialização/invocação.
- Os algoritmos de criptografia podem ser protegidos por patentes. Por exemplo, até que a patente expirasse, o RSA teve ser licenciado, aumentando o custo.

A Tabela 11.3 resume as implicações de confidencialidade e autenticação das abordagens ilustradas na Figura 11.5. Em seguida, examinamos MACs e códigos de hash com mais detalhes.

11.3 CÓDIGOS DE AUTENTICAÇÃO DE MENSAGEM

Um MAC, também conhecido como soma de verificação (*cheksum*) criptográfica, é gerado por uma função C na forma:

$$\text{MAC} = \text{C}(K, M)$$

onde M é uma mensagem de comprimento variável, K é uma chave secreta compartilhada apenas pelo emissor e receptor e $\text{C}(K,M)$ é um autenticador de comprimento fixo. O MAC é anexado à mensagem na origem em um momento em que a mensagem é suposta ou conhecida como sendo correta. O receptor autentica essa mensagem recalculando o MAC.

Nesta seção, analisamos os requisitos para a função C e depois examinamos um exemplo específico. Outros exemplos serão explicados no Capítulo 12.

Requisitos para MACs

Quando uma mensagem inteira é criptografada por confidencialidade, usando tanto criptografia simétrica quanto assimétrica, a segurança do esquema geralmente depende do comprimento em bits da chave. Exceto por algum ponto fraco no algoritmo, o oponente precisa recorrer a um ataque por força bruta usando todas as chaves possíveis. Na média, esse ataque exigirá $2^{(k-1)}$ tentativas para uma chave de k bits. Em particular, para um ataque apenas de texto cifrado, o oponente, dado o texto cifrado C, realizaria $P_i = \text{D}(K_i, C)$ para todos os valores de chave possíveis K_i até que um P_i produzido correspondesse à forma de texto claro aceitável.

No caso de um MAC, as considerações são inteiramente diferentes. Em geral, a função MAC é uma função muitos-para-um, devido à natureza muitos-para-um da função. Usando os métodos de força bruta, como um oponente tentaria descobrir uma chave? Se a confidencialidade não for empregada, o oponente terá acesso a mensagens de texto claro e seus MACs associados. Suponha que $k > n$; ou seja, suponha que o tamanho da chave seja maior que o tamanho do MAC. Então, dado um M_1 e MAC_1 conhecidos, com $MAC_1 = \text{C}(K, M_1)$, o criptoanalista pode realizar $MAC_i = \text{C}(K_i, M_1)$ para todos os valores de chave possíveis K_i. É garantido que pelo menos uma chave produza uma combinação correta $MAC_i = MAC_1$. Observe que um total de 2^k MACs serão produzidos, mas que existem apenas $2^n < 2^k$ diferentes valores de MAC. Assim, diversas chaves produzirão o MAC correto e o oponente não terá como saber qual é a chave correta. Na média, um total de $2^k/2^n = 2^{(k-n)}$ chaves produzirão uma combinação adequada. Assim, o oponente precisará repetir o ataque:

- **Rodada 1**

  Dados: $M_1, MAC_1 = \text{C}(K, M_1)$

  Calcule $MAC_1 = \text{C}(K_i, M_1)$ para todas as 2^k chaves
  Número de combinações corretas $\approx 2^{(k-n)}$

- **Rodada 2**

  Dados: $M_2, MAC_2 = \text{C}(K, M_2)$

  Calcule MAC_1 para as $2^{(k-n)}$ chaves resultantes da Rodada 1
  Número de combinações corretas $\approx 2^{(k-2\times n)}$

e assim por diante. Em média, α rodadas serão necessárias se $k = \alpha \times n$. Por exemplo, se uma chave de 80 bits for usada e o MAC tiver 32 bits de extensão, então a primeira rodada produzirá cerca de 2^{48} chaves possíveis. A segunda limitará as chaves possíveis a cerca de 2^{16} possibilidades. E a terceira deverá produzir uma única chave, que precisará ser aquela usada pelo emissor.

Se o tamanho da chave for menor ou igual ao tamanho do MAC, então é provável que uma primeira rodada produza uma única combinação correta. É possível que mais de uma chave produza essa combinação, quando o oponente precisaria realizar o mesmo teste em um novo par (mensagem, MAC).

Assim, uma tentativa por força bruta para descobrir a chave de autenticação não é menos trabalhosa, e pode ser ainda mais difícil do que a exigida para descobrir uma chave de decriptografia com o mesmo tamanho. Porém, outros ataques que não exijam a descoberta da chave são possíveis.

Considere o seguinte algoritmo MAC. Seja $M = (X_1 \| X_2 \| \ldots \| X_m)$ uma mensagem tratada como uma concatenação dos blocos de 64 bits X_i. Então defina

$$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$$
$$C(K, M) = E(K, \Delta(M))$$

onde $\oplus$ é a operação OU exclusivo (XOR) e o algoritmo de criptografia é DES no modo de livro de códigos eletrônico (*Electronic Codebook*). Assim, o comprimento da chave é de 56 bits e o comprimento do MAC é de 64 bits. Se um oponente observar $\{M \| C(K, M)\}$, uma tentativa de força bruta para determinar K exigirá pelo menos 2^{56} criptografias. Mas o oponente pode atacar o sistema substituindo X_1 a X_{m-1} por quaisquer valores desejados Y_1 a Y_{m-1} e substituindo X_m por Y_m, onde Y_m é calculado da seguinte forma:

$$Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$$

O oponente, agora, pode concatenar a nova mensagem, que consiste em Y_1 até Y_m, com o MAC original para formar uma mensagem que será aceita como autêntica pelo receptor. Com essa tática, qualquer mensagem de tamanho $64 \times (m - 1)$ bits poderá ser inserida fraudulentamente.

Assim, na avaliação da segurança de uma função MAC, precisamos considerar os tipos de ataques que podem ser montados contra ela. Com isso em mente, vamos declarar os requisitos para a função. Suponha que um oponente conheça a função MAC C, mas não conheça K. Então, a função MAC deverá satisfazer os seguintes requisitos:

1. Se um oponente observar M e $C(K, M)$, deverá ser computacionalmente inviável para o oponente construir uma mensagem M' tal que $C(K, M') = C(K, M)$.
2. $C(K, M)$ deve ser uniformemente distribuída no sentido de que, para mensagens escolhidas aleatoriamente, M e M', a probabilidade de que $C(K, M) = C(K, M')$ é 2^{-n}, onde n é o número de bits no MAC.
3. Seja M' igual a alguma transformação conhecida sobre M. Isto é, $M' = f(M)$. Por exemplo, f pode envolver a inversão de um ou mais bits específicos. Nesse caso, $\Pr[C(K, M) = C(K, M')] = 2^{-n}$.

O primeiro requisito diz respeito ao exemplo anterior, em que um oponente é capaz de construir uma nova mensagem para combinar com determinado MAC, embora o oponente não saiba e não descubra a chave. O segundo requisito lida com a necessidade de impedir a ocorrência de um ataque por força bruta baseado em um texto claro escolhido. Ou seja, se supusermos que o oponente não conhece K, mas tem acesso à função MAC e pode apresentar mensagens para geração de MAC, então o oponente pode testar várias mensagens até encontrar uma que combine corretamente com determinado MAC. Se a função de MAC tiver distribuição uniforme, então o método de força bruta exigirá, em média, $2^{(n-1)}$ tentativas antes que uma mensagem que se ajuste a determinado MAC seja encontrada.

O requisito final impõe que o algoritmo de autenticação não deve ser mais fraco em relação a certas partes ou bits da mensagem do que no restante dela. Se não fosse assim, então um oponente que tivesse M e $C(K, M)$ poderia tentar variações sobre M nos 'pontos fracos' conhecidos, provavelmente obtendo sucesso mais cedo na produção de uma nova mensagem que combinasse corretamente com o MAC antigo.

Código de autenticação de mensagens baseado no DES

O algoritmo de autenticação de dados (*DAA — Data Authentication Algorithm*), baseado no DES, tem sido um dos mais utilizados MACs por diversos anos. O algoritmo é uma publicação FIPS (FIPS PUB 113) e um padrão ANSI (X9.17). Porém, conforme explicamos no Capítulo 12, foram descobertos pontos fracos na segurança desse algoritmo, e ele está sendo substituído por algoritmos mais recentes e mais fortes.

O algoritmo pode ser definido como usando o modo de encadeamento de cifra de bloco (CBC — *Cipher Block Chaining*) de operação do DES (Figura 6.4) com um vetor de inicialização de zeros. Os dados (por exemplo, mensagem, registro, arquivo ou programa) a serem autenticados são agrupados em blocos contíguos de 64 bits: $D_1, D_2, \ldots, D_N$. Se necessário, o bloco final é preenchido à direita com zeros, para formar um bloco de 64 bits completo. Usando o algoritmo de criptografia DES, E, e uma chave secreta, K, um código de autenticação de dados (*DAC — Data Authentication Code*) é calculado da seguinte forma (Figura 11.6):

$$
\begin{aligned}
O_1 &= \mathrm{E}(K, D_1) \\
O_2 &= \mathrm{E}(K, [D_2 \oplus O_1]) \\
O_3 &= \mathrm{E}(K, [D_3 \oplus O_2]) \\
&\vdots \\
O_N &= \mathrm{E}(K, [D_N \oplus O_{N-1}])
\end{aligned}
$$

O DAC consiste no bloco inteiro O_N ou nos M bits mais à esquerda do bloco, com $16 \leq M \leq 64$.

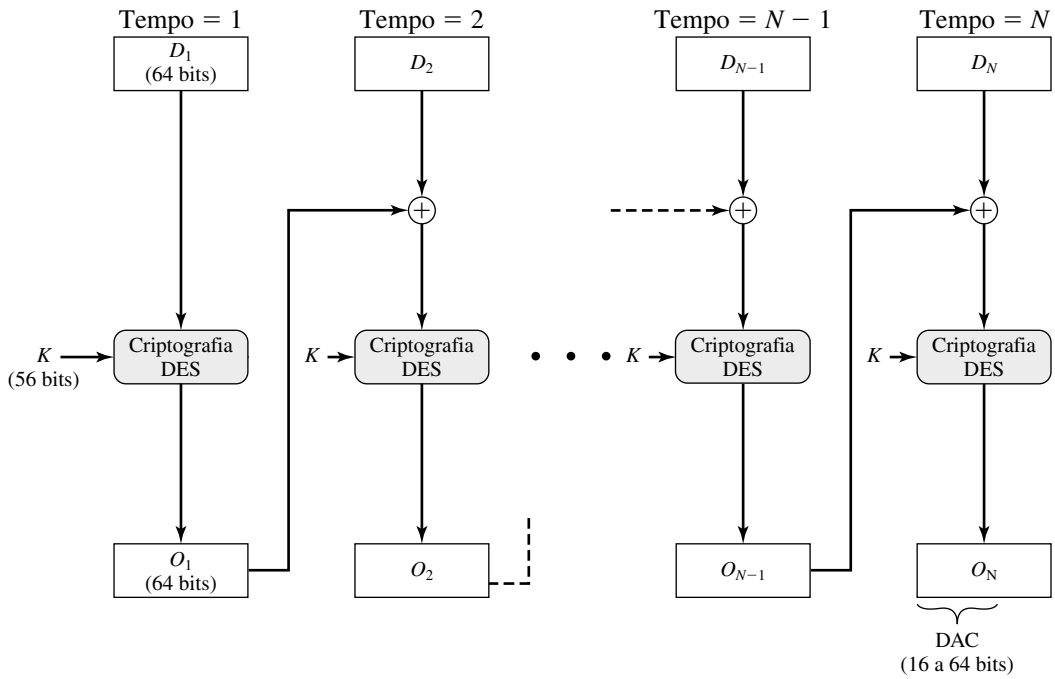


Figura 11.6 Algoritmo de autenticação de dados (FIPS PUB 113).

11.4 FUNÇÕES DE HASH

Um valor de hash h é gerado por uma função H na forma

$$h = \mathrm{H}(M)$$

onde M é uma mensagem de comprimento variável e $\mathrm{H}(M)$ é o valor de hash de comprimento fixo. O valor de hash é anexado à mensagem na origem, em um momento em que a mensagem é considerada ou reconhecida como correta. O receptor autentica essa mensagem recalculando o valor de hash. Como a própria função de hash não é considerada secreta, algum meio é necessário para proteger o valor de hash (Figura 11.5).

Começaremos examinando os requisitos para uma função de hash a ser usada para autenticação de mensagem. Como as funções de hash normalmente são muito complexas, é útil examinar algumas funções de hash muito simples para ter uma idéia das questões envolvidas. Depois, vemos as várias abordagens para o projeto da função de hash.

Requisitos para uma função de hash

A finalidade de uma função de hash é produzir uma 'impressão digital' de um arquivo, mensagem ou outro bloco de dados. Para ser útil para autenticação de mensagens, uma função de hash H precisa ter as seguintes propriedades (adaptadas da lista em [NECH92]):

1. H pode ser aplicado a um bloco de dados de qualquer tamanho.
2. H produz uma saída de comprimento fixo.

3. H(x) é relativamente fácil de calcular para qualquer x, tornando as implementações de hardware e software práticas.
4. Para qualquer valor h dado, é computacionalmente inviável encontrar x tal que $H(x) = h$. Isso às vezes é conhecido na literatura como **resistência à primeira inversão ou propriedade unidirecional** (one-way property).
5. Para qualquer bloco dado x, é computacionalmente inviável encontrar $y \neq x$ tal que $H(y) = H(x)$. Isso às vezes é conhecido como **resistência à segunda inversão ou resistência fraca a colisões** (weak collision resistance).
6. É computacionalmente inviável encontrar qualquer par (x,y) tal que $H(x) = H(y)$. Isso às vezes é conhecido como **resistência a colisões ou resistência forte a colisões** (strong collision resistance).[2]

As três primeiras propriedades são requisitos para a aplicação prática de uma função de hash para autenticação de mensagens.

A quarta propriedade, a resistência à primeira inversão, afirma que é fácil gerar um código dada uma mensagem, mas praticamente impossível gerar uma mensagem dado um código. Essa propriedade é importante se a técnica de autenticação envolver o uso de um valor secreto (Figura 11.5e). O próprio valor secreto não é enviado; porém, se a função de hash não for unidirecional, um atacante poderá facilmente descobrir o valor secreto: se o atacante puder observar ou interceptar uma transmissão, o atacante obterá a mensagem M e o código de hash $C = H(S_{AB}\|M)$. O atacante, então, inverte a função de hash para obter $S_{AB}\|M = H^{-1}(C)$ Como o atacante agora tem M e $S_{AB}\|M$, é uma questão trivial recuperar S_{AB}.

A quinta propriedade garante que dada uma mensagem, não poderá ser encontrada uma mensagem alternativa que gere um mesmo valor de hash. Isso evita a falsificação quando um valor de hash criptografado for usado (Figuras 11.5b e c). Para esses casos, o oponente pode ler a mensagem e, portanto, gerar seu código de hash. Porém, como o oponente não tem a chave secreta, ele não deverá ser capaz de alterar a mensagem sem ser descoberto. Se essa propriedade não fosse verdadeira, um atacante seria capaz de concluir a seguinte seqüência: primeiro, observar ou interceptar uma mensagem mais seu valor de hash criptografado; segundo, gerar um valor de hash não criptografado a partir da mensagem; terceiro, gerar uma mensagem alternativa com o mesmo valor de hash.

A sexta propriedade refere-se à resistência da função de hash a um tipo de ataque conhecido como o ataque de aniversário (*birthday attack*), que examinaremos em breve.

Funções de hash simples

Todas as funções de hash operam usando os seguintes princípios gerais. A entrada (mensagem, arquivo etc.) é vista como uma seqüência de blocos de n bits. A entrada é processada um bloco de cada vez, em um padrão iterativo para produzir uma função de hash de n bits.

Uma das funções de hash mais simples é o OU exclusivo (XOR) bit a bit de cada bloco. Isso pode ser expresso da seguinte forma:

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

onde

$$\begin{aligned} C_i &= i\text{-ésimo bit do valor de hash, } 1 \leq i \leq n \\ m &= \text{número de blocos de } n \text{ bits na entrada} \\ b_{ij} &= i\text{-ésimo bit no } j\text{-ésimo bloco} \\ \oplus &= \text{operação XOR} \end{aligned}$$

Essa operação produz uma paridade simples para cada posição de bit e é conhecida como uma verificação de redundância longitudinal. Ela é razoavelmente eficaz para dados aleatórios, como uma verificação de integridade de dados. Cada valor de hash de n bits é igualmente provável. Assim, a probabilidade de que um erro de dados resulte em um valor de hash inalterado é 2^{-n}. Com dados formatados mais previsivelmente, a função é menos eficaz. Por exemplo, na maioria dos arquivos de texto normais, o bit mais significativo de cada octeto é sempre zero. Assim, se um valor de hash de 128 bits for usado, em vez de uma eficácia de 2^{-128}, a função de hash desse tipo de dado possui uma eficácia de 2^{-112}.

Um modo simples de melhorar as coisas é realizar um deslocamento (*shift*) circular de um bit, ou rotação, sobre o valor de hash após cada bloco ser processado. O procedimento pode ser resumido da seguinte forma:

2. Infelizmente, esses termos não são usados de modo consistente. Termos alternativos usados na literatura são *função de hash unidirecional:* (propriedades 4 e 5); *função de hash à prova de colisões*: (propriedades 4, 5 e 6); *função de hash unidirecional fraca*: (propriedades 4 e 5); *função de hash unidirecional forte*: (propriedades 4, 5 e 6). O leitor deve ter cuidado ao consultar a literatura para determinar o significado dos termos específicos utilizados.

1. Inicialmente, defina o valor de hash de *n* bits como zero.
2. Processe cada bloco sucessivo de *n* bits de dados da seguinte forma:
   a. Faça a rotação do valor de hash atual para a esquerda em um bit.
   b. Realize o OU EXCLUSIVO (XOR) do bloco com o valor do hash.

Isso tem o efeito de tornar aleatória a entrada de forma mais completa e contornar quaisquer regularidades que apareçam na entrada. A Figura 11.7 ilustra esses dois tipos de funções de hash para valores de hash de 16 bits.

Embora o segundo procedimento ofereça uma boa medida de integridade de dados, ele é praticamente inútil para segurança de dados quando um código de hash criptografado é usado com uma mensagem de texto claro, como nas Figuras 11.5b e c. Dada uma mensagem, é uma questão fácil produzir uma nova mensagem que gere aquele código de hash: simplesmente prepare a mensagem alternativa desejada e, depois, anexe um bloco de *n* bits que force a nova mensagem e o bloco a produzirem o código de hash desejado.

Embora um XOR simples ou XOR rotacionado (RXOR) seja insuficiente se somente o código de hash estiver criptografado, você ainda pode ter a sensação de que essa função simples pode ser útil quando tanto a mensagem quanto o código de hash estiverem criptografados (Figura 11.5a). Mas é preciso ter cuidado. Uma técnica proposta originalmente pelo National Bureau of Standards usava o XOR simples, aplicado a blocos de 64 bits da mensagem e depois, uma criptografia da mensagem inteira, que usava o modo de encadeamento de cifra de bloco (CBC). Podemos definir o esquema da seguinte forma: dada uma mensagem que consiste em uma seqüência de blocos de 64 bits X_1, X_2, ..., X_N, defina o código de hash C como o XOR bloco a bloco de todos os blocos e anexe o código de hash como o bloco final:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Em seguida, criptografe a mensagem inteira, mais o código de hash, usando o modo CBC para produzir a mensagem criptografada $X_1, X_2, ..., X_{N+1}$. [JUEN85] aponta várias maneiras pelas quais o texto cifrado dessa mensagem

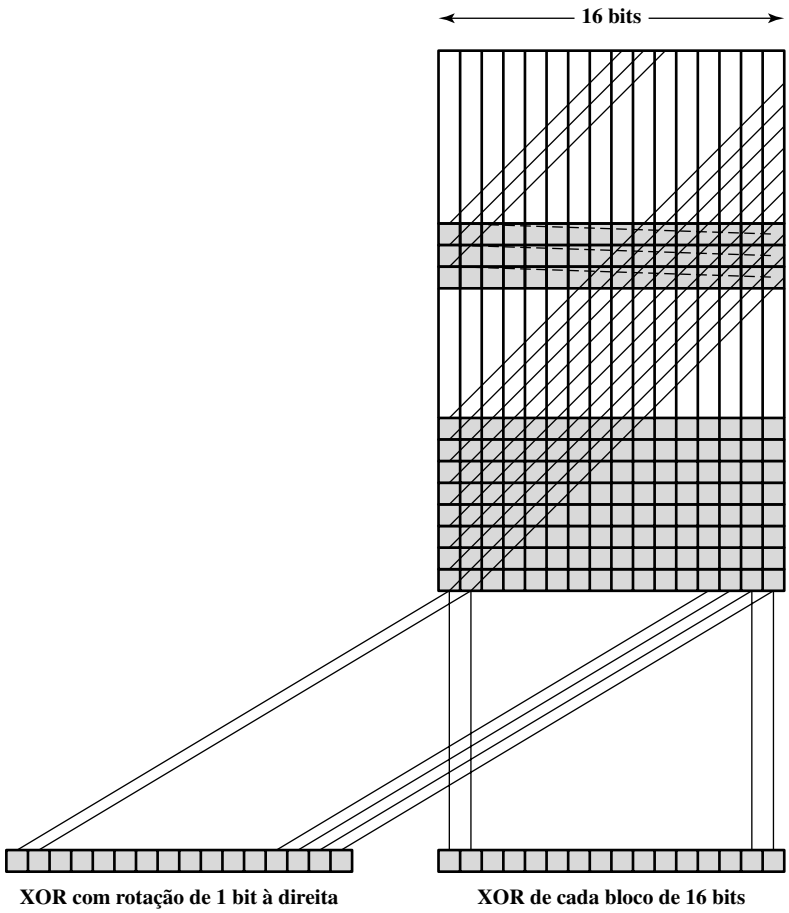


Figura 11.7 Duas funções de hash simples.

pode ser manipulado de modo que não seja detectável pelo código de hash. Por exemplo, pela definição do CBC (Figura 6.4), temos

$$\begin{aligned} X_1 &= IV \oplus \mathrm{D}(K, Y_1) \\ X_i &= Y_{i-1} \oplus \mathrm{D}(K, Y_i) \\ X_{N+1} &= Y_N \oplus \mathrm{D}(K, Y_{N+1}) \end{aligned}$$

Mas X_{N+1} é o código de hash:

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \cdots \oplus X_N \\ &= [IV \oplus \mathrm{D}(K, Y_1)] \oplus [Y_1 \oplus \mathrm{D}(K, Y_2)] \oplus \cdots \oplus [Y_{N-1} \oplus \mathrm{D}(K, Y_N)] \end{aligned}$$

Como na equação anterior pôde ser feito o OU Exclusivo dos termos em qualquer ordem, segue-se que o código de hash não mudaria se os blocos de texto cifrado fossem permutados.

Ataques de aniversário

Suponha que um código de hash de 64 bits seja utilizado. Pode-se pensar que isso é bastante seguro. Por exemplo, se um código de hash criptografado C for transmitido com a mensagem não criptografada correspondente M (Figura 11.5b ou c), então um oponente precisaria encontrar um M' tal que $\mathrm{H}(M') = \mathrm{H}(M)$ para substituir outra mensagem e enganar o receptor. Na média, o oponente teria de testar cerca de 2^{63} mensagens para encontrar uma que combinasse com o código de hash da mensagem interceptada [ver Apêndice 11A, Equação (11.1)].

```
Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}
Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}
Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}
responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}
in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}
to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the
market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the
{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}
of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}
as proof of identity. An order with his signature, which is {appended / attached}
{authorizes / allows} you to charge the cost to this company at the {above / head office}
address. We {fully / --} expect that our {level / volume} of orders will increase in
the {following / next} year and {trust / hope} that the new appointment will {be / prove}
{advantageous / an advantage} to both our companies.
```

Figura 11.8 Uma carta com 2^{37} variações [DAVI89].

Porém, um tipo diferente de ataque é possível, com base no paradoxo do aniversário (Apêndice 11A). Yuval propôs a seguinte estratégia [YUVA79]:

1. A origem, A, é preparada para 'assinar' uma mensagem anexando o código de hash apropriado de m bits e criptografando esse código de hash com a chave privada de A (Figura 11.5c).
2. O oponente gera $2^{m/2}$ variações sobre a mensagem, todas transportando basicamente o mesmo significado. O oponente prepara um número igual de mensagens, todas variações da mensagem fraudulenta a ser substituída pela mensagem real.
3. Os dois conjuntos de mensagens são comparados para encontrar um par de mensagens que produza o mesmo código de hash. A probabilidade de sucesso, pelo paradoxo do aniversário, é maior do que 0,5. Se nenhuma combinação for encontrada, mensagens válidas e fraudulentas adicionais serão geradas até que seja encontrada uma combinação.
4. O oponente oferece a variação válida a A para assinatura. Essa assinatura pode então ser anexada à variação fraudulenta para transmissão ao destinatário. Como as duas variações têm o mesmo código de hash, elas produzirão a mesma assinatura; o oponente tem garantias do sucesso, embora a chave de criptografia não seja conhecida.

Assim, se um código de hash de 64 bits for usado, o grau de esforço exigido será apenas na ordem de 2^{32} [ver Apêndice 11A, Equação (11.7)].

A geração de muitas variações que transmitem o mesmo significado não é difícil. Por exemplo, o oponente poderia inserir uma série de pares de caracteres 'espaço-espaço-backspace' entre as palavras em todo o documento. As variações poderiam então ser geradas substituindo 'espaço-espaço-backspace' nas instâncias selecionadas. Como alternativa, o oponente poderia simplesmente reformular a mensagem, mas manter o significado. A Figura 11.8 [DAVI89] contém um exemplo.

A conclusão a ser tirada disso é que o tamanho do código de hash deve ser substancial. Explicaremos isso com mais detalhes na Seção 11.5.

Técnicas de encadeamento em bloco

Diversas propostas foram feitas para funções de hash baseadas no uso de uma técnica de encadeamento de cifra de blocos, mas sem a chave secreta. Uma dessas primeiras propostas foi a de Rabin [RABI78]. Divida uma mensagem M em blocos de tamanho fixo $M_1, M_2, ..., M_N$ e use um sistema de criptografia simétrica, como DES, para calcular o código de hash G da seguinte forma:

$$\begin{aligned} H_0 &= \text{valor inicial} \\ H_i &= \mathrm{E}(M_i, H_{i-1}) \\ G &= H_N \end{aligned}$$

Isso é semelhante à técnica CBC, mas, nesse caso, não existe chave secreta. Assim como em qualquer código de hash, esse esquema está sujeito ao ataque de aniversário, e se o algoritmo de criptografia for DES e apenas um código de hash de 64 bits for produzido, então o sistema será vulnerável.

Além disso, outra versão do ataque de aniversário pode ser usada, mesmo que o oponente tenha acesso a apenas uma mensagem e sua assinatura válida, e não possa obter várias assinaturas. Eis o cenário; consideramos que o oponente intercepta uma mensagem com uma assinatura na forma de um código de hash criptografado e que o código de hash não criptografado tem m bits de extensão:

1. Use o algoritmo definido no início desta subseção para calcular o código de hash não criptografado G.
2. Construa qualquer mensagem desejada na forma $Q_1, Q_2, \ldots, Q_{N-2}$.
3. Calcule $H_i = \mathrm{E}(Q_i, H_{i-1})$ para $1 \leq i \leq (N - 2)$.
4. Gere $2^{m/2}$ blocos aleatórios; para cada bloco X, calcule $\mathrm{E}(X, H_{N-2})$. Gere $2^{m/2}$ blocos aleatórios adicionais; para cada bloco Y, calcule D(Y, G), onde D é a função de decriptografia correspondente a E.
5. Com base no paradoxo do aniversário, haverá com alta probabilidade um X e Y tais que $\mathrm{E}(X, H_{N-2}) = \mathrm{D}(Y, G)$.
6. Forme a mensagem $Q_1, Q_2, ..., Q_{N-2}, X, Y$. Essa mensagem tem o código de hash G e, portanto, pode ser usada com a assinatura criptografada interceptada.

Essa forma de ataque é conhecida como **ataque meet-in-the-middle**. Diversos pesquisadores propuseram melhorias com a finalidade de fortalecer a técnica básica de encadeamento de blocos. Por exemplo, Davies e Price [DAVI89] descrevem a seguinte variação:

$$H_i = \mathrm{E}(M_\mathrm{i}, \mathrm{H}_{i-1}) \oplus H_{i-1}$$

Outra variação, proposta em [MEYE88],

$$H_i = \mathrm{E}(H_{i-1}, M_\mathrm{i}) \oplus M_\mathrm{i}$$

Porém, esses dois esquemas têm se mostrado vulneráveis a uma série de ataques [MIYA90]. Em geral, pode-se demonstrar que alguma forma de ataque de aniversário terá sucesso contra qualquer esquema de hash envolvendo o uso do encadeamento de cifra de blocos sem uma chave secreta, desde que o código de hash resultante seja pequeno o suficiente (por exemplo, 64 bits ou menos) ou que um código de hash maior possa ser decomposto em subcódigos independentes [JUEN87].

Assim, é preciso dar atenção à descoberta de outras técnicas para o hash. Muitas delas também possuem pontos fracos [MITC92]. Examinamos duas funções de hash fortes no Capítulo 12.

11.5 SEGURANÇA DAS FUNÇÕES DE HASH E MACs

Assim como fizemos com as criptografias simétrica e de chave pública, podemos agrupar os ataques sobre as funções de hash e MACs em duas categorias: ataques por força bruta e criptoanálise.

Ataques por força bruta

A natureza dos ataques por força bruta difere um pouco das funções de hash e MACs.

Funções de hash

A força de uma função de hash contra os ataques por força bruta depende apenas do tamanho do código de hash produzido pelo algoritmo. Lembre-se, pela nossa explicação das funções de hash, que existem três propriedades desejáveis:

- **Resistência à primeira inversão ou propriedade unidirecional:** Para qualquer código h, é computacionalmente inviável encontrar x tal que $\mathrm{H}(x) = h$.
- **Resistência à segunda inversão ou resistência fraca a colisões:** Para qualquer bloco x, é computacionalmente inviável encontrar $y \neq x$ com $\mathrm{H}(y) = \mathrm{H}(x)$.
- **Resistência a colisões ou resistência forte a colisões:** É computacionalmente inviável encontrar qualquer par (x, y) tal que $\mathrm{H}(x) = \mathrm{H}(y)$.

Para um código de hash de tamanho n, o nível de esforço exigido, conforme vimos, é proporcional ao seguinte:

Resistência à primeira inversão	2^n
Resistência à segunda inversão	2^n
Resistência a colisões	$2^{n/2}$

Se a resistência a colisões for exigida (e isso é desejável para um código de hash seguro de uso geral), então o valor $2^{n/2}$ determina a força do código de hash contra os ataques por força bruta. Oorschot e Wiener [VANO94] apresentaram um projeto de uma máquina de busca de colisão para MD5 de US$ 10 milhões, que tem um tamanho de hash de 128 bits, que poderia encontrar uma colisão em 24 dias. Assim, um código de 128 bits pode ser visto como inadequado. O próximo passo à frente, se um código de hash for tratado como uma seqüência de 32 bits, é um comprimento de hash de 160 bits. Com um comprimento de 160 bits, a mesma máquina de busca exigiria mais de quatro mil anos para encontrar uma colisão. Porém, até mesmo um comprimento de 160 bits agora é considerado fraco. Voltaremos a esse assunto no Capítulo 12.

Códigos de autenticação de mensagens (MACs) Um ataque por força bruta a um MAC é uma tarefa mais difícil, pois requer pares mensagem-MAC conhecidos. Vejamos por que isso acontece. Para atacar um código de hash, podemos proceder da seguinte maneira. Dada uma mensagem fixa x com código de hash com n bits $h = \mathrm{H}(x)$, um método por força bruta para encontrar uma colisão é escolher uma string de bits aleatória y e verificar se $\mathrm{H}(y) = \mathrm{H}(x)$. O atacante pode fazer isso repetidamente off-line. Se um ataque off-line pode ser usado sobre um algoritmo MAC, isso depende do tamanho relativo da chave e do MAC.

Para prosseguir, precisamos declarar a propriedade de segurança desejada de um algoritmo MAC, que pode ser expressa da seguinte forma:

- **Resistência a computação:** Dado um ou mais pares de texto-MAC $[x_i, C(K, x_i)]$, é computacionalmente inviável calcular qualquer par texto-MAC $[x, C(K, x)]$ para qualquer nova entrada $x \neq x_i$.

Em outras palavras, o atacante gostaria de ter o código MAC válido para determinada mensagem x. Existem duas linhas de ataque possíveis: atacar todo o conjunto de valores possíveis de chave e atacar o valor MAC. Examinamos uma de cada vez.

Se um atacante puder determinar a chave MAC, então será possível gerar um valor MAC válido para qualquer entrada x. Suponha que o tamanho da chave seja de k bits e que o atacante tenha um par conhecido de texto-MAC. Então, o atacante pode calcular o MAC de n bits sobre o texto conhecido para todas as chaves possíveis. Pelo menos uma chave tem garantias de produzir o MAC correto, isto é, a chave válida que foi usada inicialmente para produzir o par texto-MAC conhecido. Essa fase do ataque exige um nível de esforço proporcional a 2^k (ou seja, a operação para cada um dos 2^k valores de chave possíveis). Porém, conforme descrevemos anteriormente, como o MAC é um mapeamento muitos-para-um, pode haver outras chaves que produzam o valor correto. Assim, se mais de uma chave for encontrada para produzir o valor correto, pares texto-MAC adicionais precisarão ser testados. Pode-se mostrar que o grau de esforço decresce rapidamente a cada par texto-MAC adicional e que o nível geral de esforço é de aproximadamente 2^k [MENE97].

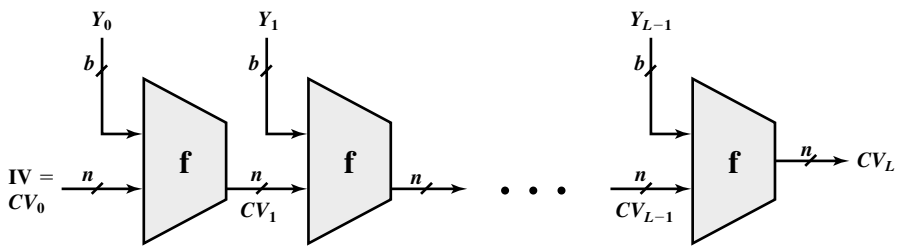
Um atacante também pode trabalhar sobre o valor MAC sem tentar recuperar a chave. Aqui, o objetivo é gerar um valor MAC válido para determinada mensagem ou encontrar uma mensagem que combine com determinado valor MAC. De qualquer forma, o nível de esforço é comparável ao do ataque de resistência à primeira inversão ou ao do ataque à segunda inversão de um código de hash, ou 2^n. No caso do MAC, o ataque não pode ser realizado off-line sem entrada adicional; o atacante exigirá pares texto-MAC escolhidos ou o conhecimento da chave.

Resumindo, a quantidade de esforço para o ataque por força bruta sobre um algoritmo MAC pode ser expressa como $\min(2^k, 2^n)$. A avaliação da força é semelhante à dos algoritmos de criptografia simétrica. Parece razoável exigir que o comprimento da chave e o comprimento do MAC satisfaçam um relacionamento como $\min(k, n) \geq N$, onde N talvez esteja no intervalo de 128 bits.

Criptoanálise

Assim como nos algoritmos de criptografia, os ataques criptoanalíticos sobre funções de hash e algoritmos MAC buscam explorar alguma propriedade do algoritmo para realizar algum ataque diferente de uma busca exaustiva. A maneira de medir a resistência de um algoritmo de hash ou de MAC à criptoanálise é comparar sua força com o esforço exigido para um ataque por força bruta. Ou seja, um algoritmo de hash ou MAC ideal exigirá um esforço criptoanalítico maior ou igual ao esforço por força bruta.

Funções de hash Nos últimos anos, tem havido um esforço considerável, e alguns sucessos, no desenvolvimento de ataques criptoanalíticos a funções de hash. Para entendê-los, precisamos examinar a estrutura geral de uma típica função de hash segura, indicada na Figura 11.9. Essa estrutura, conhecida como função de hash iterada, foi proposta por Merkle [MERK79, MERK89] e é a estrutura da maioria das funções de hash em uso atualmente, incluindo SHA e Whirlpool, as quais são explicadas no Capítulo 12. A função de hash apanha uma mensagem de entrada e a particiona em L blocos de tamanho fixo com b bits cada. Se for preciso, o bloco final é complementado até atingir b bits. O bloco final também inclui o valor do comprimento total da entrada para a função de hash. A inclusão do tamanho torna o trabalho do oponente mais difícil; assim, este terá de encontrar duas mensagens de mesmo tamanho que gerem um hash com o mesmo valor ou duas mensagens de tamanhos diferentes que, com seus comprimentos, gerem um hash com o mesmo valor.



IV = valor inicial
CV_i = Variável de encadeamento
Y_i = i-ésimo bloco de entrada
f = Função de compressão
L = Número de blocos de entrada
n = Comprimento do código de hash
b = Comprimento do bloco de entrada

Figura 11.9 Estrutura geral do código de hash seguro.

O algoritmo de hash envolve o uso repetido de uma **função de compressão**, f, que utiliza duas entradas (uma entrada de n bits da etapa anterior, chamada *variável de encadeamento*, e um bloco de b bits), e produz uma saída de n bits. No início do hash, a variável de encadeamento tem um valor inicial especificado como parte do algoritmo. O valor final da variável de encadeamento é o valor do hash. Normalmente, $b > n$; daí o termo *compressão*. A função de hash pode ser resumida da seguinte forma:

$$CV_0 = IV = \text{valor inicial de } n \text{ bits}$$
$$CV_i = \mathrm{f}(CV_{i-1}, Y_{i-1}) \qquad 1 \le i \le L$$
$$\mathrm{H}(M) = CV_L$$

onde a entrada da função de hash é uma mensagem M consistindo nos blocos $Y_0, Y_1, ..., Y_{L-1}$.

A motivação para essa estrutura iterativa vem da observação por Merkle [MERK89] e Damgard [DAMG89] de que, se a função de compressão é resistente a colisões, então o mesmo acontece com a função de hash iterada resultante.[3] Portanto, a estrutura pode ser usada para produzir uma função de hash segura para operar sobre uma mensagem de qualquer comprimento. O problema de projetar uma função de hash segura reduz-se a projetar uma função de compressão de colisão que opere sobre entradas de algum comprimento fixo.

A criptoanálise de funções de hash focaliza a estrutura interna de f e é baseada nas tentativas de encontrar técnicas eficientes para produzir colisões para uma única execução de f. Uma vez que isso esteja feito, o ataque deverá levar em consideração o valor fixo do IV. O ataque a f depende da exploração de sua estrutura interna. Normalmente, assim como nas cifras em bloco simétricas, f consiste em uma série de rodadas de processamento, de modo que o ataque envolve a análise do padrão de mudanças de bit de uma rodada para outra.

Lembre-se de que, para qualquer função de hash, deve haver colisões, pois estamos mapeando uma mensagem de comprimento pelo menos igual ao dobro do tamanho de bloco b (porque precisamos anexar um campo de comprimento) a um código de hash de tamanho n, onde $b \ge n$. O que é preciso é que seja computacionalmente inviável encontrar colisões.

Os ataques montados a funções de hash são um tanto complexos e extrapolam o escopo deste livro. Para o leitor interessado, [DOBB96] e [BELL97] são recomendados.

Códigos de autenticação de mensagens Há muito mais variedade na estrutura dos MACs do que nas funções de hash, de modo que é difícil generalizar a respeito da criptoanálise dos MACs. Além disso, tem havido muito menos trabalho no desenvolvimento desses ataques. Um estudo recente e muito útil sobre alguns métodos para MACs específicos é [PREN96].

11.6 LEITURA RECOMENDADA

[JUEN85] e [JUEN87] oferecem uma boa base sobre autenticação de mensagens, com um foco nos MACs criptográficos e funções de hash. Abordagens sólidas de funções de hash e códigos de autenticação de mensagens são encontradas em [STIN02] e [MENE97]. Um bom estudo recente é [PREN99].

JUEN85 Jueneman, R.; Matyas, S.; e Meyer, C. "Message authentication." IEEE *Communications Magazine*, setembro de 1988.
JUEN87 Jueneman, R. "Electronic document authentication." *IEEE Network Magazine*, abril de 1987.
MENE97 Menezes, A.; Oorshcot, P.; e Vanstone, S. *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997.
PREN99 Preneel, B. "The state of cryptographic hash functions." *Proceedings, EUROCRYPT '96*, 1996; publicado por Springer-Verlag.
STIN02 Stinson, D. *Cryptography: theory and practice*. Boca Raton: CRC Press, 2002.

11.7 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

ataque de aniversário
autenticação de mensagens
autenticador
código de autenticação de mensagens (MAC)
código de hash

3. A recíproca não é necessariamente verdadeira.

função de compactação
função de hash
função de hash unidirecional
paradoxo do aniversário
resistência forte a colisões ou resistência a colisões
resistência fraca a colisões ou resistência à segunda inversão
resumo de mensagem
soma de verificação criptográfica ou checksum criptográfico
valor de hash

Perguntas de revisão

11.1 Que tipos de ataques são tratados pela autenticação da mensagens?

11.2 Quais são os dois níveis de funcionalidade contidos em um mecanismo de autenticação de mensagens ou de assinatura digital?

11.3 Quais são algumas abordagens para produzir autenticação de mensagens?

11.4 Quando uma combinação de criptografia simétrica e um código de controle de erro são usados para autenticação de mensagens, em que ordem as duas funções precisam ser realizadas?

11.5 O que é um código de autenticação de mensagens?

11.6 Qual é a diferença entre um código de autenticação de mensagens e uma função de hash unidirecional?

11.7 De que maneiras um valor de hash pode ser protegido para oferecer autenticação de mensagens?

11.8 É necessário recuperar a chave secreta a fim de atacar um algoritmo de MAC?

11.9 Que características são necessárias em uma função de hash segura?

11.10 Qual é a diferença entre resistência fraca e forte a colisões?

11.11 Qual é o papel de uma função de compressão em uma função de hash?

Problemas

11.1 Se F é uma função de detecção de erros, tanto o uso interno quanto o externo (Figura 11.2) oferecerão capacidade de detecção de erros. Se algum bit da mensagem transmitida for alterado, isso será refletido em uma divergência entre a FCS (*Frame Check Sequence* ou Seqüência de Verificação de Quadro) recebida e a FCS calculada, não importa se a função FCS é realizada dentro ou fora da função de criptografia. Alguns códigos também oferecem uma capacidade de correção de erros. Dependendo da natureza da função, se um bit ou um pequeno número deles for alterado em trânsito, o código de correção de erros conterá informações redundantes suficientes para determinar o bit ou os bits errados e corrigi-los. Nitidamente, um código de correção de erros oferecerá capacidade de correção de erros quando usado externamente à função de criptografia. Ele também oferecerá essa capacidade se for usado internamente à função de criptografia?

11.2 O algoritmo de autenticação de dados, descrito na Seção 11.3, pode ser definido como usando o modo de operação de encadeamento de cifra de blocos (CBC) do DES, com um vetor de inicialização de zeros (Figura 11.6). Mostre que o mesmo resultado pode ser produzido usando o modo de cifra realimentada (CFB).

11.3 O protocolo de transporte de alta velocidade XTP (Xpress Transfer Protocol) utiliza uma função de soma de verificação (checksum) de 32 bits, definida como a concatenação de duas funções de 16 bits: XOR e RXOR, definidas na Seção 11.4 como 'duas funções de hash simples' e ilustradas na Figura 11.7.

a. Essa soma de verificação detectará todos os erros causados por um número ímpar de bits errados? Explique.

b. Essa soma de verificação detectará todos os erros causados por um número par de bits? Se não, caracterize os padrões de erro que farão a soma de verificação falhar.

c. Comente a eficácia dessa função para uso como uma função de hash para autenticação.

11.4 **a.** Considere o esquema do código de hash de Davies e Price descrito na Seção 11.4 e suponha que o DES seja usado como algoritmo de criptografia:

$$H_i = H_{i-1} \oplus \mathrm{E}(M_i, H_{i-1})$$

e lembre-se da propriedade complementar do DES (Problema 3.14): se $Y = \mathrm{E}(K, X)$, então $Y' = \mathrm{E}(K', X')$. Use essa propriedade para mostrar como uma mensagem composta pelos blocos $M_1, M_2, ..., M_N$ pode ser alterada sem que seu código de hash seja alterado.

b. Mostre que um ataque semelhante terá sucesso contra o esquema proposto em [MEYE88]:

$$H_i = M_i \oplus \mathrm{E}(H_{i-1}, M_i)$$

11.5 **a.** Considere a seguinte função de hash. As mensagens estão na forma de uma seqüência de números decimais, $M = (a_1, a_2, ..., a_t)$. O valor de hash h é calculado como $\left(\sum_{i=1}^{t} a_i\right) \bmod n$, para algum valor predefinido n. Essa função de hash satisfaz qualquer um dos requisitos para uma

função de hash listados na Seção 11.4? Explique sua resposta.

b. Repita a parte (a) para uma função de hash

$$h = \left(\sum_{i=1}^{t} (a_i)^2\right) \bmod n.$$

c. Calcule a função de hash da parte (b) para $M = (189, 632, 900, 722, 349)$ e $n = 989$.

11.6 É possível usar uma função de hash para construir uma cifra de blocos com uma estrutura semelhante à DES. Se uma função de hash é unidirecional e um bloco cifrado precisa ser reversível (para decriptografia), como isso é possível?

11.7 Agora, considere o problema oposto: usar um algoritmo de criptografia para construir uma função de hash unidirecional. Considere o uso do RSA com uma chave conhecida. Então, processe uma mensagem composta por uma seqüência de blocos da seguinte forma: criptografe o primeiro bloco, faça o XOR do resultado com o segundo bloco e criptografe novamente etc. Mostre que esse esquema não é seguro, solucionando o problema a seguir. Dada uma mensagem de dois blocos B1, B2 e seu hash

$$\text{RSAH(B1, B2)} = \text{RSA(RSA(B1)} \oplus \text{B2)}$$

Dado um bloco qualquer C1, escolha C2 de modo que RSAH(C1, C2) = RSAH(B1, B2). Assim, a função de hash não satisfaz a resistência à segunda inversão.

11.8 Suponha que $H(m)$ seja uma função de hash resistente a colisões que mapeia uma mensagem de comprimento arbitrário em um valor de hash de n bits. É verdade que, para todas as mensagens x, x' com $x \neq x'$, temos $H(x) \neq H(x')$? Explique sua resposta.

APÊNDICE 11A BASE MATEMÁTICA DO ATAQUE DO ANIVERSÁRIO

Neste apêndice, derivamos a justificativa matemática para o ataque do aniversário. Começaremos com um problema relacionado e depois examinaremos o problema do qual o nome 'ataque do aniversário' deriva.

Problema relacionado

Um problema geral relacionado a funções de hash é o seguinte. Dada uma função de hash H, com n saídas possíveis e um valor específico $H(x)$, se H é aplicado a K entradas aleatórias, qual deve ser o valor de k de modo que a probabilidade de que pelo menos uma entrada y satisfaça $H(y) = H(x)$ seja 0,5?

Para um único valor de y, a probabilidade de que $H(y) = H(x)$ é de apenas $1/n$. Reciprocamente, a probabilidade de que $H(y) \neq H(x)$ é $[1 - (1/n)]$. Se gerarmos k valores aleatórios de y, então a probabilidade de que nenhum deles tenha valores de hash iguais é apenas o produto das probabilidades de que cada valor individual não combine, ou $[1 - (1/n)]^k$. Assim, a probabilidade de que exista pelo menos uma combinação é $1 - [1 - (1/n)]^k$.

O teorema binomial pode ser indicado da seguinte forma:

$$(1 - a)^k = 1 - ka + \frac{k(k-1)}{2!}a^2 - \frac{k(k-1)(k-2)}{3!}a^3 \ldots$$

Para valores muito pequenos de a, isso pode ser aproximado como $(1 - ka)$. Assim, a probabilidade de pelo menos uma combinação é aproximada como $1 - [1 - (1/n)]^k \approx 1 - [1 - (k/n)] = k/n$. Para uma probabilidade de 0,5, temos $k = n/2$.

Em particular, para um código de hash de m bits, o número de códigos possíveis é 2^m e o valor de k que produz uma probabilidade de 0,5 é

$$k = 2^{(m-1)} \tag{11.1}$$

The Birthday Paradox

O paradoxo do aniversário normalmente é apresentado em cursos de probabilidade básica para demonstrar que os resultados da probabilidade às vezes são contra-intuitivos. O problema pode ser afirmado da seguinte forma: qual é o valor mínimo de k, tal que a probabilidade de que pelo menos duas pessoas em um grupo de k pessoas façam aniversário no mesmo dia seja maior que 0,5? Ignore 29 de fevereiro e considere que cada dia do aniversário seja igualmente provável. Para responder, vamos definir

$P(n, k)$ = Probabilidade pelo menos uma duplicata em k itens, com cada item capaz de assumir um dos n valores igualmente prováveis entre 1 e n]

Assim, estamos procurando o menor valor de k tal que $P(365, k) \geq 0{,}5$. É mais fácil derivar primeiro a probabilidade de que não existam duplicatas, o que designamos como $Q(365, k)$. Se $k > 365$, então é impossível que todos os valores sejam diferentes. Assim, consideramos $k \leq 365$. Agora, considere o número de maneiras diferentes, N, de acordo com as quais podemos ter k valores sem duplicatas. Podemos escolher qualquer um dos 365 valores para o primeiro item, qualquer um dos 364 números restantes para o segundo item, e assim por diante. Logo, o número de maneiras diferentes é

$$N = 365 \times 364 \times \ldots (365 - k + 1) = \frac{365!}{(365 - k)!} \tag{11.2}$$

Se removermos a restrição de que não existem duplicatas, então cada item pode ser qualquer um dos 365 valores, e o número total de possibilidades é 365^k. Assim, a probabilidade de nenhuma duplicata é simplesmente a fração de conjuntos de valores que não possuem duplicatas a partir de todos os conjuntos possíveis de valores:

$$Q(365, k) = \frac{365!/(365 - k)!}{(365)^k} = \frac{365!}{(365 - k)!(365)^k}$$

e

$$P(365, k) = 1 - Q(365, k) = 1 - \frac{365!}{(365 - k)!(365)^k} \tag{11.3}$$

O gráfico dessa função está na Figura 11.10. A probabilidade pode parecer surpreendentemente grande para qualquer um que não tenha considerado o problema antes. Muitas pessoas diriam que, para ter uma probabilidade maior que 0,5 de que haja pelo menos uma duplicata, o número de pessoas no grupo teria de ser aproximadamente 100. Na verdade, o número é 23, com $P(365, 23) = 0{,}5073$. Para $k = 100$, a probabilidade de pelo menos uma duplicata é 0,9999997.

Talvez o motivo para que o resultado pareça tão surpreendente é que, se você considerar uma determinada pessoa em um grupo, a probabilidade de que alguma outra pessoa no grupo faça aniversário no mesmo dia é pequena. Mas a probabilidade com que estamos preocupados é a de que *qualquer* par de pessoas no grupo faça aniversário no mesmo dia. Em um grupo de 23, existem $(23(23 - 1))/2 = 253$ diferentes pares de pessoas. Daí as altas probabilidades.

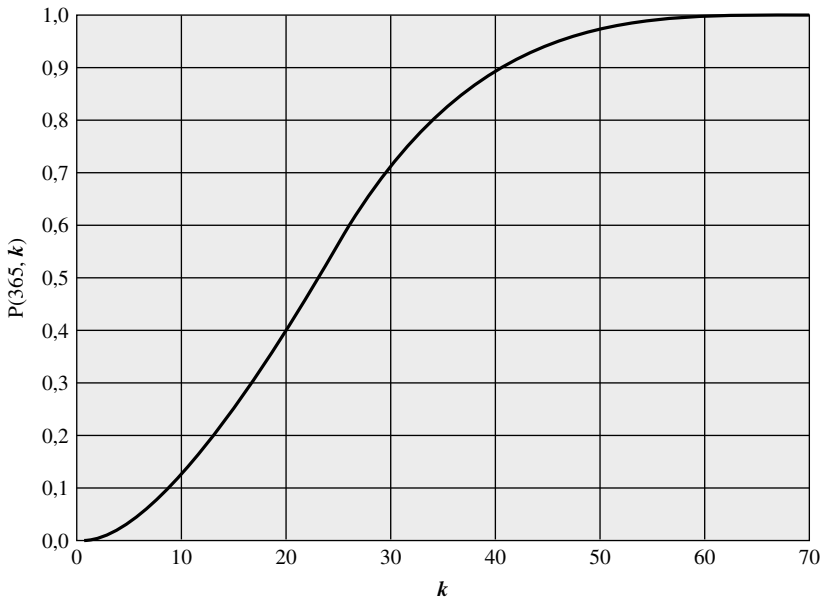


Figura 11.10 O paradoxo do aniversário.

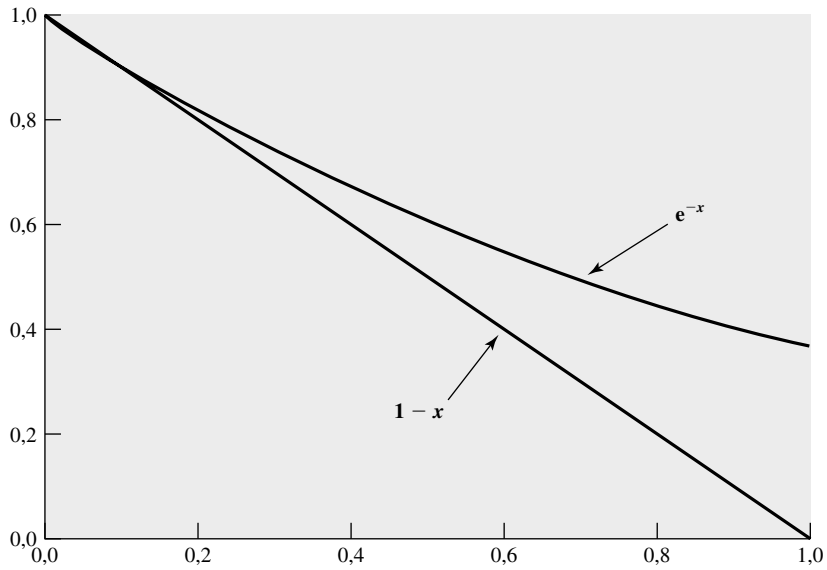


Figura 11.11 Uma desigualdade útil.

Desigualdade útil

Antes de desenvolvermos uma generalização do problema do aniversário, derivamos uma desigualdade que será necessária:

$$(1 - x) \leq e^{-x} \qquad \text{para todo } x \geq 0 \tag{11.4}$$

A Figura 11.11 ilustra a desigualdade. Para verificar se a desigualdade permanece, observe que a linha inferior é a tangente de e^{-x} em $x = 0$. A inclinação dessa linha é apenas a derivada de e^{-x} em $x = 0$:

$$\begin{aligned} f(x) &= e^{-x} \\ f'(x) &= \frac{d}{dx}e^{-x} = -e^{-x} \\ f'(0) &= -1 \end{aligned}$$

A tangente é uma linha reta na forma $ax + b$, com $a = -1$, e a tangente em $x = 0$ precisa ser igual a $e^{-0} = 1$. Assim, a tangente é a função $(1 - x)$, confirmando a desigualdade da Equação (11.4). Além disso, observe que, para um x pequeno, temos $(1 - x) \approx e^{-x}$.

O caso geral das duplicações

O problema do aniversário pode ser generalizado para o seguinte problema: dada uma variável aleatória, que é um inteiro com distribuição uniforme entre 1 e n e uma seleção de k ocorrências ($k \leq n$) da variável aleatória, qual é a probabilidade, $P(n, k)$, de que exista pelo menos uma duplicata? O problema do aniversário é apenas um caso especial com $n = 365$. Pelo mesmo raciocínio de antes, temos a seguinte generalização da Equação (11.3):

$$P(n, k) = 1 - \frac{n!}{(n - k)!n^k} \tag{11.5}$$

Podemos reescrever como

$$\begin{aligned} P(n, k) &= 1 - \frac{n \times (n - 1) \times \ldots \times (n - k + 1)}{n^k} \\ &= 1 - \left[\frac{n - 1}{n} \times \frac{n - 2}{n} \times \ldots \times \frac{n - k + 1}{n}\right] \\ &= 1 - \left[\left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times \ldots \times \left(1 - \frac{k - 1}{n}\right)\right] \end{aligned}$$

Usando a desigualdade da Equação (11.4):

$$\begin{aligned} P(n, k) &> 1 - \left[(e^{-1/n}) \times (e^{-2/n}) \times \ldots \times (e^{-(k-1)/n}) \right] \\ &> 1 - e^{-[(1/n)+(2/n)+\ldots+((k-1)/n)]} \\ &> 1 - e^{-(k\times(k-1))/2n} \end{aligned}$$

Agora, vamos formular a pergunta: Que valor de k é necessário tal que $P(n, k) > 0{,}5$? Para satisfazer o requisito, temos

$$\begin{aligned} 1/2 &= 1 - e^{-(k\times(k-1))/2n} \\ 2 &= e^{(k\times(k-1))/2n} \\ \ln 2 &= \frac{k \times (k-1)}{2n} \end{aligned}$$

Para um k grande, podemos substituir $k \times (k - 1)$ por k^2, e obtemos

$$k = \sqrt{2(\ln 2)n} = 1{,}18\sqrt{n} \approx \sqrt{n} \qquad \textbf{(11.6)}$$

Como uma verificação, para $n = 365$, obtemos $k = 1{,}18 \times \sqrt{365} = 22{,}54$, que é muito próximo da resposta correta de 23.

Agora, podemos colocar a base do ataque do aniversário nos seguintes termos: suponha que tenhamos uma função H, com 2^m saídas possíveis, ou seja, uma saída de m bits. Se H for aplicado a k entradas aleatórias, qual deverá ser o valor de k para que haja a probabilidade de pelo menos uma duplicata [ou seja, $H(x) = H(y)$ para algumas entradas (x, y)]? Usando a aproximação da Equação (11.6):

$$k = \sqrt{2^m} = 2^{m/2} \qquad \textbf{(11.7)}$$

Sobreposição entre dois conjuntos

Existe um problema relacionado ao caso geral de duplicatas que também é relevante para nossas explicações. O problema é este: dada uma variável aleatória inteira com distribuição uniforme entre 1 e n e dois conjuntos de ocorrências k ($k \leq n$) da variável aleatória, qual é a probabilidade, $R(n, k)$ de que os dois conjuntos não sejam disjuntos; ou seja, qual é a probabilidade de que haja pelo menos um valor encontrado nos dois conjuntos?

Vamos chamar os dois conjuntos de X e Y, com elementos $\{x_1, x_2, ..., x_k\}$ e $\{y_1, y_2, ..., y_k\}$, respectivamente. Dado o valor de x_1, a probabilidade de que $y_1 = x_1$ é apenas $1/n$ e, portanto, a probabilidade de que y_1 não corresponda a x_1 é de $[1 - (1/n)]$. Se gerarmos os k valores aleatórios em Y, a probabilidade de que nenhum desses valores seja igual a x é $[1 - (1/n)]^k$. Assim, a probabilidade de que haja pelo menos uma combinação com x_1 é $1 - [1 - (1/n)]^k$.

Para prosseguir, vamos supor que todos os elementos de X sejam distintos. Se n for grande e se k também for grande (por exemplo, na ordem de $\sqrt{n}$), então essa é uma boa aproximação. Na verdade, pode haver umas poucas duplicatas, mas a maioria dos valores será distinta. Com essa suposição, podemos fazer a seguinte derivação:

$$\Pr[\text{nenhuma combinação em } Y \text{ com } x_1] = \left(1 - \frac{1}{n}\right)^k$$

$$\Pr[\text{nenhuma combinação em } Y \text{ com } X] = \left(\left(1 - \frac{1}{n}\right)^k\right)^k = \left(1 - \frac{1}{n}\right)^{k^2}$$

$$R(n, k) = \Pr[\text{pelo menos uma combinação em } Y \text{ com } X] = 1 - \left(1 - \frac{1}{n}\right)^{k^2}$$

Usando a desigualdade da Equação (11.4):

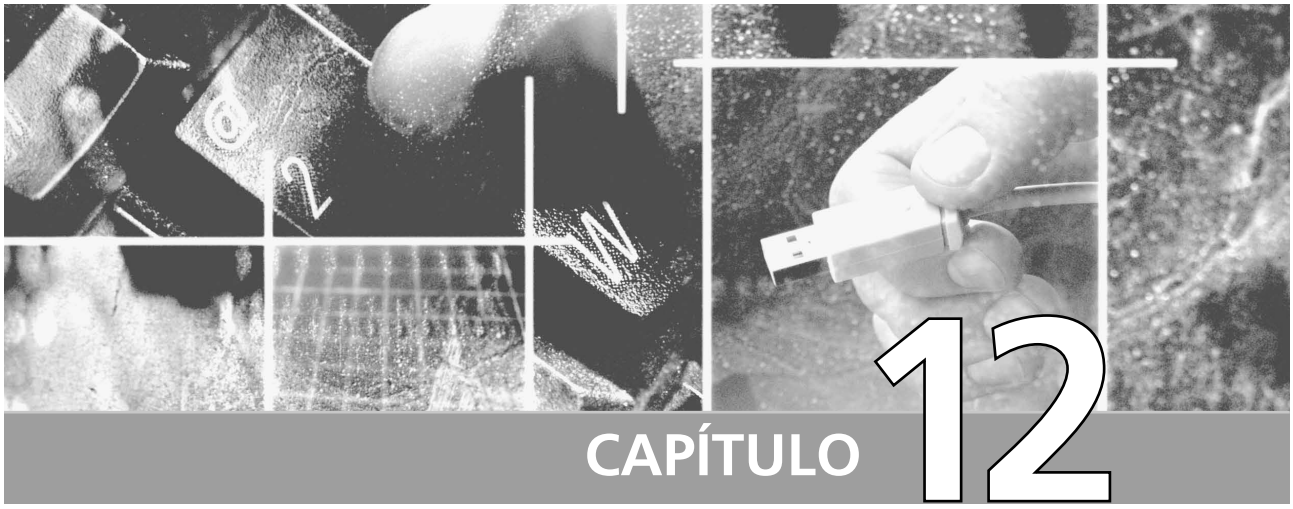
$$\begin{aligned} R(n, k) &> 1 - (e^{-1/n})^{k^2} \\ R(n, k) &> 1 - (e^{-k^2/n}) \end{aligned}$$

Vamos formular a pergunta: Que valor de k é necessário tal que $R(n, k) > 0{,}5$? Para satisfazer o requisito, temos

$$1/2 = 1 - (e^{-k^2/n})$$
$$2 = e^{k^2/n}$$
$$\ln(2) = \frac{k^2}{n}$$
$$k = \sqrt{(\ln(2))n} = 0{,}83\sqrt{n} \approx \sqrt{n} \qquad \textbf{(11.8)}$$

Podemos colocar isso em termos relacionados aos ataques do aniversário da seguinte forma. Suponha que tenhamos uma função H, com 2^m saídas possíveis (ou seja, uma saída de m bits). Aplique H a k entradas aleatórias para produzir o conjunto X e novamente a k entradas aleatórias adicionais para produzir o conjunto Y. Qual deverá ser o valor de k para que haja a probabilidade de pelo menos 0,5 de existir uma combinação entre os dois conjuntos (ou seja, $H(x) = H(y)$ para algumas entradas $x \in X, y \in Y$)? Usando a aproximação da equação (11.8):

$$k = \sqrt{2^m} = 2^{m/2}$$



Algoritmos de hash e de MAC

12.1 **Algoritmo de hash seguro**
12.2 **Whirlpool**
12.3 **HMAC**
12.4 **CMAC**
12.5 **Leitura e sites Web recomendados**
12.6 **Principais termos, perguntas de revisão e problemas**

Cada uma das mensagens, como todas que ele já tinha lido dos comandos de Stern, começava com um número e terminava com um número ou seqüência de números. Nenhum esforço de Mungo ou de qualquer um de seus peritos tinha sido capaz de quebrar o código de Stern, nem havia qualquer pista quanto ao significado do número inicial e dos números finais.

Talking to strange men, Ruth Rendell

O esquilo Douglas tem um hábito de alimentação diferente. Ele normalmente come pinhões de baixo para cima. Os pinhões parcialmente comidos podem indicar a presença desses esquilos se tiverem sido comidos por baixo primeiro. Se, ao invés disso, o pinhão tiver sido comido de cima para baixo, é mais provável que um cruza-bico tenha feito a refeição.

Squirrels: a wildlife handbook, Kim Long

PONTOS PRINCIPAIS

- Praticamente todos os algoritmos de hash seguros possuem a estrutura geral mostrada na Figura 11.9.
- A função de compressão usada nos algoritmos de hash seguros podem estar em uma dentre duas categorias: uma função projetada especificamente para a função de hash ou uma cifra de bloco simétrica. SHA e Whirlpool são exemplos dessas duas abordagens, respectivamente.
- Os códigos de autenticação de mensagens também são classificados em duas categorias; aqueles baseados no uso de um algoritmo de hash seguro e aqueles baseados no uso de uma cifra de bloco simétrica. HMAC e CMAC são exemplos dessas duas abordagens, respectivamente.

Neste capítulo, vemos importantes exemplos de algoritmos de hash seguros e de códigos de autenticação de mensagens (MACs). As funções de hash modernas mais importantes seguem a estrutura básica da Figura 11.9. Esta tem provado ser uma estrutura fundamentalmente consistente, e projetos mais novos simplesmente refinam a estrutura e aumentam o tamanho do código de hash. Dentro dessa estrutura básica, duas abordagens têm sido usadas no projeto da função de compressão, que é o elemento básico da função de hash. Tradicionalmente, a maioria das funções de hash que alcançou uso generalizado conta com uma função de compressão projetada especificamente para a função de hash. Normalmente, essa função utiliza aritmética modular e operações lógicas binárias. Outra técnica é usar uma cifra de bloco simétrica como função de compressão. Neste capítulo, examinamos os exemplos possivelmente mais importantes de cada uma destas abordagens: o Secure Hash Algorithm (SHA) e o Whirlpool.

Os MACs, convenientemente, também possuem duas categorias, baseadas em seu elemento fundamental. Uma técnica comum é usar um algoritmo de hash, por exemplo SHA, como núcleo do algoritmo MAC. Outra técnica é usar uma cifra de bloco simétrico em um modo de encadeamento de cifra de bloco. Novamente, vemos o exemplo talvez mais importante de cada técnica: HMAC e CMAC.

12.1 ALGORITMO DE HASH SEGURO

O *Secure Hash Algorithm* (SHA) foi desenvolvido pelo NIST (*National Institute of Standards and Technology*) dos EUA e publicado como um padrão nacional para processamento de informações (FIPS 180) em 1993; uma versão revisada foi lançada como FIPS 180-1 em 1995, geralmente é conhecida como SHA-1. A norma real é intitulada *Secure Hash Standard*. SHA baseia-se na função de hash MD4 e seu projeto modela de perto essa função. O SHA-1 também é especificado na RFC 3174, que basicamente duplica o material do FIPS 180-1, mas acrescenta uma implementação de código em C.

O SHA-1 produz um valor de hash de 160 bits. Em 2002, o NIST produziu uma versão revisada do padrão, FIPS 180-2, que definiu três novas versões do SHA, com tamanhos de valor de hash de 256, 384 e 512 bits, conhecidas como SHA-256, SHA-384 e SHA-512 (Tabela 12.1). Essas novas versões têm a mesma estrutura básica e usam os mesmos tipos de aritmética modular e operações lógicas binárias do SHA-1. Em 2005, o NIST anunciou a intenção de retirar gradualmente a aprovação do SHA-1 e que passará a contar com outras versões do SHA por volta de 2010. Pouco tempo depois, uma equipe de pesquisa descreveu um ataque em que duas mensagens diferentes poderiam oferecer o mesmo hash SHA-1 usando 2^{69} operações, muito menos do que as 2^{80} operações que anteriormente acreditava-se serem necessárias para encontrar uma colisão com um hash SHA-1 [WANG05]. Esse resultado deverá apressar a transição para as outras versões do SHA.

Nesta seção, oferecemos uma descrição do SHA-512. As outras versões são muito semelhantes.

Lógica do SHA-512

O algoritmo aceita como entrada uma mensagem com um tamanho máximo de 2^{128} bits e produz como saída um resumo da mensagem de 512 bits. A entrada é processada em blocos de 1.024 bits. A Figura 12.1 representa o processamento geral de uma mensagem para produzir um resumo. Este segue a estrutura geral representada na Figura 11.9. O processamento consiste nas seguintes etapas:

- **Etapa 1: Anexar bits de complementação.** A mensagem é complementada de modo que seu tamanho seja congruente a 896 módulo 1.024 [comprimento ≡ 896 (mod 1.024)]. A complementação sempre é realizada, mesmo que a mensagem já tenha o tamanho desejado. Assim, o número de bits de complementação está no intervalo de 1 a 1.024. O complemento consiste de um único bit 1 seguido pelo número necessário de bits 0.

Tabela 12.1 Comparação de parâmetros do SHA

	SHA-1	SHA-256	SHA-384	SHA-512
Tamanho do resumo de mensagem	160	256	384	512
Tamanho da mensagem	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Tamanho do bloco	512	512	1024	1024
Tamanho da palavra (word)	32	32	64	64
Número de etapas	80	64	80	80
Segurança	80	128	192	256

Notas: 1. Todos os tamanhos são medidos em bits.

2. A segurança refere-se ao fato de que um ataque do aniversário a um resumo da mensagem de comprimento n produz uma colisão com um fator de trabalho de aproximadamente $2^{n/2}$.

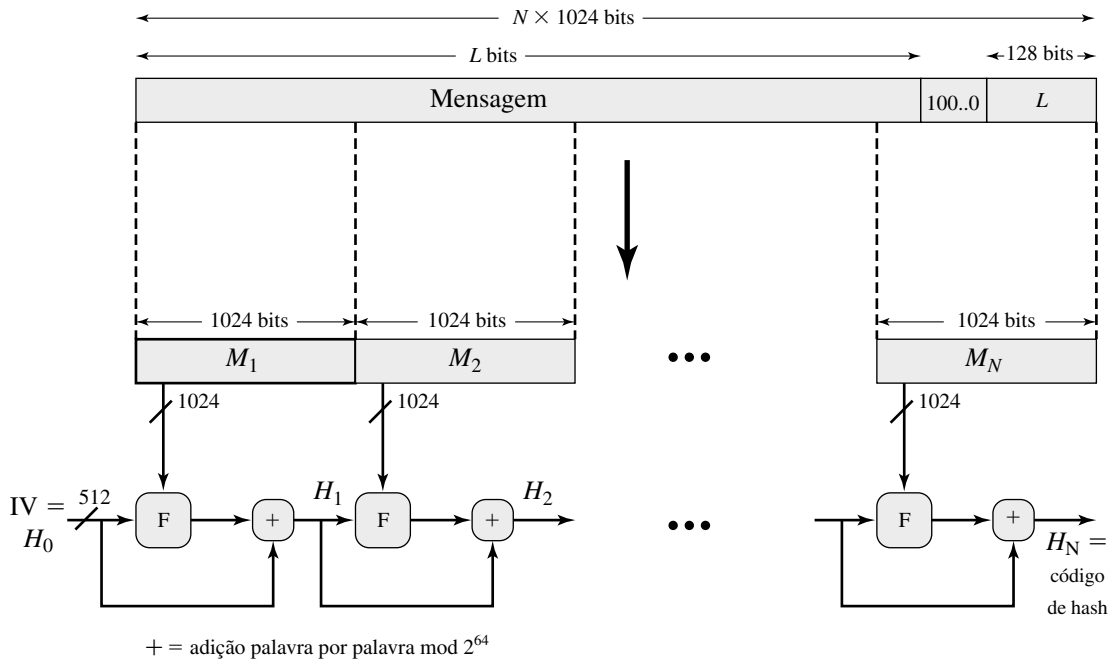


Figura 12.1 Geração de resumo da mensagem usando SHA-512.

- **Etapa 1: Anexar bits de complementação.** A mensagem é complementada de modo que seu tamanho seja congruente a 896 módulo 1.024 [comprimento ≡ 896 (mod 1.024)]. A complementação sempre é realizada, mesmo que a mensagem já tenha o tamanho desejado. Assim, o número de bits de complementação está no intervalo de 1 a 1.024. O complemento consiste de um único bit 1 seguido pelo número necessário de bits 0.
- **Etapa 2: Anexar o comprimento.** Um bloco de 128 bits é anexado à mensagem. Esse bloco é tratado como um inteiro sem sinal de 128 bits (byte mais significativo primeiro) e contém o tamanho da mensagem original (antes da complementação).

  O resultado das duas primeiras etapas produz uma mensagem que é um múltiplo inteiro de 1.024 bits de extensão. Na Figura 12.1, a mensagem expandida é representada como uma seqüência de blocos de 1.024 bits $M_1, M_2, ..., M_N$, de modo que o tamanho total da mensagem expandida é N X 1.024 bits.
- **Etapa 3: Inicializar buffer de hash.** Um buffer de 512 bits é usado para manter resultados intermediários e finais da função de hash. O buffer pode ser representado como oito registradores de 64 bits (a, b, c, d, e, f, g, h). Esses registradores são inicializados com os seguintes inteiros de 64 bits (valores hexadecimais):

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CDI9137E2179

  Estes valores são armazenados no formato big-endian, que é o byte mais significativo de uma palavra (*word*) na posição de byte de endereço baixo (mais à esquerda). Essas palavras foram obtidas tomando-se os primeiros 64 bits das partes fracionárias das raízes quadradas dos oito primeiros números primos.
- **Etapa 4: Processar mensagem em blocos de 1.024 bits (128 palavras).** O núcleo do algoritmo é um módulo que consiste de 80 rodadas; esse módulo é chamado de F na Figura 12.1. A lógica é ilustrada na Figura 12.2.

Cada rodada tem como entrada o buffer de 512 bits abcdefgh, e atualiza o conteúdo do buffer. Na entrada da primeira rodada, o buffer contém o valor do hash intermediário, H_{i-1}. Cada rodada t utiliza um valor de 64 bits W_t, derivado do bloco atual de 1.024 bits sendo processado (M_i). Esses valores são derivados usando-se um schedule de mensagem descrito mais adiante. Cada rodada também utiliza uma constante aditiva K_t, onde $0 \leq t \leq 79$ indica uma das 80 rodadas. Essas palavras representam os primeiros 64 bits das partes fracionárias das raízes cúbicas dos 80 primeiros números primos. As constantes oferecem um conjunto de padrões de 64 bits aleatórios, o que deverá eliminar quaisquer regularidades nos dados da entrada.

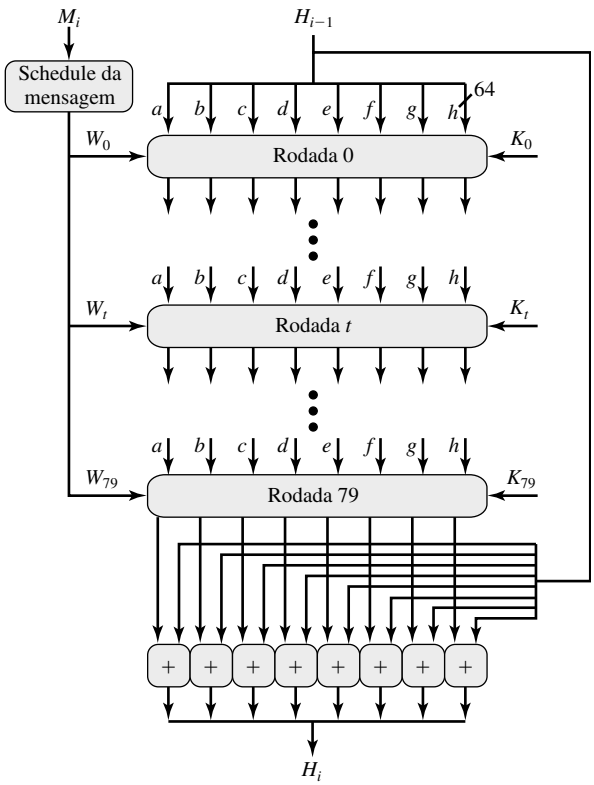


Figura 12.2 Processamento SHA-512 de um único bloco de 1.024 bits.

A saída da octogésima rodada é somada à entrada da primeira rodada (H_{i-1}) para produzir H_i. A adição é feita independentemente para cada uma das oito palavras no buffer com cada uma das palavras correspondentes em H_{i-1}, usando a adição módulo 2^{64}.

- **Etapa 5: Saída.** Depois que todos os N blocos de 1.024 bits tiverem sido processados, a saída do estágio N é o resumo de mensagem de 512 bits.
  Podemos resumir o comportamento do SHA-512 da seguinte forma:

$$\begin{aligned} H_0 &= \text{IV} \\ H_i &= \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i) \\ MD &= H_N \end{aligned}$$

onde

IV	= valor inicial do buffer abcdefgh, definido na etapa 3
abcdefgh_i	= a saída da última rodada de processamento do *i-ésimo* bloco de mensagem
N	= o número de blocos na mensagem (incluindo campos de complementação e de comprimento)
SUM_{64}	= adição módulo 2^{64} realizada separadamente em cada palavra do par de entradas
MD	= valor final do resumo de mensagem

Função de rodada do SHA-512

Vejamos com mais detalhes a lógica em cada uma das 80 etapas do processamento de um bloco de 512 bits (Figura 12.3). Cada rodada é definida pelo seguinte conjunto de equações:

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum\nolimits_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\sum\nolimits_0^{512} a\right) + \text{Maj}(a, b, c)$$

$$
\begin{aligned}
a &= T_1 + T_2 \\
b &= a \\
c &= b \\
d &= c \\
e &= d + T_1 \\
f &= e \\
g &= f \\
h &= g
\end{aligned}
$$

onde

t	= número da etapa; $0 \le t \le 79$
$\mathrm{Ch}(e, f, g)$	= (e AND f) $\oplus$ (NOT e AND g) *a função condicional:* If e then f else g
$\mathrm{Maj}(a, b, c)$	= (a AND b) $\oplus$ (a AND c) $\oplus$ (b AND c) *a função é verdadeira somente se a maioria (dois ou três) dos argumentos for verdadeira.*
$\left(\sum_0^{512} a\right)$	$= \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$
$\left(\sum_1^{512} e\right)$	$= \mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$
$\mathrm{ROTR}^n(x)$	= deslocamento circular (rotação) à direita do argumento de 64 bits x por n bits
W_t	= uma palavra de 64 bits derivada do bloco de entrada atual de 512 bits
K_i	= uma constante aditiva de 64 bits
$+$	= adição módulo 2^{64}

Resta indicar como os valores de palavra de 64 bits W_t são derivados da mensagem de 1.024 bits. A Figura 12.4 ilustra o mapeamento. Os primeiros 16 valores de w_t são tomados diretamente das 16 palavras do bloco atual. Os valores restantes são definidos da seguinte forma:

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

onde

$$
\begin{aligned}
\sigma_0^{512}(x) &= \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x) \\
\sigma_1^{512}(x) &= \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)
\end{aligned}
$$

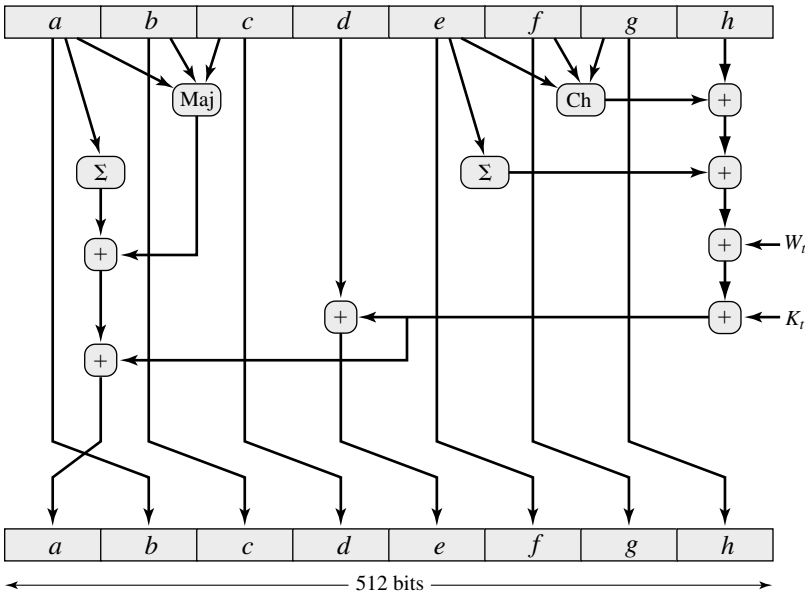


Figura 12.3 Operação elementar do SHA-512 (única rodada).

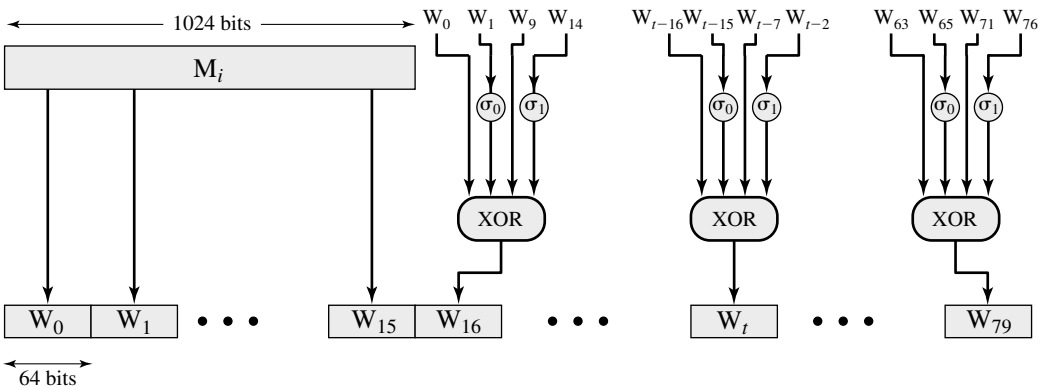


Figura 12.4 Criação da seqüência de entrada de 80 palavras para processamento de único bloco do SHA-512.

$\text{ROTR}^n(x)$ = deslocamento circular à direita (rotação) do argumento de 64 bits x por n bits
$\text{SHR}^n(x)$ = deslocamento à esquerda do argumento de 64 bits x por n bits com complementação por zeros à direita

Assim, nas primeiras 16 etapas de processamento, o valor de W_t é igual à palavra correspondente no bloco da mensagem. Para as 64 etapas restantes, o valor de W_t é composto pelo deslocamento circular à esquerda por um bit do XOR (OU EXCLUSIVO) de quatro dos valores anteriores de W_t com dois desses valores sujeitos a operações de deslocamento e rotação. Isso introduz uma grande quantidade de redundância e interdependência nos blocos da mensagem comprimidos, o que complica a tarefa de encontrar um bloco de mensagem diferente que seja mapeado na mesma saída de função de compressão.

12.2 WHIRLPOOL[1]

Nesta seção, examinamos a função de hash Whirlpool [BARR03], da qual um dos projetistas também é co-inventor do Rijndael, algoritmo adotado como padrão Advanced Encryption Standard (AES). Whirlpool é uma das duas únicas funções de hash recomendadas pelo NESSIE (*New European Schemes for Signatures, Integrity, and Encryption*).[2] O projeto NESSIE é um empreendimento patrocinado pela União Européia para propor um conjunto de primitivas criptográficas fortes de vários tipos.

O Whirlpool é baseado no uso de uma cifra de bloco para a função de compressão. Conforme mencionamos no Capítulo 11, tem havido tradicionalmente pouco interesse no uso das funções de hash baseadas na cifra de bloco, devido às vulnerabilidades de segurança demonstradas pela estrutura. A seguir estão os inconvenientes em potencial:

1. As cifras de bloco não possuem as propriedades das funções aleatórias. Por exemplo, elas são reversíveis. Essa falta de aleatoriedade pode conduzir a fragilidades que poderão ser exploradas.
2. As cifras de bloco normalmente exibem outras regularidades ou fraquezas. Por exemplo, [MIYA90] demonstra como comprometer muitos esquemas de hash baseados em propriedades inerentes da cifra de bloco.
3. Normalmente, as funções de hash baseadas em cifra de bloco são significativamente mais lentas do que as funções de hash baseadas em uma função de compressão projetada especificamente para a função de hash.
4. Uma medida principal da força de uma função de hash é o comprimento do código de hash em bits. Para códigos de hash baseados na cifra de bloco, os projetos propostos têm um tamanho de código de hash igual a uma ou duas vezes o tamanho da cifra de bloco. Tradicionalmente, o tamanho da cifra de bloco tem sido limitado a 64 bits (por exemplo, DES, triple DES), resultando em um código de hash de força questionável.

Porém, desde a adoção do AES, tem havido interesse renovado no desenvolvimento de uma função de hash segura baseada em uma cifra de bloco forte e exibindo bom desempenho. Whirlpool é uma função de hash baseada em cifra de bloco, com segurança e desempenho comparáveis, se não melhores, ao que encontramos nas funções de hash não baseadas em cifra de bloco, tais como o SHA. O Whirlpool tem as seguintes características:

1. O tamanho do código de hash é de 512 bits, igualando o código de hash mais longo disponível com SHA.
2. A estrutura geral da função de hash tem se mostrado resistente aos ataques comuns a códigos de hash baseados em cifra em bloco.

1. A maior parte do material desta seção foi publicada inicialmente em [STAL O6].
2. O outro esquema recomendado consiste de três variantes do SHA: SHA-256, SHA-384 e SHA-512.

3. A cifra de bloco interna é baseada no AES e é projetada para oferecer implementação compacta e com bom desempenho tanto em software quanto em hardware.

O projeto do Whirlpool coloca os seguintes objetivos de segurança: suponha que tomamos como resultado do hash o valor de qualquer substring de n bits da saída completa do Whirlpool.

- A carga de trabalho esperada para a geração de uma colisão é da ordem de $2^{n/2}$ execuções do Whirlpool.
- Dado um valor de n bits, a carga de trabalho esperada para encontrar uma mensagem que crie um hash com esse valor é da ordem de 2^n execuções do Whirlpool.
- Dada uma mensagem e seu resultado de hash de n bits, a carga de trabalho esperada para encontrar uma segunda mensagem que gere o mesmo valor de hash é da ordem de 2^n execuções do Whirlpool.
- É inviável detectar correlações sistemáticas entre qualquer combinação linear de bits de entrada e qualquer combinação linear de bits do resultado do hash, ou prever que bits do resultado do hash mudarão de valor quando certos bits de entrada forem invertidos (isso significa resistência a ataques lineares e diferenciais).

Os projetistas se dizem confiantes de que esses objetivos foram atendidos com uma margem de segurança considerável. Porém, os objetivos não são suscetíveis a uma prova formal.

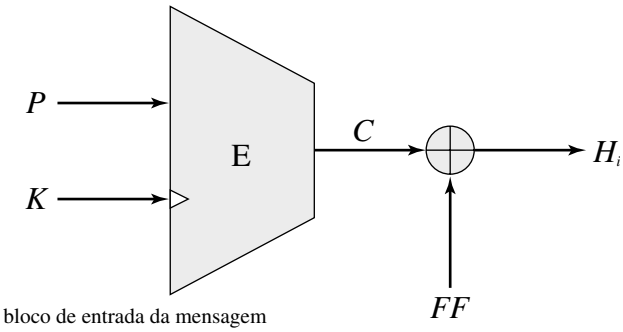
Começamos com uma explicação da estrutura da função de hash geral e, depois, examinamos a cifra em bloco usada como elemento básico.

Estrutura de hash do Whirlpool

Fundamentos A estrutura geral de hash repetida, proposta por Merkle (Figura 11.9), é usada em praticamente todas as funções de hash seguras. Porém, conforme dissemos, existem dificuldades no projeto de uma função de hash repetida verdadeiramente segura quando a função de compressão é uma cifra de bloco. Preneel [PREN93a, PREN93b] realizou uma análise sistemática das funções de hash baseadas em cifra de bloco, usando o modelo representado na Figura 12.5. Nesse modelo, o tamanho do código de hash é igual ao da cifra de bloco. Problemas adicionais de segurança são introduzidos e a análise se torna mais difícil se o tamanho do código de hash for maior que o da cifra de bloco. Preneel formulou 64 permutações possíveis do modelo básico, com base em qual entrada serviria como chave de criptografia e qual serviria como texto claro e em qual entrada, caso houvesse, combinada com o texto cifrado para produzir o código de hash intermediário. Com base em sua análise, ele concluiu que eram seguros somente os esquemas em que o texto claro era passado adiante e combinado com o texto cifrado. Esse arranjo torna a função de compactação difícil de ser invertida. [BLAC02] confirmou esses resultados, mas apontou o problema de segurança de usar uma cifra de bloco estabelecida, como AES: o valor do código de hash de 128 bits, resultante do uso do AES ou de outro esquema com o mesmo tamanho de bloco, pode ser inadequado para a segurança.

Lógica do Whirlpool Dada uma mensagem que consiste em uma seqüência de blocos $m_1, m_2, ..., m_t$, a função de hash do Whirlpool é expressa da seguinte forma:

$$
\begin{aligned}
H_0 &= \text{valor inicial} \\
H_i &= \mathrm{E}(H_{i-1}, m_i) \oplus H_{i-1} \oplus m_i = \text{valor intermediário} \\
H_t &= \text{valor do código de hash}
\end{aligned}
$$



m_i = i-ésimo bloco de entrada da mensagem
H_i = i-ésimo valor de hash intermediário
P = texto claro; K = chave de criptografia; C = texto cifrado
FF = valor passado adiante
P, K e FF podem ser escolhidos a partir do conjunto $(0, m_i, H_{i-1}, m_i \oplus H_{i-1})$

Observação: O triângulo indica a entrada da chave de criptografia.

Figura 12.5 Modelo da única iteração da função de hash (código de hash igual ao tamanho do bloco).

Em termos do modelo da Figura 12.5, a entrada de chave de criptografia para cada iteração é o valor de hash intermediário obtido da iteração anterior; o texto claro é o bloco de mensagem atual, e o valor passado adiante é o XOR bit a bit entre o bloco de mensagem atual e o valor de hash intermediário obtido na iteração anterior.

O algoritmo utiliza como entrada uma mensagem com um tamanho máximo menor do que 2^{256} bits e produz como saída um resumo de mensagem de 512 bits. A entrada é processada em blocos de 512 bits. A Figura 12.6 representa o processamento geral de uma mensagem para produzir um resumo. Este segue a estrutura geral representada na Figura 11.9. O processamento consiste nas seguintes etapas:

- **Etapa 1: Anexar bits de complementação.** A mensagem é complementada de modo que seu tamanho em bits seja um múltiplo ímpar de 256. A complementação é sempre acrescentada, mesmo que a mensagem já tenha o tamanho desejado. Por exemplo, se a mensagem tiver $256 \times 3 = 768$ bits de extensão, ela será complementada com 512 bits, totalizando um comprimento de $256 \times 5 = 1280$ bits. Assim, o número de bits de complementação está na faixa de 1 a 512.

  A complementação consiste de um único bit 1 seguido do número necessário de bits 0.
- **Etapa 2: Anexar comprimento.** Um bloco de 256 bits é anexado à mensagem. Esse bloco é tratado como um inteiro de 256 bits sem sinal (byte mais significativo primeiro) e contém o comprimento em bits da mensagem original (antes da complementação).

  O resultado das duas primeiras etapas produz uma mensagem que é um múltiplo inteiro de 512 bits de extensão. Na Figura 12.6, a mensagem expandida é representada como uma seqüência de blocos de 512 bits $m_1, m_2, ..., m_t$, de modo que o tamanho total da mensagem expandida é $t \times 512$ bits. Esses blocos são vistos externamente como vetores de bytes que agrupam seqüencialmente os bits em trechos de 8 bits. Contudo, internamente, o estado de hash H_i é visto como uma matriz de bytes de dimensão 8×8. A transformação entre os dois é explicada mais adiante.
- **Etapa 3: Inicializar a matriz de hash.** Uma matriz de bytes é usada para manter tanto os resultados intermediários quanto os finais da função de hash. A matriz é inicializada com todos os bits em 0.
- **Etapa 4: Processar mensagem em blocos de 512 bits (64 bytes).** O centro do algoritmo é a cifra de bloco W.

Cifra de bloco W

Diferente de praticamente todas as outras propostas para uma função de hash baseada na cifra de bloco, o Whirlpool utiliza uma cifra de bloco projetada especificamente para uso em função de hash e que é improvável de ser algum dia

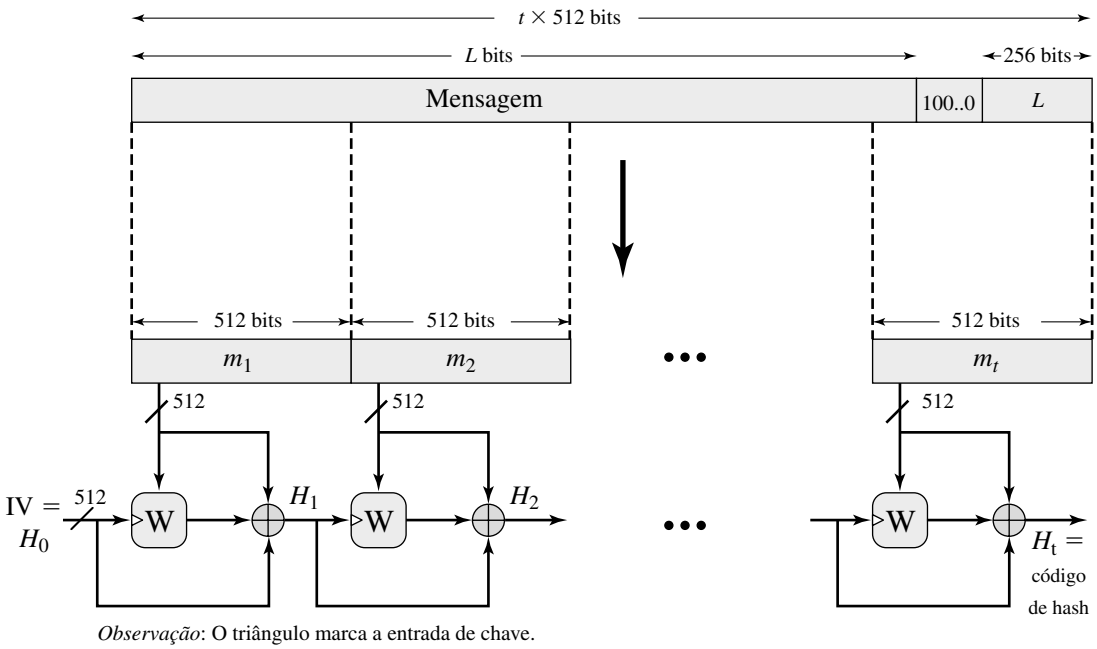


Figura 12.6 Geração de resumo da mensagem usando Whirlpool.

Tabela 12.2 Comparação da cifra de bloco do Whirlpool W e AES.

	W	AES
Tamanho do bloco (bits)	512	128
Tamanho da chave (bits)	512	128, 192 ou 256
Orientação da matriz	Entrada é mapeada por linha	Entrada é mapeada por coluna
Número de rodadas	10	10, 12 ou 14
Expansão de chave	Função de rodada W	Algoritmo de expansão dedicado
Polinômio GF(2^8)	$x^8 + x^4 + x^3 + x^2 + 1$ (011D)	$x^8 + x^4 + x^3 + x + 1$ (011B)
Origem da caixa-S	Estrutura recursiva	Inverso multiplicativo em G(2^8) mais transformação afim
Origem das constantes da rodada	Entradas sucessivas de caixa-S	Elementos 2^t de GF(2^8)
Camada de difusão	Multiplicação à direita por matriz MDS circulante 8 × 8 (1, 1, 4, 1, 8, 5, 2, 9) – linhas mistas	Multiplicação à esquerda por matriz MDS circulante 4 × 4 (2, 3, 1, 1) – colunas mistas
Permutação	Deslocar colunas	Deslocar linhas

usada como uma função de criptografia independente. O motivo para isso é que os projetistas quiseram utilizar uma cifra de bloco com a segurança e a eficiência do AES, mas com um tamanho de hash que fornecesse uma segurança em potencial igual a do SHA-512. O resultado é a cifra de bloco W, que tem uma estrutura semelhante ao AES e usa as mesmas funções elementares, mas com um tamanho de bloco e, também, um tamanho de chave de 512 bits. A Tabela 12.2 compara o AES com o W.

Embora o W seja semelhante ao AES, ele não é simplesmente uma extensão. Lembre-se de que a proposta do Rijndael para o AES definiu uma cifra na qual o tamanho do bloco e o da chave podem ser especificados como 128, 192 ou 256 bits. A especificação AES utiliza as mesmas alternativas de tamanho de chave, mas limita o tamanho do bloco a 128 bits. O AES opera sobre um estado de 4 × 4 bytes. O Rijndael com tamanho de bloco de 192 bits opera sobre um estado de 4 × 6 bytes. O Rijndael com um tamanho de bloco de 256 bits opera sobre um estado de 4 X 8 bytes. O W opera sobre um estado de 8 × 8 bytes. Quanto mais a representação do estado difere de um quadrado, mais lenta a difusão e de mais rodadas a cifra precisa. Para um tamanho de bloco de 512 bits, os desenvolvedores do Whirlpool poderiam ter definido um Rijndael operando sobre um estado de 4 × 16 bytes, mas essa cifra exigiria muitas rodadas e seria muito lenta.

Como a Tabela 12.2 indica, o W usa uma matriz orientada por linha, enquanto o AES utiliza uma matriz orientada por coluna. Não existe motivo técnico para preferir uma orientação em relação à outra, pois pode-se facilmente construir uma descrição equivalente da mesma cifra trocando linhas com colunas.

Estrutura geral

A Figura 12.7 mostra a estrutura geral de W. O algoritmo de criptografia utiliza um bloco de 512 bits de texto claro e uma chave de 512 bits como entrada e produz um bloco de 512 bits de texto cifrado como saída. O algoritmo de criptografia envolve o uso de quatro funções diferentes, ou transformações: acrescentar chave (AK), substituir bytes (SB), deslocar colunas (SC) e embaralhar colunas (MR), cujas operações são explicadas mais adiante. W consiste em uma única aplicação de AK seguida por 10 rodadas que envolvem todas as quatro funções. Podemos expressar resumidamente a operação de uma rodada r como uma RF (*round function*), que é uma composição de funções:

$$\mathrm{RF}(\boldsymbol{K}_r) = \mathrm{AK}[\boldsymbol{K}_r] \circ \mathrm{MR} \circ \mathrm{SC} \circ \mathrm{SB} \tag{12.1}$$

onde $\boldsymbol{K}_r$ é a matriz de chave da rodada para a rodada r. O algoritmo geral, com a entrada de chave **K**, pode ser definido da seguinte forma:

$$\mathrm{W}(K) = \left(\bigcirc_{r=1}^{10} \mathrm{RF}(\boldsymbol{K}_r)\right) \circ \mathrm{AK}(\boldsymbol{K}_0)$$

onde o círculo grande indica a iteração da função de composição com índice r variando de 1 a 10.

A entrada de texto claro para W é um único bloco de 512 bits. Esse bloco é tratado como uma matriz quadrada de 8 × 8 bytes, chamada **CState**. A Figura 12.8 ilustra que a ordenação dos bytes dentro de uma matriz é por linha. Assim, por exemplo, os primeiros oito bytes de uma entrada de texto claro de 512 bits para a cifra de criptografia ocupam a primeira linha da matriz interna **CState**, os próximos oito bytes ocupam a segunda linha, e assim por diante. A representação do fluxo linear de bytes como uma matriz quadrada pode ser expressa concisamente como uma função de mapeamento μ. Para um vetor de bytes linear X com elementos $x_k (0 \leq k \leq 63)$, a matriz correspondente A com elementos $a_{i,j} (0 \leq i, j \leq 7)$, temos a seguinte correspondência:

$$\mathbf{A} = \mu(\mathrm{X}) \Leftrightarrow a_{i,j} = x_{8i} + j$$

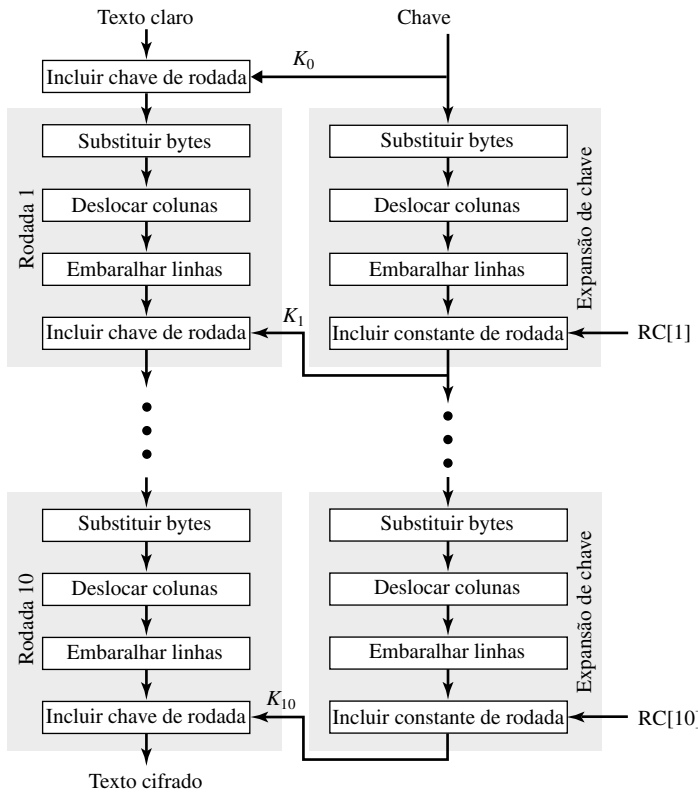


Figura 12.7 Cifra W do Whirlpool.

in_0	in_1	in_2	in_3	in_4	in_5	in_6	in_7
in_8	in_9	in_{10}	in_{11}	in_{12}	in_{13}	in_{14}	in_{15}
in_{16}	in_{17}	in_{18}	in_{19}	in_{20}	in_{21}	in_{22}	in_{23}
in_{24}	in_{25}	in_{26}	in_{27}	in_{28}	in_{29}	in_{30}	in_{31}
in_{32}	in_{33}	in_{34}	in_{35}	in_{36}	in_{37}	in_{38}	in_{39}
in_{40}	in_{41}	in_{42}	in_{43}	in_{44}	in_{45}	in_{46}	in_{47}
in_{48}	in_{49}	in_{50}	in_{51}	in_{52}	in_{53}	in_{54}	in_{55}
in_{56}	in_{57}	in_{58}	in_{59}	in_{60}	in_{61}	in_{62}	in_{63}

String de bytes de entrada

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$a_{4,6}$	$a_{4,7}$
$a_{5,0}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	$a_{5,7}$
$a_{6,0}$	$a_{6,1}$	$a_{6,2}$	$a_{6,3}$	$a_{6,4}$	$a_{6,5}$	$a_{6,6}$	$a_{6,7}$
$a_{7,0}$	$a_{7,1}$	$a_{7,2}$	$a_{7,3}$	$a_{7,4}$	$a_{7,5}$	$a_{7,6}$	$a_{7,7}$

Matriz de cifra interna **CState**

Figura 12.8 Estrutura de matriz do Whirlpool.

De modo semelhante, a chave de 512 bits é representada como uma matriz quadrada **KState** de bytes. Essa chave é usada como entrada para a função AK inicial. A chave também é expandida para um conjunto de 10 chaves de rodada, conforme explicaremos mais adiante.

Agora, vejamos as diferentes funções que fazem parte de W.

A camada não linear SB A função substituir byte (SB) é uma pesquisa de tabela simples, que oferece um mapeamento não linear. W define uma matriz 16 × 16 de valores de byte, chamada caixa-S (Tabela 12.3), que contém uma permutação de todos os 256 valores de 8 bits. Cada byte individual de **CState** é mapeado para um novo byte da seguinte maneira: os 4 bits mais à esquerda do byte são usados como um valor de linha e os 4 bits mais à direita são

usados como um valor de coluna. Esses valores de linha e coluna servem como índices para a caixa-S, para selecionar um valor de saída de 8-bits único. Por exemplo, o valor hexadecimal[3] {95} referencia a linha 9, coluna 5 da caixa-S, que contém o valor {BA}. Por conseguinte, o valor {95} é mapeado para o valor {BA}. A função SB pode ser expressa pela correspondência a seguir, para uma matriz de entrada **A** e uma matriz de saída **B**:

$$\mathbf{B} = \mathrm{SB}(\mathbf{A}) \Leftrightarrow b_{i,j} = \mathrm{S}[a_{i,j}], \quad 0 \leq i,j \leq 7$$

onde S[x] refere-se ao mapeamento do byte de entrada x no byte de saída S[x] pela caixa-S.

A caixa-S pode ser gerada pela estrutura da Figura 12.9. Ela consiste em duas camadas não lineares, cada uma contendo duas caixas-S 4×4, separadas por uma caixa 4×4 gerada aleatoriamente. Cada uma das caixas relaciona uma entrada de 4 bits a uma saída de 4 bits. A caixa E é definida como $\mathrm{E}(u) = \{\mathrm{B}\}^u$ se $u \neq \{\mathrm{F}\}$ e $\mathrm{E}(\{\mathrm{F}\}) = 0$, onde a aritmética é realizada sobre o corpo finito $\mathrm{GF}(2^4)$ com o polinômio irredutível $f(x) = x^4 + x + 1$.

A função SB foi projetada para introduzir não-linearidade ao algoritmo. Isso significa que a função SB não deverá exibir correlações entre combinações lineares dos bits de uma entrada e combinações lineares dos bits de uma saída. Além disso, as diferenças entre conjuntos de bits de entrada não devem se propagar em diferenças semelhantes entre os bits de saída correspondentes. Colocando de outra maneira, pequenas mudanças na entrada devem causar grandes mudanças na saída. Essas duas propriedades ajudam a tornar o W à prova de criptoanálise linear e diferencial.

Tabela 12.3 Whirlpool Caixa-S.

(a) Caixa-S

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	18	23	C6	E8	87	B8	01	4F	36	A6	D2	F5	79	6F	91	52
1	60	BC	9B	8E	A3	0C	7B	35	1D	E0	D7	C2	2E	4B	FE	57
2	15	77	37	E5	9F	F0	4A	CA	58	C9	29	0A	B1	A0	6B	85
3	BD	5D	10	F4	CB	3E	05	67	E4	27	41	8B	A7	7D	95	C8
4	FB	EE	7C	66	DD	17	47	9E	CA	2D	BF	07	AD	5A	83	33
5	63	02	AA	71	C8	19	49	C9	F2	E3	5B	88	9A	26	32	B0
6	E9	0F	D5	80	BE	CD	34	48	FF	7A	90	5F	20	68	1A	AE
7	B4	54	93	22	64	F1	73	12	40	08	C3	EC	DB	A1	8D	3D
8	97	00	CF	2B	76	82	D6	1B	B5	AF	6A	50	45	F3	30	EF
9	3F	55	A2	EA	65	BA	2F	C0	DE	1C	FD	4D	92	75	06	8A
A	B2	E6	0E	1F	62	D4	A8	96	F9	C5	25	59	84	72	39	4C
B	5E	78	38	8C	C1	A5	E2	61	B3	21	9C	1E	43	C7	FC	04
C	51	99	6D	0D	FA	DF	7E	24	3B	AB	CE	11	8F	4E	B7	EB
D	3C	81	94	F7	B9	13	2C	D3	E7	6E	C4	03	56	44	7F	A9
E	2A	BB	C1	53	DC	0B	9D	6C	31	74	F6	46	AC	89	14	E1
F	16	3A	69	09	70	B6	C0	ED	CC	42	98	A4	28	5C	F8	86

(b) E mini-box

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E(u)	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0

(c) E^{-1} mini-box

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\mathrm{E}^{-1}(u)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6

(d) R mini-box

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
R(u)	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

3. Como fizemos para o AES, um número hexadecimal será delimitado por chaves quando isso for necessário por motivo de clareza.

A camada de permutação SC A camada de permutação (deslocar colunas) causa um deslocamento circular para baixo de cada coluna de **CState**, exceto a primeira coluna. Para a segunda coluna, é realizado um deslocamento circular para baixo de 1 byte; para a terceira coluna, é realizado um deslocamento circular para baixo de 3 bytes, e assim por diante. A função SC pode ser expressa pela seguinte correspondência, para uma matriz de entrada **A** e uma matriz de saída **B**:

$$\mathbf{B} = \mathrm{SC}(\mathbf{A}) \Leftrightarrow b_{i,j} = a_{(i-j) \bmod 8, j} \quad 0 \leq \mathrm{i}, \mathrm{j} \leq 7$$

A transformação deslocar coluna é mais substancial do que pode parecer a princípio. Isso porque **CState** é tratado como um vetor de 8 linhas de 8 bytes. Assim, na criptografia, os primeiros 8 bytes do texto claro são copiados para a primeira linha de **CState**, e assim por diante. Um deslocamento de coluna move um byte individual de uma linha para outra, que é uma distância linear de um múltiplo de 8 bytes. Observe também que a transformação garante que os 8 bytes de uma linha sejam espalhados para 8 linhas diferentes.

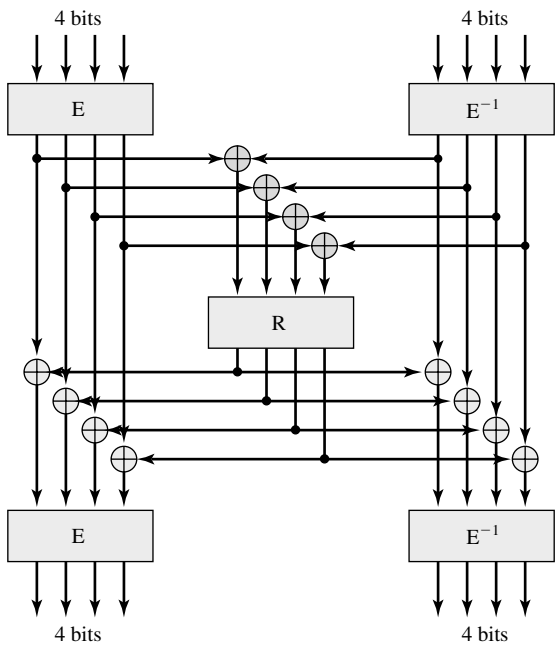


Figura 12.9 Implementação da caixa-S do Whirlpool.

A camada de difusão MR Lembre-se, do Capítulo 3, de que para uma função que exibe difusão, a estrutura estatística da entrada é dissipada em estatísticas de longa duração da saída. Isso é obtido fazendo com que cada bit afete o valor de muitos bits de saída; geralmente, isso resulta em cada bit de saída sendo afetado por muitos bits de entrada. A camada de difusão (embaralhar linhas) obtém a difusão dentro de cada linha individualmente. Cada byte de uma linha é mapeado em um novo valor que é uma função de todos os 8 bytes nessa linha. A transformação pode ser definida pela multiplicação de matriz: **B** = **AC**, onde **A** é a matriz de entrada, **B** é a matriz de saída e **C** é a matriz de transformação:

$$\mathrm{C} = \begin{bmatrix} 01 & 01 & 04 & 01 & 08 & 05 & 02 & 09 \\ 09 & 01 & 01 & 04 & 01 & 08 & 05 & 02 \\ 02 & 09 & 01 & 01 & 04 & 01 & 08 & 05 \\ 05 & 02 & 09 & 01 & 01 & 04 & 01 & 08 \\ 08 & 05 & 02 & 09 & 01 & 01 & 04 & 01 \\ 01 & 08 & 05 & 02 & 09 & 01 & 01 & 04 \\ 04 & 01 & 08 & 05 & 02 & 09 & 01 & 01 \\ 01 & 04 & 01 & 08 & 05 & 02 & 09 & 01 \end{bmatrix}$$

Cada elemento na matriz de produtos é a soma dos produtos dos elementos de uma linha e uma coluna. Nesse caso, as adições e multiplicações individuais[4] são realizadas em GF(2^8) com o polinômio irredutível $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. Como um exemplo da multiplicação de matriz envolvida, o primeiro elemento da matriz de saída é

$$b_{0,0} = a_{0,0} \oplus (9 \cdot a_{0,1}) \oplus (2 \cdot a_{0,2}) \oplus (5 \cdot a_{0,3}) \oplus (8 \cdot a_{0,4}) \oplus a_{0,5} \oplus (4 \cdot a_{0,6}) \oplus a_{0,7}$$

Observe que cada linha de C é construída por meio de um deslocamento circular à direita da linha anterior. C é projetado para ser uma matriz de **distância máxima separável** (MDS). Na área de códigos de correção de erro, um código MDS tem como entrada uma string de bits de tamanho fixo e produz uma string de saída expandida de modo que haja a distância máxima de Hamming entre os pares de strings de saída. Com um código MDS, até mesmo vários erros de bit resultam em um código mais próximo do valor correto do que algum outro valor. No contexto de cifras em bloco, uma matriz de transformação construída usando um código MDS oferece um alto grau de difusão [JUNO04]. O uso de códigos MDS para oferecer alta difusão foi proposto inicialmente em [RIJM96].

A matriz C é uma matriz MDS que possui tantos elementos 1 quanto possível (3 por linha). Em geral, os coeficientes em C oferecem implementação de hardware eficiente.

A camada adicionar chave AK Na camada adicionar chave, é feito um OU EXCLUSIVO bit a bit entre os 512 bits de **CState** e os 512 bits da chave da rodada. A função AK pode ser expressa pela seguinte correspondência, para uma matriz de entrada **A**, uma matriz de saída **B** e uma chave de rodada K_i:

$$\mathbf{B} = \mathrm{AK}[K_i](\mathbf{A}) \Leftrightarrow b_{i,\,j} = a_{i,j} \oplus k_{i,j}, \quad 0 \le i, j \le 7$$

Expansão de chave para a cifra de bloco W Como vemos na Figura 12.7, a expansão de chave é obtida por meio da própria cifra de bloco, com uma constante de rodada servindo como chave de rodada para a expansão. A constante para a rodada $r(1 \le r \le 10)$ é uma matriz $\mathbf{RC}[r]$ em que somente a primeira linha é diferente de zero, e é definida da seguinte forma:

$$\mathrm{rc}[r]_{0,j} = \mathrm{S}[8(r-1) + j], \quad 0 \le j \le 7, 1 \le r \le 10$$

$$\mathrm{rc}[r]_{i,j} = 0, \qquad 1 \le i \le 7, 0 \le j \le 7, 1 \le r \le 10$$

Cada elemento da primeira linha é um mapeamento usando a caixa-S. Assim, a primeira linha de **RC**[1] é

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]	=
18	23	C6	E8	87	B8	01	4F	

Usando as constantes de rodada, o schedule de chave expande a chave de cifra de 512 bits K para uma seqüência de chaves de rodada $\mathbf{K}_0, \mathbf{K}_1, ..., \mathbf{K}_{10}$:

$$K_0 = K$$
$$K_r = \mathrm{RF}[\mathbf{RC}[r]](K_{r-1})$$

onde RF é a função de rodada definida na Equação (12.1). Observe que, para a fase AK de cada rodada, somente a primeira linha de **KState** é alterada.

Desempenho do Whirlpool

Ainda existe muito pouca experiência de implementação com o Whirlpool. Lembre-se de que a avaliação do NIST sobre o Rijndael determinou que ele mostrava bom desempenho em hardware e em software e que era adequado a requisitos de espaço restrito (pouca memória). Esses critérios foram importantes na seleção do Rijndael para o AES. Como o Whirlpool utiliza os mesmos elementos básicos funcionais do AES e tem a mesma estrutura, podemos esperar características semelhantes de desempenho e de requisitos de espaço.

Um estudo realizado encontra-se em [KITS04]. Os autores compararam o Whirlpool com uma série de outras funções de hash seguras, incluindo todas as versões do SHA. Os autores desenvolveram diversas implementações de hardware de cada função de hash e concluíram que, em comparação com o SHA-512, o Whirlpool exige mais recursos de hardware, mas funciona muito melhor em termos de saída (*throughput*).

4. Da mesma forma como procedemos para o AES, usaremos os símbolos · para indicar multiplicação sobre o corpo finito GF(2^8) e ⊕ para indicar XOR bit a bit, que corresponde à adição em GF(2^8).

12.3 HMAC

No Capítulo 11, examinamos um exemplo de um código de autenticação de mensagem (MAC) com base no uso de uma cifra de bloco simétrica, a saber, o Data Authentication Algorithm, definido no FIPS PUB 113. Essa, tradicionalmente tem sido a técnica mais comum para construir um MAC. Nos últimos anos, tem havido um interesse cada vez maior no desenvolvimento de um MAC derivado de uma função de hash criptográfica. As motivações para esse interesse são:

1. Funções de hash criptográficas, como MD5 e SHA-1, geralmente são executadas mais rapidamente no software do que as cifras de bloco simétricas, como DES.
2. Existe muito código de biblioteca à disposição para funções de hash criptográficas.

Com o desenvolvimento do AES e a disponibilidade maior do código para algoritmos de criptografia, essas considerações são menos significativas, mas os MACs baseados em hash continuam a ser bastante utilizados.

Uma função de hash como SHA não foi projetada para uso como um MAC e não pode ser usada diretamente para essa finalidade, pois não depende de uma chave secreta. Tem havido diversas propostas para a incorporação de uma chave secreta em um algoritmo de hash existente. A técnica que obteve o maior suporte é HMAC [BELL96a, BELL96b]. O HMAC foi publicado como a RFC 2104, escolhido como MAC de implementação obrigatória para segurança IP, e é usado em outros protocolos da Internet, como SSL. O HMAC também foi publicado como um padrão do NIST (FIPS 198).

Objetivos de projeto do HMAC

A RFC 2104 relaciona os seguintes objetivos de projeto para o HMAC:

- Usar, sem modificações, as funções de hash disponíveis, particularmente aquelas que tenham bom desempenho em software, e para as quais o código esteja ampla e gratuitamente disponível.
- Permitir a substituição fácil da função de hash embutida caso sejam encontradas ou exigidas funções de hash mais rápidas ou mais seguras.
- Preservar o desempenho original da função de hash sem incorrer em degradação significativa.
- Usar e gerenciar as chaves de maneira simples.
- Ter uma análise criptográfica bem compreendida quanto à força do mecanismo de autenticação com base em suposições razoáveis sobre a função de hash embutida.

Os dois primeiros objetivos são importantes para a aceitação do HMAC. O HMAC trata a função de hash como uma 'caixa-preta'. Isso tem dois benefícios. Primeiro, uma implementação existente de uma função de hash pode ser usada como um módulo na implementação do HMAC, assim a maior parte do código HMAC estará pré-empacotado e pronto para uso sem modificação. Segundo, se algum dia for desejado substituir determinada função de hash em uma implementação HMAC, será preciso apenas remover o módulo da função de hash existente e incluir o novo módulo. Isto poderá ser feito se uma função de hash mais rápida for desejada. Mais importante: se a segurança da função de hash embutida for comprometida, a segurança do HMAC poderá ser retomada simplesmente substituindo-se a função de hash embutida por uma mais segura (por exemplo, substituindo o SHA pelo Whirlpool).

O último objetivo de projeto na lista anterior é, na verdade, a principal vantagem do HMAC em relação a outros esquemas baseados em hash. Pode-se comprovar que o HMAC é seguro desde que a função de hash embutida tenha uma força criptográfica razoável. Retornaremos a esse ponto mais adiante, nesta seção, mas primeiramente examinaremos a estrutura do HMAC.

Algoritmo HMAC

A Figura 12.10 ilustra a operação geral do HMAC. Definimos os seguintes termos:

H	= função de hash embutida (por exemplo, MD5, SHA-1, RIPEMD-160)
IV	= entrada de valor inicial para a função de hash
M	= entrada de mensagem para HMAC (incluindo a complementação especificada na função de hash embutida)
Y_i	= *i-ésimo* bloco de M, $0 \leq i \leq (L - 1)$
L	= número de blocos em M
b	= número de bits em um bloco
n	= comprimento do código de hash produzido pela função de hash embutida
K	= tamanho recomendado da chave secreta é $\geq n$; se o tamanho da chave for maior que b; a função de hash será aplicada à chave para produzir uma chave de n bits
K^+	= K complementado com zeros à esquerda de modo que o resultado tenha b bits de extensão
ipad	= 00110110 (36 em hexadecimal) repetido $b/8$ vezes
opad	= 01011100 (5C em hexadecimal) repetido $b/8$ vezes

Então, o HMAC pode ser expresso da seguinte forma:

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

1. Acrescente zeros à extremidade esquerda de K para criar uma string de b bits K^+ (por exemplo, se K tiver o tamanho de 160 bits e $b = 512$, então K será anexado com 44 bytes zero 0 X 00).
2. Faça o XOR (OU exclusivo bit a bit) com ipad para produzir o bloco de b bits S_i.
3. Anexe M a S_i.
4. Aplique H ao fluxo gerado na etapa 3.
5. Faça o OU EXCLUSIVO de K^+ com opad para produzir o bloco de b bits S_o.
6. Anexe o resultado do hash da etapa 4 a S_o.
7. Aplique H ao fluxo gerado na etapa 6 e apresente o resultado na saída.

Observe que o XOR com ipad resulta na inversão de metade dos bits de K. De modo semelhante, o XOR com opad resulta na inversão de metade dos bits de K, mas um conjunto diferente de bits. Na verdade, passando S_i e S_o pela função de compressão do algoritmo de hash, temos duas chaves geradas pseudo-aleatoriamente a partir de K.

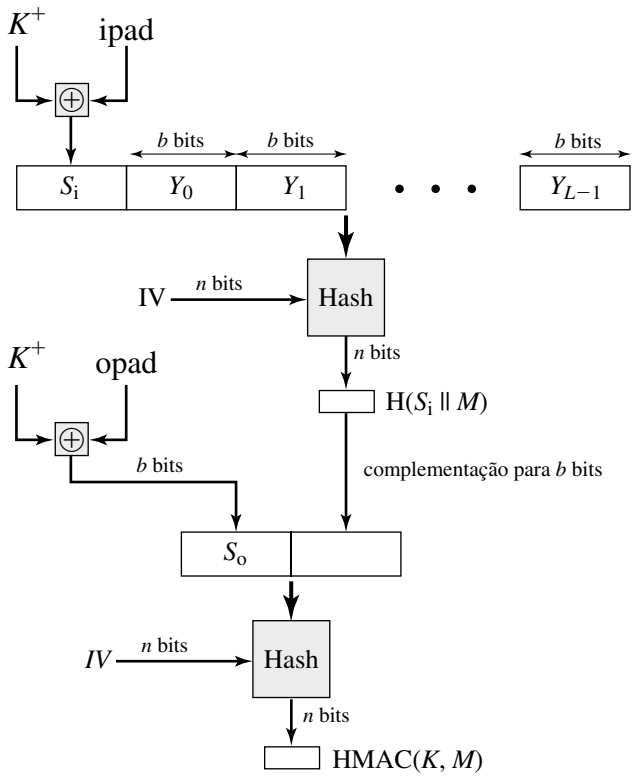


Figura 12.10 Estrutura do HMAC.

HMAC deverá ser executado aproximadamente ao mesmo tempo em que a função de hash embutida para mensagens longas. O HMAC acrescenta três execuções da função de compressão de hash (para S_i, software e o bloco produzido pelo hash interno).

Uma implementação mais eficiente é possível, como mostra a Figura 12.11. Duas quantidades são pré-calculadas:

$$\text{f}(\text{IV}, (K^+ \oplus \text{ipad}))$$
$$\text{f}(\text{IV}, (K^+ \oplus \text{opad}))$$

onde f(cv, bloco) é a função de compressão para a função de hash, que usa como argumentos uma variável encadeada de n bits e um bloco de b bits e produz uma variável encadeada de n bits. Essas quantidades só precisam ser calculadas inicialmente e toda vez em que a chave mudar. Na realidade, as quantidades pré-calculadas substituem o valor inicial (IV) na função de hash. Com essa implementação, somente uma instância adicional da função de compressão é acrescentada ao processamento normalmente produzido pela função de hash. Essa implementação mais eficiente é especialmente valiosa se a maioria das mensagens para as quais um MAC é calculado for curta.

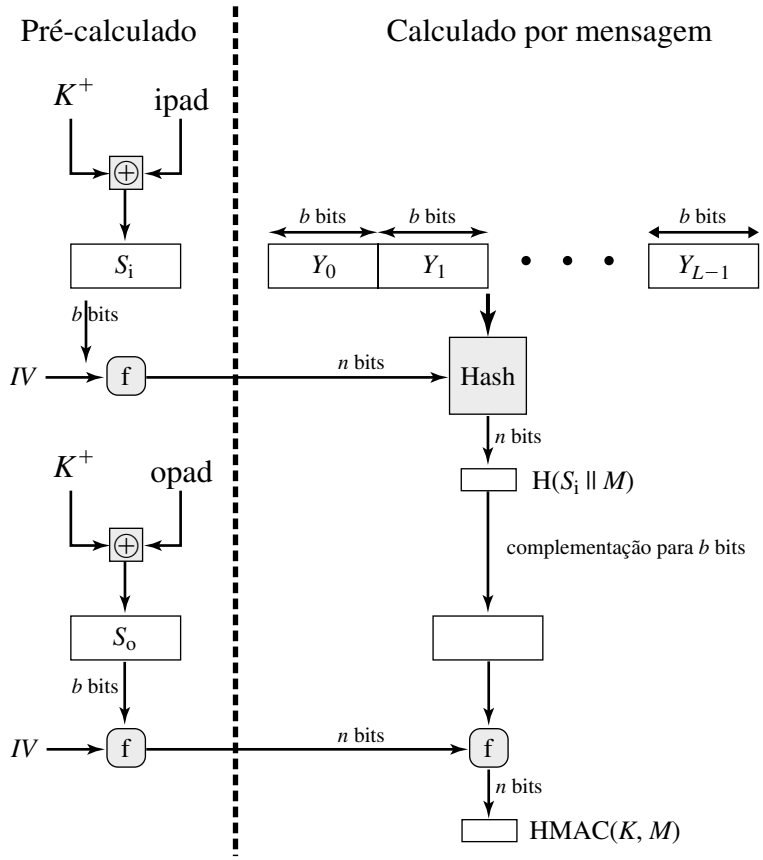


Figura 12.11 Implementação eficiente do HMAC.

Segurança do HMAC

A segurança de qualquer função MAC baseada em uma função de hash embutida depende de alguma maneira da força criptográfica da função de hash utilizada internamente. O atrativo do HMAC é que seus projetistas foram capazes de provar um relacionamento exato entre a força da função de hash embutida e a força do HMAC.

A segurança de uma função MAC geralmente é expressa em termos da probabilidade de falsificação bem-sucedida com um determinado tempo gasto pelo falsificador e um determinado número de pares mensagem-MAC criados com a mesma chave. Basicamente, é provado em [BELL96a] que, para determinado grau de esforço (tempo, pares mensagem-MAC) para mensagens geradas por um usuário legítimo e visualizadas pelo atacante, a probabilidade de um ataque bem-sucedido ao HMAC é equivalente a um dos seguintes ataques sobre a função de hash embutida:

1. O atacante é capaz de calcular uma saída da função de compressão mesmo com um IV aleatório, secreto e desconhecido pelo atacante.
2. O atacante encontra colisões na função de hash mesmo quando o IV é aleatório e secreto.

No primeiro ataque, podemos ver a função de compressão como equivalente à função de hash aplicada a uma mensagem que consiste em um único bloco de b bits. Para esse ataque, o IV da função de hash é substituído por um valor secreto e aleatório de n bits. Um ataque a essa função de hash requer um ataque por força bruta à chave, que é um esforço da ordem de 2^n, ou um ataque de aniversário, que é um caso especial do segundo ataque, explicado em seguida.

No segundo caso, o atacante está procurando duas mensagens M e M', que produzam o mesmo hash. Esse é um ataque de aniversário, explicado no Capítulo 11. Mostramos que isso exige um esforço de $2^{n/2}$ para um tamanho de hash de n. Com base nisso, a segurança do MD5 é colocada em questão, pois um esforço de 2^{64} parece compatível com a tecnologia de hoje. Isso significa que uma função de hash de 128 bits, como MD5, é inadequada para o HMAC? A resposta é não, porque, para atacar o MD5, o atacante pode escolher qualquer conjunto de mensagens e trabalhar nelas off-line, em uma instalação de computação dedicada para encontrar uma colisão. Como o atacante conhece o algoritmo de hash e o IV padrão, ele pode obter o código de hash para cada uma das mensagens que gerar. Porém, ao atacar o HMAC, o atacante não pode gerar pares de mensagem/código off-line, pois ele não conhece K. Portanto, o atacante terá de observar uma seqüência de mensagens geradas pelo HMAC sob a mesma chave e realizar o ata-

que a essas mensagens conhecidas. Para um tamanho de código de hash de 128 bits, são necessários 2^{64} blocos observados (2^{72} bits), gerados usando-se a mesma chave. Em um link de 1 Gbps, seria preciso observar um fluxo contínuo de mensagens sem mudança na chave por cerca de 150.000 anos para ter sucesso. Assim, se a velocidade for um problema, é totalmente aceitável usar MD5 em vez de SHA-1 como função de hash embutida para HMAC.

12.4 CMAC

O Data Authentication Algorithm definido no FIPS PUB 113, também conhecido como CBC-MAC (*cipher block chaining message authentication code*), é descrito no Capítulo 11. Esse MAC baseado em cifra tem sido muito adotado pelo governo e pela indústria.[BELL00] demonstrou que esse MAC é seguro sob um conjunto razoável de critérios de segurança, com a seguinte restrição: somente mensagens de um tamanho fixo de mn bits são processadas, onde n é o tamanho do bloco cifrado e m é um inteiro positivo fixo. Como exemplo, observe que, dado o CBC-MAC de uma mensagem de um bloco X, digamos $T = \text{MAC}(K,X)$, o adversário sabe imediatamente o CBC-MAC para a mensagem de dois blocos $X \| (X \oplus T)$, pois este é mais uma vez T.

Black e Rogaway [BLAC00] demonstraram que essa limitação pode ser contornada com o uso de três chaves: uma chave de comprimento k, para ser usada em cada etapa do encadeamento da cifra de bloco, e duas chaves de comprimento n, onde k é o tamanho da chave e n é o tamanho do bloco cifrado. A construção proposta foi refinada por Iwata e Kurosawa, de modo que as duas chaves de n bits poderiam ser derivadas da chave de criptografia, em vez de serem fornecidas separadamente [IWAT03]. Essa melhoria foi adotada pelo modo de operação CMAC (*cipher-based message authentication code*) do NIST, para uso com o AES e o triple DES. Ela é especificada na NIST Special Publication 800-38B.

Primeiramente, vamos considerar a operação do CMAC quando a mensagem é um múltiplo inteiro n do comprimento da cifra de bloco b. Para o AES, $b = 128$, e para o triple DES, $b = 64$. A mensagem é dividida em n blocos, $M_1, M_2, ..., M_n$. O algoritmo utiliza uma chave de criptografia de k bits, K, e uma constante de n bits, K_1. Para o AES, o tamanho da chave k é 128, 192 ou 256 bits; para o triple DES, o tamanho da chave é 112 ou 168 bits. CMAC é calculado da seguinte forma (Figura 12.12a):

$$
\begin{aligned}
C_1 &= \text{E}(K, M_1) \\
C_2 &= \text{E}(K, [M_2 \oplus C_1]) \\
C_3 &= \text{E}(K, [M_3 \oplus C_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
C_n &= \text{E}(K, [M_N \oplus C_{n-1} \oplus K_1]) \\
T &= \text{MSB}_{Tlen}(C_n)
\end{aligned}
$$

onde

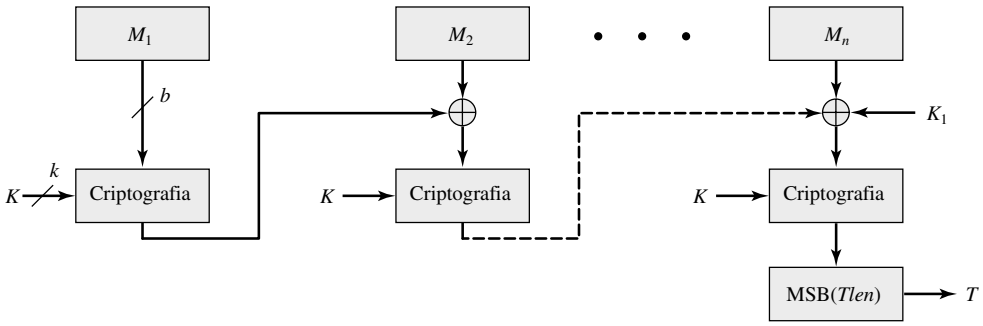
T = código de autenticação de mensagem, também conhecido como tag
$Tlen$ = comprimento de T em bits
$\text{MSB}_s(X)$ = os s bits mais à esquerda da string de bits X

Se a mensagem não for um múltiplo inteiro do tamanho do bloco de cifra, então o bloco final será complementado à direita (bits menos significativos) com 1 e tantos 0s quantos forem necessários para que o bloco final também seja de tamanho b. A operação CMAC, então, prossegue como antes, exceto pelo uso de uma chave de n bits diferente, K_2, no lugar de K_1.

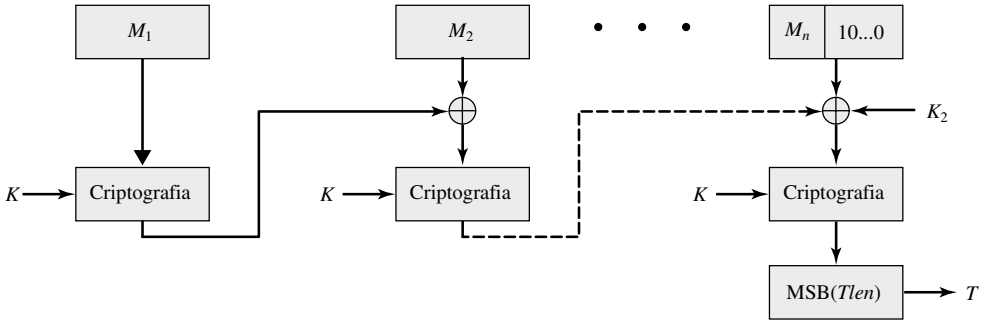
As duas chaves de n bits são derivadas da chave de criptografia de k bits da seguinte forma:

$$
\begin{aligned}
L &= \text{E}(K, 0^n) \\
K_1 &= L \bullet x \\
K_2 &= L \bullet x^2 = (L \bullet x) \bullet x
\end{aligned}
$$

onde a multiplicação ($\bullet$) é feita no corpo finito $\text{GF}(2^n)$ e x e x^2 são polinômios de primeiro e segundo graus, que são elementos de $\text{GF}(2^n)$. Assim, a representação binária de x consiste em $n - 2$ zeros seguidos por 10; a representação binária de x^2 consiste em $n - 3$ zeros seguidos por 100. O corpo finito é definido com relação a um polinômio irredutível que vem lexicograficamente primeiro entre todos os polinômios com o menor número possível de termos diferentes de zero. Para os dois tamanhos de bloco aprovados, os polinômios são $x^{64} + x^4 + x^3 + x + 1$ e $x^{128} + x^7 + x^2 + x + 1$.



(a) O tamanho da mensagem é múltiplo inteiro do tamanho do bloco



(b) O tamanho da mensagem não é múltiplo inteiro do tamanho do bloco

Figura 12.12 Cipher-based message authentication code (CMAC).

Para gerar K_1 e K_2, a cifra em bloco é aplicada ao bloco que consiste inteiramente em bits 0. A primeira subchave é derivada do texto cifrado resultante do deslocamento esquerdo de um bit e, condicionalmente, pelo XOR de uma constante que depende do tamanho do bloco. A segunda subchave é, da mesma maneira, derivada da primeira. Essa propriedade dos corpos finitos da forma GF(2^n) foi explicada em MixColumns, no Capítulo 5.

12.5 LEITURA E SITES WEB RECOMENDADOS

[GILB03] examina a segurança do SHA-256 através do SHA-512. Visões gerais do HMAC podem ser encontradas em [BELL96a] e [BELL96b].

BELL96a Bellare, M.; Canetti, R.; e Krawczyk, H. "Keying hash functions for message authentication". *Proceedings, CRYPTO '96*, agosto de 1996; publicado por Springer-Verlag. Uma versão ampliada está disponível em **http://www-cse.ucsd.edu/users/mihir**.

BELL96b Bellare, M.; Canetti, R.; e Krawczyk, H. "The HMAC construction". *CryptoBytes*, Spring 1996.

GILB03 Gilbert, H. e Handschuh, H. "Security analysis of SHA-256 and sisters". Proceedings, CRYPTO '03, 2003; publicado por Springer-Verlag.

Sites Web recomendados

- **NIST Secure Hashing Page:** SHA FIPS e documentos relacionados
- **Whirlpool:** várias informações sobre Whirlpool
- **Block cipher modes of operation:** página do NIST com todas as informações sobre CMAC

12.6 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

big endian	little endian	SHA-256
CMAC	MD4	SHA-384
função de compressão	MD5	SHA-512
HMAC	SHA-1	Whirlpool

Perguntas de revisão

12.1 Qual é a diferença entre os formatos little-endian e big-endian?

12.2 Que funções aritméticas e lógicas básicas são usadas no SHA?

12.3 Que funções aritméticas e lógicas básicas são usadas no Whirlpool?

12.4 Por que tem havido interesse no desenvolvimento de um código de autenticação de mensagens derivado de uma função de hash criptográfica, ao contrário de um derivado de uma cifra simétrica?

12.5 Que mudanças no HMAC são exigidas para substituir uma função de hash interna por outra?

Problemas

12.1 Na Figura 12.4, supusemos que um vetor de 80 palavras de 64 bits está disponível para armazenar os valores de W_t, de modo que possam ser pré-calculados no início do processamento de um bloco. Agora, considere que o espaço é restrito. Como alternativa, considere o uso de um buffer circular de 16 palavras que é carregado inicialmente com W_0 até W_{15}. Crie um algoritmo que, para cada etapa t, calcule o valor de entrada exigido W_t.

12.2 Para o SHA-512, mostre as equações dos valores de $W_{16}, W_{17}, W_{18}, W_{19}$.

12.3 Suponha que $a_1\ a_2\ a_3\ a_4$ sejam os 4 bytes em uma palavra de 32 bits. Cada a_i pode ser visto como um inteiro no intervalo de 0 a 255, representado em binário. Em uma arquitetura big-endian, essa palavra representa o inteiro

$$a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$$

Em uma arquitetura little-endian, essa palavra representa o inteiro

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$$

a. Algumas funções de hash, como MD5, assumem uma arquitetura little-endian. É importante que o resumo da mensagem seja independente da arquitetura utilizada. Portanto, para realizar a operação de adição módulo 2 do MD5 ou RIPEMD-160 em uma arquitetura big-endian, é preciso fazer um ajuste. Suponha que $X = x_1\ x_2\ x_3\ x_4$ e $Y = y_1\ y_2\ y_3\ y_4$. Mostre como a operação de adição do MD5 (X + Y) seria executada em uma máquina big-endian.

b. O SHA assume uma arquitetura big-endian. Mostre como a operação (X + Y) para o SHA seria executada em uma máquina little-endian.

12.4 Este problema introduz uma função de hash semelhante ao SHA a qual opera sobre letras em vez de dados binários. Ela é chamado de *toy tetragraph hash* (tth).[5] Dada uma mensagem que consiste em uma seqüência de letras, o tth produz um valor de hash que consiste de quatro letras. Primeiro, o tth divide a mensagem em blocos de 16 letras, ignorando espaços, pontuação e letras maiúsculas iniciais. Se o tamanho da mensagem não for divisível por 16, ela será complementada com nulos. Um total acumulado de quatro números é mantido, começando com o valor (0, 0, 0, 0); isso é colocado como entrada na função de compactação para o processamento do primeiro bloco. A função de compactação consiste em duas rodadas. **Rodada 1:** Apanhe o próximo bloco de texto, organize-o como um bloco de texto de 4 × 4 linha por linha e converta-o em números (A = 0, B = 1 etc.). Por exemplo, para o bloco ABCDEFGHIJKLMNOP, temos:

A	B	C	D	0	1	2	3
E	F	G	H	4	5	6	7
I	J	K	L	8	9	10	11
M	N	O	P	12	13	14	15

Depois, some cada coluna mod 26 e some o resultado ao total acumulado, mod 26. Neste exemplo, o total acumulado é (24, 2, 6, 10). **Rodada 2:** Usando a matriz da rodada 1, gire a primeira linha à esquerda por 1, a segunda linha à esquerda por 2, a terceira linha à esquerda por 3 e inverta a ordem da quarta linha. Em nosso exemplo:

B	C	D	A	1	2	3	0
G	H	E	F	6	7	4	5
L	I	J	K	11	8	9	10
P	O	N	M	15	14	13	12

Agora, some cada coluna mod 26 e some o resultado ao total acumulado. O novo total acumulado é (5, 7, 9, 11). Esse total acumulado, agora, é a entrada para a primeira rodada da função de compressão para o próximo bloco de texto. Depois que o bloco final for processado, converta o total acumulado final em letras. Por exemplo, se a mensagem for ABCDEFGHIJKLMNOP, então o hash será FHJL.

5. Agradeço a William K. Mason, da equipe da revista *The Cryptogram*, por fornecer este exemplo.

a. Desenhe figuras semelhantes às Figuras 12.1 e 12.2 para representar a lógica tth geral e a lógica da função de compactação.
b. Calcule a função de hash para a mensagem de 48 letras "I leave twenty million dollars to my friendly cousin Bill".
c. Para demonstrar os pontos fracos do tth, encontre um bloco de 48 letras que produza o mesmo hash que acabamos de obter. *Dica:* use muitos 'A'.

12.5 Desenvolva uma tabela semelhante à Tabela 4.8 para $GF(2^8)$ com $m(x) = x^8 + x^4 + x^3 + x^2 + 1$.

12.6 Mostre as minicaixas E e E^{-1} na Tabela 12.3 no formato tradicional de matriz quadrada da caixa-S, como a da Tabela 5.4.

12.7 Verifique se a Figura 12.9 é uma implementação válida da caixa-S mostrada na Tabela 12.3a. Faça isso mostrando os cálculos envolvidos para três valores de entrada: 00, 55, 1E.

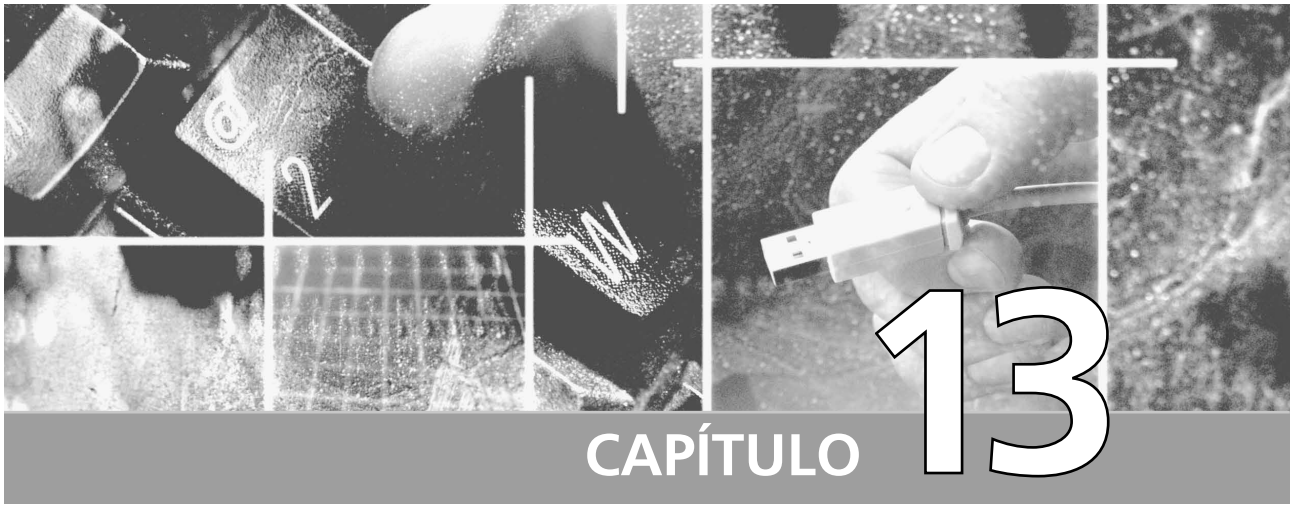
12.8 Forneça uma expressão booleana que defina a funcionalidade da caixa-S da Figura 12.9.

12.9 O Whirlpool utiliza a construção $H_i = E(H_{i-1}, M_i) \oplus H_{i-1} \oplus M_i$. Outra construção mostrada por Preneel como sendo segura é $H_i = E(H_{i-1}, M_i) \oplus M_i$. Agora observe que o schedule da chave para o Whirlpool é semelhante à criptografia da chave cifrada sob uma pseudochave definida pelas constantes da rodada, de modo que o núcleo do processo de hashing poderia ser visto formalmente como duas linhas de criptografia interagindo. Considere a criptografia $E(H_{i-1}, M_i)$. Poderíamos escrever a chave da rodada final para esse bloco como $K_{10} = E(RC, H_{i-1})$. Agora, mostre que as duas construções de hash são essencialmente equivalentes devido ao modo como o schedule da chave é definido.

12.10 No começo da Seção 12.4, foi observado que, dado o CBC-MAC de uma mensagem de um bloco X, digamos $T = MAC(K,X)$, o adversário imediatamente conhece o CBC-MAC para a mensagem de dois blocos $X \| (X \oplus T)$, pois esta é mais uma vez T. Justifique essa afirmativa.

12.11 Neste problema, demonstramos que, para o CMAC, uma variante que realiza o XOR da segunda chave após aplicar a criptografia final não funciona. Vamos considerar isso no caso de a mensagem ser um múltiplo inteiro do tamanho do bloco. Então, a variante pode ser expressa como $VMAC(K, M) = CBC(K, M) \oplus K_1$. Agora, suponha que um adversário seja capaz de solicitar os MACs das três mensagens: a mensagem $\mathbf{0} = 0^n$, onde n é o tamanho do bloco cifrado; a mensagem $\mathbf{1} = 1^n$, e a mensagem $\mathbf{1} \| \mathbf{0}$. Como resultado dessas três consultas, o adversário consegue $T_0 = CBC(K, \mathbf{0}) \oplus K_1$; $T_1 = CBC(K, \mathbf{1}) \oplus K_1$, e $T_2 = CBC(K, [CBC(K, \mathbf{1})]) \oplus K_1$. Mostre que o adversário pode calcular o MAC correto para a mensagem (não pesquisada) $\mathbf{0} \| (T_0 \oplus T_1)$.

12.12 Na explicação sobre a geração de subchave no CMAC, há a afirmação de que a cifra de bloco é aplicada ao bloco que consiste inteiramente de bits 0. A primeira subchave é derivada da string resultante por um deslocamento à esquerda de um bit e, condicionalmente, fazendo o XOR com uma constante que depende do tamanho do bloco. A segunda subchave é derivada da mesma maneira da primeira subchave.
a. Que constantes são necessárias para os tamanhos de bloco 64 e 128 bits?
b. Explique como o deslocamento à esquerda e o XOR permitem chegar ao resultado desejado.



Assinaturas digitais e protocolos de autenticação

13.1 **Assinaturas digitais**

13.2 **Protocolos de autenticação**

13.3 **Padrão de assinatura digital**

13.4 **Leitura e sites Web recomendados**

13.5 **Principais termos, perguntas de revisão e problemas**

Proteger-se contra a influência maligna exercida por estranhos é, portanto, um preceito elementar de prudência selvagem. Logo, antes que estranhos tenham permissão para entrar em um distrito, ou pelo menos antes que tenham permissão para se misturar livremente com seus habitantes, certas cerimônias normalmente são realizadas pelos nativos do país, com a finalidade de tirar dos estranhos seus poderes mágicos, ou desinfectar, por assim dizer, a atmosfera contaminada pela qual eles, supostamente, estão cercados.

The golden bough, Sir James George Frazer

PONTOS PRINCIPAIS

- Uma assinatura digital é um mecanismo de autenticação que permite ao criador de uma mensagem anexar um código que atue como uma assinatura. A assinatura é formada tomando o hash da mensagem e criptografando-a com a chave privada do criador. A assinatura garante a origem e a integridade da mensagem.
- Protocolos de autenticação mútua permitem que as partes da comunicação se convençam mutuamente da identidade umas das outras e também para a troca de chaves de sessão.
- Na autenticação unidirecional, o destinatário deseja alguma garantia de que uma mensagem vem realmente do emissor alegado.
- O padrão de assinatura digital (DSS — *Digital Signature Standard*) é um padrão do NIST que usa o algoritmo de hash seguro (SHA — Secure Hash Algorithm).

O desenvolvimento mais importante do trabalho em criptografia de chave pública é a assinatura digital. A assinatura digital oferece um conjunto de recursos de segurança que seria difícil implementar de qualquer outra maneira. Iniciamos este capítulo com uma visão geral das assinaturas digitais. Depois, examinamos os protocolos de autenticação, muitos dos quais dependem do uso da assinatura digital. Finalmente, introduzimos o *Digital Signature Standard* (DSS).

13.1 ASSINATURAS DIGITAIS

Requisitos

A autenticação da mensagem protege duas partes que trocam mensagens contra uma terceira parte. Porém, ela não protege as duas partes uma da outra e diferentes formas de disputa são possíveis entre as duas.

Por exemplo, suponha que John envie uma mensagem autenticada a Mary, usando um dos esquemas apresentados na Figura 11.4. Considere as seguintes disputas que poderiam surgir:

1. Mary pode forjar uma mensagem diferente e reivindicar que ela veio de John. Mary simplesmente teria de criar uma mensagem e anexar um código de autenticação usando a chave que John e Mary compartilham.
2. John pode negar o envio da mensagem. Uma vez que é possível para Mary falsificar uma mensagem, não há como provar que John realmente a enviou.

Esses dois cenários são preocupações legítimas. Eis um exemplo do primeiro cenário: ocorre uma transferência eletrônica de fundos e o receptor aumenta a quantidade de fundos transferidos e afirma que o valor maior chegou do emissor. Um exemplo do segundo cenário é que um e-mail contém instruções para um agente de valores para uma transação que mais tarde resultou desfavorável. O emissor finge que o e-mail nunca foi enviado.

Em situações nas quais não existe confiança mútua e completa entre emissor e receptor, é necessário algo mais do que a autenticação. A solução mais adequada para esse problema é a assinatura digital, que é semelhante à assinatura escrita a mão. Ela precisa ter as seguintes características:

- Deve verificar o autor, a data e hora da assinatura.
- Deve autenticar o conteúdo no momento da assinatura.
- Deve ser verificável por terceiros, para resolver disputas.

Assim, a função de assinatura digital inclui a função de autenticação.

Com base nessas propriedades, podemos formular os seguintes requisitos para uma assinatura digital:

- Ela precisa ser um padrão de bits que dependa da mensagem que será assinada.
- Precisa usar alguma informação exclusiva do emissor, para impedir tanto a falsificação quanto a retratação.
- Deve ser relativamente fácil produzi-la.
- Deve ser relativamente fácil reconhecê-la e verificá-la.
- Deve ser computacionalmente inviável falsificá-la, seja construindo uma nova mensagem para uma assinatura digital existente seja construindo uma assinatura digital fraudulenta para determinada mensagem.
- Deve ser prático armazenar uma cópia da assinatura digital.

Uma função de hash segura, embutida em um esquema como o da Figura 11.5c ou d, satisfaz esses requisitos.

Diversas técnicas foram propostas para a função de assinatura digital e estão classificadas em duas categorias: direta e arbitrada.

Assinatura digital direta

A assinatura digital direta envolve apenas as partes em comunicação (origem, destino). Considera-se que o destino conhece a chave pública da origem. Uma assinatura digital pode ser formada criptografando-se a mensagem inteira com a chave privada do emissor (Figura 11.1c) ou criptografando-se um código de hash da mensagem com a chave privada do emissor (Figura 11.5c).

A confidencialidade pode ser obtida pela criptografia adicional da mensagem inteira mais a assinatura com a chave pública do receptor (criptografia de chave pública) ou com uma chave secreta compartilhada (criptografia simétrica); como exemplo, veja as Figuras 11.1d e 11.5d. Observe que é importante realizar a função de assinatura primeiro e, depois, uma função de confidencialidade externa. No caso de disputa, algum terceiro deverá ver a mensagem e sua assinatura. Se a assinatura for calculada sobre uma mensagem criptografada, então o terceiro também precisará acessar a chave de decriptografia para ler a mensagem original. Porém, se a assinatura for a operação interna, então o destinatário poderá armazenar a mensagem em texto claro e sua assinatura para uso posterior na solução da disputa.

Todos os esquemas diretos descritos até aqui compartilham um ponto fraco comum: a validade do esquema depende da segurança da chave privada do emissor. Se um emissor, mais tarde, quiser negar o envio de uma determi-

nada mensagem, ele poderá reivindicar que a chave privada foi perdida ou roubada, e que um outro falsificou sua assinatura. Controles administrativos relacionados à segurança das chaves privadas podem ser empregados para afastar ou, pelo menos, enfraquecer essa tática, mas a ameaça ainda existe, pelo menos até certo ponto. Um exemplo é exigir que cada mensagem assinada inclua um carimbo de tempo (data e hora) e exigir um relato imediato das chaves comprometidas a uma autoridade central.

Outra ameaça é que alguma chave privada possa realmente ser roubada de X no momento T. O oponente pode, então, enviar uma mensagem assinada com a assinatura de X e 'carimbada' com uma hora antes ou igual a T.

Assinatura digital arbitrada

Os problemas associados às assinaturas digitais diretas podem ser resolvidos com o uso de um árbitro.

Assim como os esquemas de assinatura direta, existem diversos esquemas de assinatura arbitrada. Em termos gerais, todos eles operam da seguinte forma: cada mensagem assinada de um emissor X para um receptor Y vai primeiro para um árbitro A, o qual submete a mensagem e sua assinatura a uma série de testes para verificar sua origem e conteúdo. A mensagem, então, é datada e enviada a Y com uma indicação de que foi verificada e aceita pelo árbitro. A presença de A resolve o problema enfrentado pelos esquemas de assinatura direta: que X poderia negar a mensagem.

O árbitro desempenha um papel crítico e crucial nesse tipo de esquema, e todas as partes precisam ter muita confiança de que o mecanismo de arbitragem esteja funcionando corretamente. O uso de um sistema confiável, descrito no Capítulo 20, poderia satisfazer esse requisito.

A Tabela 13.1, baseada em cenários descritos em [AKL83] e [MITC92], oferece vários exemplos de assinaturas digitais arbitradas.[1] No primeiro, a criptografia simétrica é utilizada. Considera-se que o emissor X e o árbitro A compartilham uma chave secreta K_{xa}, e que A e Y compartilham a chave secreta K_{ay}; X constrói uma mensagem M e calcula seu valor de hash $\mathrm{H}(M)$. Depois, X transmite a mensagem mais uma assinatura para A. A assinatura consiste em um identificador ID_X de X mais o valor de hash, tudo criptografado usando K_{xa}. A decriptografa a assinatura e verifica o valor de hash para validar a mensagem. Depois, A transmite uma mensagem para Y, criptografada com K_{ay}. A mensagem inclui ID_X, a mensagem original de X, a assinatura e um carimbo de tempo. Y pode decriptografar isto para recuperar a mensagem e a assinatura. O carimbo de tempo informa a Y que essa mensagem é oportuna e não uma repetição. Y pode armazenar M e a assinatura. No caso de disputa, Y, que afirma ter recebido M de X, envia a seguinte mensagem para A:

$$\mathrm{E}(K_{ay}, [ID_X \| M \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)])])$$

Tabela 13.1 Técnicas de assinatura digital arbitrada

(1) $\mathrm{X} \rightarrow \mathrm{A}$: $M \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)])$
(2) $\mathrm{A} \rightarrow \mathrm{Y}$: $\mathrm{E}(K_{ay}, [ID_X \| M \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)]) \| T])$

(a) Criptografia convencional, árbitro vê mensagem

(1) $\mathrm{X} \rightarrow \mathrm{A}$: $ID_X \| \mathrm{E}(K_{xy}, M) \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(\mathrm{E}(K_{xy}, M))])$
(2) $\mathrm{A} \rightarrow \mathrm{Y}$: $\mathrm{E}(K_{ay}, [ID_X \| \mathrm{E}(K_{xy}, M)]) \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(\mathrm{E}(K_{xy}, M)) \| T])$

(b) Criptografia convencional, árbitro não vê mensagem

(1) $\mathrm{X} \rightarrow \mathrm{A}$: $ID_X \| \mathrm{E}(PR_x, [ID_X \| \mathrm{E}(PU_y, \mathrm{E}(PR_x, M))])$
(2) $\mathrm{A} \rightarrow \mathrm{Y}$: $\mathrm{E}(PR_a, [ID_X \| \mathrm{E}(PU_y, \mathrm{E}(PR_x, M)) \| T])$

(c) Criptografia de chave pública, árbitro não vê mensagem

Legenda
X = emissor
Y = receptor
A = Árbitro
M = Mensagem
T = Carimbo de tempo

O árbitro usa K_{ay} para recuperar ID_X, M e a assinatura, e depois usa K_{xa} para decriptografar a assinatura e verificar o código de hash. Nesse esquema, Y não pode verificar diretamente a assinatura de X; a assinatura existe unicamente para resolver disputas. Y considera a mensagem de X autêntica, pois vem por meio de A. Nesse cenário, os dois lados precisam ter um alto grau de confiança em A:

- X precisa confiar em A para não revelar K_{xa} e não gerar assinaturas falsas na forma $\mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)])$.

1. O seguinte formato é utilizado: uma etapa de comunicação, em que P envia uma mensagem M para Q, é representada como $\mathrm{P} \rightarrow \mathrm{Q}$: M.

- Y precisa confiar em A para enviar $E(K_{ay}, [ID_X \| M \| E(K_{xa}, [ID_X \| H(M)]) \| T])$ apenas se o valor de hash estiver correto e a assinatura tiver sido gerada por X.
- Os dois lados precisam acreditar que A resolverá disputas com justiça.

Se o árbitro estiver à altura dessa confiança, então X tem garantias de que ninguém poderá falsificar sua assinatura e Y tem garantias de que X não poderá negar sua assinatura.

O cenário anterior também significa que A é capaz de ler mensagens de X para Y e, na realidade, que qualquer espião também seja capaz de fazer isso. A Tabela 13.1b mostra um cenário que oferece a arbitragem como antes, mas também garante a confidencialidade. Nesse caso, considera-se que X e Y compartilham a chave secreta K_{xy}. Agora, X transmite um identificador, uma cópia da mensagem criptografada com K_{xy} e uma assinatura para A. A assinatura consiste no identificador mais o valor de hash da mensagem criptografada, tudo criptografado usando K_{xa}. Como antes, A decriptografa a assinatura e verifica o valor de hash para validar a mensagem. Nesse caso, A está funcionando apenas com a versão criptografada da mensagem e é impedido de lê-la. A, então, transmite tudo que é recebido de X, mais um carimbo de tempo, tudo criptografado com K_{ay}, para Y.

Embora seja incapaz de ler a mensagem, o árbitro ainda está em posição de impedir fraude da parte de X ou Y. Outro problema que resta, compartilhado com o primeiro cenário, é que o árbitro poderia formar uma aliança com o emissor para rejeitar a mensagem assinada, ou com o receptor, para falsificar a assinatura do emissor.

Todos os problemas que acabamos de discutir podem ser resolvidos passando-se para um esquema de chave pública, sendo que uma versão aparece na Tabela 13.1c. Nesse caso, X realiza a criptografia dupla de uma mensagem M primeiro com a chave privada de X, PR_x, e depois com a chave pública de Y, PU_y. Esta é uma versão assinada e secreta da mensagem. Essa mensagem assinada, com o identificador de X, é criptografada novamente com PR_x e, com o ID_x, é enviada para A. A mensagem interna, com criptografia dupla, é protegida contra o árbitro (e de todos os outros, exceto Y). Porém, A pode decriptografar a criptografia externa para se certificar de que a mensagem tenha vindo de X (pois somente X tem PR_x). A verifica se o par de chaves privada/pública de X ainda é válido e, se for, verifica a mensagem. Depois, A transmite uma mensagem para Y, criptografada com PR_a. A mensagem inclui ID_X, a mensagem com criptografia dupla, e um carimbo de tempo.

Este esquema tem diversas vantagens em relação aos dois esquemas anteriores. Primeiro, nenhuma informação é compartilhada entre as partes antes da comunicação, evitando alianças para fraudar. Segundo, nenhuma mensagem datada incorretamente pode ser enviada, mesmo que PR_x seja comprometida, supondo que PR_a não esteja comprometida. Finalmente, o conteúdo da mensagem de X para Y é secreto para A e qualquer outro. Porém, esse esquema final envolve a criptografia da mensagem duas vezes com um algoritmo de chave pública. Discutiremos técnicas mais práticas posteriormente.

13.2 PROTOCOLOS DE AUTENTICAÇÃO

As ferramentas básicas descritas no Capítulo 11 são usadas em diversas aplicações, incluindo a assinatura digital discutida na Seção 13.1. Existem muitos outros usos, e a quantidade continua crescendo. Nesta seção, focalizamos duas áreas gerais (autenticação mútua e autenticação unidirecional) e examinamos algumas das implicações das técnicas de autenticação em ambas.

Autenticação mútua

Uma área de aplicação importante é a dos protocolos de autenticação mútua. Esses protocolos permitem que as partes em comunicação convençam-se mutuamente a respeito da identidade um do outro e troquem chaves de sessão. Esse assunto foi examinado na Seção 7.3 (técnicas simétricas) e na Seção 10.1 (técnicas de chave pública). Lá, o foco era a distribuição de chaves. Retornamos a esse assunto aqui para considerar as implicações mais amplas da autenticação.

Existem duas questões centrais relativas ao problema de troca de chave autenticada: confidencialidade e adequação no tempo *(timeliness)*. Para evitar disfarce e impedir comprometimento das chaves de sessão, informações essenciais de identificação e de chave de sessão precisam ser comunicadas de forma criptografada. Isso exige a existência anterior de chaves secretas e públicas que possam ser usadas para essa finalidade. A segunda questão, adequação no tempo, é importante devido à ameaça de repetições de mensagem. Essas repetições, no pior dos casos, poderiam permitir que um oponente comprometesse uma chave de sessão ou personificasse outra parte com sucesso. No mínimo, uma repetição bem-sucedida poderia interromper as operações apresentando às partes mensagens que parecem ser genuínas, mas não são.

[GONG93] lista os seguintes exemplos de ataques por repetição:

- **Repetição simples:** o oponente simplesmente copia uma mensagem e a repete mais tarde.
- **Repetição que pode ser registrada em log:** um oponente pode repetir uma mensagem com carimbo de tempo dentro da janela de tempo válida.

- **Repetição que não pode ser detectada:** esta situação poderia surgir porque a mensagem original poderia ter sido suprimida e, assim, não ter chegado ao seu destino; somente a mensagem por repetição chega.
- **Repetição inversa sem modificação:** esta é uma repetição de volta ao emissor da mensagem. Esse ataque é possível se a criptografia simétrica for usada e o emissor, com base no conteúdo, não puder reconhecer facilmente a diferença entre mensagens enviadas e mensagens recebidas.

Uma técnica para lidar com ataques por repetição é anexar um número de seqüência a cada mensagem usada em uma troca de autenticação. Uma nova mensagem é aceita somente se seu número de seqüência estiver na ordem correta. A dificuldade com essa técnica é que ela exige que cada parte registre o último número de seqüência para cada requerente com quem precisa lidar. Devido a esse overhead, os números de seqüência geralmente não são usados para autenticação e troca de chave. Em vez disso, uma das duas técnicas gerais a seguir é utilizada:

- **Carimbos de tempo:** a parte A aceita uma mensagem como sendo nova somente se esta tiver um carimbo de tempo que, no julgamento de A, é próxima o suficiente do conhecimento de A da hora atual. Essa técnica exige que os relógios entre os diversos participantes estejam sincronizados.
- **Desafio/resposta:** a parte A, esperando uma mensagem nova de B, primeiro envia a B um nonce (desafio) e exige que a mensagem subseqüente (resposta) recebida de B contenha o valor nonce correto.

Pode-se argumentar (por exemplo, [LAM92a]) que a técnica de carimbo de tempo não deva ser usada para aplicações orientadas a conexão, devido às dificuldades inerentes a essa técnica. Primeiro, algum tipo de protocolo é necessário para manter sincronismo entre os diversos clocks do processador. Este protocolo precisa ser tolerante a falhas, para lidar com erros de rede, e ser seguro, para enfrentar ataques hostis. Segundo, a oportunidade para um ataque bem-sucedido surgirá se houver uma perda temporária de sincronismo resultante de uma falha no mecanismo de clock de uma das partes. Finalmente, devido à natureza variável e imprevisível dos atrasos da rede, os relógios (clocks) distribuídos não podem manter sincronismo preciso. Portanto, qualquer procedimento baseado em carimbo de tempo precisa permitir uma janela de tempo suficientemente grande para acomodar atrasos de rede ainda que suficientemente pequenos para minimizar a oportunidade de ataque.

Por outro lado, a técnica de desafio-resposta é inadequada para um tipo de aplicação sem conexão, pois requer o overhead de um handshake antes de qualquer transmissão sem conexão, negando efetivamente a principal característica de uma transação sem conexão. Para essas aplicações, contar com algum tipo de servidor de tempo seguro e uma tentativa constante de cada parte para manter seus relógios em sincronismo pode ser a melhor técnica (por exemplo, [LAM92b]).

Técnicas de criptografia simétrica Conforme discutimos na Seção 7.3, uma hierarquia de dois níveis de chaves de criptografia simétricas pode ser usada para fornecer confidencialidade para a comunicação em um ambiente distribuído. Em geral, essa estratégia envolve o uso de um centro de distribuição de chave – key distribution center (KDC) confiável. Cada parte na rede compartilha uma chave secreta, conhecida como chave-mestra, com o CDC. O CDC é responsável por gerar chaves a serem usadas por um curto período em uma conexão entre duas partes, conhecidas como chaves de sessão, e por distribuir essas chaves usando as chaves-mestras para proteger a distribuição. Essa técnica é muito comum. Como exemplo, examinamos o sistema Kerberos no Capítulo 14. A discussão nesta subseção é relevante para um conhecimento dos mecanismos do Kerberos.

A Figura 7.9 ilustra uma proposta apresentada inicialmente por Needham e Schroeder [NEED78] para distribuição de chave secreta usando um CDC que, conforme mencionamos no Capítulo 7, inclui características de autenticação. O protocolo pode ser resumido da seguinte forma:

1. $A \rightarrow KDC$: $ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A])$
4. $B \rightarrow A$: $E(K_s, N_2)$
5. $A \rightarrow B$: $E(K_s, f(N_2))$

As chaves secretas K_a e K_b são compartilhadas entre A e o CDC, e B e o CDC, respectivamente. A finalidade do protocolo é distribuir com segurança uma chave de sessão K_s para A e B. A adquire de modo seguro uma nova chave de sessão na etapa 2. A mensagem na etapa 3 pode ser decriptografada, e portanto entendida apenas por B. A etapa 4 reflete o conhecimento de B quanto a K_s e a etapa 5 garante a B o conhecimento de K_s por A, garantindo a B que esta é uma mensagem nova, devido ao uso do nonce N_2. Lembre-se, pela nossa discussão no Capítulo 7, que a finalidade das etapas 4 e 5 é impedir um certo tipo de ataque por repetição. Em particular, se um oponente for capaz de capturar a mensagem na etapa 3 e repeti-la, isso poderia de alguma maneira perturbar as operações em B.

Apesar do handshake das etapas 4 e 5, o protocolo ainda é vulnerável a uma forma de ataque por repetição. Suponha que um oponente, X, seja capaz de comprometer uma chave de sessão antiga. Evidentemente, esta é uma

ocorrência muito mais improvável do que a de um oponente ter simplesmente observado e registrado a etapa 3. Apesar disso, esse é um risco de segurança em potencial. X pode personificar A e enganar B para usar a chave antiga simplesmente replicando a etapa 3. A menos que B lembre-se indefinidamente de todas as chaves de sessão anteriores usadas com A, B não poderá determinar que essa é uma repetição. Se X puder interceptar a mensagem de handshake, etapa 4, então ele poderá personificar a resposta de A, etapa 5. A partir desse ponto, X pode enviar mensagens falsas para B, que aparecem para ele como se viessem de A, usando uma chave de sessão autenticada.

Denning [DENN81, DENN82] propõe contornar esse ponto fraco com uma modificação no protocolo de Needham/Schroeder, que inclui a adição de um carimbo de tempo às etapas 2 e 3. Sua proposta supõe que as chaves-mestras, K_a e K_b, são seguras, e consiste das seguintes etapas:

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A \| T])$
4. $B \rightarrow A$: $E(K_s, N_1)$
5. $A \rightarrow B$: $E(K_s, f(N_1))$

T é um carimbo de tempo que garante a A e a B que a chave de sessão acabou de ser gerada. Assim, tanto A quanto B sabem que a distribuição de chave é uma troca nova. A e B podem analisar a adequação ao tempo verificando se

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

onde Δt_1 é a discrepância normal estimada entre o clock do CDC e o clock local (em A ou B) e Δt_2 é o tempo esperado do atraso de rede. Cada nó pode definir seu clock usando alguma fonte de referência padrão. Como o carimbo de tempo T está criptografado usando as chaves-mestras seguras, um oponente, mesmo com o conhecimento de uma chave de sessão antiga, não pode ser bem-sucedido, pois uma repetição da etapa 3 será detectada por B como inoportuna.

Um último ponto: as etapas 4 e 5 não foram incluídas na apresentação original [DENN81], mas foram incluídas depois [DENN82]. Essas etapas confirmam o recebimento da chave de sessão em B.

O protocolo de Denning parece oferecer um maior grau de segurança em comparação com o protocolo de Needham/Schroeder. Porém, surge uma nova preocupação: esse novo esquema depende de clocks sincronizados pela rede. [GONG92] aponta um risco envolvido. O risco é baseado no fato de que os clocks distribuídos podem perder o sincronismo como resultado de sabotagem ou falhas nos clocks ou no mecanismo de sincronismo.[2] O problema ocorre quando o clock de um emissor estiver adiantado em relação ao clock do destinatário desejado. Neste caso, um oponente pode interceptar uma mensagem do emissor e repeti-la mais adiante, quando o carimbo de tempo na mensagem se tornar atual na localização do destinatário. Essa repetição poderia causar resultados inesperados. Gong refere-se a esses ataques como **ataques suppress-replay**.

Um modo de impedir ataques suppress-replay é impor o requisito de que as partes verifiquem regularmente seus clocks contra o clock do CDC. A outra possibilidade, que evita a necessidade de sincronismo de clock, é contar com os protocolos de handshaking usando nonces. Essa última alternativa não é vulnerável a um ataque suppress-replay, pois os nonces que o destinatário escolherá no futuro são imprevisíveis ao emissor. O protocolo de Needham/Schroeder conta apenas com nonces, mas, como vimos, possui outras vulnerabilidades.

Em [KEHN92], há uma tentativa de responder aos problemas sobre ataques suppress-replay e ao mesmo tempo consertar os problemas no protocolo de Needham/Schroeder. Posteriormente, foi observada uma inconsistência nesse último protocolo, e uma estratégia melhorada foi apresentada em [NEUM93a].[3] O protocolo é o seguinte:

1. $A \rightarrow B$: $ID_A \| N_a$
2. $B \rightarrow KDC$: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A$: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

Vamos acompanhar essa troca passo a passo:

1. A inicia a troca de autenticação gerando um nonce, N_a, e enviando isso mais seu identificador para B em texto claro. Esse nonce será retornado para A em uma mensagem criptografada, que inclui a chave da sessão, garantindo a adequação ao tempo para A.

2. Essas coisas podem acontecer e acontecem. Nos últimos anos, chips com defeitos foram usados em diversos computadores e outros sistemas eletrônicos para rastrear hora e data. Os chips tinham uma tendência a avançar um dia [NEUM90].
3. É realmente difícil acertar essas coisas.

2. B alerta o CDC de que uma chave de sessão é necessária. Sua mensagem para o CDC inclui seu identificador e um nonce, N_b. Esse nonce será retornado a B em uma mensagem criptografada, que inclui a chave da sessão, garantindo sua adequação ao tempo para B. A mensagem de B para o CDC também inclui um bloco criptografado com a chave secreta compartilhada por B e o CDC. Esse bloco é usado para instruir o CDC a emitir credenciais a A; o bloco especifica o destinatário pretendido quanto às credenciais, um tempo de expiração sugerido para as credenciais e o nonce recebido de A.
3. O CDC passa para A o nonce de B e um bloco criptografado com a chave secreta que B compartilha com o CDC. O bloco serve como um 'ticket' que pode ser usado por A para autenticações subseqüentes, conforme veremos. O CDC também envia para A um bloco criptografado com a chave secreta compartilhada por A e o CDC. Esse bloco verifica se B recebeu a mensagem inicial de A (ID_B) e se esta é uma mensagem dentro do prazo, e não uma repetição (N_a), e oferece a A uma chave de sessão (K_s) e o limite de tempo sobre seu uso (T_b).
4. A transmite o ticket para B, junto com o nonce de B, com este último criptografado com a chave da sessão. O ticket dá a B a chave secreta, que é usada para decriptografar E (K_s, N_b) para recuperar o nonce. O fato de o nonce de B estar criptografado com a chave de sessão autentica que a mensagem veio de A e não é uma repetição.

Esse protocolo oferece um meio eficaz e seguro para A e B estabelecerem uma sessão com uma chave de sessão segura. Além disso, o protocolo deixa A em posse de uma chave que pode ser usada para autenticação subseqüente perante B, evitando a necessidade de contatar o servidor de autenticação repetidamente. Suponha que A e B estabeleçam uma sessão usando o protocolo mencionado e depois terminem essa sessão. Posteriormente, mas dentro do limite de tempo estabelecido pelo protocolo, A requisita uma nova sessão com B. O resultado é o seguinte protocolo:

1. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| N'_a$
2. $B \rightarrow A$: $N'_b \| E(K_s, N'_a)$
3. $A \rightarrow B$: $E(K_s, N'_b)$

Quando B recebe a mensagem na etapa 1, ele verifica se o ticket não expirou. Os nonces recém-gerados N'_a e N'_b garantem a cada parte que não existe um ataque por repetição.

Em todo o texto anterior, o tempo especificado em T_b é um tempo relativo ao clock de B. Assim, este carimbo de tempo não exige clocks sincronizados, pois B só verifica carimbos de tempo que ele mesmo gerou.

Técnicas de criptografia de chave pública No Capítulo 10, apresentamos uma técnica para o uso da criptografia de chave pública para fins de distribuição de chave de sessão (Figura 10.6). Esse protocolo pressupõe que cada uma das duas partes está de posse da chave pública atual da outra. Pode não ser prático exigir esta hipótese.

Um protocolo usando carimbo de tempo aparece em [DENN81]:

1. $A \rightarrow AS$: $ID_A \| ID_B$
2. $AS \rightarrow A$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
3. $A \rightarrow B$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
   $\| E(PU_b, E(PR_a, [K_s \| T]))$

Nesse caso, o sistema central é conhecido como um servidor de autenticação (AS, *Authentication Server*), pois não é realmente responsável pela distribuição da chave secreta. Em vez disso, o AS oferece certificados de chave pública. A chave de sessão é escolhida e criptografada por A; logo, não existe risco de exposição pelo AS. O carimbo de tempo protege contra repetições de chaves comprometidas.

Esse protocolo é compacto mas, como antes, exige sincronismo dos clocks. Outra técnica, proposta por Woo e Lam [WOO92a], utiliza nonces. O protocolo consiste nas seguintes etapas:

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B$: $E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC$: $ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_B]))$
6. $B \rightarrow A$: $E(PU_a, E(PR_{auth}, [(N_a \| K_s \| ID_B)]) \| N_b]))$
7. $A \rightarrow B$: $E(K_s, N_b)$

Na etapa 1, A informa ao CDC sobre sua intenção de estabelecer uma conexão segura com B. O CDC retorna a A uma cópia do certificado de chave pública de B (etapa 2). Usando a chave pública de B, A informa a B quanto ao seu desejo de comunicar e envia um nonce N_a (etapa 3). Na etapa 4, B perde pede ao CDC o certificado de chave pública de A e solicita uma chave de sessão; B inclui o nonce de A de modo que o CDC possa estampar a chave de

sessão com esse nonce. O nonce é protegido usando a chave pública do CDC. Na etapa 5, o CDC retorna a B uma cópia do certificado de chave pública de A, mais a informação $\{N_a, Ks, ID_B\}$. Essa informação basicamente diz que K_s é uma chave secreta gerada pelo CDC em favor de B e ligada a N_a; o vínculo de K_s e N_a garantirá a A que K_s é recente. Essa tripla é criptografada, usando a chave privada do CDC, para permitir que B verifique se a tripla é de fato do CDC. Ela também é criptografada usando a chave pública de B, para que nenhuma outra entidade possa usar a tripla em uma tentativa de estabelecer uma conexão fraudulenta com A. Na etapa 6, a tripla $\{Na, Ks, IDB\}$, ainda criptografada com a chave privada do CDC, é repassada para A, junto com um nonce N_b gerado por B. Tudo isso é criptografado usando a chave pública de A. A apanha a chave de sessão K_s e a utiliza para criptografar N_b e retorná-la a B. Essa última mensagem garante a B sobre o conhecimento da chave da sessão por A.

Esse parece ser um protocolo seguro, que leva em conta os diversos ataques. Porém, os próprios autores localizaram uma falha e submeteram uma versão revisada do algoritmo em [WOO92b]:

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B$: $E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC$: $ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]))$
6. $B \rightarrow A$: $E(PU_a, E(PR_{auth}, [(N_a \| K_s \| ID_A \| ID_B) \| N_b]))$
7. $A \rightarrow B$: $E(K_s, N_b)$

O identificador de A, ID_A, é adicionado ao conjunto de itens criptografados com a chave privada do CDC nas etapas 5 e 6. Isso vincula a chave da sessão K_s às identidades das duas partes que estarão envolvidas na sessão. Essa inclusão de ID_A considera o fato de que o valor de nonce N_a é considerado exclusivo somente entre todos os nonces gerados por A, e não entre todos os nonces gerados por todas as partes. Assim, é o par $\{ID_A, N_a\}$ que identifica exclusivamente o pedido de conexão de A.

Neste exemplo e nos protocolos descritos anteriormente, os protocolos que pareciam seguros foram revisados após análise adicional. Esses exemplos realçam a dificuldade de acertar as coisas na área da autenticação.

Autenticação unidirecional

Uma aplicação para a qual a criptografia está crescendo em popularidade é o correio eletrônico (e-mail). A própria natureza do correio eletrônico, e seu principal benefício, é que não é preciso que emissor e receptor estejam on-line ao mesmo tempo. Em vez disso, a mensagem de e-mail é encaminhada à caixa de correio eletrônica do receptor, onde é mantida em buffer até que este esteja disponível para lê-la.

O 'envelope' ou cabeçalho da mensagem de e-mail precisa estar às claras, para que a mensagem possa ser entregue pelo protocolo de armazenamento e encaminhamento do e-mail, como o Simple Mail Transfer Protocol (SMTP) ou X.400. Porém, normalmente é desejável que o protocolo de tratamento de correio não exija acesso ao formato de texto claro da mensagem, pois isso exigiria confiar no mecanismo de tratamento de correio. Por conseguinte, a mensagem de e-mail deveria ser criptografada, de modo que o sistema de tratamento de correio não esteja de posse da chave de decriptografia.

Um segundo requisito é o de autenticação. Normalmente, o destinatário deseja alguma garantia de que a mensagem é do emissor alegado.

Técnica de criptografia simétrica Usando a criptografia simétrica, o cenário de distribuição descentralizada de chaves, ilustrado na Figura 7.11, é impraticável. Esse esquema exige que o emissor emita uma solicitação para o destinatário desejado, espere uma resposta que inclua uma chave de sessão e somente então envie a mensagem.

Com alguma melhoria, a estrutura do CDC, ilustrada na Figura 7.9, é candidata para e-mail criptografado. Como queremos evitar a necessidade de que o destinatário (B) esteja on-line ao mesmo tempo em que o emissor (A), as etapas 4 e 5 precisam ser eliminadas. Para uma mensagem com conteúdo M, a seqüência é a seguinte:

1. $A \rightarrow KDC$: $ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

Essa técnica garante que somente o destinatário pretendido de uma mensagem poderá lê-la. Isso também oferece um grau de autenticação de que o emissor é A. Conforme especificado, o protocolo não protege contra repetições. Alguma medida de defesa poderia ser obtida incluindo-se um carimbo de tempo com a mensagem. Porém, devido aos atrasos em potencial no processo de e-mail, esses carimbos de tempo podem ter utilidade limitada.

Técnicas de criptografia de chave pública Já apresentamos as técnicas de criptografia de chave pública que são adequadas para o e-mail, incluindo a criptografia direta da mensagem inteira por confidencialidade (Figura 11.1b), autenticação (Figura 11.1c) ou ambos (Figura 11.1d). Essas técnicas exigem que o emissor conheça a chave pública do destinatário (confidencialidade) ou o destinatário identifique a chave pública do emissor (autenticação) ou ambos (confidencialidade mais autenticação). Além disso, o algoritmo de chave pública precisa ser aplicado uma ou duas vezes ao que pode ser uma mensagem longa.

Se a confidencialidade for a preocupação principal, então o seguinte poderá ser mais eficiente:

$$A \rightarrow B: E(PU_b, K_s) \| E(K_s, M)$$

Nesse caso, a mensagem é criptografada com uma chave secreta de uso único. A também criptografa essa chave de uso único com a chave pública de B. Somente B será capaz de usar a chave privada correspondente para recuperar a chave de uso único e depois usar essa chave para decriptografar a mensagem. Esse esquema é mais eficiente do que simplesmente criptografar a mensagem inteira com a chave pública de B.

Se a autenticação for a principal preocupação, então uma assinatura digital poderá ser suficiente, como foi ilustrado na Figura 11.5c:

$$A \rightarrow B: M \| E(PR_a, H(M))$$

Esse método garante que, mais tarde, A não poderá negar que enviou a mensagem. Porém, essa técnica está aberta a outro tipo de fraude. Bob redige uma mensagem à sua chefe Alice, que contém uma idéia para economizar o dinheiro da empresa. Ele anexa sua assinatura digital e a envia para o sistema de e-mail. Depois de algum tempo, por fim, a mensagem será entregue na caixa de correio de Alice. Mas suponha que Max tenha ouvido falar da idéia de Bob e obtenha acesso à fila de e-mail antes da entrega. Ele encontra a mensagem de Bob, remove sua assinatura, anexa a dele e reenvia a mensagem para a fila, para ser entregue a Alice. Max recebe o crédito pela idéia de Bob.

Para enfrentar esse esquema, a mensagem e a assinatura podem ser criptografadas com a chave pública do destinatário:

$$A \rightarrow B: E(PU_b, [M \| E(PR_a, H(M))])$$

Os dois últimos esquemas exigem que B conheça a chave pública de A e esteja convencido de que ela é recente. Um modo eficiente de fornecer essa garantia é o certificado digital, descrito no Capítulo 10. Agora, temos

$$A \rightarrow B: M \| E(PR_a, H(M)) \| E(PR_{as}, [T \| ID_A \| PU_a])$$

Além da mensagem, A envia para B a assinatura, criptografada com a chave privada de A, e o certificado de A, criptografado com a chave privada do servidor de autenticação. O destinatário da mensagem, primeiro, usa o certificado para obter a chave pública do emissor e verificar se ela é autêntica e, depois, usa a chave pública para verificar a própria mensagem. Se a confidencialidade for exigida, então a mensagem inteira pode ser criptografada com a chave pública de B. Como alternativa, a mensagem inteira pode ser criptografada com uma chave secreta de uso único; a chave secreta também é transmitida, criptografada com a chave pública de B. Essa técnica é explorada no Capítulo 15.

13.3 PADRÃO DE ASSINATURA DIGITAL

O National Institute of Standards and Technology (NIST) publicou o Federal Information Processing Standard FIPS 186, conhecido como padrão de assinatura digital (Digital Signature Standard — DSS). O DSS utiliza o Secure Hash Algorithm (SHA), descrito no Capítulo 12, e apresenta uma nova técnica de assinatura digital, o Digital Signature Algorithm (DSA). O DSS foi proposto originalmente em 1991 e revisado em 1993 em resposta ao feedback público com relação à segurança do esquema. Houve uma outra revisão secundária em 1996. Em 2000, uma versão expandida do padrão foi publicada como FIPS 186-2. Esta versão mais recente também incorpora os algoritmos de assinatura digital baseados no RSA e na criptografia de curva elíptica. Nesta seção, discutimos o algoritmo DSS original.

A técnica do DSS

O DSS utiliza um algoritmo que foi projetado para oferecer apenas a função de assinatura digital. Diferentemente do RSA, ele não pode ser usado para criptografia ou troca de chave. Apesar disso, essa é uma técnica de chave pública.

A Figura 13.1 compara a técnica do DSS para gerar assinaturas digitais com aquela que é usada no RSA. Na técnica do RSA, a mensagem a ser assinada é inserida em uma função de hash que produz um código de hash seguro, de tamanho fixo. Esse código de hash é, então, criptografado usando a chave privada do emissor para formar a assinatura. Tanto a mensagem quanto a assinatura são então transmitidas. O destinatário pega a mensagem e produz um código de hash. O destinatário também decriptografa a assinatura, usando a chave pública do emissor. Se o código de hash

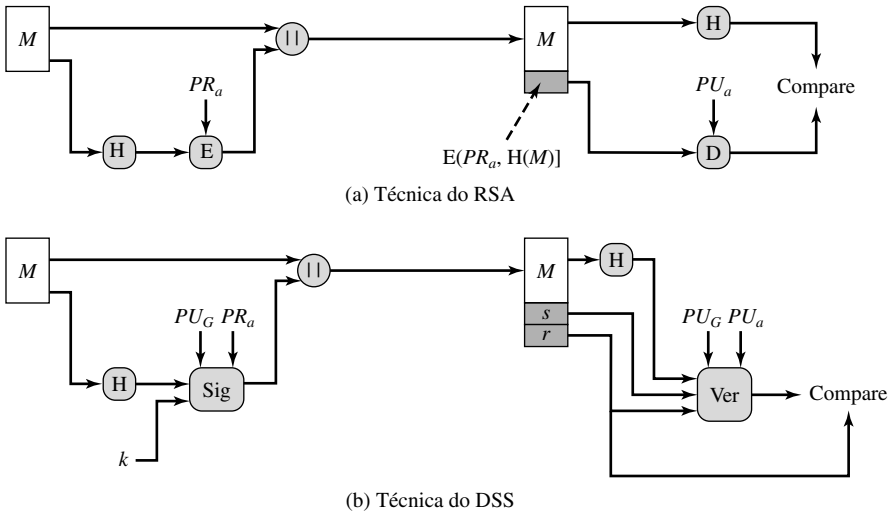


Figura 13.1 Duas técnicas para assinaturas digitais.

calculado combinar adequadamente com a assinatura decriptografada, a assinatura é aceita como válida. Como somente o emissor conhece a chave privada, somente ele poderia ter produzido uma assinatura válida.

A técnica do DSS também usa uma função de hash. O código de hash é fornecido como entrada para uma função de assinatura, com um número aleatório k, gerado para essa assinatura em particular. A função de assinatura também depende da chave privada do emissor (PR_a) e de um conjunto de parâmetros conhecidos de um grupo de membros em comunicação. Podemos considerar esse conjunto como constituído de uma chave pública global (PU_G).[4] O resultado é uma assinatura que consiste em dois componentes, rotulados com s e r.

Na ponta receptora, o código de hash da mensagem que chega é gerado. Isso, juntamente com a assinatura, é inserido em uma função de verificação. A função de verificação também depende da chave pública global, além da chave pública do emissor (PU_a), que é o correspondente da chave privada do emissor. A saída da função de verificação é um valor igual ao componente de assinatura r se a assinatura for válida. A função de assinatura é tal que somente o emissor, com conhecimento da chave privada, poderia ter produzido a assinatura válida.

Agora, vejamos os detalhes do algoritmo.

O algoritmo de assinatura digital

O DSA é baseado na dificuldade de se calcular logaritmos discretos (ver Capítulo 8) e nos esquemas apresentados originalmente por ElGamal [ELGA85] e Schnorr [SCHN91].

A Figura 13.2 resume o algoritmo. Existem três parâmetros públicos e que podem ser comuns a um grupo de usuários. Um número primo de 160 bits q é escolhido. Em seguida, um número primo p é selecionado, com um tamanho entre 512 e 1024 bits, tal que q divida $(p - 1)$. Finalmente, g é escolhido para estar na forma $h^{(p-1)/q} \bmod p$, onde h é um inteiro entre 1 e $(p - 1)$, com a restrição de que g deverá ser maior que 1.[5]

Com estes números em mãos, cada usuário seleciona uma chave privada e gera uma chave pública. A chave privada x precisa ser um número entre 1 e $(q - 1)$ e deveria ser escolhida aleatória ou pseudo-aleatoriamente. A chave pública é calculada a partir da chave privada como $y = g^x \bmod p$. O cálculo de y dado x é relativamente simples. Porém, dada a chave pública y, acredita-se ser computacionalmente inviável determinar x, que é o logaritmo discreto de y na base g, mod p (ver Capítulo 8).

Para criar uma assinatura, um usuário calcula duas quantidades, r e s, que são funções dos componentes de chave pública (p, q, g), da chave privada do usuário (x), do código de hash da mensagem, $H(M)$ e do inteiro k **adicional**, que deveria ser gerado aleatória ou pseudo-aleatoriamente e ser exclusivo para cada assinatura.

Na ponta receptora, a verificação é realizada usando-se as fórmulas mostradas na Figura 13.2, O receptor gera uma quantidade v que é uma função dos componentes da chave pública, da chave pública do emissor e do código de hash da mensagem que chega. Se essa quantidade combinar adequadamente com o componente r da assinatura, então a assinatura é validada.

A Figura 13.3 representa as funções de assinatura e de verificação.

4. Também é possível permitir que esses parâmetros adicionais variem com cada usuário, para que sejam uma parte da chave pública de um usuário. Na prática, é mais provável que seja usada uma chave pública global, separada da chave pública de cada usuário.
5. De acordo com a teoria dos números, g é da ordem de q mod p; ver Capítulo 8.

Componentes globais da chave pública

p número primo onde $2^{L-1} < p < 2^L$
para $512 \leq L \leq 1024$ e L um múltiplo de 64;
ou seja, o tamanho entre 512 e 1024 bits
em incrementos de 64 bits

q divisor primo de $(p - 1)$, where $2^{159} < q < 2^{160}$;
isto é, comprimento de 160 bits

g $= h^{(p-1)/q} \bmod p$,
onde h é qualquer inteiro com $1 < h < (p - 1)$
tal que $h^{(p-1)/q} \bmod p > 1$

Chave privada do usuário

x random or pseudorandom integer with $0 < x < q$

Chave pública do usuário

$y = g^x \bmod p$

Número secreto por mensagem do usuário

k = inteiro aleatório ou pseudoaleatório com $0 < k < q$

Assinatura

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1} (\mathrm{H(M)} + xr)] \bmod q$

Signature = (r, s)

Verificação

$w = (s')^{-1} \bmod q$

$u_1 = [\mathrm{H(M')}w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

TESTe: $v = r'$

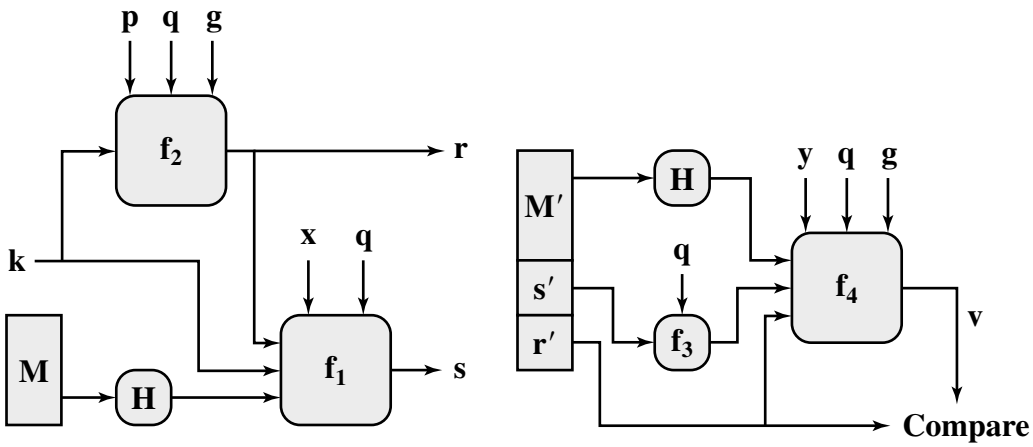
M = mensagem a ser assinada

$\mathrm{H}(M)$ = hash de M usando SHA-1

M', r', s' = versões recebidas de M, r, s

Figura 13.2 O algoritmo de assinatura digital (DSA).

A estrutura do algoritmo, como revelada na Figura 13.3, é muito interessante. Observe que o teste no final é sobre o valor r, que não depende de forma alguma da mensagem. Em vez disto, r é uma função de k e dos três componentes globais de chave pública. O inverso multiplicativo de k (mod q) é passado a uma função que também tem como entradas o código hash da mensagem e a chave privada do usuário. A estrutura dessa função é tal que o receptor pode recuperar r usando a mensagem que chega e a assinatura, a chave pública do usuário, e a chave pública global. Certamente, não é óbvio pela Figura 13.2 ou pela Figura 13.3, que esse esquema funcionaria. Uma prova é fornecida no site Web deste livro.



$s = f_1(\mathrm{H(M)}, k, x, r, q) = (k^{-1} (\mathrm{H(M)} + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Assinatura

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, \mathrm{H(M')}, w, r')$

$= ((g^{(\mathrm{H(M')}w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verificação

Figura 13.3 Assinatura e verificação do DSS.

Dada a dificuldade de apanhar logaritmos discretos, é inviável para um oponente recuperar k a partir de r ou recuperar x a partir de s.

Outro ponto que merece ser mencionado é que a única tarefa computacionalmente exigente na geração da assinatura é o cálculo exponencial $g^k \bmod p$. Como esse valor não depende da mensagem a ser assinada, ele pode ser calculado antes. Na realidade, um usuário poderia calcular previamente diversos valores de r para serem usados para assinar documentos conforme a necessidade. A única outra tarefa um pouco exigente é determinar um inverso multiplicativo, k^{-1}. Mais uma vez, vários valores destes podem ser previamente calculados.

13.4 LEITURA E SITES WEB RECOMENDADOS

[AKL83] é o artigo clássico sobre assinaturas digitais e ainda é altamente relevante. Um estudo mais recente, e excelente, é [MITC92].

AKL83 Akl, S. "Digital signatures: a tutorial survey." *Computer*, fevereiro de 1983.

MITC92 Mitchell, C.; Piper, F. ; and Wild, P. "Digital signatures." In SIMM92a].

Site Web recomendado:

- **Digital Signatures:** Página do NIST com informações sobre opções de assinatura digital aprovadas pelo NIST.

13.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

árbitro
assinatura digital arbitrada
assinatura digital direta
assinatura digital
autenticação mútua
ataque por repetição
ataque suppress-replay
autenticação unidirecional
carimbo de tempo
Digital Signature Algorithm (DSA)
Digital Signature Standard (DSS)
nonce

Perguntas de revisão

13.1 Relacione duas disputas que podem surgir no contexto da autenticação de mensagem.

13.2 Que propriedades uma assinatura digital deve ter?

13.3 Que requisitos um esquema de assinatura digital deve satisfazer?

13.4 Qual é a diferença entre assinatura digital direta e arbitrada?

13.5 Em que ordem a função de assinatura e a função de confidencialidade deveriam ser aplicadas a uma mensagem e por quê?

13.6 Quais são algumas ameaças associadas a um esquema de assinatura digital direta?

13.7 Dê exemplos de ataques por repetição.

13.8 Liste três técnicas gerais para lidar com ataques de repetição.

13.9 O que é um ataque suppress-replay?

Problemas

13.1 Modifique as técnicas de assinatura digital da Tabela 13.1a e b para permitir que o receptor verifique a assinatura.

13.2 Modifique a técnica de assinatura digital da Tabela 13.1c para evitar a criptografia tripla da mensagem inteira.

13.3 Na discussão sobre a Tabela 13.1c, foi dito que eram possíveis alianças para realizar fraudes. De fato, existe uma possibilidade. Descreva-a e explique por que haveria tão pouca credibilidade a ponto de podermos ignorá-la com segurança.

13.4 Na Seção 13.2, esboçamos o esquema de chave pública proposto em [WOO92a] para a distribuição de chaves secretas. A versão revisada inclui ID_A nas etapas 5 e 6. Que ataque, especificamente, é combatido por essa revisão?

13.5 O protocolo referenciado no Problema 13.4 pode ser reduzido de sete etapas para cinco, tendo a seguinte seqüência:

(1) A → B:
(2) B → KDC:
(3) KDC → B:
(4) B → A:
(5) A → B:

Mostre a mensagem transmitida em cada etapa. *Dica:* A mensagem final nesse protocolo é a mesma mensagem final no protocolo original.

13.6 Com referência ao ataque suppress-replay descrito na Seção 13.2:

a. Dê um exemplo de ataque quando o relógio de uma parte está adiantado em relação ao do CDC.

b. Dê um exemplo de ataque em que o relógio de uma parte está adiantado em relação ao de outra parte.

13.7 Existem três formas típicas de usar nonces como desafios. Suponha que N_a seja um nonce gerado por A, A e B compartilhem a chave K, e f() seja uma função como incremento. Os três estágios são :

Uso 1
(1) A → B: N_a
(2) B → A: E(K, N_a)

Uso 2
(1) A → B: E(K, N_a)
(2) B → A: N_a

Uso 3
(1) A → B: E(K, N_a)
(2) B → A: E(K, f(N_a))

Descreva situações para as quais cada uso é apropriado.

13.8 Dr. Watson pacientemente esperou até que Sherlock Holmes terminasse. "Algum problema interessante para resolver, Holmes?", perguntou ele quando Holmes finalmente fez o logout.

"Ah, não exatamente. Simplesmente verifiquei meu e-mail e depois fiz algumas experiências na rede em vez das minhas experiências químicas normais. Só tenho um cliente agora e já resolvi seu problema. Se me lembro bem, certa vez você mencionou a criptologia entre os seus outros hobbies, e talvez isso possa interessar-lhe."

"Bem, sou apenas um criptólogo amador, Holmes. Mas certamente estou interessado no problema. De que se trata?"

"Meu cliente é Mr. Hosgrave, diretor de um banco pequeno mas promissor. O banco é totalmente computadorizado e, naturalmente, usa muitas comunicações em rede. O banco já usa RSA para proteger seus dados e assinar digitalmente os documentos que são comunicados. Agora, o banco deseja introduzir algumas mudanças em seus procedimentos; em particular, ele precisa assinar digitalmente alguns documentos por *dois* signatários, para que:

1. O primeiro signatário prepare o documento, forme sua assinatura e passe o documento ao segundo signatário.

2. O segundo signatário, como um primeiro passo, deve verificar se o documento foi realmente assinado pelo primeiro signatário. Depois, ele incorpora sua assinatura na assinatura do documento, para que o destinatário, além de qualquer membro do público, possa verificar se o documento foi realmente assinado por dois signatários. Além disso, apenas o segundo signatário deverá ser capaz de verificar a assinatura do documento depois da etapa (1); ou seja, o destinatário (ou qualquer membro do público) deveria ser capaz de verificar apenas o documento completo, com as assinaturas de ambos os signatários, mas não o documento em sua forma intermediária, onde apenas um signatário o assinou. Além disso, o banco gostaria de utilizar seus módulos existentes que admitem assinaturas digitais em estilo RSA."

"Hum, entendo como o RSA pode ser usado para assinar documentos digitalmente por *um* signatário, Holmes. Acho que você resolveu o problema de Mr. Hosgrave pela generalização apropriada das assinaturas digitais do RSA."

"Exatamente, Watson", sinalizou Sherlock Holmes. "Originalmente, a assinatura digital do RSA era formada pela criptografia do documento pela chave de decriptografia privada do signatário, 'd', e a assinatura podia ser verificada por qualquer um por intermédio de sua decriptografia usando a chave de criptografia conhecida publicamente, 'e'. Pode-se verificar se a assinatura S foi realizada pela pessoa que conhece d, que supostamente é o único signatário. Agora, o problema de Mr. Hosgrave pode ser resolvido da mesma forma pela ligeira generalização do processo, ou seja..."

Termine a explicação.

13.9 O DSA especifica que, se o processo de geração de assinatura resultar em um valor de $s = 0$, um novo valor de k deverá ser gerado e a assinatura deverá ser recalculada. Por quê?

13.10 O que acontece se um valor k usado na criação de uma assinatura DSA for comprometido?

13.11 O documento DSS inclui um algoritmo recomendado para testar se um número é primo, da seguinte forma:

(1) **[Escolha *w*]** Seja w um inteiro ímpar aleatório. Então $(w - 1)$ é par e pode ser expresso na forma $2^a m$, com m ímpar. Ou seja, 2^a é a maior potência de 2 que divide $(w - 1)$.

(2) **[Gere *b*]** Seja b um inteiro aleatório no intervalo $1 < b < w$.

(3) **[Faça a exponenciação]** Defina $j = 0$ e $z = b^m \bmod w$.

(4) [**[Achou?]** Se $j = 0$ e $z = 1$, ou se $z = w - 1$, então w passa no teste e pode ser primo; vá para a etapa 8.

(5) [**[Terminou?]** Se $j > 0$ e $z = 1$, então w não é primo; termine o algoritmo para esse w.

(6) [**[Aumente *j*]** Defina $j = j + 1$. Se $j < a$, defina $z = z^2 \bmod w$ e vá para a etapa 4.
(7) [**[Termine]** w não é primo; termine o algoritmo para esse w.
(8) [**[Testar novamente?]** Se valores aleatórios suficientes de b tiverem sido testados, então aceite w como primo e termine o algoritmo; caso contrário, vá para a etapa 2.

a. Explique como o algoritmo funciona.
b. Mostre que ele é equivalente ao teste de Miller-Rabin descrito no Capítulo 8.

13.12 Com o DSS, como o valor de k é gerado para cada assinatura, mesmo que a mesma mensagem seja assinada duas vezes em diferentes ocasiões, as assinaturas diferirão. Isso não é verdade com assinaturas do RSA. Qual é a implicação prática desta diferença?

13.13 Considere o problema de criar parâmetros de domínio para o DSA. Suponha que já tenhamos encontrado os primos p e q tais que $q|(p - 1)$. Agora, precisamos encontrar $g \in Z_p$ com g da ordem de q mod p. Considere os dois algoritmos a seguir:

Algoritmo 1	**Algoritmo 2**
repeat	**repeat**
select $g \in Z_p$	**select** $h \in Z_p$
h ß $g^q \bmod p$	g ß $h^{(p-1)/p} \bmod p$
until ($h = 1$ e $g \neq 1$)	**until** ($g \neq 1$)
return g	**return** g

a. Prove que o valor retornado pelo Algoritmo 1 possui ordem q.
b. Prove que o valor retornado pelo Algoritmo 2 possui ordem q.
c. Suponha que $p = 40193$ e $q = 157$. Quantas iterações de loop você espera que o Algoritmo 1 faça antes de encontrar um gerador?
d. Se p é 1024 bits e q é 160 bits, você recomendaria o uso do Algoritmo 1 para encontrar g? Explique.
e. Suponha que $p = 40193$ e $q = 157$. Qual é a probabilidade de que o Algoritmo 2 calcule um gerador em sua primeira iteração de loop? (se for útil, você poderá usar o fato de que $\sum_{d|n} \varphi(d) = n$ ao responder esta pergunta).

13.14 É tentadora a idéia de tentar desenvolver uma variação de Diffie-Hellman que possa ser usada como uma assinatura digital. Aqui está uma que é mais simples do que DSA e que não exige um número aleatório secreto além da chave privada.

Elementos públicos:

q número primo
α $\alpha < q$ e α é uma raiz primitiva de q

Chave privada:

X $X < q$

Chave pública:

$Y = \alpha^X \bmod q$

Para assinar uma mensagem M, calcule $h = \mathrm{H}(M)$, o código de hash da mensagem. Exigimos que $\mathrm{mdc}(h, q - 1) = 1$. Se não, anexe o hash à mensagem e calcule um novo hash. Continue esse processo até que seja produzido um código de hash relativamente primo de $(q - 1)$. Depois, calcule Z para satisfazer a Z X $h \equiv X(\bmod\ q - 1)$. A assinatura da mensagem é α^Z. Para verificar a assinatura, um usuário verifica se $Y = (\alpha^Z)^h = \alpha^X \bmod q$.

a. Mostre que esse esquema funciona, ou seja, mostre que o processo de verificação produz uma igualdade se a assinatura for válida.
b. Mostre que o esquema é inaceitável descrevendo uma técnica simples para forjar a assinatura de um usuário em uma mensagem qualquer.

13.15 Uma proposta antiga para um esquema de assinatura digital usando criptografia simétrica é baseada no seguinte: Para assinar uma mensagem de n bits, o emissor gera aleatoriamente com antecedência 2n chaves criptográficas de 56 bits:

$$k1, K1, k2, K2, ..., kn, Kn$$

que são mantidas secretas. O emissor prepara com antecedência dois conjuntos correspondentes de parâmetros de validação de 64 bits não secretos, que são tornados públicos:

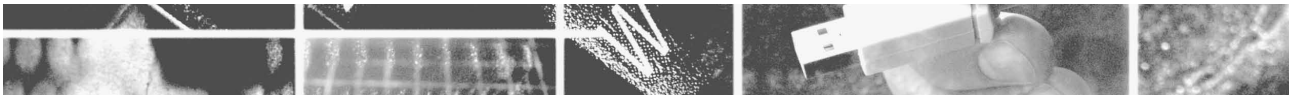
$$u1, V1, u2, V2, ..., un, Vn \text{ e } v1, V1, v2, V2, ..., vn, Vn$$

onde

$$vi = \mathrm{E}(ki, ui), Vi = \mathrm{E}(ki, Ui)$$

A mensagem M é assinada da seguinte forma. Para o i-ésimo bit da mensagem, ki ou Ki é conectado à mensagem, dependendo se o bit de mensagem for 0 ou 1. Por exemplo, se os três primeiros bits da mensagem forem 011, então as três primeiras chaves da assinatura serão $k1$, $K2$, $K3$.

a. Como o receptor valida a mensagem?
b. A técnica é segura?
c. Quantas vezes o mesmo conjunto de chaves secretas pode ser usado com segurança para diferentes mensagens?
d. Que problemas práticos, se houver algum, este esquema apresenta?



PARTE TRÊS

Aplicações de segurança de rede

Na prática, a eficácia de uma contramedida normalmente depende de como ela é usada; o melhor cofre do mundo é inútil se ninguém se lembrar de fechar a porta.

Computers at risk: safe computing in the information age,
National Research Council, 1991

O uso crescente das redes de computadores e de comunicações, bem como os crescentes conhecimentos sobre computadores e dependência da tecnologia da informação aumentaram o risco de a indústria dos EUA perder informações proprietárias para a espionagem econômica. Em parte para reduzir o risco, a indústria está usando com mais freqüência hardware e software com capacidade de criptografia.

Communications privacy: federal policy and actions.
General Accounting Office Report GAO/OSI-94-2, novembro de 1993

Nas duas primeiras partes, examinamos diversas cifras e seu uso para confidencialidade, autenticação, troca de chaves e funções relacionadas. A Parte 3 examina importantes ferramentas e aplicações de segurança de rede, que fazem uso dessas funções. Essas aplicações podem ser usadas em uma única rede, em uma intranet corporativa ou na Internet.

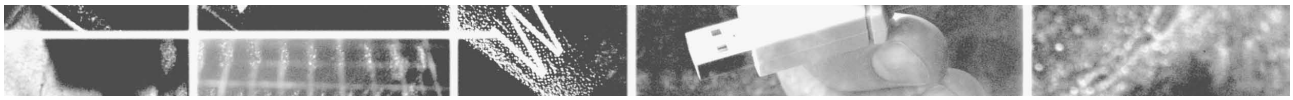
ROTEIRO DA PARTE TRÊS

Capítulo 14: Aplicações de autenticação
O Capítulo 14 é um estudo das duas especificações de autenticação mais importantes em uso atualmente. Kerberos é um protocolo de autenticação baseado na criptografia convencional, que tem recebido um amplo suporte e é usado em diversos sistemas. X.509 especifica um algoritmo de autenticação e define uma estrutura de certificação. Esta última permite que os usuários obtenham certificados de chaves públicas de modo que uma comunidade de usuários possa ter confiança na validade das chaves públicas. Essa estrutura é empregada como um elemento básico em diversas aplicações.

Capítulo 15: Segurança de e-mail
A aplicação distribuída utilizada com mais intensidade é o e-mail, e há um interesse crescente em fornecer serviços de autenticação e confidencialidade como parte de uma estrutura de correio eletrônico. O Capítulo 15 examina as duas técnicas que provavelmente dominarão a segurança do e-mail em um futuro próximo. Pretty Good Privacy (PGP) é um esquema bastante utilizado, que não depende de qualquer organização ou autoridade. Assim, ele é bem adequado para uso individual, pessoal, assim como para incorporação em configurações de rede operadas pelas organizações. S/MIME (Secure/Multipurpose Internet Mail Extension) foi desenvolvido especificamente para ser um padrão para Internet.

Capítulo 16: Segurança de IP
O Internet Protocol (IP) é o elemento central na Internet e intranets privadas. A segurança no nível de IP, conseqüentemente, é importante para o projeto de qualquer esquema de segurança baseado em interligação de rede. O Capítulo 16 examina o esquema de segurança IP que foi desenvolvido para operar tanto com o IP atual quanto com o IP da próxima geração, conhecido como IPv6.

Capítulo 17: Segurança da Web
O crescimento explosivo no uso da World Wide Web para comércio eletrônico e para disseminar informações gerou a necessidade de uma segurança forte baseada na Web. O Capítulo 17 oferece uma análise dessa nova importante área de segurança e examina dois padrões fundamentais: Secure Sockets Layer (SSL) e Secure Electronic Transaction (SET).



Aplicações de autenticação

14.1 **Kerberos**

14.2 **Serviço de autenticação X.509**

14.3 **Infra-estrutura de chave pública**

14.4 **Leitura e sites Web recomendados**

14.5 **Principais termos, perguntas de revisão e problemas**

Apêndice 14A Técnicas de criptografia do Kerberos

Não podemos fazer aliança com príncipes vizinhos até que saibamos de suas intenções.
—*The art of war,* Sun Tzu

PONTOS PRINCIPAIS

- O Kerberos é um serviço de autenticação projetado para uso em um ambiente distribuído.
- Ele utiliza um serviço de autenticação de terceiros confiável, que permite que clientes e servidores estabeleçam comunicação autenticada.
- O X.509 define o formato para certificados de chave pública. Esse formato é bastante utilizado em diversas aplicações.
- Uma infra-estrutura de chave pública (PKI, Public Key Infrastructure) é definida como o conjunto de hardware, software, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais com base na criptografia assimétrica.
- Normalmente, as implementações de PKI utilizam certificados X.509.

Este capítulo examina algumas das funções de autenticação que foram desenvolvidas para dar suporte à autenticação em nível de aplicação e assinaturas digitais.

Começamos examinando um dos primeiros e mais amplamente utilizados serviços: Kerberos. Em seguida, examinamos o serviço de autenticação de diretório X.509. Esse padrão é importante como parte do serviço de diretório que ele suporta e também é um elemento básico utilizado em outros padrões, como S/MIME, explicado no Capítulo 15. Finalmente, este capítulo examina o conceito de uma infra-estrutura de chave pública (PKI).

14.1 KERBEROS

Kerberos[1] é um serviço de autenticação desenvolvido como parte do projeto Athena no MIT. O problema de que o Kerberos trata é o seguinte: considere um ambiente distribuído aberto, em que os usuários nas estações de trabalho desejem acessar serviços nos servidores distribuídos pela rede. Gostaríamos que os servidores pudessem restringir o acesso a usuários autorizados e fossem capazes de autenticar as solicitações de serviço. Nesse ambiente, não se pode garantir que uma estação de trabalho identifique corretamente seus usuários nos serviços da rede. Em particular, existem as três ameaças a seguir:

- Um usuário pode ganhar acesso a determinada estação de trabalho e fingir ser outro usuário operando a partir dessa estação de trabalho.
- Um usuário pode alterar o endereço de rede de uma estação de trabalho de modo que as solicitações enviadas a partir da estação de trabalho modificada pareçam vir da estação de trabalho original.
- Um usuário pode espionar as trocas e usar um ataque por repetição para obter acesso a um servidor ou perturbar as operações.

Em qualquer um desses casos, um usuário não autorizado pode ser capaz de obter acesso a serviços e dados aos quais ele não tem autorização. Em vez de se basear em complexos protocolos de autenticação incorporados em cada servidor, o Kerberos oferece um servidor de autenticação centralizado, cuja função é autenticar usuários perante os servidores e os usuários. Diferentemente da maioria dos outros esquemas de autenticação descritos neste livro, o Kerberos conta exclusivamente com a criptografia simétrica, não utilizando qualquer criptografia de chave pública.

Duas versões do Kerberos são de uso corrente. Implementações da versão 4 [MILL88, STEI88] ainda existem. A versão 5 [KOHL94] corrige algumas das deficiências de segurança da versão 4 e foi publicada como um padrão proposto para Internet (RFC 1510).[2]

Começamos esta seção com uma rápida explicação sobre a motivação para a abordagem usada no Kerberos. Depois, devido às complexidades do Kerberos, é melhor começar com uma descrição do protocolo de autenticação utilizado na versão 4. Isso nos permitirá ver a essência da estratégia do Kerberos sem considerar alguns dos detalhes exigidos para lidar com ameaças de segurança sutis. Finalmente, examinamos a versão 5.

Motivação

Se um conjunto de usuários receber computadores pessoais dedicados que não possuem conexão em rede, então os recursos e arquivos de cada usuário poderão ser protegidos como proteção física de cada computador pessoal. Quando esses usuários, em vez disso, são atendidos por um sistema centralizado de tempo compartilhado, o sistema operacional de tempo compartilhado precisa oferecer a segurança. O sistema operacional pode impor políticas de controle de acesso com base na identidade do usuário e usar o procedimento de logon para identificar os usuários.

Hoje, nenhum desses cenários é típico. O mais comum é uma arquitetura distribuída consistindo em estações de trabalho de usuário dedicadas (clientes) e servidores distribuídos ou centralizados. Nesse ambiente, três técnicas de segurança podem ser identificadas:

1. Confiar em cada estação de trabalho cliente individual para garantir a identidade de seu usuário ou usuários e confiar em cada servidor para impor uma política de segurança com base na identificação do usuário (ID).
2. Exigir que os sistemas clientes autentiquem-se perante os servidores, mas confiar no sistema cliente com relação à identidade de seu usuário.
3. Exigir que o usuário prove sua identidade para cada serviço chamado. Além disso, exigir que os servidores provem sua identidade aos clientes.

1. "Na mitologia grega, um cão de muitas cabeças, normalmente três, talvez com um rabo de serpente, o guardião da entrada do Hades". Em *Dictionary of subjects and symbols in art*, de James Hall, Harper & Row, 1979. Assim como o Kerberos grego tem três cabeças, o Kerberos moderno foi idealizado com três componentes para proteger as entradas de uma rede: autenticação, contabilidade e auditoria. As duas últimas cabeças nunca foram implementadas.
2. As versões de 1 a 3 foram versões internas de desenvolvimento. A Versão 4 é o Kerberos "original".

Em um ambiente pequeno e fechado, em que todos os sistemas pertencem a uma única organização e são operados por ela, a primeira ou talvez a segunda estratégia podem ser suficientes.[3] Mas, em um ambiente mais aberto, em que as conexões de rede com outras máquinas são admitidas, a terceira técnica é necessária para proteger a informação do usuário e os recursos armazenados no servidor. O Kerberos admite essa terceira técnica. Ele assume uma arquitetura cliente/servidor distribuída e emprega um ou mais servidores Kerberos para oferecer um serviço de autenticação.

O primeiro relatório publicado sobre o Kerberos [STEI88] listou os seguintes requisitos:

- **Seguro:** Um espião da rede não poderá obter a informação necessária para imitar um usuário. Em geral, o Kerberos deverá ser forte o suficiente para que um oponente em potencial não o considere como sendo o elo mais fraco.
- **Confiável:** Para todos os serviços que contam com o Kerberos para controle de acesso, a falta de disponibilidade do serviço Kerberos significa falta de disponibilidade dos serviços admitidos. Logo, o Kerberos deverá ser altamente confiável e deverá empregar uma arquitetura de servidor distribuída, com um sistema capaz de garantir o outro em caso de falha.
- **Transparente:** O ideal é que o usuário não perceba que a autenticação está ocorrendo, além do requisito da entrada de uma senha.
- **Expansível:** O sistema deverá ser capaz de suportar uma grande quantidade de clientes e servidores. Isso sugere uma arquitetura modular, distribuída.

Para dar suporte a estes requisitos, o esquema geral do Kerberos é de um serviço de autenticação de terceiros confiável, que usa um protocolo baseado naquele proposto por Needham e Schroeder [NEED78], conforme explicado no Capítulo 7. Ele é confiável no sentido de que os clientes e os servidores confiam no Kerberos para mediar sua autenticação mútua. Supondo que o protocolo Kerberos seja bem projetado, então o serviço de autenticação será seguro se o próprio servidor Kerberos for seguro.[4]

Kerberos Versão 4

A versão 4 do Kerberos utiliza o DES, em um protocolo algo elaborado, para oferecer o serviço de autenticação. Vendo o protocolo como um todo, é difícil constatar a necessidade dos muitos elementos lá contidos. Portanto, adotamos uma estratégia usada por Bill Bryant do projeto Athena [BRYA88] e construímos gradualmente todo o protocolo examinando primeiro os diversos diálogos hipotéticos. Cada diálogo sucessivo aumenta a complexidade para atacar as vulnerabilidades de segurança reveladas no diálogo anterior.

Depois de examinar o protocolo, vemos alguns outros aspectos da versão 4.

Um diálogo de autenticação simples

Em um ambiente de rede desprotegido, qualquer cliente pode solicitar um serviço de qualquer servidor. O risco de segurança óbvio é o de imitação. Um oponente pode fingir ser outro cliente e obter privilégios não autorizados nos servidores. Para conter essa ameaça, os servidores precisam ser capazes de confirmar as identidades dos clientes que solicitam os serviços. Cada servidor pode ter que passar por essa tarefa para cada interação entre cliente/servidor, mas, em um ambiente aberto, isso impõe uma carga substancial a cada servidor.

Uma alternativa é usar um servidor de autenticação (AS – Authentication Server) que conheça as senhas de todos os usuários e armazene-as em um banco de dados centralizado. Além disso, o AS compartilha uma chave secreta exclusiva com cada servidor. Essas chaves foram distribuídas fisicamente ou de alguma outra maneira segura. Considere o seguinte diálogo hipotético:[5]

$$\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: \quad ID_C \| P_C \| ID_V$$

$$\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: \quad Ticket$$

$$\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: \quad ID_C \| Ticket$$

$$Ticket = \mathrm{E}(K_v, [ID_C \| AD_C \| ID_V])$$

3. Porém, até mesmo um ambiente fechado sofre a ameaça de ataque por um funcionário insatisfeito.

4 Lembre-se de que a segurança do servidor Kerberos não deve ser presumida automaticamente, mas deve ser protegida com cuidado (por exemplo, em uma sala trancada). É bom lembrar o destino do Kerberos grego, capturado por Hércules sob ordem de Euristeu em seu décimo segundo trabalho: "Hércules encontrou o grande cão em sua corrente e o agarrou pela garganta. Imediatamente, as três cabeças tentaram atacar, e Kerberos chicoteou seu poderoso rabo. Hércules segurou firmemente, e Kerberos relaxou inconsciente. Euristeu ficou surpreso ao ver Hércules vivo. Quando viu as três cabeças babando e o imenso cão a quem elas pertenciam, ele se assustou e pulou para a segurança do seu grande jarro de bronze." Em *The Hamlyn concise dictionary of greek and roman mythology*, de Michael Stapleton, Hamlyn, 1982.

5 A parte à esquerda do sinal de dois pontos indica o emissor e o receptor; a parte à direita indica o conteúdo da mensagem, e o símbolo || indica concatenação

onde

C	= cliente
AS	= servidor de autenticação
V	= servidor
ID_C	= identificador do usuário em C
ID_V	= identificador de V
P_C	= senha do usuário em C
AD_C	= endereço de rede de C
K_v	= chave de criptografia secreta compartilhada por AS e V

Nesse cenário, o usuário efetua o logon em uma estação de trabalho e solicita acesso ao servidor V. O módulo cliente C na estação de trabalho do usuário solicita a senha do usuário e depois envia uma mensagem ao AS, que inclui o ID do usuário, o ID do servidor e a senha do usuário. O AS verifica seu banco de dados para saber se o usuário forneceu a senha correta para esse ID de usuário e se esse usuário tem permissão para acessar o servidor V. Se os dois testes forem bem-sucedidos, o AS aceitará o usuário como autêntico e agora precisará convencer o servidor de que o usuário é autêntico. Para fazer isso, o AS cria um ticket que contém o ID e o endereço de rede do usuário, e o ID do servidor. Esse ticket é criptografado usando a chave secreta compartilhada pelo AS e por pelo servidor. O ticket é então enviado de volta a C. Como o ticket é criptografado, ele não pode ser alterado por C ou por um oponente.

Com esse ticket, C pode agora solicitar o serviço de V. C envia a mensagem a V contendo o ID de C e o ticket. V decriptografa o ticket e verifica se o ID do usuário no ticket é o mesmo que o ID do usuário não criptografado na mensagem. Se os dois coincidirem, o servidor considerará o usuário autenticado e concederá o serviço solicitado.

Cada um dos ingredientes da mensagem (3) é significativo. O ticket é criptografado para impedir alteração ou falsificação. O ID do servidor (ID_V) é incluído no ticket, para que o servidor possa verificar se decriptografou o ticket corretamente. O ID_C é incluído no ticket para indicar que este foi emitido em favor de C. Finalmente, AD_C serve para combater a ameaça seguinte. Um oponente poderia captar o ticket transmitido na mensagem (2), depois usar o nome ID_C $_C$ e transmitir uma mensagem na forma (3) a partir de outra estação de trabalho. O servidor receberia um ticket válido que, coincidindo com o ID do usuário, concederia acesso ao usuário nessa outra estação de trabalho. Para impedir esse ataque, o AS inclui no ticket o endereço de rede do qual veio a solicitação original. Agora, o ticket só é válido se for transmitido da mesma estação de trabalho que solicitou o ticket inicialmente.

Um diálogo de autenticação mais seguro Embora o cenário anterior solucione alguns dos problemas de autenticação em um ambiente de rede aberto, os problemas permanecem. Dois em particular se destacam. Primeiro, gostaríamos de minimizar o número de vezes que um usuário precisa digitar uma senha. Suponha que cada ticket só possa ser usado uma vez. Se o usuário C efetuar o logon em uma estação de trabalho de manhã e quiser verificar seu e-mail em um servidor de correio, C precisará fornecer uma senha para obter um ticket para o servidor de e-mail. Se C quiser verificar seus e-mails várias vezes durante o dia, cada tentativa exigirá a redigitação da senha. Podemos melhorar as coisas dizendo que os tickets são reutilizáveis. Para uma única sessão de logon, a estação de trabalho pode armazenar o ticket do servidor de correio depois de tê-la recebido e usado em favor do usuário por múltiplos acessos ao servidor de correio.

Porém, sob esse esquema, um usuário continuará precisando de um novo ticket para cada serviço diferente. Se um usuário quiser acessar um servidor de impressão, um servidor de e-mail, um servidor de arquivos etc., a primeira instância de cada acesso exigiria um novo ticket e, portanto, exigiria que o usuário digitasse a senha.

O segundo problema é que o cenário anterior envolveu uma transmissão da senha em texto claro [mensagem (1)]. Um espião poderia captar a senha e usar qualquer serviço acessível à vítima.

Para resolver esses problemas adicionais, apresentamos um esquema para evitar senhas de texto claro e um novo servidor, conhecido como servidor de concessão de ticket (TGS – *Ticket-Granting Server*). O novo, porém ainda hipotético, cenário é o seguinte:

Uma vez por sessão de logon do usuário:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Uma vez por tipo de serviço:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Uma vez por sessão do serviço:

(5) C → V: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| TempodeVida_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| D_v \| TS_2 \| TempodeVida_2])$$

O novo serviço, TGS, emite tickets aos usuários autenticados no AS. Assim, o usuário primeiro solicita um ticket de concessão de ticket ($Ticket_{tgs}$) do AS. O módulo cliente na estação de trabalho do usuário salva esse ticket. Toda vez que o usuário solicitar acesso a um novo serviço, o cliente se submeterá ao TGS, usando o ticket para se autenticar. O TGS, então, concede um ticket para o serviço específico. O cliente salva cada ticket de concessão de serviço e o utiliza para autenticar seu usuário em um servidor toda vez que um determinado serviço for solicitado. Vamos examinar os detalhes desse esquema:

1. O cliente solicita um ticket de concessão de ticket em favor do usuário enviando seu ID de usuário para o AS, junto com o ID do TGS, indicando uma solicitação para usar o serviço do TGS.
2. O AS responde com um ticket que é criptografado com uma chave derivada da senha do usuário (K_c), a qual já está armazenada no AS. Quando essa resposta chega ao cliente, este pede a senha do usuário, gera a chave e tenta decriptografar a mensagem que chega. Se a senha correta for fornecida, o ticket será recuperado com sucesso.

Como apenas o cliente correto deverá conhecer a senha, apenas o usuário correto poderá recuperar o ticket. Assim, usamos a senha para obter credenciais do Kerberos sem ter de transmitir a senha em texto claro. O próprio ticket consiste no ID e endereço de rede do usuário, e o ID do TGS. Isso corresponde ao primeiro cenário. A idéia é que o cliente possa usar esse ticket para solicitar vários tickets de concessão de serviço. Assim, o ticket de concessão de ticket deve ser reutilizável. Porém, não queremos que um oponente seja capaz de captar o ticket e usá-lo. Considere o cenário a seguir: um oponente capta o ticket de login e espera até que o usuário efetue um logoff de sua estação de trabalho. Então o oponente obtém acesso a essa estação de trabalho ou configura sua estação com o mesmo endereço de rede da vítima. O oponente pode reutilizar o ticket para enganar o TGS. Para impedir isso, o ticket inclui um carimbo de tempo, indicando a data e a hora em que o ticket foi emitido, e um tempo de vida, indicando o tempo pelo qual o ticket é válido (por exemplo, oito horas). Assim, o cliente agora possui um ticket reutilizável e não precisa incomodar o usuário solicitando uma senha para cada nova solicitação de serviço. Finalmente, observe que o ticket de concessão de ticket é criptografado com uma chave secreta conhecida apenas pelo AS e pelo TGS. Isso impede a alteração do ticket. O ticket é recriptografado com uma chave, com base na senha do usuário. Isso garante que o ticket possa ser recuperado apenas pelo usuário correto, oferecendo a autenticação.

Agora que o cliente tem um ticket de concessão de ticket, o acesso a qualquer servidor pode ser obtido com as etapas 3 e 4:

3. O cliente solicita um ticket de concessão de serviço em favor do usuário. Para essa finalidade, o cliente transmite uma mensagem ao TGS contendo o ID do usuário, o ID do serviço desejado e o ticket de concessão de ticket.
4. O TGS decriptografa o ticket recebido usando uma chave compartilhada apenas pelo AS e pelo TGS (K_{tgs}) e verifica o sucesso da decriptografia pela presença de seu ID. Ele verifica para ter certeza de que o tempo de vida não expirou. Depois, compara o ID do usuário e o endereço de rede com a informação que chega para autenticar o usuário. Se o usuário tiver acesso permitido ao servidor V, o TGS emitirá um ticket para conceder acesso ao serviço solicitado.

O ticket de concessão de serviço tem a mesma estrutura do ticket de concessão de ticket. Realmente, como o TGS é um servidor, poderíamos esperar que os mesmos elementos fossem necessários para autenticar um cliente no TGS e para autenticar um cliente em servidor de aplicação. Novamente, o ticket contém um carimbo de tempo e tempo de vida. Se o usuário quiser acessar o mesmo serviço mais tarde, o cliente poderá simplesmente usar o ticket de concessão de serviço adquirido anteriormente e não precisará incomodar o usuário com uma solicitação de senha. Observe que o ticket é criptografado com uma chave secreta (K_v), conhecida apenas pelo TGS e pelo servidor, impedindo alteração.

Finalmente, com um ticket de concessão de serviço específico, o cliente pode ter acesso ao serviço correspondente com a etapa 5:

5. O cliente solicita acesso ao serviço em favor do usuário. Para essa finalidade, o cliente transmite uma mensagem ao servidor, contendo o ID do usuário e o ticket de concessão de serviço. O servidor realiza a autenticação usando o conteúdo do ticket.

Esse novo cenário satisfaz os dois requisitos de somente uma solicitação de senha por sessão do usuário e proteção da senha do usuário.

O diálogo de autenticação da versão 4 Embora o cenário anterior melhore a segurança em comparação com a primeira tentativa, permanecem dois problemas adicionais. O núcleo do primeiro problema é o tempo de vida associado ao ticket de concessão de ticket. Se esse tempo de vida for muito curto (por exemplo, minutos), então será

solicitado repetidamente que o usuário informe uma senha. Se o tempo de vida for longo (por exemplo, horas), então um oponente terá maior oportunidade para realizar um ataque por repetição. Um oponente poderia espionar a rede e captar uma cópia do ticket de concessão de ticket e depois esperar que o usuário legítimo saia do sistema. Então o oponente poderia falsificar o endereço de rede do usuário legítimo e enviar a mensagem da etapa (3) ao TGS. Isso daria ao oponente acesso ilimitado aos recursos e arquivos disponíveis ao usuário legítimo.

De modo semelhante, se um oponente capturar um ticket de concessão de serviço e usá-lo antes que ele expire, o oponente terá acesso ao serviço correspondente.

Assim, chegamos a um requisito adicional. Um serviço de rede (o TGS ou um serviço de aplicação) precisa ser capaz de provar que a pessoa usando um ticket é a mesma pessoa para quem esse ticket foi emitido.

O segundo problema é que pode haver um requisito para os servidores se autenticarem com os usuários. Sem essa autenticação um oponente poderia sabotar a configuração de modo que as mensagens para um servidor fossem direcionadas a outro local. O servidor falso, então, estaria em posição de atuar como um servidor real e captar qualquer informação do usuário e negar o verdadeiro serviço ao usuário.

Examinamos esses problemas um de cada vez nos referindo à Tabela 14.1, que mostra o protocolo Kerberos real.

Tabela 14.1 Resumo das trocas de mensagem do Kerberos versão 4

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$ **(2) AS → C** $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| TempodeVida_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| TempodeVida_2])$

(a) Troca de serviço de autenticação para obter ticket de concessão de ticket

(3) C → TGS $ID_v \| Ticket_{tgs} \| Autenticador_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| TempodeVida_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| TempodeVida_4])$

$Autenticador_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

(b) Troca do serviço de concessão de ticket para obter ticket de concessão de serviço

(5) C → V $Ticket_v \| Autenticador_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (para autenticação mútua)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| TempodeVida_4])$

$Autenticador_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Troca de autenticação cliente/servidor para obter serviço

Primeiro, considere o problema de tickets de concessão de ticket captados e a necessidade de determinar se o apresentador do ticket é o mesmo que o cliente para quem o ticket foi emitido. A ameaça é de que um oponente roube o ticket e o use antes que expire. Para contornar esse problema, vamos fazer com que o AS ofereça ao cliente e ao TGS uma informação secreta de maneira segura. Então, o cliente poderá provar sua identidade ao TGS revelando a informação secreta, novamente de uma maneira segura. Um modo eficiente de fazer isso é usar uma chave de criptografia como informação segura; esta é conhecida como chave de sessão no Kerberos.

A Tabela 14.1a mostra a técnica para distribuir a chave de sessão. Como antes, o cliente envia uma mensagem ao AS solicitando acesso ao TGS. O AS responde com uma mensagem, criptografada com uma chave derivada da senha do usuário (K_c), que contém o ticket. A mensagem criptografada também contém uma cópia da chave da sessão, $K_{c,tgs}$, onde os subscritos indicam que essa é uma chave de sessão para C e TGS. Como essa chave de sessão está dentro da mensagem criptografada com K_c, somente o cliente do usuário poderá lê-la. A mesma chave de sessão está incluída no ticket, que só pode ser lido pelo TGS. Assim, a chave de sessão foi entregue com segurança a C e ao TGS.

Observe que várias informações adicionais foram acrescentadas a essa primeira fase do diálogo. A mensagem (1) inclui um carimbo de tempo, de modo que o AS sabe que a mensagem é recente. A mensagem (2) inclui vários elementos do ticket em uma forma acessível a C. Isso permite que C confirme que esse ticket é para o TGS e descubra sua hora de expiração.

Armado com o ticket e a chave de sessão, C está pronto para chegar ao TGS. Como antes, C envia ao TGS uma mensagem que inclui o ticket mais o ID do serviço solicitado (mensagem (3) na Tabela 14.1b). Além disso, C transmite um autenticador, que inclui o ID e o endereço do usuário de C e um carimbo de tempo. Diferentemente do ticket, que é reutilizável, o autenticador serve para ser usado apenas uma vez e possui um tempo de vida muito curto. O TGS pode decriptografar o ticket com a chave que ele compartilha com o AS. Esse ticket indica que o usuário C recebeu a chave de sessão $K_{c,tgs}$. De fato, o ticket diz: "Qualquer um que use $K_{c,tgs}$ deve ser C". O TGS usa a chave de sessão para

decriptografar o autenticador. O TGS pode, então, verificar o nome e o endereço do autenticador com o do ticket e com o endereço de rede da mensagem que chega. Se tudo coincidir, então o TGS tem garantias de que o emissor do ticket é realmente o proprietário real do ticket. Conseqüentemente, o autenticador diz: "No instante TS_3, eu uso $K_{c,tgs}$". Observe que o ticket não prova a identidade de ninguém, mas é um modo de distribuir chaves com segurança. É o autenticador que prova a identidade do cliente. Como o autenticador só pode ser usado uma vez e tem um tempo de vida curto, a ameaça de um oponente roubar o ticket e o autenticador para apresentação posterior é frustrada.

A resposta do TGS, na mensagem (4), segue a forma da mensagem (2). A mensagem é criptografada com a chave de sessão compartilhada pelo TGS e C, e inclui uma chave de sessão a ser compartilhada entre C e o servidor V, o ID de V e o carimbo de tempo do ticket. O próprio ticket inclui a mesma chave de sessão.

C agora tem um ticket de concessão de serviço reutilizável para V. Quando C apresenta esse ticket, como mostrado na mensagem (5), ele também envia um autenticador. O servidor pode decriptografar o ticket, recuperar a chave de sessão e decriptografar o autenticador.

Se a autenticação mútua for exigida, o servidor pode responder conforme mostrado na mensagem (6) da Tabela 14.1. O servidor retorna o valor do carimbo de tempo a partir do autenticador, incrementado em 1, e criptografado na chave de sessão. C pode decriptografar essa mensagem para recuperar o carimbo de tempo incrementada. Como a mensagem foi criptografada pela chave da sessão, C tem garantias de que ela só poderia ter sido criada por V. O conteúdo da mensagem garante a C que essa não é uma repetição de uma resposta antiga.

Finalmente, na conclusão desse processo, o cliente e o servidor compartilham uma chave secreta. Essa chave pode ser usada para criptografar mensagens futuras entre os dois ou para trocar uma nova chave de sessão aleatória para essa finalidade.

A Tabela 14.2 resume a justificativa para cada um dos elementos no protocolo Kerberos, e a Figura 14.1 contém uma visão geral simplificada da ação.

Tabela 14.2 Razões para os elementos do protocolo Kerberos versão 4

Mensagem (1)	O cliente solicita o ticket de concessão de ticket
ID_C	Diz ao AS a identidade do usuário a partir desse cliente
ID_{tgs}	Diz ao AS que o usuário solicita acesso ao TGS
TS_1	Permite que o AS verifique se o clock deste cliente está sincronizado com o do AS
Mensagem (2)	O AS retorna o ticket de concessão de ticket
K_c	A criptografia é baseada na senha do usuário, permitindo que o AS e o cliente verifiquem a senha, e protegendo o conteúdo da mensagem (2)
$K_{c,tgs}$	Cópia da chave de sessão acessível ao cliente, criada pelo AS para permitir a troca segura entre cliente e TGS sem exigir que compartilhem uma chave permanente
ID_{tgs}	Confirma que esse ticket é para o TGS
TS_2	Informa ao cliente sobre a hora em que esse ticket foi emitido
$TempodeVida_2$	Informa ao cliente sobre o tempo de vida desse ticket
$Ticket_{tgs}$	Ticket a ser usado pelo cliente para acessar o TGS

(a) Troca do serviço de autenticação

Mensagem (3)	O cliente solicita o ticket de concessão de serviço
ID_V	Diz ao TGS que o usuário solicita acesso ao servidor V
$Ticket_{tgs}$	Garante ao TGS que esse usuário foi autenticado pelo AS
$Autenticador_c$	Gerado pelo cliente para validar o ticket
Mensagem (4)	O TGS retorna o ticket de concessão de serviço
$K_{c,tgs}$	Chave compartilhada apenas por C e TGS protege conteúdo da mensagem (4)
$K_{c,y}$	Cópia da chave de sessão acessível ao cliente, criada pelo TGS para permitir a troca segura entre cliente e servidor, sem exigir que compartilhem uma chave permanente
ID_V	Confirma que esse ticket é para o servidor V
TS_4	Informa ao cliente sobre a hora em que esse ticket foi emitido
$Ticket_V$	O ticket a ser usado pelo cliente para acessar o servidor V
$Ticket_{tgs}$	Reutilizável de modo que o usuário não precise reinserir a senha
K_{tgs}	O ticket é criptografado com a chave conhecida apenas pelo AS e pelo TGS, para impedir modificações
$K_{c,tgs}$	Cópia da chave de sessão acessível ao TGS, usada para decriptografar o autenticador, autenticando assim o ticket

ID_C	Indica o proprietário que tem direito a esse ticket
AD_C	Impede o uso do ticket a partir de uma estação de trabalho que não seja aquela que solicitou o ticket inicialmente
ID_{tgs}	Garante ao servidor que ele decriptografou o ticket corretamente
TS_2	Informa ao TGS sobre a hora em que esse ticket foi emitido
$TempodeVida_2$	Impede a repetição depois da expiração do ticket
$Autenticador_c$	Garante ao TGS que o apresentador do ticket é o mesmo cliente a quem o ticket foi emitido; tem muito pouco tempo de vida, para impedir repetição
$K_{c,tgs}$	O autenticador é criptografado com a chave conhecida apenas pelo cliente e pelo TGS, para impedir modificação
ID_C	Precisa coincidir com o ID no ticket para autenticar o ticket
AD_C	Precisa coincidir com o endereço no ticket para autenticar o ticket
TS_3	Informa ao TGS sobre a hora em que esse autenticador foi gerado

(b) Troca de serviço de concessão de ticket

Mensagem (5)	O cliente solicita o serviço
$Ticket_V$	Garante ao servidor que esse usuário foi autenticado pelo AS
$Autenticador_c$	Gerado pelo cliente para validar o ticket
Mensagem (6)	Autenticação opcional do servidor ao cliente
$K_{c,v}$	Garante a C que a mensagem é de V
$TS_5 + 1$	Garante a C que essa não é uma repetição de uma resposta antiga
$Ticket_v$	Reutilizável de modo que o cliente não precisa solicitar um novo ticket do TGS para cada acesso ao mesmo servidor
K_v	O ticket é criptografado com chave conhecida apenas pelo TGS e pelo servidor, para impedir modificações
$K_{c,v}$	Cópia da chave de sessão acessível ao cliente; usada para decriptografar o autenticador, autenticando assim o ticket
ID_C	Indica o proprietário que tem direito a esse ticket
AD_C	Impede o uso do ticket a partir de uma estação de trabalho que não seja aquela que solicitou o ticket inicialmente
ID_V	Garante ao servidor que ele decriptografou o ticket corretamente
TS_4	Informa ao servidor sobre a hora em que o ticket foi emitido
$TempodeVida$	Impede a repetição depois da expiração do ticket
$Autenticador_c$	Garante ao servidor que o apresentador do ticket é o mesmo cliente a quem o ticket foi emitido; tem curto tempo de vida, para impedir repetição
$K_{c,v}$	O autenticador é criptografado com chave conhecida apenas pelo cliente e pelo servidor, para impedir modificações
ID_C	Precisa coincidir com ID no ticket para autenticar o ticket
AD_C	Precisa coincidir com o endereço no ticket para autenticar o ticket
TS_5	Informa ao servidor sobre a hora em que o autenticador foi gerado

(c) Troca de autenticação cliente/servidor

Domínios do Kerberos e Kerberi múltiplos Um ambiente Kerberos de serviço completo, consistindo em um servidor Kerberos, diversos clientes e diversos servidores de aplicação, exige o seguinte:

1. O servidor Kerberos precisa ter o ID de usuário e o hash das senhas de todos os usuários participantes em seu banco de dados. Todos os usuários são registrados com o servidor Kerberos.
2. O servidor Kerberos precisa compartilhar uma chave secreta com cada servidor. Todos os servidores são registrados com o servidor Kerberos.

Esse ambiente é conhecido como **domínio Kerberos**. O conceito de *domínio* pode ser explicado da seguinte maneira. Um domínio Kerberos é um conjunto de nós gerenciados que compartilham o mesmo banco de dados Kerberos. O banco de dados Kerberos reside no sistema de computador Kerberos mestre, que deve ser mantido em uma sala fisicamente segura. Uma cópia somente de leitura do banco de dados Kerberos também pode residir em

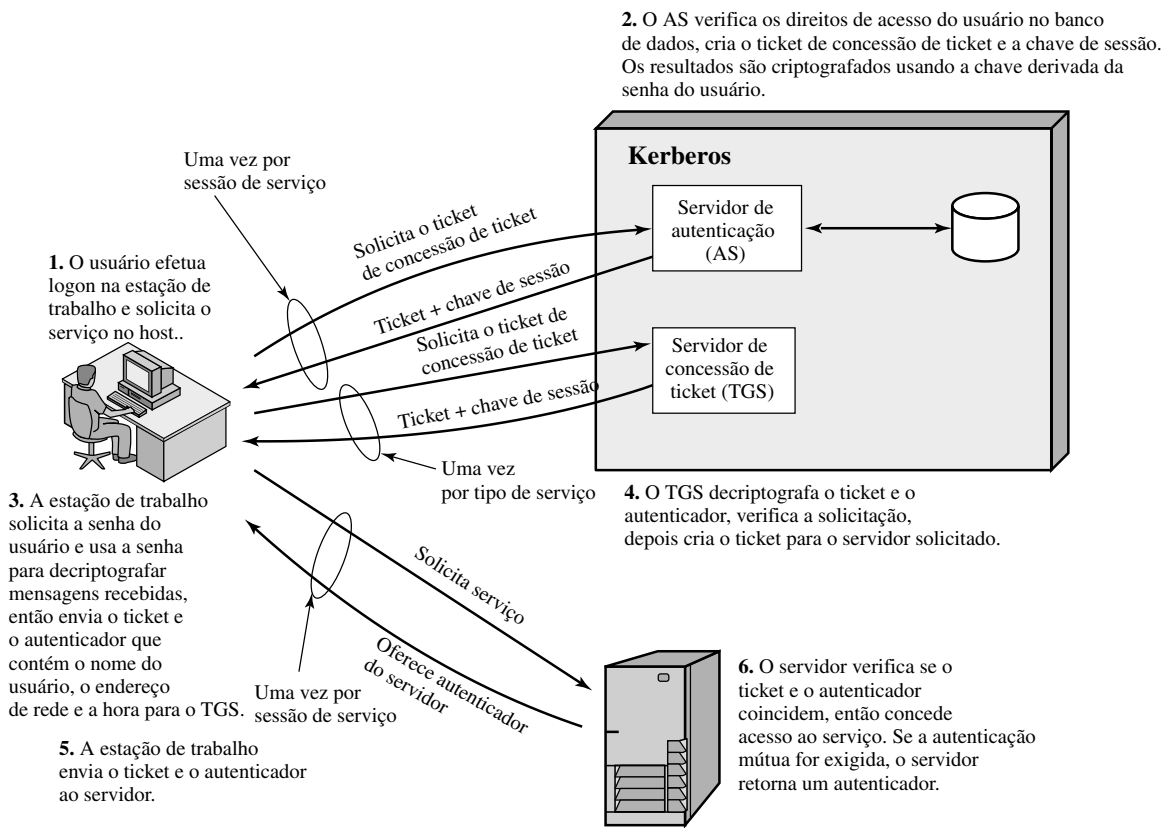


Figura 14.1 Visão geral do Kerberos.

outros sistemas de computador Kerberos. Porém, todas as mudanças no banco de dados precisam ser feitas no sistema de computador-mestre. Para alterar ou acessar o conteúdo de um banco de dados Kerberos, é preciso da senha-mestra do Kerberos. Um conceito relacionado é o de um **Kerberos principal**, que é um serviço ou usuário conhecido do sistema Kerberos. Cada Kerberos principal é identificado por seu nome principal. Nomes principais consistem em três partes: um nome de serviço ou usuário, um nome de instância e um nome de domínio.

Redes de clientes e servidores sob diferentes organizações administrativas normalmente constituem domínios diferentes. Ou seja, geralmente não é prático, ou não está de acordo com a política administrativa, que usuários e servidores em um domínio administrativo se registrem em um servidor Kerberos em outro lugar. Porém, os usuários em um domínio podem precisar de acesso aos servidores em outros domínios e alguns servidores podem querer fornecer serviço aos usuários de outros domínios, desde que esses usuários sejam autenticados.

O Kerberos oferece um mecanismo para dar suporte a essa autenticação entre domínios. Para dois domínios admitirem a autenticação entre domínios, um terceiro requisito é incluído:

3. O servidor Kerberos em cada domínio interoperacional compartilha uma chave secreta com o servidor no outro domínio. Os dois servidores Kerberos são registrados um no outro.

O esquema exige que o servidor Kerberos em um domínio confie no servidor Kerberos no outro domínio para autenticar seus usuários. Além disso, os servidores participantes do segundo domínio também precisam confiar no servidor Kerberos no primeiro domínio.

Com essas regras básicas estabelecidas, podemos descrever o mecanismo da seguinte maneira (Figura 14.2): Um usuário desejando serviço em um servidor em outro domínio precisa de um ticket para esse servidor. O cliente do usuário segue os procedimentos normais para ganhar acesso ao TGS local e depois solicita um ticket de concessão de ticket a um TGS remoto (TGS em outro domínio). O cliente pode, então, solicitar ao TGS remoto um ticket de concessão de serviço para o servidor desejado no domínio do TGS remoto.

Os detalhes das trocas ilustradas na Figura 14.2 são os seguintes (compare com a Tabela 14.1):

(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C: $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| TempodeVida\ e_2 \| Ticket_{tgs}])$

(3) C → TGS: $ID_{tgsrem} \| Ticket_{tgs} \| Autenticador_c$
(4) TGS → C: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$
(5) C → TGS_{rem}: $ID_{vrem} \| Ticket_{tgsrem} \| Autenticador_c$
(6) TGS_{rem} → C: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$
(7) C → V_{rem}: $Ticket_{vrem} \| Autenticador_c$

O ticket apresentado ao servidor remoto (V_{rem}) indica o domínio em que o usuário foi autenticado originalmente. O servidor escolhe se deve atender a solicitação remota.

Um problema apresentado pela técnica anterior é que ela não se expande muito bem para muitos domínios. Se houver N domínios, então deverá haver $N(N - 1)/2$ trocas de chave seguras, de modo que cada domínio Kerberos possa interoperar com todos os outros domínios Kerberos.

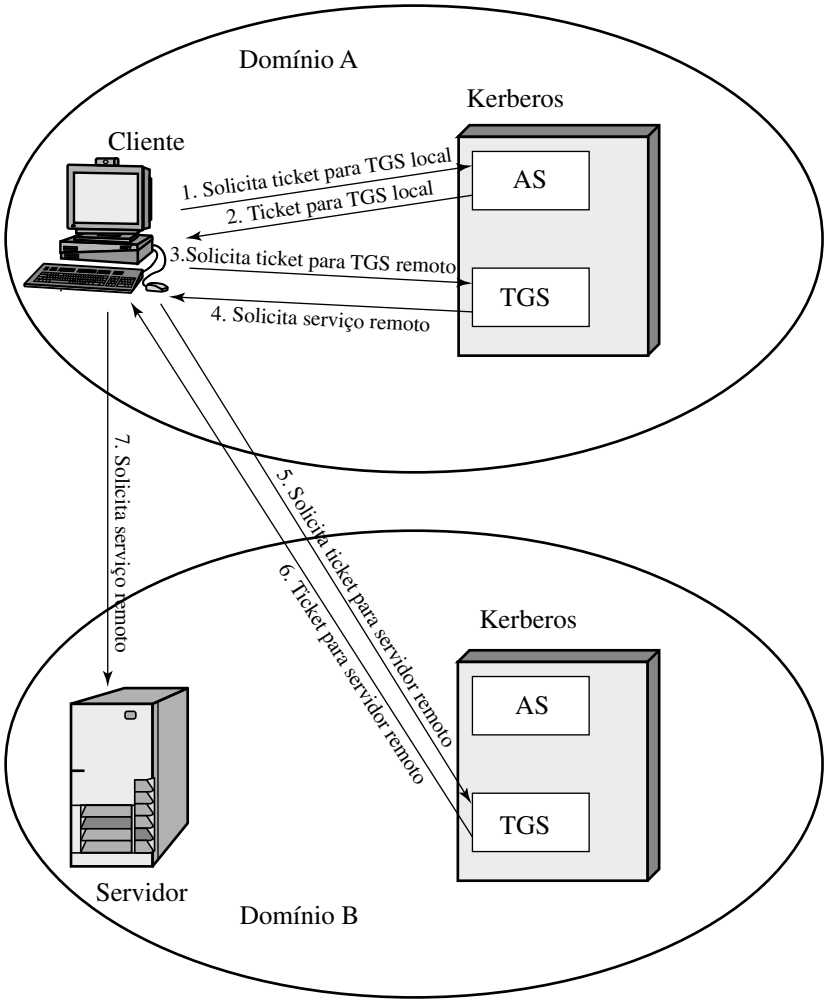


Figura 14.2 Solicitação de serviço em outro domínio.

Kerberos versão 5

O Kerberos versão 5 é especificado na RFC 1510 e oferece diversas melhorias em relação à versão 4 [KOHL94]. Para começar, oferecemos uma visão geral das mudanças da versão 4 para a versão 5 e depois examinamos o protocolo da versão 5.

Diferenças entre as versões 4 e 5 A versão 5 tem como objetivo resolver as limitações da versão 4 em duas áreas: limitações ambientais e deficiências técnicas. Vamos resumir rapidamente as melhorias em cada área.[6]

6. A explicação a seguir segue a apresentação em [KOHL94].

O Kerberos versão 4 foi desenvolvido para uso dentro do ambiente do projeto Athena e, por conseguinte, não atendia totalmente a necessidade de ser de uso geral. Isto levou às seguintes limitações ambientais:

1. **Dependência do sistema de criptografia:** A versão 4 exige o uso do DES. A restrição de exportação no DES, além de dúvidas sobre sua força, era uma preocupação. Na versão 5, o texto cifrado é marcado com um identificador de tipo de criptografia para que qualquer técnica de criptografia possa ser usada. As chaves de criptografia são marcadas com um tipo e um tamanho, permitindo que a mesma chave seja usada em diferentes algoritmos e permitindo a especificação de diferentes variações sobre determinado algoritmo.
2. **Dependência do protocolo da Internet:** A versão 4 requer o uso de endereços Internet Protocol (IP). Outros tipos de endereço, como o endereço de rede ISO, não são aceitos. Os endereços de rede da versão 5 são marcados com tipo e tamanho, permitindo que qualquer tipo de endereço de rede seja utilizado.
3. **Ordenação de byte da mensagem:** Na versão 4, o emissor de uma mensagem emprega uma ordenação de byte de sua própria escolha e marca a mensagem para indicar o byte menos significativo no endereço mais baixo ou o byte mais significativo no endereço mais baixo. Essa técnica funciona, mas não segue convenções estabelecidas. Na versão 5, todas as estruturas de mensagem são definidas usando Abstract Syntax Notation One (ASN.1) e Basic Encoding Rules (BER), que oferecem uma ordenação de byte sem ambigüidade.
4. **Tempo de vida do ticket:** Os valores de tempo de vida na versão 4 são codificados em uma quantidade de 8 bits em unidades de cinco minutos. Assim, o tempo de vida máximo que pode ser expresso é $2^8 \times 5 =$ 1.280 minutos, ou pouco mais de 21 horas. Isso pode ser inadequado para algumas aplicações (por exemplo, uma simulação de longa duração, que exija credenciais Kerberos válidas durante toda a execução). Na versão 5, os tickets incluem tempo de início e tempo de fim explícitos, permitindo tickets com tempos de vida arbitrários.
5. **Encaminhamento de autenticação:** A versão 4 não permite que credenciais emitidas para um cliente sejam encaminhadas para algum outro host e usadas por algum outro cliente. Essa capacidade permitiria que um cliente acessasse um servidor e fizesse esse servidor acessar outro servidor em nome do cliente. Por exemplo, um cliente emite uma solicitação a um servidor de impressão, que depois acessa o arquivo do cliente a partir de um servidor de arquivos, usando as credenciais do cliente para ter acesso. A versão 5 oferece essa capacidade.
6. **Autenticação entre domínios:** Na versão 4, a interoperabilidade entre N domínios requer algo na ordem de N^2 relacionamentos Kerberos-para-Kerberos, conforme descrito anteriormente. A versão 5 admite um método que requer menos relacionamentos, conforme explicaremos em breve.

Fora essas limitações ambientais, existem **deficiências técnicas** no próprio protocolo da versão 4. A maioria dessas deficiências foi documentada em [BELL90], e a versão 5 tenta resolvê-las. As deficiências são as seguintes:

1.s **Criptografia dupla:** Observe na Tabela 14.1 [mensagens (2) e (4)] que os tickets fornecidos aos clientes são criptografados duas vezes, uma com a chave secreta do servidor de destino e depois novamente com uma chave secreta conhecida do cliente. A segunda criptografia não é necessária e é computacionalmente dispendiosa.

2. **Criptografia PCBC:** A criptografia na versão 4 utiliza um modo fora do padrão do DES conhecido como *propagating cipher block chaining* (PCBC).[7] Tem sido demonstrado que esse modo é vulnerável a um ataque envolvendo o intercâmbio de blocos de texto cifrado [KOHL89]. O PCBC foi criado para oferecer uma verificação de integridade como parte da operação de criptografia. A versão 5 oferece mecanismos de integridade explícitos, permitindo que o modo CBC padrão seja usado para criptografia. Em particular, uma soma de verificação ou código hash é anexada à mensagem antes da criptografia usando CBC.
3. **Chaves de sessão:** Cada ticket inclui uma chave de sessão, que é usada pelo cliente para criptografar o autenticador enviado ao serviço associado a esse ticket. Além disso, a chave de sessão pode ser usada posteriormente pelo cliente e o servidor para proteger mensagens passadas durante essa sessão. Porém, como o mesmo ticket pode ser usado repetidamente para obter serviços de um determinado servidor, existe o risco de que um oponente replique mensagens de uma sessão antiga para o cliente ou servidor. Na versão 5, é possível que cliente e servidor negociem uma chave de subsessão, que só deve ser usada para essa única conexão. Um novo acesso pelo cliente resultaria no uso de uma nova chave de subsessão.
4. **Ataques de senha:** As duas versões são vulneráveis a um ataque de senha. A mensagem do AS para o cliente inclui material criptografado com uma chave baseada na senha do cliente.[8] Um oponente pode captar essa mensagem e tentar decriptografá-la experimentando diversas senhas. Se o resultado de uma decriptografia de teste estiver no formato apropriado, então o oponente terá descoberto a senha do cliente e poderá usá-la

7. Isso é descrito no Apêndice 14A.
8. O Apêndice 14A descreve o mapeamento entre senhas e chaves de criptografia.

subseqüentemente para obter credenciais de autenticação do Kerberos. Esse é o mesmo tipo de ataque de senha descrito no Capítulo 18, com os mesmos tipos de contramedidas sendo aplicáveis. A versão 5 oferece um mecanismo conhecido como pré-autenticação, que deverá tornar os ataques de senha mais difíceis, mas não os impedir

O diálogo de autenticação da versão 5 A Tabela 14.3 resume o diálogo básico da versão 5. Isso é mais bem explicado com uma comparação com a versão 4 (Tabela 14.1).

Tabela 14.3 Resumo das trocas de mensagem do Kerberos versão 5

(1) C → AS $\text{Options} \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS → C $Realm_c \| ID_C \| Ticket_{tgs} \| \text{E}(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$$Ticket_{tgs} = \text{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

(a) Troca de serviço de autenticação para obter o ticket de concessão de ticket

(3) C → TGS $\text{Options} \| ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $Realm_c \| ID_C \| Ticket_v \| \text{E}(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$$Ticket_{tgs} = \text{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

$$\text{Ticket}_v = \text{E}(\text{K}_v, [\text{Flags} \| \text{K}_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_C \| \text{Times}])$$

$$Authenticator_c = \text{E}(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$$

(b) Troca do serviço de concessão de ticket para obter o ticket de concessão de serviço

(5) C → V $\text{Options} \| \text{Ticket}_\text{v} \| \text{Authenticator}_\text{c}$

(6) V → C $\text{E}_{\text{K}_{c,v}}[\text{TS}_2 \| \text{Subkey} \| \text{Seq\#}]$

$$\text{Ticket}_v = \text{E}(\text{K}_v, [\text{Flags} \| \text{K}_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_C \| \text{Times}])$$

$$Authenticator_c = \text{E}(K_{c,v}, [ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#])$$

(c) Troca de autenticação cliente/servidor para obter o serviço

Primeiramente, considere a **troca do serviço de autenticação**. A mensagem (1) é uma solicitação do cliente para um ticket de concessão de ticket. Como antes, ela inclui o ID do usuário e o TGS. Os novos elementos a seguir são acrescentados:

- **Domínio:** indica o domínio do usuário
- **Opções:** usado para solicitar que certos flags sejam marcados no ticket retornado
- **Tempos:** usado pelo cliente para solicitar as seguintes definições de tempo no ticket:
  — from: a hora inicial desejada para o ticket solicitado
  — till: a hora de expiração solicitada para o ticket solicitado
  — rtime: o tempo-limite de renovação solicitado (*renew-till time*)
- **Nonce:** um valor aleatório a ser repetido na mensagem (2) para garantir que a resposta é recente e não foi repetida por um oponente

A mensagem (2) retorna um ticket de concessão de ticket, identificando a informação para o cliente, e um bloco criptografado usando a chave de criptografia baseada na senha do usuário. Esse bloco inclui a chave de sessão a ser usada entre o cliente e o TGS, tempos específicos na mensagem (1), o nonce da mensagem (1) e informação de identificação do TGS. O ticket em si inclui a chave de sessão, informação de identificação para o cliente, os valores de tempo solicitados e flags que refletem o status desse ticket e das opções solicitadas. Esses flags introduzem nova funcionalidade significativa à versão 5. Por enquanto, adiamos uma explicação desses flags e nos concentramos na estrutura geral do protocolo da versão 5.

Vamos agora comparar a **troca do serviço de concessão de ticket** para as versões 4 e 5. Vemos que a mensagem (3) para as duas versões inclui um autenticador, um ticket e o nome do serviço solicitado. Além disso, a versão 5 inclui os tempos solicitados e opções para o ticket e um nonce, tudo com funções semelhantes àquelas da mensagem (1). O próprio autenticador é basicamente o mesmo que o usado na versão 4.

A mensagem (4) tem a mesma estrutura da mensagem (2), retornando um ticket mais informações necessárias para o cliente, sendo a última criptografada com a chave de sessão agora compartilhada pelo cliente e o TGS.

Finalmente, para a **troca de autenticação cliente/servidor**, vários recursos novos aparecem na versão 5. Na mensagem (5), o cliente pode solicitar como opção que a autenticação mútua seja exigida. O autenticador inclui vários campos novos, como:

- **Subchave:** A escolha do cliente para uma chave de criptografia a ser usada para proteger essa sessão de aplicação específica. Se esse campo for omitido, a chave de sessão do ticket ($K_{c,v}$) será usada.
- **Número de seqüência:** Um campo opcional que especifica o número de seqüência inicial a ser usado pelo servidor para mensagens enviadas ao cliente durante essa sessão. As mensagens podem ser numeradas em seqüência para detectar repetições.

Se for exigida a autenticação mútua, o servidor responderá com a mensagem (6). Essa mensagem inclui o carimbo de tempo do autenticador. Observe que, na versão 4, o carimbo de tempo foi incrementado em um. Isso não é necessário na versão 5, pois a natureza do formato das mensagens é tal que não é possível que um oponente crie a mensagem (6) sem conhecimento das chaves de criptografia apropriadas. O campo de subchave, se estiver presente, substitui o campo de subchave, se estiver presente, na mensagem (5). O campo opcional de número de seqüência especifica o número de seqüência inicial a ser usado pelo cliente.

Flags do ticket O campo de flags incluído nos tickets da versão 5 admite funcionalidade expandida em comparação com o que existe na versão 4. A Tabela 14.4 resume os flags que podem ser incluídos em um ticket.

Tabela 14.4 Flags Kerberos versão 5

INITIAL	Este ticket foi emitido usando o protocolo AS e não emitido com base em um ticket de concessão de ticket.
PRE-AUTHENT	Durante a autenticação inicial, o cliente foi autenticado pelo KDC antes que um ticket fosse emitido.
HW-AUTHENT	O protocolo empregado para autenticação inicial exigia o uso de hardware que deveria ser de posse apenas do cliente indicado.
RENEWABLE	Diz ao TGS que este ticket pode ser usado para obter um ticket substituto que expira em uma data posterior.
MAY-POSTDATE	Diz ao TGS que um ticket pós-datado pode ser emitido com base nesse ticket de concessão de ticket.
POSTDATED	Indica que este ticket foi pós-datado; o servidor final pode verificar o campo authtime para verificar quando ocorreu a autenticação original.
INVALID	O ticket é inválido e precisa ser validado pelo KDC antes do uso.
PROXIABLE	Diz ao TGS que um novo ticket de concessão de serviço, com um endereço de rede diferente, pode ser emitido com base no ticket apresentado. PROXY indica que este ticket é um proxy.
FORWARDABLE	Diz ao TGS que um novo ticket de concessão de ticket, com um endereço de rede diferente, pode ser emitido com base no ticket de concessão de ticket.
FORWARDED	Indica que o ticket foi encaminhado ou emitido com base na autenticação envolvendo um ticket de concessão de ticket encaminhado.

O flag INITIAL indica que o ticket foi emitido pelo AS, e não pelo TGS. Quando um cliente solicita um ticket de concessão de serviço do TGS, ele apresenta um ticket de concessão de ticket obtido do AS. Na versão 4, essa era a única maneira de obter um ticket de concessão de serviço. A versão 5 oferece a capacidade adicional de o cliente poder obter um ticket de concessão de serviço diretamente do AS. A utilidade disso é a seguinte: um servidor, como um servidor de mudança de senha, pode querer saber se a senha do cliente foi testada recentemente.

O flag PRE-AUTHENT, se marcado, indica que, quando o AS recebeu a solicitação inicial [mensagem (1)], ele autenticou o cliente antes de emitir um ticket. A forma exata dessa pré-autenticação não é especificada. Como exemplo, a implementação do MIT para a versão 5 possui pré-autenticação do carimbo de tempo criptografado, ativada como padrão. Quando um usuário deseja obter um ticket, ele precisa enviar ao AS um bloco de pré-autenticação contendo um misturador aleatório, um número de versão e um caminho de tempo, criptografados com a chave baseada na senha do cliente. O AS decriptografa o bloco e não enviará um ticket de concessão de ticket de volta a menos que o carimbo de tempo no bloco de pré-autenticação esteja dentro do período de tempo permissível (intervalo de tempo considerando variação de clock e atrasos da rede). Outra possibilidade é o uso de um cartão inteligente que gera senhas que mudam continuamente, incluídas nas mensagens pré-autenticadas. As senhas geradas pelo cartão podem ser baseadas na senha de um usuário, mas transformadas pelo cartão de modo que, de fato, senhas arbitrárias sejam

usadas. Isso impede um ataque baseado em senhas de fácil adivinhação. Se um cartão inteligente ou um dispositivo semelhante for usado, isso será indicado pelo flag HW-AUTHENT.

Quando um ticket tem um tempo de vida longo, existe a possibilidade de que ele seja roubado e usado por um oponente por um período considerável. Se um tempo de vida curto for usado para diminuir a ameaça, então o overhead é envolvido na aquisição de novos tickets. No caso de um ticket de concessão de ticket, o cliente teria de armazenar a chave secreta do usuário, o que certamente é arriscado, ou pedir repetidamente uma senha ao usuário. Um meio-termo é o uso de tickets renováveis. Um ticket com o flag RENEWABLE marcado inclui dois tempos de expiração: um para esse ticket específico e um que é o último valor permissível para um tempo de expiração. Um cliente pode ter o ticket renovado apresentando ao TGS uma solicitação com um novo tempo de expiração. Se o novo tempo estiver dentro do limite do maior valor permissível, o TGS pode emitir um novo ticket com um novo tempo de sessão e um novo tempo de expiração específica. A vantagem desse mecanismo é que o TGS pode se recusar a renovar um ticket informado como roubado.

Um cliente pode solicitar que o AS forneça um ticket de concessão de ticket com o flag MAY-POSTDATE marcado. O cliente pode, então, usar esse ticket para solicitar um ticket marcado como POSTDATED e INVALID a partir do TGS. Posteriormente, o cliente pode enviar o ticket pós-datado para validação. Esse esquema pode ser útil para executar uma tarefa longa em batch em um servidor que exige um ticket periodicamente. O cliente pode obter diversos tickets para essa sessão de uma só vez, com valores de tempo espalhados. Todos além do primeiro ticket são inicialmente inválidos. Quando a execução chegar a um ponto no tempo em que um novo ticket for solicitado, o cliente poderá obter o ticket apropriadamente validado. Com essa técnica, o cliente não precisa usar repetidamente seu ticket de concessão de ticket para obter um ticket de concessão de serviço.

Na versão 5, é possível um servidor atuar como um proxy em favor de um cliente, adotando assim as credenciais e os privilégios do cliente para solicitar um serviço de outro servidor. Se um cliente quiser usar esse mecanismo, ele solicita um ticket de concessão de ticket com o flag PROXIABLE marcado. Quando esse ticket for apresentado ao TGS, o TGS terá permissão para emitir um ticket de concessão de serviço com um endereço de rede diferente; esse último ticket terá seu flag PROXY marcado. Uma aplicação recebendo esse ticket poderá aceitá-lo ou exigir autenticação adicional para fornecer uma trilha de auditoria.[9]

O conceito de proxy é um caso limitado do procedimento mais poderoso de encaminhamento. Se um ticket for definido com o flag FORWARDABLE, um TGS pode emitir ao solicitante um ticket de concessão de ticket com um endereço de rede diferente e o flag FORWARDED marcado. Esse ticket pode então ser apresentado ao TGS remoto. Essa capacidade permite que um cliente obtenha acesso a um servidor em outro domínio sem exigir que cada Kerberos mantenha uma chave secreta com servidores Kerberos em cada um dos outros domínios. Por exemplo, os domínios poderiam ser estruturados hierarquicamente. Então, um cliente poderia subir na árvore até um nó comum e depois retroceder para chegar a um domínio de destino. Cada etapa da caminhada envolveria o encaminhamento de um ticket de concessão de ticket ao próximo TGS no caminho.

14.2 SERVIÇO DE AUTENTICAÇÃO X.509

A recomendação X.509 do ITU-T faz parte da série de recomendações X.500 que definem um serviço de diretório. O diretório é, na verdade, um servidor ou conjunto de servidores distribuídos que mantém um banco de dados de informações sobre os usuários. As informações incluem um mapeamento entre o nome de usuário e o endereço de rede, além de outros atributos e informações sobre os usuários.

O X.509 define uma estrutura para a provisão de serviços de autenticação pelo diretório X.500 aos seus usuários. O diretório pode servir como um repositório de certificados de chave pública do tipo explicado no Capítulo 9. Cada certificado contém a chave pública de um usuário e é assinado com a chave privada de uma autoridade de certificação confiável. Além disso, o X.509 define protocolos de autenticação alternativos com base no uso de certificados de chave pública.

O X.509 é um padrão importante porque a estrutura de certificado e os protocolos de autenticação definidos no X.509 são usados em vários contextos. Por exemplo, o formato de certificado X.509 é usado em S/MIME (Capítulo 15), IP Security (Capítulo 16), e SSL/TLS e SET (Capítulo 17).

O X.509 foi emitido inicialmente em 1988. O padrão foi revisado posteriormente para resolver alguns dos problemas de segurança documentados em [IANS90] e [MITC90]; uma recomendação revisada foi emitida em 1993. Uma terceira versão foi emitida em 1995 e revisada em 2000.

O X.509 é baseado no uso da criptografia de chave pública e assinaturas digitais. O padrão não dita o uso de um algoritmo específico, mas recomenda o RSA. Presume-se que o esquema de assinatura digital exija o uso de uma função hash. Novamente, o padrão não dita um algoritmo hash específico. A recomendação de 1988 incluía a descrição de um algoritmo hash recomendado; esse algoritmo desde então provou ser inseguro e foi retirado da recomendação de 1993. A Figura 14.3 ilustra a geração de um certificado de chave pública.

9. Para obter uma explicação de alguns dos usos possíveis da capacidade de proxy, consulte [NEUM93b].

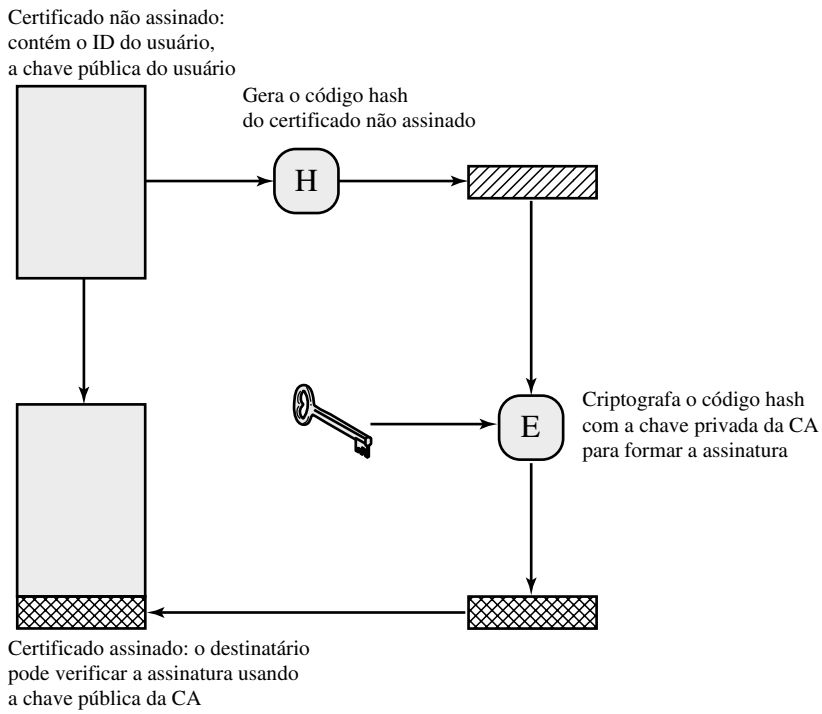


Figura 14.3 Uso do certificado de chave pública.

Certificados

O núcleo do esquema X.509 é o certificado de chave pública associado a cada usuário. Esses certificados de usuário são considerados como sendo criados por alguma autoridade certificadora (CA) confiável e colocados no diretório pela CA ou pelo usuário. O próprio servidor de diretório não é responsável pela criação das chaves públicas ou pela função de certificação; ele simplesmente oferece um local de fácil acesso para os usuários obterem certificados.

A Figura 14.4a mostra o formato geral de um certificado, que inclui os seguintes elementos:

- **Versão:** Diferencia entre versões sucessivas do formato do certificado; o padrão é a versão 1. Se o Identificador Exclusivo do Emissor ou o Identificador Exclusivo do Titular estiverem presentes, o valor precisa ser versão 2. Se uma ou mais extensões estiverem presentes, a versão precisa ser a versão 3.
- **Número de série:** Um valor inteiro, exclusivo dentro da CA emitente, que é associado sem ambigüidades a esse certificado.
- **Identificador do algoritmo de assinatura:** O algoritmo usado para assinar o certificado, juntamente com quaisquer parâmetros associados. Como essa informação é repetida no campo Assinatura, ao final do certificado, o campo tem pouca ou nenhuma utilidade.
- **Nome do emissor:** O nome X.500 da CA que criou e assinou o certificado.
- **Período de validade:** Consiste em duas datas: a primeira e a última em que o certificado é válido.
- **Nome do titular:** O nome do usuário a quem o certificado se refere. Ou seja, o certificado certifica a chave pública do titular que mantém a chave privada correspondente.
- **Informação de chave pública do titular:** A chave pública do titular, mais um identificador do algoritmo para o qual a chave deve ser usada, juntamente com quaisquer parâmetros associados.
- **Identificador exclusivo do emissor:** Um campo de seqüência de bits opcional usado para identificar exclusivamente a CA emissora caso o nome X.500 tenha sido reutilizado para entidades diferentes.
- **Identificador exclusivo do titular:** Um campo de seqüência de bits opcional, usado para identificar exclusivamente o titular caso o nome X.500 tenha sido reutilizado para diferentes entidades.
- **Extensões:** Um conjunto de um ou mais campos de extensão. As extensões foram adicionadas na versão 3 e são explicadas mais adiante nesta seção.
- **Assinatura:** Abrange todos os outros campos do certificado; ela contém o código hash dos outros campos, criptografados com a chave privada da CA. Este campo inclui o identificador do algoritmo de assinatura.

Os campos de identificadores exclusivos foram adicionados na versão 2 para lidar com a possível reutilização dos nomes de titular e/ou emissor com o passar do tempo. Esses campos raramente são usados.

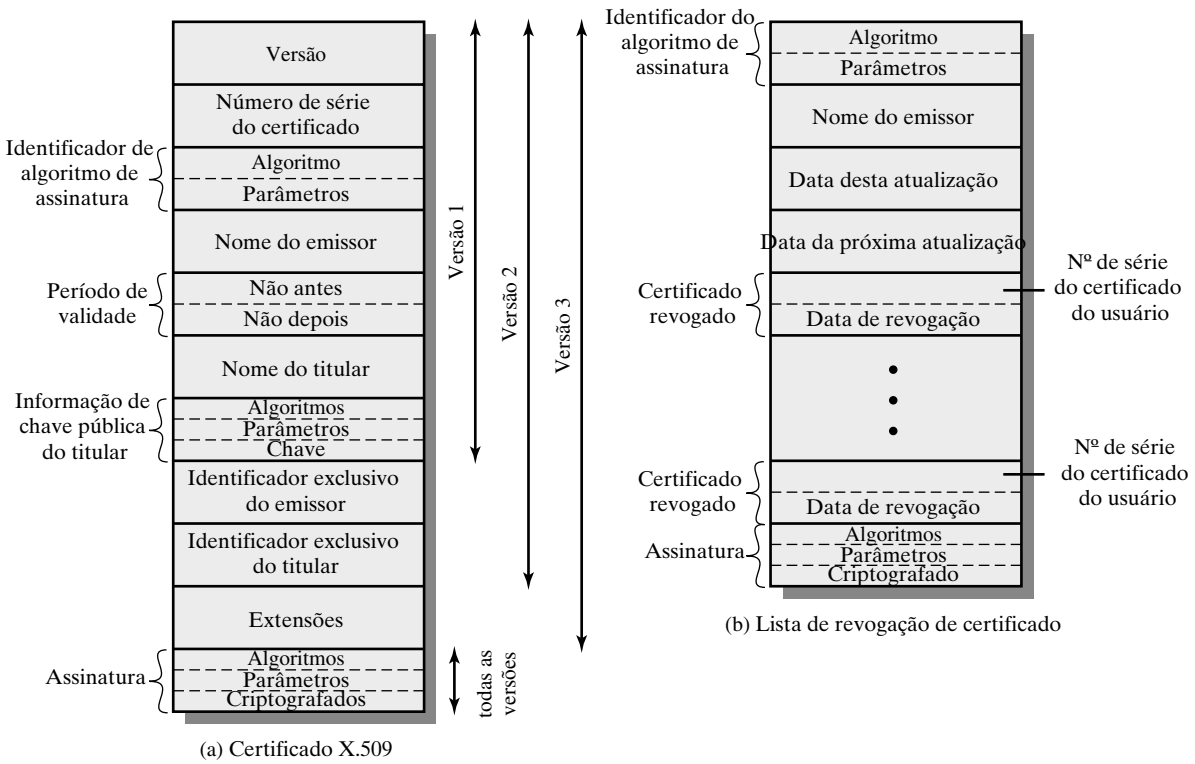


Figura 14.4 Formatos X.509.

O padrão usa a seguinte notação para definir um certificado:

$$CA \ll A \gg = CA\ \{V, SN, AI, CA, T_A, A, Ap\}$$

onde

$Y \ll X \gg$ = o certificado do usuário X emitido pela autoridade certificadora Y
$Y\{I\}$ = a assinatura de I por Y. Consiste em I com um código hash criptografado anexado

A CA assina o certificado com sua chave privada. Se a chave pública correspondente for conhecida por um usuário, então esse usuário poderá verificar se o certificado assinado pela CA é válido. Essa é a técnica de assinatura digital típica ilustrada na Figura 11.5c.

Obtendo o certificado de um usuário Certificados do usuário gerados por uma CA têm as seguintes características:

- Qualquer usuário com acesso à chave pública da CA pode verificar a chave pública do usuário, que foi certificada.
- Nenhuma parte além da autoridade certificadora pode modificar o certificado sem que isso seja detectado.

Como os certificados não podem ser falsificados, eles podem ser colocados em um diretório sem a necessidade de esforços especiais para protegê-los.

Se todos os usuários assinarem a mesma CA, então haverá uma confiança comum nessa CA. Todos os certificados do usuário podem ser colocados no diretório para que sejam acessados por todos os usuários. Além disso, um usuário pode transmitir seu certificado diretamente a outros usuários. De qualquer maneira, quando B está de posse do certificado de A, B pode confiar que as mensagens que ele criptografa com a chave pública de A serão protegidas contra espionagem e que as mensagens assinadas com a chave privada de A não serão falsificadas.

Se houver uma grande comunidade de usuários, pode não ser prático que todos os usuários se inscrevam na mesma CA. Como é a CA que assina os certificados, cada usuário participante precisa ter uma cópia da própria chave pública da CA para verificar assinaturas. Essa chave pública precisa ser fornecida a cada usuário de uma maneira absolutamente segura (com relação à integridade e autenticidade), de modo que o usuário tenha confiança nos certi-

ficados associados. Assim, com muitos usuários, pode ser mais prático que haja diversas CAs, cada qual oferecendo seguramente sua chave pública a uma parte dos usuários.

Agora suponha que A tenha obtido um certificado da autoridade certificadora X_1 e B tenha obtido um certificado da CA X_2. Se A não conhece com segurança a chave pública de X_2, então o certificado de B, emitido por X_2, é inútil para A. A pode ler o certificado de B, mas A não pode verificar a assinatura. Porém, se as duas CAs tiverem trocado seguramente suas próprias chaves públicas, o procedimento a seguir permitirá que A obtenha a chave pública de B:

1. A obtém, pelo diretório, o certificado de X_2 assinado por X_1. Como A conhece com segurança a chave pública de X_1, A pode obter a chave pública de X_2 a partir do certificado dele e verificá-la por meio da assinatura de X_1 no certificado.
2. A, então, volta para o diretório e obtém o certificado de B assinado por X_2. Como A agora tem uma cópia confiável da chave pública de X_2, A pode verificar a assinatura e obter a chave pública de B com segurança.

A usou uma cadeia de certificados para obter a chave pública de B. Na notação X.509, essa cadeia é expressa como

$$X_1 \langle\langle X_2 \rangle\rangle X_2 \langle\langle B \rangle\rangle$$

Da mesma maneira, B pode obter a chave pública de A com a cadeia inversa:

$$X_2 \langle\langle X_1 \rangle\rangle X_1 \langle\langle A \rangle\rangle$$

Esse esquema não precisa ser limitado a uma cadeia de dois certificados. Um caminho de CAs arbitrariamente longo pode ser seguido para produzir uma cadeia. Uma cadeia com *N* elementos seria expressa como

$$X_1 \langle\langle X_2 \rangle\rangle X_2 \langle\langle X_3 \rangle\rangle \cdots X_N \langle\langle B \rangle\rangle$$

Nesse caso, cada par de CAs na cadeia (X_i, X_{i+1}) precisa ter criado certificados para cada uma das outras.

Todos esses certificados de CAs por CAs precisam aparecer no diretório, e o usuário precisa saber como eles estão ligados para seguir um caminho para um outro certificado de chave pública de usuário. O X.509 sugere que as CAs sejam organizadas em uma hierarquia de modo que a navegação seja direta.

A Figura 14.5, tirada do X.509, é um exemplo dessa hierarquia. Os círculos conectados indicam o relacionamento hierárquico entre as CAs; as caixas associadas indicam certificados mantidos no diretório para cada entrada de CA. A entrada de diretório para cada CA inclui dois tipos de certificados:

- **Certificados diretos:** Certificados de X gerados por outras CAs
- **Certificados reversos:** Certificados gerados por X que são os certificados das outras CAs

Nesse exemplo, o usuário A pode adquirir os seguintes certificados a partir do diretório para estabelecer um caminho de certificação para B:

$$X \langle\langle W \rangle\rangle W \langle\langle V \rangle\rangle V \langle\langle Y \rangle\rangle Y \langle\langle Z \rangle\rangle Z \langle\langle B \rangle\rangle$$

Quando A tiver obtido esses certificados, ele poderá abrir o caminho de certificação na seqüência para recuperar uma cópia confiável da chave pública de B. Usando essa chave pública, A pode enviar mensagens criptografadas para B. Se A quiser receber de volta mensagens criptografadas de B, então B exigirá a chave pública de A, que pode ser obtida pelo seguinte caminho de certificação:

$$Z \langle\langle Y \rangle\rangle Y \langle\langle V \rangle\rangle V \langle\langle W \rangle\rangle W \langle\langle X \rangle\rangle X \langle\langle A \rangle\rangle$$

B pode obter esse conjunto de certificados a partir do diretório, ou A pode fornecê-los como parte de sua mensagem inicial para B.

Revogação de certificados Lembre-se, como mostrado na Figura 14.4, de que cada certificado inclui um período de validade, muito semelhante a um cartão de crédito. Normalmente, um novo certificado é emitido imediatamente antes da expiração do antigo. Além disso, pode-se desejar, na ocasião, revogar um certificado antes que ele expire, por um dos seguintes motivos:

1. A chave privada do usuário foi considerada comprometida.
2. O usuário não é mais certificado pela CA.
3. O certificado da CA foi considerado comprometido.

Cada CA precisa manter uma lista consistindo em todos os certificados revogados, porém não expirados, emitidos por ela, incluindo aqueles emitidos aos usuários e a outras CAs. Essa lista também deve ser postada no diretório.

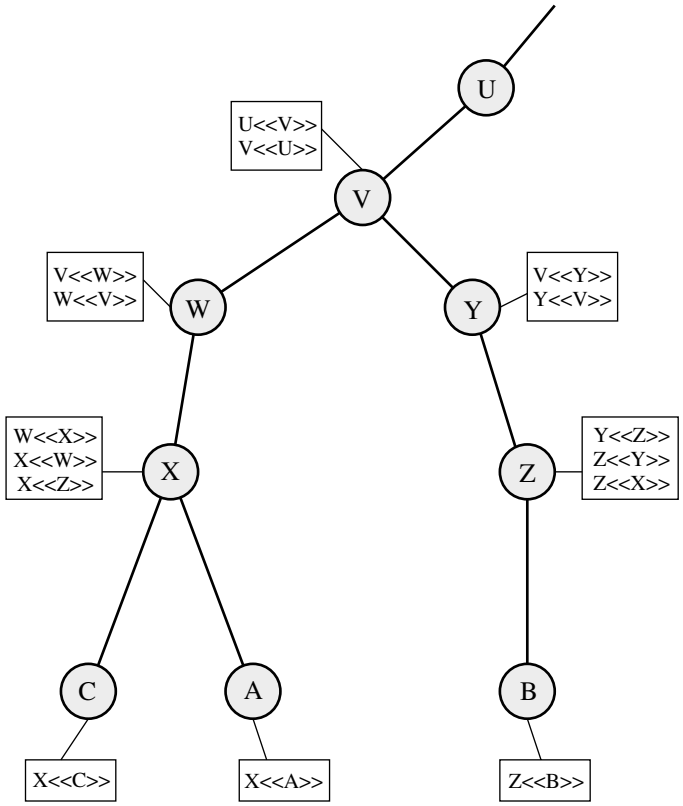


Figura 14.5 Hierarquia X.509: um exemplo hipotético.

Cada lista de revogação de certificado (CRL) postada no diretório é assinada pelo emissor e inclui (Figura 14.4b) o nome do emissor, a data em que a lista foi criada, a data em que a próxima CRL está agendada para ser emitida e uma entrada para cada certificado revogado. Cada entrada consiste no número de série de um certificado e a data de revogação para esse certificado. Como os números de série são exclusivos dentro de uma CA, o número de série é suficiente para identificar o certificado.

Quando um usuário recebe um certificado em uma mensagem, o usuário precisa determinar se o certificado foi revogado. O usuário pode verificar o diretório toda vez em que um certificado for recebido. Para evitar atrasos (e possíveis custos) associados às buscas de diretório, é recomendável que o usuário mantenha um cache local de certificados e listas de certificados revogados.

Procedimentos de autenticação

O X.509 também inclui três procedimentos de autenticação alternativos para serem usados por diversas aplicações. Todos esses procedimentos utilizam assinaturas de chave pública. Considera-se que as duas partes conhecem a chave pública uma da outra, seja obtendo os certificados umas das outras pelo diretório ou porque o certificado está incluído na mensagem inicial de cada lado.

A Figura 14.6 ilustra os três procedimentos.

Autenticação de uma via A autenticação de uma via envolve uma única transferência de informações de um usuário (A) para outro (B), e estabelece o seguinte:

1. A identidade de A e que a mensagem foi gerada por A
2. Que a mensagem foi destinada a B
3. A integridade e a originalidade (ela não foi enviada várias vezes) da mensagem

Observe que somente a identidade da entidade que inicia é verificada nesse processo, e não a da entidade que responde.

No mínimo, a mensagem inclui um carimbo de tempo t_A, um nonce r_A e a identidade de B, e é assinada com a chave privada de A. O carimbo de tempo consiste em uma hora de geração (opcional) e em uma hora de expiração. Isso impede a entrega adiada de mensagens. O nonce pode ser usado para detectar ataques por repetição. O valor do nonce precisa ser exclusivo dentro do período de expiração da mensagem. Assim, B pode armazenar o nonce até que expire e rejeitar quaisquer novas mensagens com o mesmo nonce.

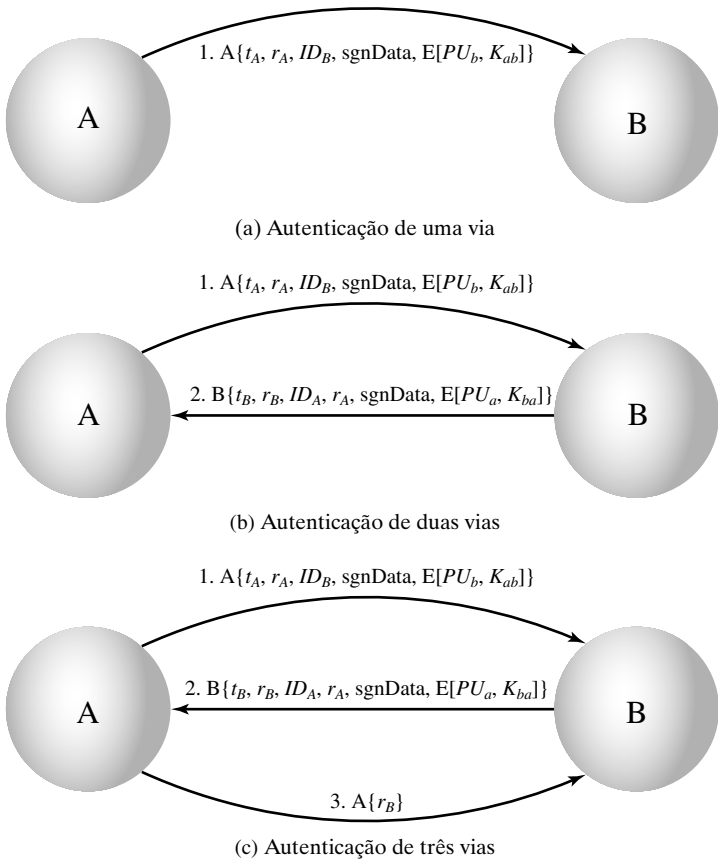


Figura 14.6 Procedimentos de autenticação forte X.509.

Por pura autenticação, a mensagem é usada simplesmente para apresentar credenciais para B. A mensagem também pode incluir informações a serem transportadas. Essas informações, *sgnData*, estão incluídas dentro do escopo da assinatura, garantindo sua autenticidade e integridade. A mensagem também pode ser usada para transportar uma chave de sessão para B, criptografada com a chave pública de B.

Autenticação de duas vias Além dos três elementos que acabamos de listar, a autenticação de duas vias estabelece os seguintes elementos:

4. A identidade de B e que a mensagem de resposta foi gerada por B
5. Que a mensagem foi destinada a A
6. A integridade e a originalidade da resposta

A autenticação de duas vias, assim, permite que ambas as partes de uma comunicação verifiquem a identidade uma da outra.

A mensagem de resposta inclui o nonce de A, para validar as resposta. Ela também inclui um carimbo de tempo e o nonce gerado por B. Como antes, a mensagem pode incluir informações adicionais assinadas e uma chave de sessão criptografada com a chave pública de A.

Autenticação de três vias Na autenticação de três vias, uma mensagem final de A para B é incluída, contendo uma cópia assinada do nonce r_B. A intenção desse projeto é que as marcas de data/hora não tenham de ser verificadas: como os dois nonces são repetidos de volta pelo outro lado, cada lado pode verificar o nonce retornado para detectar ataques por repetição. Essa técnica é necessária quando clocks sincronizados não estão disponíveis.

X.509 versão 3

O formato X.509 versão 2 não transporta todas as informações que os projetos e implementações mais recentes têm mostrado serem necessárias. [FORD95] lista os seguintes requisitos não satisfeitos pela versão 2:

1. O campo Titular (*subject*) é inadequado para transportar a identidade de um proprietário de chave a um usuário de chave pública. No X.509 os nomes podem ser relativamente curtos e sem detalhes de identificação óbvios, que podem ser necessários para o usuário.
2. O campo Titular também é inadequado para muitas aplicações, que normalmente reconhecem entidades por um endereço de e-mail da Internet, um URL ou alguma outra identificação relacionada à Internet.
3. Existe uma necessidade de indicar informações de política de segurança. Isso permite que uma aplicação ou função de segurança, como IPSec, relacione um certificado X.509 a determinada política.
4. Há necessidade de limitar o dano que pode resultar de uma CA defeituosa ou maliciosa, por meio da definição de restrições sobre a aplicabilidade de um determinado certificado.
5. É importante ser capaz de identificar diferentes chaves usadas pelo mesmo proprietário em diferentes ocasiões. Esse recurso admite gerenciamento de ciclo de vida da chave, em particular, a capacidade de atualizar pares de chave para usuários e CAs regularmente ou sob circunstâncias excepcionais.

Em vez de continuar a incluir campos para um formato fixo, os desenvolvedores de padrões sentiram que uma técnica mais flexível era necessária. Assim, a versão 3 inclui diversas extensões opcionais que podem ser acrescidas ao formato da versão 2. Cada extensão consiste em um identificador de extensão, um indicador de importância e um valor de extensão. O indicador de importância indica se uma extensão pode ser ignorada com segurança. Se esse indicador tiver um valor TRUE e uma implementação não reconhecer a extensão, ela deverá tratar o certificado como inválido.

As extensões de certificado podem ser de três categorias gerais: informações de chave e política, atributos de titular e emissor, e restrições de caminho de certificação.

Informações de chave e política Essas extensões carregam informações adicionais sobre as chaves do titular e do emissor, mais indicadores da política de certificado. Uma política de certificado é um conjunto nomeado de regras que indicam a aplicabilidade de um certificado a determinada comunidade e/ou classe de aplicação com requisitos de segurança comuns. Por exemplo, uma política poderia ser aplicável à autenticação de transações de intercâmbio eletrônico de dados (EDI) para o comércio de bens dentro de determinada faixa de preço.

Essa área inclui o seguinte:

- **Identificador da chave de autoridade:** Identifica a chave pública a ser usada para verificar a assinatura nesse certificado ou CRL. Permite que chaves distintas da mesma CA sejam diferenciadas. Um uso deste campo é para tratar da atualização do par de chaves da CA.
- **Identificador da chave do titular:** Identifica a chave pública sendo certificada. Útil para atualização do par de chaves do titular. Além disso, um titular pode ter vários pares de chaves e, por conseguinte, diferentes certificados para diferentes finalidades (por exemplo, assinatura digital e acordo de chaves de criptografia).
- **Uso de chave:** Indica uma restrição imposta como a finalidade para a qual, e as políticas sob as quais, a chave pública certificada pode ser usada. Pode indicar um ou mais dos seguintes: assinatura digital, irretratabilidade (*nonrepudiation*), criptografia de chave, criptografia de dados, acordo de chaves, verificação de assinatura da CA nos certificados, verificação de assinatura da CA nas CRLs.
- **Período de uso da chave privada:** Indica o período de uso da chave privada correspondente à chave pública. Normalmente, a chave privada é usada por um período diferente da validade da chave pública. Por exemplo, com as chaves de assinatura digital, o período de uso para a chave privada de assinatura normalmente é mais curto que para a chave pública de verificação.
- **Políticas de certificado:** Os certificados podem ser usados em ambientes onde diversas políticas se aplicam. Essa extensão lista políticas que o certificado reconhecidamente suporta, juntamente com informações qualificadoras opcionais.
- **Mapeamentos de política:** Usados apenas em certificados para CAs emitidos por outras CAs. Os mapeamentos de política permitem que uma CA emissora indique que uma ou mais das políticas do emissor podem ser consideradas equivalentes a outra política usada no domínio da CA subordinada.

Titular do certificado e atributos do emissor Essas extensões admitem nomes alternativos, em formatos alternativos, para um titular ou emissor do certificado, e podem carregar informações adicionais sobre o titular do certificado, para aumentar a confiança do usuário de que o titular do certificado é uma determinada pessoa ou entidade. Por exemplo, informações como endereço postal, cargo na empresa ou fotografia podem ser exigidas.

Os campos de extensão nesta área incluem os seguintes:

- **Nome alternativo do titular:** Contém um ou mais nomes alternativos, usando qualquer uma de diversas formas. Esse campo é importante para dar suporte a certas aplicações, como e-mail, EDI e IPSec, que podem empregar suas próprias formas de nome.
- **Nome alternativo do emissor:** Contém um ou mais nomes alternativos, usando qualquer uma de diversas formas.
- **Atributos de diretório do titular:** Transporta quaisquer valores de atributo de diretório X.500 para o titular desse certificado.

Restrições do caminho de certificação Essas extensões permitem que especificações de restrição sejam incluídas nos certificados emitidos para CAs por outras CAs. As restrições podem restringir os tipos de certificados que podem ser emitidos pela CA subordinada ou que podem ocorrer subseqüentemente em uma cadeia de certificação.

Os campos de extensão nessa área incluem os seguintes:

- **Restrições básicas:** Indica se o titular pode atuar como uma CA. Caso sim, poderá ser especificada uma restrição de tamanho do caminho de certificação.
- **Restrições de nome:** Indica um espaço de nomes dentro do qual todos os nomes de titular nos certificados subseqüentes em um caminho de certificação precisam estar localizados.
- **Restrições de política:** Especifica restrições que podem exigir identificação de política de certificado explícita ou inibir mapeamento de política pelo restante do caminho de certificação.

14.3 INFRA-ESTRUTURA DE CHAVE PÚBLICA

A RFC 2822 (*Internet Security Glossary*) define a infra-estrutura de chave pública (PKI – Public-Key Infrastructure) como o conjunto de hardware, software, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais com base na criptografia assimétrica. O objetivo principal para desenvolver uma PKI é permitir a aquisição segura, conveniente e eficiente de chaves públicas. O grupo de trabalho Public Key Infrastructure X.509 (PKIX) da Internet Engineering Task Force (IETF) tem sido a força motriz por trás da preparação de um modelo formal (e genérico) baseado no X.509, que seja adequado para a implantação de uma arquitetura baseada em certificado na Internet. Esta seção descreve o modelo PKIX.

A Figura 14.7 mostra o inter-relacionamento entre os principais elementos do modelo PKIX. Esses elementos são

- **Entidade final:** Um termo genérico usado para indicar os usuários finas, dispositivos (por exemplo, servidores, roteadores) ou qualquer outra entidade que possa ser identificada no campo titular de um certificado de chave pública. As entidades finais normalmente consomem e/ou dão suporte a serviços relacionados a PKI.
- **Autoridade de certificação (CA):** O emissor dos certificados e (normalmente) listas de revogação de certificado (CRLs). Também pode dar suporte a diversas funções administrativas, embora estas geralmente sejam delegadas a uma ou mais autoridades de registro.
- **Autoridade de registro (RA):** Um componente opcional que pode assumir diversas funções administrativas da CA. A RA normalmente está associada ao processo de registro da entidade final, mas também pode auxiliar em várias outras áreas.
- **Emissor da CRL:** Um componente opcional que uma CA pode delegar para publicar CRLs.
- **Repositório:** Um termo genérico usado para indicar qualquer método para armazenar certificados e CRLs de modo que possam ser recuperados por entidades finais.

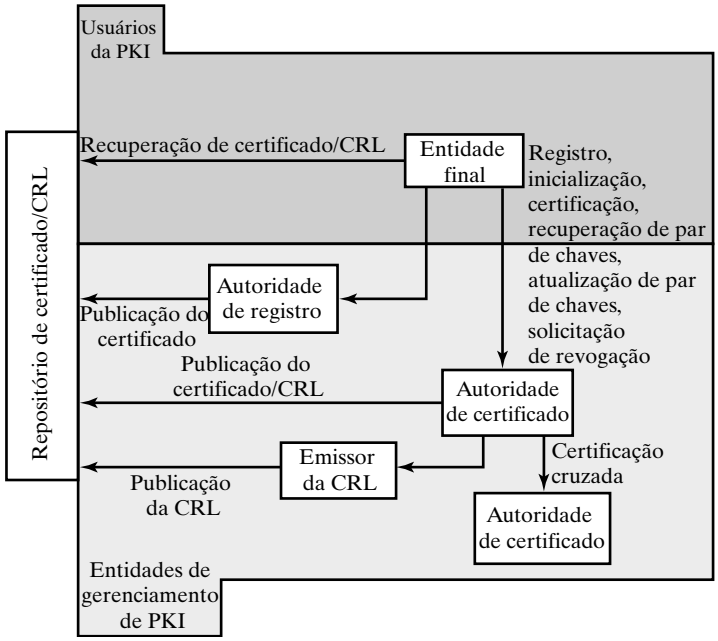


Figura 14.7 Modelo arquitetônico do PKIX.

Funções de gerenciamento do PKIX

O PKIX identifica diversas funções de gerenciamento que potencialmente precisam ser admitidas pelos protocolos de gerenciamento. Estas são indicadas na Figura 14.7 e incluem o seguinte:

- **Registro:** Este é o processo pelo qual um usuário primeiramente se torna conhecido por uma CA (diretamente, ou por meio de uma RA), antes que a CA emita um certificado ou certificados para o usuário. O registro inicia o processo de matrícula em uma PKI. O registro normalmente envolve algum procedimento off-line ou on-line para autenticação mútua. Normalmente, a entidade final recebe uma ou mais chaves secretas compartilhadas para posterior autenticação.
- **Inicialização:** Antes que um sistema cliente possa operar com segurança, é preciso instalar materiais-chave que possuem o relacionamento apropriado com as chaves armazenadas em outro lugar na infra-estrutura. Por exemplo, o cliente precisa ser inicializado com segurança com a chave pública e outras informações garantidas da(s) CA(s) confiáveis, para serem usadas na validação dos caminhos de certificação.
- **Certificação:** Este é o processo no qual uma CA emite um certificado para a chave pública de um usuário, e retorna esse certificado ao sistema cliente do usuário e/ou posta o certificado em um repositório.
- **Recuperação do par de chaves:** Os pares de chaves podem ser usados para dar suporte à criação e verificação de assinatura digital, criptografia e decriptografia, ou ambos. Quando um par de chaves é usado para criptografia/decriptografia, é importante oferecer um mecanismo para recuperar as chaves de decriptografia necessárias quando o acesso normal ao material de chave não for mais possível, ou então não será possível recuperar os dados criptografados. A perda de acesso à chave de decriptografia pode resultar de senhas/PINs esquecidos, unidades de disco corrompidas, danos a tokens de hardware etc. A recuperação do par de chaves permite que as entidades finais restaurem seu par de chaves de criptografia/decriptografia a partir de uma facilidade de backup de chave autorizada (normalmente, a CA que emitiu o certificado da entidade final).
- **Atualização do par de chaves:** Todos os pares de chaves precisam ser atualizados regularmente (ou seja, substituídos por um novo par de chaves) e novos certificados emitidos. A atualização é exigida quando o tempo de vida do certificado expira e como resultado da revogação do certificado.
- **Solicitação de revogação:** Uma pessoa autorizada avisa uma CA sobre uma situação anormal, exigindo revogação de certificado. Os motivos para revogação incluem comprometimento de chave privada, mudança na afiliação e troca de nome.
- **Certificação cruzada:** Duas CAs trocam informações usadas no estabelecimento de um certificado cruzado. Um certificado cruzado é aquele emitido por uma CA para outra CA, que contém uma chave de assinatura da CA usada para emissão de certificados.

Protocolos de gerenciamento de PKIX

O grupo de trabalho PKIX define dois protocolos de gerenciamento alternativos entre entidades PKIX que admitem funções de gerenciamento listadas na subseção anterior. A RFC 2510 define os protocolos de gerenciamento de certificado (CMP – *Certificate Management Protocols*). Dentro do CMP, cada uma das funções de gerenciamento é identificada explicitamente por trocas de protocolo específicas. O CMP foi elaborado para ser um protocolo flexível, capaz de acomodar diversos modelos técnicos, operacionais e comerciais.

A RFC 2797 define mensagens de gerenciamento de certificado sobre CMS (CMC), sendo que CMS refere-se à RFC 2630, sintaxe de mensagem criptográfica (Cryptographic Message Syntax). O CMC é baseado no trabalho anterior e tem como finalidade alavancar as implementações existentes. Embora todas as funções PKIX sejam admitidas, nem todas as funções possuem correspondência com trocas de protocolo específicas.

14.4 LEITURA E SITES WEB RECOMENDADOS

Um modo fácil de compreender os conceitos do Kerberos pode ser encontrado em [BRYA88]. Uma das melhores abordagens de Kerberos é [KOHL94]. [TUNG99] descreve o Kerberos do ponto de vista de um usuário.

[PERL99] analisa diversos modelos confiáveis que podem ser usados em uma PKI. [GUTM02] destaca as dificuldades no uso da PKI e recomenda técnicas para uma PKI eficaz.

BRYA88 Bryant, W. *Designing an authentication system: a dialogue in four scenes*. Documento do projeto Athena, fevereiro de 1988. Disponível em **http://web.mit.edu/kerberos/www/dialogue.html.**

GUTM02 Gutmann, P. "PKI: it's not dead, just resting." *Computer*, agosto de 2002.

KOHL94 Kohl, J.; Neuman, B.; e Ts'o, T. "The evolution of the Kerberos Authentication service." em Brazier, F., e Johansen, D. *Distributed Open Systems*. Los Alamitos: IEEE Computer Society Press, 1994. Disponível em **http://web.mit.edu/kerberos/www/papers.html**.

PERL99 Perlman, R. "An overview of PKI trust models." *IEEE Network*, novembro/dezembro de 1999.

TUNG99 Tung, B. *Kerberos: a network authentication system*. Reading: Addison-Wesley, 1999.

Sites Web recomendados

- **MIT Kerberos Site:** Informações sobre o Kerberos, incluindo a FAQ, artigos e documentos e indicadores de sites de produtos comerciais
- **USC/ISI Kerberos Page:** Outra boa fonte de material sobre o Kerberos
- **Kerberos Working Group:** Grupo do IETF desenvolvendo padrões com base no Kerberos
- **Public-Key Infrastructure Working Group:** Grupo do IETF desenvolvendo padrões com base no X.509v3
- **Verisign:** Um dos principais fornecedores de produtos relacionados a X.509; artigos e outros materiais valiosos neste site
- **NIST PKI Program:** Boa fonte de informações

14.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

autenticação
certificado de chave pública
certificado X.509
domínio
Kerberos
Kerberos, domínio
modo PCBC (Propagating Cipher Block Chaining)
nonce
número de seqüência
servidor de autenticação
servidor de concessão de ticket (TGS - Ticket-Granting Server)
subchave
tempo de vida
ticket

Perguntas de revisão

14.1 O Kerberos foi criado para resolver que problema?
14.2 Quais são as três ameaças associadas à autenticação do usuário por uma rede ou pela Internet?
14.3 Liste três técnicas para proteger a autenticação do usuário em um ambiente distribuído.
14.4 Quais os quatro requisitos definidos para o Kerberos?
14.5 Que entidades constituem um ambiente Kerberos com serviço completo?
14.6 No contexto do Kerberos, o que é um domínio?
14.7 Quais são as principais diferenças entre a versão 4 e a versão 5 do Kerberos?
14.8 Qual é a finalidade do padrão X.509?
14.9 O que é uma cadeia de certificados?
14.10 Como um certificado X.509 é revogado?

Problemas

14.1 Mostre que um erro aleatório em um bloco de texto cifrado é propagado para todos os blocos subseqüentes de texto claro no modo PCBC (Figura 14.9).

14.2 Suponha que, no modo PCBC, os blocos Ci e $Ci+1$ sejam trocados durante a transmissão. Mostre que isso afeta apenas os blocos decriptografados Pi e $Pi+1$, mas não os blocos subseqüentes.

14.3 O procedimento de autenticação de três vias original para X.509, ilustrado na Figura 14.6c, contém uma falha de segurança. A essência do protocolo é a seguinte:

$$\begin{aligned} A \rightarrow B: &\quad A\{t_A, r_A, ID_B\} \\ B \rightarrow A: &\quad B\{t_B, r_B, ID_A, r_A\} \\ A \rightarrow B: &\quad A\{r_B\} \end{aligned}$$

O texto do X.509 afirma que verificar as marcas de data/hora t_A e t_B é opcional para a autenticação de três vias. Mas considere o exemplo a seguir: suponha que A e B tenham usado o protocolo anterior em alguma ocasião, e que o oponente C tenha interceptado as três mensagens anteriores. Além disso, suponha que as marcas de data/hora não sejam usadas e sejam todas definidas como 0. Finalmente, suponha que C queira imitar A para B. C inicialmente envia a primeira mensagem captada para B:

$$C \rightarrow B: \quad A\{0, r_A, ID_B\}$$

B responde pensando que está falando com A, mas na realidade está falando com C:

$$B \rightarrow C: \quad B\{0, r'_B, ID_A, r_A\}$$

Nesse meio tempo, C faz com que A inicie a autenticação com C por algum meio. Como resultado, A envia o seguinte para C:

$$A \rightarrow C: \quad A\{0, r'_A, ID_C\}$$

C responde a A usando o mesmo nonce fornecido a C por B.

$$C \rightarrow A: \quad C\{0, r'_B, ID_A, r'_A\}$$

A responde com

$$A \rightarrow C: \quad A\{r'_B\}$$

É exatamente disso que C precisa para convencer B de que este está falando com A, de modo que C agora repete a mensagem que chega de volta para B.

$$C \rightarrow B: \quad A\{r'_B\}$$

Assim, B acreditará que está falando com A enquanto na realidade está falando com C. Sugira uma solução simples para esse problema, que não envolva o uso de carimbos de tempo.

14.4 A versão do X.509 de 1988 lista propriedades que as chaves RSA precisam satisfazer para serem seguras, dado o conhecimento atual sobre a dificuldade de fatorar números grandes. A explicação termina com uma restrição sobre o expoente público e o módulo n:

> Deve-se garantir que $e > \log_2 n$ para impedir o ataque apanhando-se a raiz de ordem e mod n para divulgar o texto claro.

Embora a restrição esteja correta, o motivo dado para exigi-la está incorreto. O que está errado com o motivo dado e qual é o motivo correto?

APÊNDICE 14A TÉCNICAS DE CRIPTOGRAFIA DO KERBEROS

O Kerberos inclui uma biblioteca de criptografia que admite várias operações relacionadas à criptografia. Estas foram incluídas na especificação Kerberos 5 e são comuns nas implementações comerciais. Em fevereiro de 2005, o IETF emitiu as RFCs 3961 e 3962, que expandem as opções das técnicas criptográficas. Neste apêndice, descrevemos as técnicas da RFC 1510.

Transformação de senha para chave

No Kerberos, as senhas são limitadas ao uso dos caracteres que podem ser representados em um formato ASCII de 7 bits. Essa senha, de tamanho arbitrário, é convertida em uma chave de criptografia que é armazenada no banco de dados Kerberos. A Figura 14.8 ilustra o procedimento.

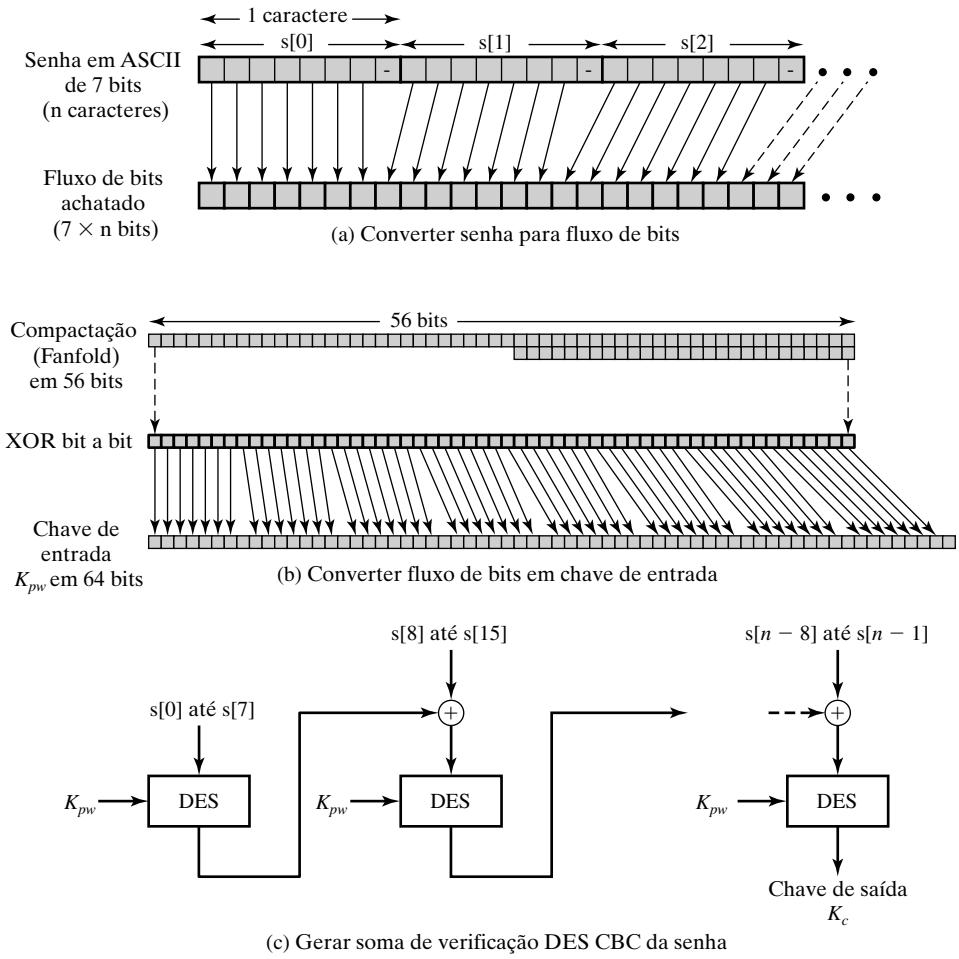


Figura 14.8 Geração de chave de criptografia a partir da senha.

Primeiro, a string de caracteres, s, é compactada para uma string de bits, b, de modo que o primeiro caractere seja armazenado nos 7 primeiros bits, o segundo caractere nos próximos 7 bits e assim por diante. Isso pode ser expresso como

$$
\begin{aligned}
& b[0] = \text{bit 0 de } s[0] \\
& \ldots \\
& b[6] = \text{bit 6 de } s[0] \\
& b[7] = \text{bit 0 de } s[1] \\
\ldots \quad & \\
& b[7i + m] = \text{bit } m \text{ de } s[i] \qquad 0 \leq m \leq 6
\end{aligned}
$$

Em seguida, a string de bits é compactada para 56 bits alinhando-se os bits de um modo 'sanfonado' e realizando-se o XOR bit a bit. Por exemplo, se a string de bits tiver comprimento 59, então

$$
\begin{aligned}
b[55] &= b[55] \oplus b[56] \\
b[54] &= b[54] \oplus b[57] \\
b[53] &= b[53] \oplus b[58]
\end{aligned}
$$

Isso cria uma chave DES de 56 bits. Para estar em conformidade com o formato de chave esperado de 64 bits, a string é tratada como uma seqüência de oito blocos de 7 bits e é mapeada para oito blocos de 8 bits para formar uma chave de entrada K_{pw}.

Finalmente, a senha original é criptografada usando-se o modo de encadeamento de bloco cifrado (CBC) do DES com chave K_{pw}. O último bloco de 64 bits retornado desse processo, conhecido como soma de verificação CBC, é a chave de saída associada a essa senha.

O algoritmo inteiro pode ser visto como uma função hash que mapeia uma senha arbitrária para um código hash de 64 bits.

Modo PCBC (Propagating Cipher Block Cipher)

Como explicado no Capítulo 6, no modo CBC do DES a entrada para o algoritmo DES em cada estágio consiste no XOR do bloco de texto claro atual com o bloco de texto cifrado anterior, com a mesma chave usada para cada bloco (Figura 6.4). A vantagem desse modo em relação ao modo Electronic CodeBook (ECB), em que cada bloco de texto claro é criptografado independentemente, é esta: com o CBC, o mesmo bloco de texto claro, se repetido, produz diferentes blocos de texto cifrado.

O CBC tem a propriedade de que, se houver um erro na transmissão do bloco de texto cifrado C_I, então esse erro se propaga para os blocos de texto claro recuperado P_I e P_{I+1}.

A versão 4 do Kerberos usa uma extensão do CBC, chamada modo *Propagating CBC* (PCBC) [MEYE82]. Esse modo tem a propriedade de que um erro em um bloco de texto cifrado é propagado para todos os blocos decriptografados subseqüentes da mensagem, tornando cada bloco inútil. Assim, a criptografia e integridade dos dados são combinadas em uma operação. (Veja uma exceção no Problema 14.2.)

O PCBC é ilustrado na Figura 14.9. Nesse esquema, a entrada do algoritmo de criptografia é o XOR do bloco de texto claro atual, o bloco de texto cifrado anterior e o bloco de texto claro anterior:

$$C_n = E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])$$

Na decriptografia, cada bloco de texto cifrado é passado pelo algoritmo de decriptografia. Depois, a saída passa por um XOR com o bloco de texto cifrado anterior e o bloco de texto claro anterior. Podemos demonstrar que esse esquema funciona, da seguinte forma:

$$
\begin{aligned}
D(K, C_n) &= D(K, E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])) \\
D(K, C_n) &= C_{n-1} \oplus P_{n-1} \oplus P_n \\
C_{n-1} \oplus P_{n-1} \oplus D(K, C_n) &= P_n
\end{aligned}
$$

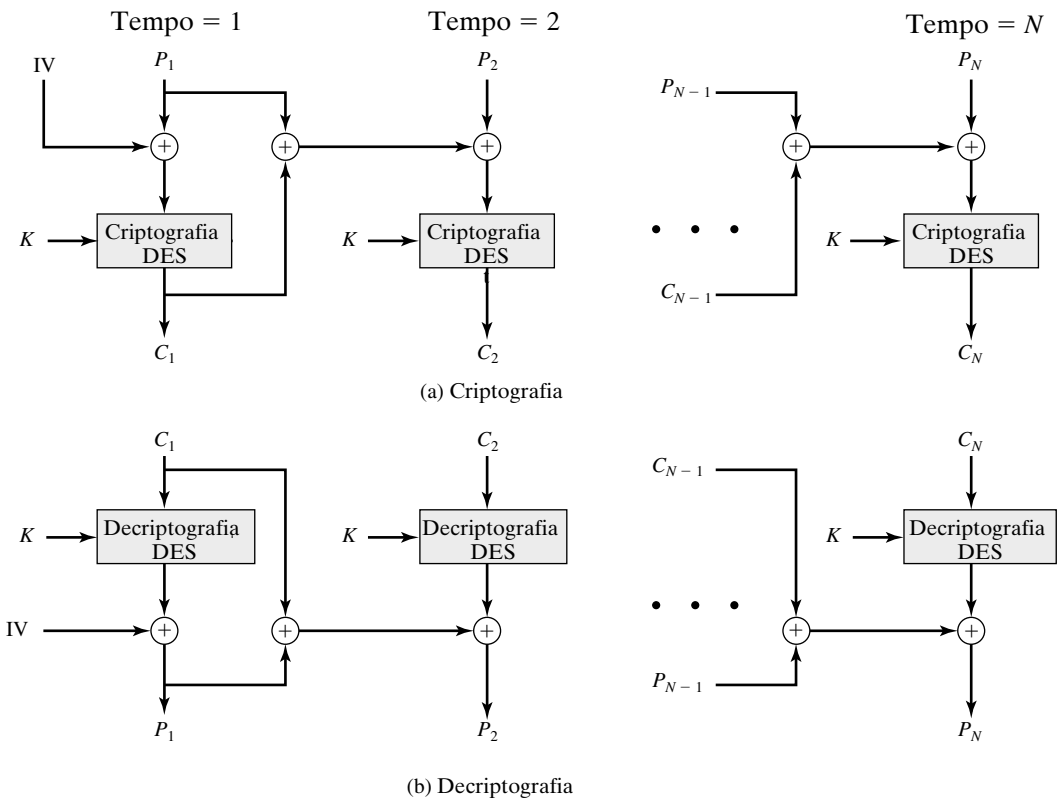
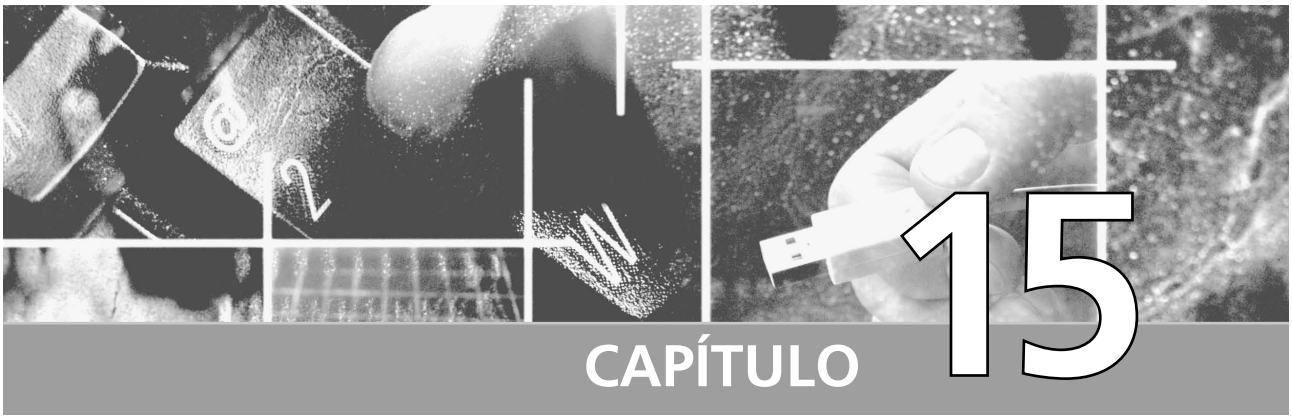


Figura 14.9 Modo Propagating Cipher Block Chaining (PCBC).



Segurança de e-mail

15.1 **Pretty Good Privacy**

15.2 **S/MIME**

15.3 **Principais termos, perguntas de revisão e problemas**

Apêndice 15A Compressão de dados usando ZIP

Apêndice 15B Conversão radix64

Apêndice 15C Geração de números aleatórios PGP

Apesar do vice-almirante Poindexter e do Tenente-Coronel North se recusarem a aparecer, o acesso da Comissão a outras fontes de informação preencheu grande parte dessa lacuna. O FBI forneceu documentos retirados dos arquivos do National Security Advisor e membros importantes do Conselho de Segurança Nacional dos EUA, incluindo mensagens do sistema PROF entre Poindexter e North. As mensagens do PROF eram conversas por computador, escritas na época em que os eventos ocorreram e consideradas pelos autores como protegidas contra divulgação. Nesse sentido, elas oferecem um relato de primeira mão e contemporâneo dos eventos.

— O relatório da comissão Tower ao presidente Reagan sobre o caso Irã-Contras, 1987

PONTOS PRINCIPAIS

- O PGP é um pacote de software de código-fonte aberto, disponível gratuitamente, para segurança de e-mail. Ele oferece autenticação por meio do uso da assinatura digital; confidencialidade pelo uso da criptografia simétrica; compressão usando o algoritmo ZIP; compatibilidade de e-mail usando o esquema de codificação radix-64 e segmentação e remontagem para acomodar e-mails longos.
- O PGP incorpora ferramentas para desenvolver um modelo de confiança de chave pública e gerenciamento de certificados de chave pública.
- S/MIME é uma técnica padrão da Internet para segurança do e-mail, que incorpora a mesma funcionalidade do PGP.

Em praticamente todos os ambientes distribuídos, o e-mail é a aplicação de rede mais utilizada. Ela também é a única aplicação distribuída que é amplamente usada por todas as arquiteturas e plataformas de fornecedor. Os usuários esperam poder, e podem, enviar e-mails para outros que estejam conectados direta ou indiretamente à Internet, independente do sistema operacional ou do pacote de comunicações do host.

Com a confiança cada vez maior no e-mail para todo tipo de propósito imaginável, cresce uma demanda por serviços de autenticação e confidencialidade. Dois esquemas se destacam como técnicas que gozam de uso generalizado: Pretty Good Privacy (PGP) e S/MIME. Examinamos ambos neste capítulo.

15.1 PRETTY GOOD PRIVACY

O PGP é um fenômeno notável. Em grande parte esforço de uma única pessoa, Phil Zimmermann, o PGP oferece um serviço de confidencialidade e autenticação que pode ser usado para aplicações de e-mails e armazenamento de arquivos. Basicamente, Zimmermann fez o seguinte:

1. Selecionou os melhores algoritmos criptográficos disponíveis como elementos básicos
2. Integrou esses algoritmos em uma aplicação de uso geral, independente do sistema operacional e processador, e baseada em um pequeno conjunto de comandos fáceis de usar
3. Tornou o pacote e sua documentação, incluindo o código-fonte, livremente disponíveis por meio da Internet, BBSs e redes comerciais como AOL (America On Line)
4. Entrou em um acordo com uma empresa (Viacrypt, agora Network Associates) para oferecer uma versão totalmente compatível, de baixo custo, do PGP

O PGP experimentou uma explosão de crescimento e agora é bastante utilizado. Diversos motivos podem ser citados para esse crescimento:

1. Ele está disponível gratuitamente no mundo inteiro em versões para diversas plataformas, incluindo Windows, UNIX, Macintosh e muito mais. Além disso, a versão comercial satisfaz os usuários que desejam um produto que vem com suporte do fornecedor.
2. Ele é baseado em algoritmos que sobreviveram a uma ampla crítica pública e são considerados extremamente seguros. Especificamente, o pacote inclui RSA, DSS e Diffie-Hellman para criptografia de chave pública; CAST-128, IDEA e 3DES para criptografia simétrica; e SHA-1 para codificação de hash.
3. Ele possui uma grande gama de aplicabilidade, desde corporações que desejam selecionar e impor um esquema padronizado para criptografar arquivos e mensagens até indivíduos que desejam se comunicar com segurança com outros no mundo inteiro pela Internet e outras redes.
4. Ele não foi desenvolvido nem é controlado por qualquer organização do governo ou de padrões. Para aqueles que sempre desconfiam do 'sistema', isso torna o PGP atraente.
5. O PGP agora está em processo para tornar-se um padrão da Internet (RFC 3156). Apesar disso, PGP ainda tem fama de um esforço anti-sistema.

Começamos com uma visão geral do funcionamento do PGP. Em seguida, examinamos como as chaves criptográficas são criadas e armazenadas. Depois, tratamos da questão vital do gerenciamento de chaves públicas.

Notação

A maior parte da notação usada neste capítulo já foi usada antes, mas alguns termos são novos. Talvez seja melhor resumi-los no início. Os seguintes símbolos são usados:

A documentação do PGP normalmente usa o termo *chave secreta* para se referir a uma chave que faz par com uma chave pública em um esquema de criptografia de chave pública. Como já dissemos, essa prática pode gerar confusão com uma chave secreta usada para a criptografia simétrica. Logo, usaremos o termo *chave privada* em seu lugar.

K_s = chave de sessão usada no esquema de criptografia simétrica
PR_a = chave privada do usuário A, usada no esquema de criptografia de chave pública
PU_a = chave pública do usuário A, usada no esquema de criptografia de chave pública
EP = criptografia de chave pública
DP = decriptografia de chave pública
EC = criptografia simétrica
DC = decriptografia simétrica
H = função de hash
|| = concatenação
Z = compressão usando algoritmo ZIP
R64 = conversão no formato ASCII radix 64

Descrição operacional

A operação real do PGP, ao contrário do gerenciamento de chaves, consiste em cinco serviços: autenticação, confidencialidade, compressão, compatibilidade de e-mail e segmentação (Tabela 15.1). Examinamos cada um destes por sua vez.

Tabela 15.1 Resumo dos serviços do PGP

Função	Algoritmo utilizado	Descrição
Assinatura digital	DSS/SHA ou RSA/SHA	Um código de hash de uma mensagem é criado usando SHA-1. Este resumo de mensagem é criptografado usando DSS ou RSA, com a chave privada do emissor e incluído com a mensagem.
Criptografia de mensagem	CAST ou IDEA ou Triple DES ou RSA	Uma mensagem é criptografada usando CAST-128 ou IDEA ou com três chaves, com Diffie-Hellman utilização gerada pelo emissor. A chave de sessão é criptografada usando Diffie-Hellman ou RSA com a chave pública do destinatário e incluída com a mensagem.
Compressão	ZIP	Uma mensagem pode ser comprimida, para armazenamento ou transmissão, usando ZIP.
Compatibilidade de e-mail	Conversão radix 64	Para oferecer transparência a aplicações de e-mail, uma mensagem criptografada pode ser convertida em uma string ASCII usando-se a conversão radix 64.
Segmentação	—	Para acomodar limitações de tamanho máximo da mensagem, o PGP realiza segmentação e remontagem

Autenticação A Figura 15.1a ilustra o serviço de assinatura digital fornecido pelo PGP. Esse é o esquema de assinatura digital explicado no Capítulo 13 e ilustrado na Figura 11.5c. A seqüência é a seguinte:

1. O emissor cria uma mensagem.
2. SHA-1 é usado para gerar um código hash de 160 bits da mensagem.
3. O código de hash é criptografado com RSA usando a chave privada do emissor, e o resultado é anexado ao início da mensagem.
4. O receptor usa RSA com a chave pública do emissor para decriptografar e recuperar o código de hash.
5. O receptor gera um novo código de hash para a mensagem e o compara com o código de hash decriptografado. Se os dois forem iguais, a mensagem será aceita como autêntica.

A combinação de SHA-1 e RSA oferece um esquema de assinatura digital eficaz. Devido à força do RSA, o destinatário tem garantia de que somente o possuidor da chave privada correspondente pode gerar a assinatura. Devido à força do SHA-1, o destinatário tem garantia de que ninguém mais pode gerar uma nova mensagem que tenha o mesmo código de hash e, portanto, a mesma assinatura da mensagem original.

Como alternativa, as assinaturas podem ser geradas usando DSS/SHA-1.

Embora as assinaturas normalmente sejam anexadas à mensagem ou arquivo que elas assinam, isso nem sempre acontece: assinaturas separadas são aceitas. Uma assinatura separada pode ser armazenada e transmitida separadamente da mensagem que ela assina. Isso é útil em vários contextos. Um usuário pode querer manter um registro de assinatura separado de todas as mensagens enviadas ou recebidas. Uma assinatura separada de um programa executável pode detectar infecção posterior por vírus. Finalmente, assinaturas separadas podem ser usadas quando mais de uma parte tiver de assinar um documento, como um contrato jurídico. A assinatura de cada pessoa é independente e, portanto, aplicada apenas ao documento. Caso contrário, as assinaturas teriam de ser aninhadas, com o segundo assinante assinando o documento e a primeira assinatura, e assim por diante.

Confidencialidade Outro serviço básico fornecido pelo PGP é a confidencialidade, que é obtida a partir da criptografia das mensagens a serem transmitidas ou armazenadas localmente como arquivos. Nos dois casos, o algoritmo de criptografia simétrica CAST-128 pode ser usado. Como alternativa, IDEA ou 3DES podem ser usados. O modo de feedback cifrado (CFB) de 64 bits é utilizado.

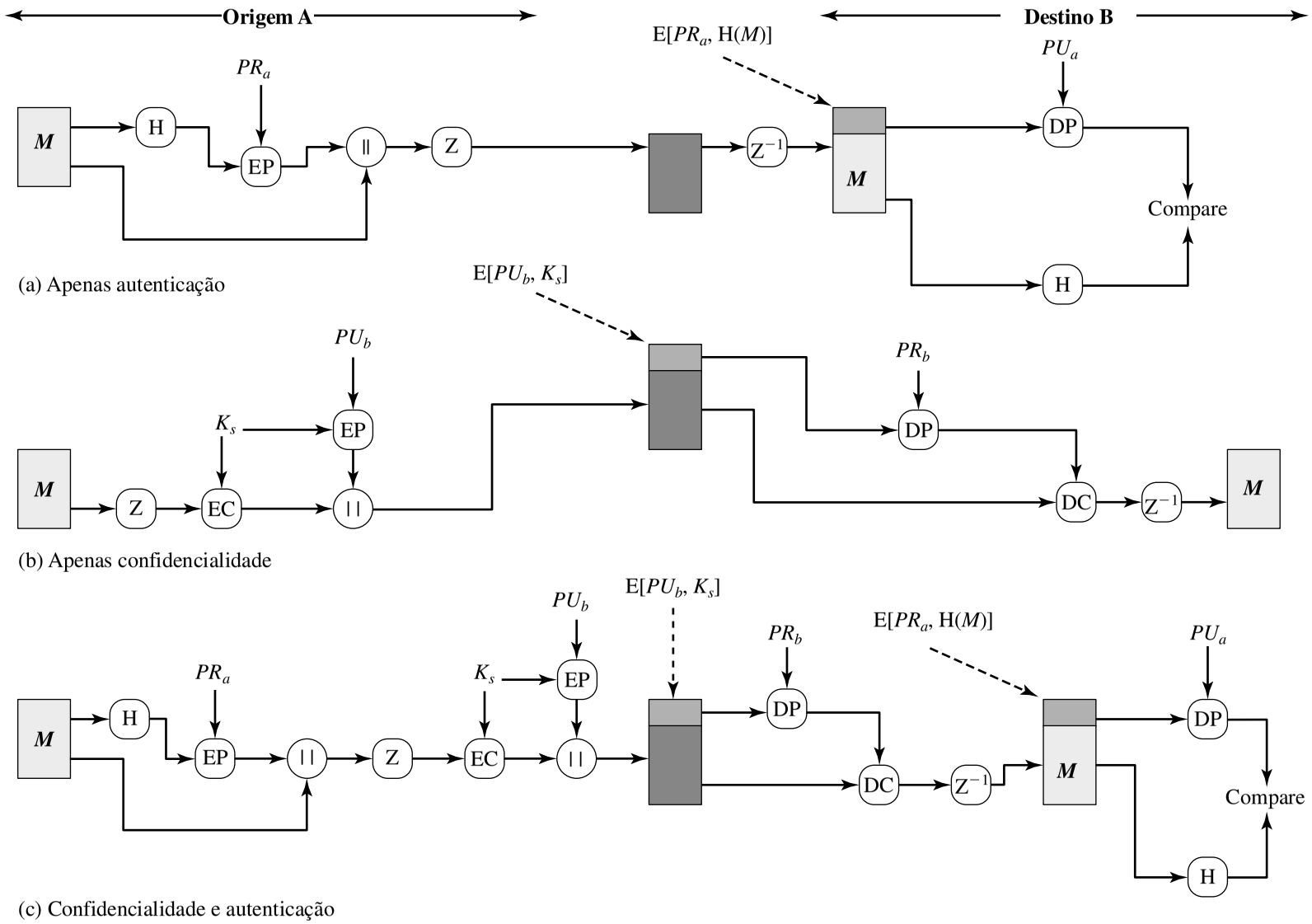


Figura 15.1 Funções criptográficas do PGP.

Como sempre, é preciso resolver o problema de distribuição de chaves. No PGP, cada chave simétrica é usada apenas uma vez. Ou seja, uma nova chave é gerada como um número aleatório de 128 bits para cada mensagem. Assim, embora isso seja conhecido na documentação como uma chave de sessão, na realidade é uma chave de uso único (*one-time key*). Por ser usada apenas uma vez, a chave de sessão está vinculada à mensagem e é transmitida com ela. Para proteger a chave, esta é criptografada com a chave pública do receptor. A Figura 15.1b ilustra a seqüência, que pode ser descrita desta maneira:

1. O emissor gera uma mensagem e um número aleatório de 128 bits a ser usado como chave de sessão apenas para esta mensagem.
2. A mensagem é criptografada, usando CAST-128 (ou IDEA ou 3DES) com a chave de sessão.
3. A chave de sessão é criptografada com RSA, usando a chave pública do destinatário, e é anexada ao início da mensagem.
4. O receptor usa RSA com sua chave privada para decriptografar e recuperar a chave de sessão.
5. A chave de sessão é usada para decriptografar a mensagem.

Como alternativa ao uso do RSA para a criptografia de chave, o PGP oferece uma opção conhecida como *Diffie-Hellman*. Conforme explicamos no Capítulo 10, Diffie-Hellman é um algoritmo para a troca de chaves. Na verdade, o PGP usa uma variante do Diffie-Hellman que oferece criptografia/decriptografia, conhecida como ElGamal (ver Problema 10.6).

Podemos fazer várias observações. Primeiro, para reduzir o tempo de criptografia, a combinação de criptografia simétrica e de chave pública é usada preferencialmente a simplesmente usar RSA ou ElGamal para criptografar a mensagem diretamente: CAST-128 e os outros algoritmos simétricos são substancialmente mais rápidos que RSA ou ElGamal. Segundo, o uso do algoritmo de chave pública soluciona o problema de distribuição de chave de sessão, pois somente o destinatário é capaz de recuperar a chave de sessão que está vinculada à mensagem. Observe que não precisamos de um protocolo de troca de chave de sessão do tipo explicado no Capítulo 10, pois não estamos iniciando uma sessão contínua. Em vez disso, cada mensagem é um evento independente único, com sua própria chave. Além disso, dada a natureza armazenar-e-encaminhar (*store-and-forward*) do e-mail, o uso do handshaking para garantir que os dois lados tenham a mesma chave de sessão não é prático. Finalmente, o uso de chaves simétricas de uso único fortalece o que já é uma técnica de criptografia simétrica forte. Apenas uma pequena quantidade de texto claro é criptografada com cada chave, e não existe relacionamento entre as chaves. Assim, até o ponto em que o algoritmo de chave pública é seguro, o esquema inteiro é seguro. Para esse fim, o PGP oferece ao usuário uma série de opções de tamanho de chave, de 768 a 3.072 bits (a chave DSS para assinaturas é limitada a 1.024 bits).

Confidencialidade e autenticação Conforme ilustra a Figura 15.1c, os dois serviços podem ser usados para a mesma mensagem. Primeiro, uma assinatura é gerada para a mensagem de texto claro e anexada ao início da mensagem. Depois, a mensagem em texto claro mais a assinatura é criptografada usando CAST-128 (ou IDEA ou 3DES), e a chave de sessão é criptografada usando RSA (ou ElGamal). Essa seqüência é preferível ao oposto: criptografar a mensagem e depois gerar uma assinatura para a mensagem criptografada. Geralmente, é mais conveniente armazenar uma assinatura com uma versão de texto claro de uma mensagem. Além disso, para fins de verificação de terceiros, se a assinatura for realizada primeiro, um terceiro não precisará se preocupar com a chave simétrica ao verificar a assinatura.

Resumindo, quando os dois serviços são usados, o emissor primeiramente assina a mensagem com sua própria chave privada, depois criptografa a mensagem com uma chave de sessão, e depois criptografa a chave de sessão com a chave pública do destinatário.

Compressão Como padrão, o PGP comprime a mensagem depois de aplicar a assinatura, mas antes da criptografia. Isso tem o benefício de economizar espaço para transmissão de e-mail e para armazenamento de arquivos.

O posicionamento do algoritmo de compressão, indicado por Z para compressão e Z^{-1} para descompressão na Figura 15.1, é muito importante:

1. A assinatura é gerada antes da compressão por dois motivos:
   a. É preferível assinar uma mensagem não comprimida de modo que seja possível armazenar apenas a mensagem descomprimida juntamente com a assinatura para verificação futura. Se alguém assinasse um documento comprimido, então seria preciso armazenar uma versão comprimida da mensagem para verificação posterior ou recomprimir a mensagem quando a verificação fosse necessária.
   b. Mesmo que alguém quisesse gerar dinamicamente uma mensagem recomprimida para verificação, o algoritmo de compressão do PGP apresentaria uma dificuldade. O algoritmo não é determinístico; várias implementações do algoritmo obtém diferentes velocidades de execução em relação à razão de compressão e, como resultado, produzem diferentes formas comprimidas. Porém, esses diferentes algoritmos de compressão são interoperáveis, pois qualquer versão do algoritmo pode descomprimir corretamente a saída de qualquer outra versão. A aplicação da função de hash e assinatura após a compressão restringiria todas as implementações do PGP à mesma versão do algoritmo de compressão.

2. A criptografia de mensagem é aplicada após a compressão para fortalecer a segurança criptográfica. Como a mensagem comprimida tem menos redundância do que o texto claro original, a criptoanálise é mais difícil.

O algoritmo de compressão utilizado é ZIP, que é descrito no Apêndice 15A.

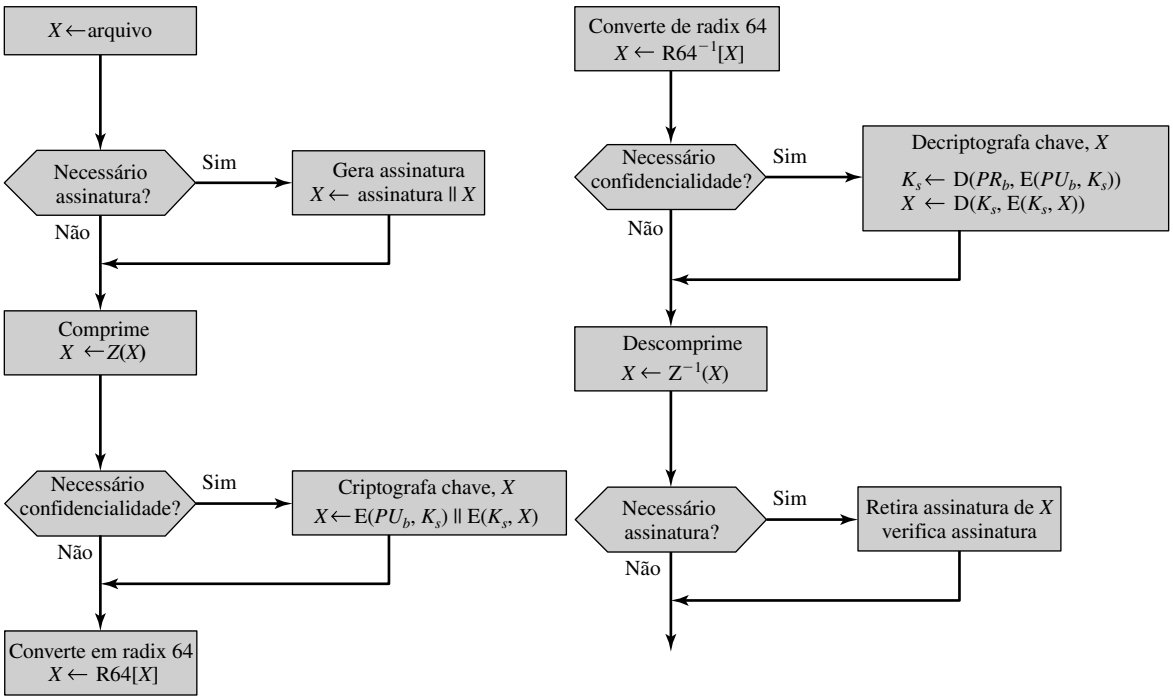
Compatibilidade de e-mail Quando o PGP é utilizado, pelo menos parte do bloco a ser transmitido é criptografado. Se apenas o serviço de assinatura for utilizado, então o resumo da mensagem é criptografado (com a chave privada do emissor). Se o serviço de confidencialidade for usado, a mensagem mais assinatura (se estiver presente) são criptografadas (com uma chave simétrica de uso único). Assim, parte ou todo o bloco resultante consiste em um fluxo de octetos arbitrários de 8 bits. Porém, muitos sistemas de e-mail só permitem o uso de blocos de texto ASCII. Para lidar com essa restrição, o PGP oferece o serviço de conversão do fluxo binário bruto de 8 bits em um fluxo de caracteres ASCII imprimíveis.

O esquema usado para essa finalidade é a conversão radix 64. Cada grupo de três octetos de dados binários é mapeado para quatro caracteres ASCII. Esse formato também anexa um CRC para detectar erros de transmissão. Veja uma descrição no Apêndice 15B.

O uso de radix 64 expande uma mensagem em torno de 33%. Felizmente, as partes de chave de sessão e assinatura da mensagem são relativamente compactas, e a mensagem de texto claro foi comprimida. Na verdade, a compressão deveria ser mais do que suficiente para compensar a expansão radix 64. Por exemplo, [HELD96] informa uma razão de compressão média de cerca de 2,0 usando ZIP. Se ignorarmos os componentes relativamente pequenos de assinatura e chave, o efeito geral típico da compressão e expansão de um arquivo de tamanho X seria $1{,}33 \times 0{,}5 \times X = 0{,}665 \times X$. Assim, ainda existe uma compressão geral de cerca de um terço.

Um aspecto digno de nota do algoritmo radix 64 é que ele converte cegamente o fluxo de entrada no formato radix 64, independente do conteúdo, mesmo que a entrada seja texto ASCII. Assim, se uma mensagem estiver assinada, mas não criptografada, e a conversão for aplicada ao bloco inteiro, a saída será ilegível ao observador casual, o que oferece um certo nível de confidencialidade. Como opção, o PGP pode ser configurado para converter no formato radix 64 somente a parte de assinatura das mensagens de texto claro assinadas. Isso permite que o destinatário humano leia a mensagem sem usar o PGP. Para verificar a assinatura, o PGP ainda teria de ser utilizado.

A Figura 15.2 mostra o relacionamento entre os quatro serviços explicados até aqui. Na transmissão, se exigido, será gerada uma assinatura usando um código de hash do texto claro não comprimido. Depois, o texto claro mais a assinatura, se estiver presente, serão comprimidos. Em seguida, se a confidencialidade for exigida, o bloco (texto claro comprimido ou assinatura mais texto claro comprimidos) será criptografado e anexado ao início com a chave de criptografia simétrica criptografada com a chave pública. Finalmente, o bloco inteiro é convertido no formato radix 64.



(a) Diagrama de transmissão genérico (de A) (b) Diagrama de recepção genérico (para B)

Figura 15.2 Transmissão e recepção de mensagens PGP.

Na recepção, o bloco que chega é primeiro convertido do formato radix 64 em binário novamente. Depois, se a mensagem estiver criptografada, o destinatário recupera a chave de sessão e decriptografa a mensagem. O bloco resultante é, então, descomprimido. Se a mensagem estiver assinada, o destinatário recupera o código de hash transmitido e o compara com seu próprio cálculo do código de hash.

Segmentação e remontagem Os recursos de e-mail normalmente são restritos a um tamanho máximo de mensagem. Por exemplo, muitos dos recursos acessíveis pela Internet impõem um tamanho máximo de 50.000 octetos. Qualquer mensagem maior que isso precisa ser dividida em segmentos menores, cada um remetido separadamente.

Para lidar com essa restrição, o PGP subdivide automaticamente uma mensagem muito grande em segmentos pequenos o suficiente para envio por e-mail. A segmentação é feita depois de todo o outro processamento, incluindo a conversão radix 64. Assim, o componente de chave de sessão e o componente de assinatura aparecem apenas uma vez, no início do primeiro segmento. Na ponta receptora, o PGP precisa remover todos os cabeçalhos de e-mail e remontar o bloco original inteiro antes de realizar as etapas ilustradas na Figura 15.2b.

Chaves criptográficas e lista de chaves

O PGP faz uso de quatro tipos de chaves: chaves simétricas de sessão de uso único, chaves públicas, chaves privadas e chaves simétricas baseadas em frase secreta (explicadas posteriormente). Três requisitos distintos podem ser identificados com relação a essas chaves:

1. Um meio de gerar chaves de sessão imprevisíveis é necessário.
2. Queremos permitir que um usuário tenha vários pares de chave pública/chave privada. Um motivo é que o usuário pode querer mudar seu par de vez em quando. Quando isso acontece, quaisquer mensagens que estejam sendo processadas serão construídas com uma chave obsoleta. Além disso, os destinatários só terão conhecimento da antiga chave pública até que uma atualização os alcance. Além da necessidade de mudar as chaves com o tempo, um usuário pode querer ter vários pares de chaves em determinado momento para interagir com diferentes grupos de correspondentes ou simplesmente para melhorar a segurança, limitando a quantidade de material criptografado com uma única chave qualquer. O resultado de tudo isso é que não existe uma correspondência um a um entre usuários e suas chaves públicas. Assim, é preciso haver algum meio para identificar chaves específicas.
3. Cada entidade do PGP precisa manter um arquivo com seus próprios pares de chaves pública/privada, além de um arquivo com as chaves públicas dos correspondentes.

Examinamos cada um desses requisitos por vez.

Geração da chave de sessão Cada chave de sessão está associada a uma única mensagem, e é usada apenas para fins de criptografia e decriptografia dessa mensagem. Lembre-se de que a criptografia/decriptografia de mensagens é feita com um algoritmo de criptografia simétrico. CAST-128 e IDEA usam chaves de 128 bits; 3DES utiliza uma chave de 168 bits. Para a explicação a seguir, consideramos CAST-128.

Números aleatórios de 128 bits são gerados pelo próprio CAST-128. A entrada do gerador de números aleatórios consiste em uma chave de 128 bits e dois blocos de 64 bits que são tratados como texto claro a ser criptografado. Usando o modo de feedback de cifra, o criptografador CAST-128 produz dois blocos de texto cifrado de 64 bits, que são concatenados para formar a chave de sessão de 128 bits. O algoritmo usado é baseado naquele especificado no ANSI X12.17.

A entrada de 'texto claro' para o gerador de números aleatórios, consistindo em dois blocos de 64 bits, é derivada de um fluxo de números aleatórios de 128 bits. Esses números são baseados na entrada de toque de tecla do usuário. Tanto o tempo dos toques de tecla quanto as teclas pressionadas são usadas para gerar o fluxo aleatório. Assim, se o usuário pressionar teclas ao acaso em seu ritmo normal, uma entrada razoavelmente 'aleatória' será gerada. Essa entrada aleatória também é combinada com a saída da chave de sessão anterior do CAST-128 para formar a entrada de chave para o gerador. O resultado, dado o embaralhar eficaz do CAS-128, é produzir uma seqüência de chaves de sessão efetivamente imprevisível.

O Apêndice 15C explica as técnicas de geração de números aleatórios do PGP com mais detalhes.

Identificadores de chave Conforme já explicamos, uma mensagem criptografada é acompanhada de uma forma criptografada da chave de sessão, que foi usada para a criptografia da mensagem. A própria chave de sessão é criptografada com a chave pública do destinatário. Logo, somente o destinatário poderá recuperar a chave de sessão e, portanto, recuperar a mensagem. Se cada usuário empregar um único par de chaves pública/privada, então o destinatário saberá automaticamente qual chave usar para decriptografar a chave de sessão: a chave privada exclusiva do destinatário. Porém, colocamos um requisito de que qualquer usuário poderá ter vários pares de chave pública/privada.

Como, então, o destinatário saberá qual de suas chaves públicas foi usada para criptografar a chave de sessão? Uma solução simples seria transmitir a chave pública com a mensagem. O destinatário poderia, então, verificar se essa é, na realidade, uma de suas chaves públicas, e prosseguir. Esse esquema funcionaria, mas desperdiçaria espaço des-

necessariamente. Uma chave pública RSA pode ter centenas de dígitos decimais de extensão. Outra solução seria associar um identificador a cada chave pública, que fosse único para pelo menos um dado usuário. Isto é, a combinação de ID de usuário e ID de chave seria suficiente para identificar uma chave exclusivamente. Então, somente a ID de chave muito mais curta teria de ser transmitida. Essa solução, no entanto, gera um problema de gerenciamento e de overhead: as IDs de chave precisam ser atribuídas e armazenadas de modo que o emissor e o destinatário possam fazer a correspondência entre ID de chave e chave pública. Isso parece desnecessariamente oneroso.

A solução adotada pelo PGP é atribuir uma ID de chave a cada chave pública que, muito provavelmente, é exclusiva dentro de uma ID de usuário.[1] A ID de chave associada a cada chave pública consiste em seus 64 bits menos significativos. Ou seja, a ID da chave pública PU_a é (PU_a mod 2^{64}). Esse é um tamanho suficiente de modo que a probabilidade de IDs de chave duplicadas seja muito pequena.

Uma ID de chave também é exigida para a assinatura digital do PGP. Como um emissor pode usar uma dentre diversas chaves privadas para criptografar o resumo da mensagem, o destinatário precisa saber qual chave pública deverá ser usada. Por conseguinte, o componente de assinatura digital de uma mensagem inclui a ID de chave de 64 bits da chave pública exigida. Quando a mensagem é recebida, o destinatário verifica se a ID de chave é para uma chave pública que ele conhece para esse emissor e depois prossegue para verificar a assinatura.

Agora que o conceito de ID de chave foi apresentado, podemos examinar com mais detalhes o formato de uma mensagem transmitida, que aparece na Figura 15.3. Uma mensagem consiste em três componentes: o componente da mensagem, uma assinatura (opcional) e um componente de chave de sessão (opcional).

O **componente de mensagem** inclui os dados reais a serem armazenados ou transmitidos, além de um nome de arquivo e carimbo de tempo que especifica a hora da criação.

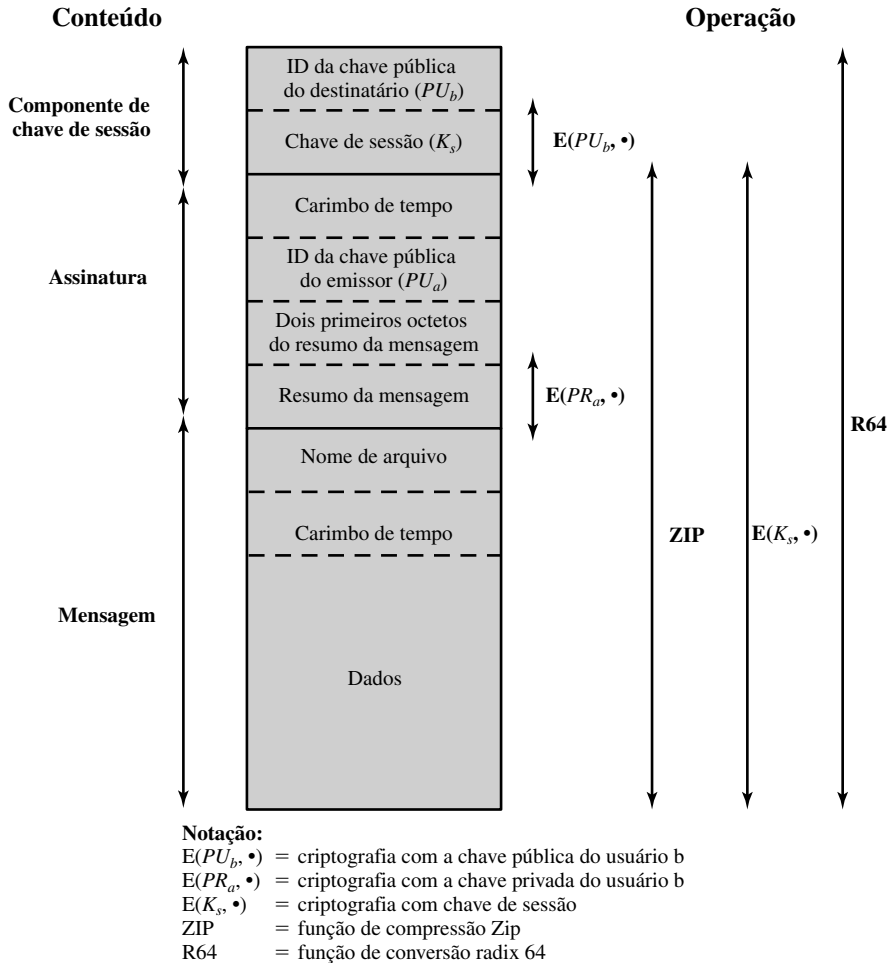


Figura 15.3 Formato geral da mensagem PGP (de A para B).

1. Vimos essa introdução dos conceitos probabilísticos antes, na Seção 8.3, para determinar se um número é primo. Normalmente acontece, no projeto de algoritmos, que o uso de técnicas probabilísticas resulta em uma solução menos demorada ou menos complexa, ou ambos.

O **componente de assinatura** inclui o seguinte:

- **Carimbo de tempo:** A hora em que a assinatura foi criada.
- **Resumo criptográfico:** O resumo SHA-1 de 160 bits, criptografado com a chave de assinatura privada do emissor. O resumo é calculado sobre o carimbo de tempo da assinatura concatenada com a parte de dados do componente da mensagem. A inclusão do carimbo de tempo da assinatura no resumo impede ataques do tipo repetição. A exclusão das partes de nome de arquivo e carimbo de tempo do componente da mensagem garante que as assinaturas separadas sejam exatamente as mesmas que as assinaturas anexadas ao início da mensagem. As assinaturas separadas são calculadas em um arquivo separado que não possui nenhum dos campos de cabeçalho do componente de mensagem.
- **Dois primeiros octetos do resumo da mensagem:** Para permitir que o destinatário determine se a chave pública correta foi usada para decriptografar o resumo da mensagem para autenticação, comparando essa cópia em texto claro dos dois primeiros octetos com os dois primeiros octetos do resumo decriptografado. Esses octetos também servem como uma seqüência de verificação de frame de 16 bits para a mensagem.
- **ID da chave pública do emissor:** Identifica a chave pública que deve ser usada para decriptografar o resumo da mensagem e, portanto, identifica a chave privada usada para decriptografar o resumo da mensagem.

O componente de mensagem e o componente de assinatura opcional podem ser comprimidos usando ZIP, e podem ser criptografados usando uma chave de sessão.

O **componente de chave de sessão** inclui a chave de sessão e o identificador da chave pública do destinatário usado pelo emissor para criptografar a chave de sessão.

O bloco inteiro normalmente é codificado com radix 64.

Lista de chaves Vimos como as IDs de chave são críticas para a operação do PGP, e que duas IDs de chave são incluídas em qualquer mensagem PGP que ofereça confidencialidade e autenticação. Essas chaves precisam ser armazenadas e organizadas de maneira sistemática para uso eficaz e efetivo por todas as partes. O esquema usado no PGP é oferecer um par de estruturas de dados em cada nó, uma para armazenar os pares de chaves pública/privada pertencentes a esse nó e outra para armazenar as chaves públicas de outros usuários conhecidos nesse nó. Essas estruturas de dados são conhecidas, respectivamente, como lista de chaves privadas (*private-key ring* – "chaveiro" ou "molho" de chaves privadas) e lista de chaves públicas (*public-key ring* – "chaveiro" ou "molho" de chaves públicas).

A Figura 15.4 mostra a estrutura geral de **uma lista de chaves privadas**. Podemos ver a lista como uma tabela, em que cada linha representa um dos pares de chave pública/privada pertencentes a esse usuário. Cada linha contém as seguintes entradas:

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	PU_i mod 2^{64}	PU_i	$E(H(P_i), PR_i)$	User i
• • •	• • •	• • •	• • •	• • •

Public-Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	PU_i mod 2^{64}	PU_i	$trust_flag_i$	User i	$trust_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = Campo utilizado para indexação da tabela

Figura 15.4 Estrutura geral de uma lista de chaves privada e de uma pública.

- **Carimbo de tempo (Timestamp):** A data/hora em que esse par de chaves foi gerado.
- **ID de chave (Key ID):** Os 64 bits menos significativos da chave pública para essa entrada.
- **Chave pública (Public Key):** A chave pública do par.
- **Chave privada (Private Key):** A chave privada do par; este campo é criptografado.
- **ID de usuário (User ID):** Normalmente, esse será o endereço de e-mail do usuário (por exemplo, stallings@acm.org). Porém, o usuário pode escolher associar um nome diferente a cada par (por exemplo, Stallings, WStallings, WilliamStallings etc.) ou reutilizar a mesma ID de usuário mais de uma vez.

A lista de chaves privadas pode ser indexado por ID de usuário ou por ID de chave; mais adiante veremos a necessidade dos dois meios de indexação.

Embora a lista de chaves privadas fique armazenada normalmente apenas na máquina do usuário que criou e possui os pares de chaves, e esteja acessível apenas a este usuário, faz sentido tornar o valor da chave privada o mais seguro possível. Por conseguinte, a própria chave privada não é armazenada na lista de chaves. Em vez disso, essa chave é criptografada usando-se CAST-128 (ou IDEA ou 3DES). O procedimento é o seguinte:

1. O usuário seleciona uma frase secreta a ser usada para criptografar chaves privadas.
2. Quando o sistema gera um novo par de chaves pública/privada usando RSA, ele pede uma frase secreta do usuário. Usando SHA-1, um código de hash de 160 bits é gerado a partir da frase secreta, e esta é descartada.
3. O sistema criptografa a chave privada usando CAST-128 com os 128 bits do código de hash como chave. O código de hash é então descartado, e a chave privada criptografada (encripted privet key) é armazenada na lista de chaves privadas.

Posteriormente, quando um usuário acessa a lista de chaves privadas para recuperar uma dessas chaves, ele precisa fornecer a chave secreta. O PGP recuperará a chave privada criptografada, gerará o código de hash da frase secreta e decriptografará a chave privada usando CAST-128 com o código de hash.

Esse é um esquema bastante compacto e eficaz. Assim como em qualquer sistema baseado em senhas, a segurança desse sistema depende da segurança da senha. Para evitar a tentação de escrevê-la, o usuário deverá usar uma frase secreta que não seja facilmente adivinhada, mas facilmente lembrada.

A Figura 15.4 também mostra a estrutura geral de uma lista **de chaves públicas**. Essa estrutura de dados é usada para armazenar chaves públicas de outros usuários conhecidas por esse usuário. Por um momento, vamos ignorar alguns campos mostrados na tabela e descrever os seguintes campos:

- **Carimbo de tempo:** A data/hora em que essa entrada foi gerada.
- **ID de chave:** Os 64 bits menos significativos da chave pública para essa entrada.
- **Chave pública:** A chave pública para essa entrada.
- **ID do usuário:** Identifica o proprietário dessa chave. Várias IDs de usuário podem ser associadas a uma única chave pública.

A lista de chaves públicas pode ser indexada por ID de usuário ou ID de chave; veremos a necessidade dos dois meios de indexação mais adiante.

Agora estamos em posição de mostrar como essas listas de chaves são usadas na transmissão e recepção de mensagens. Para simplificar, ignoramos a compressão e a conversão radix 64 na explicação a seguir. Primeiro, considere a transmissão de mensagem (Figura 15.5) e suponha que a mensagem está assinada e criptografada. A entidade de PGP de envio realiza as seguintes etapas:

1. Assinar a mensagem
   a. O PGP recupera a chave privada do emissor a partir da lista de chaves privadas usando sua_idusuário como índice. Se sua_idusuário não foi fornecida no comando, a primeira chave privada na lista é recuperada.
   b. O PGP pede a frase secreta do usuário para recuperar a chave privada não criptografada.
   c. O componente de assinatura da mensagem é criado.
2. Criptografar a mensagem
   a. O PGP gera uma chave de sessão e criptografa a mensagem.
   b. O PGP recupera a chave pública do destinatário a partir do anel de chave pública, usando idusuário_dela como índice.
   c. O componente de chave de sessão da mensagem é construído.

A entidade receptora do PGP realiza as seguintes etapas (Figura 15.6):

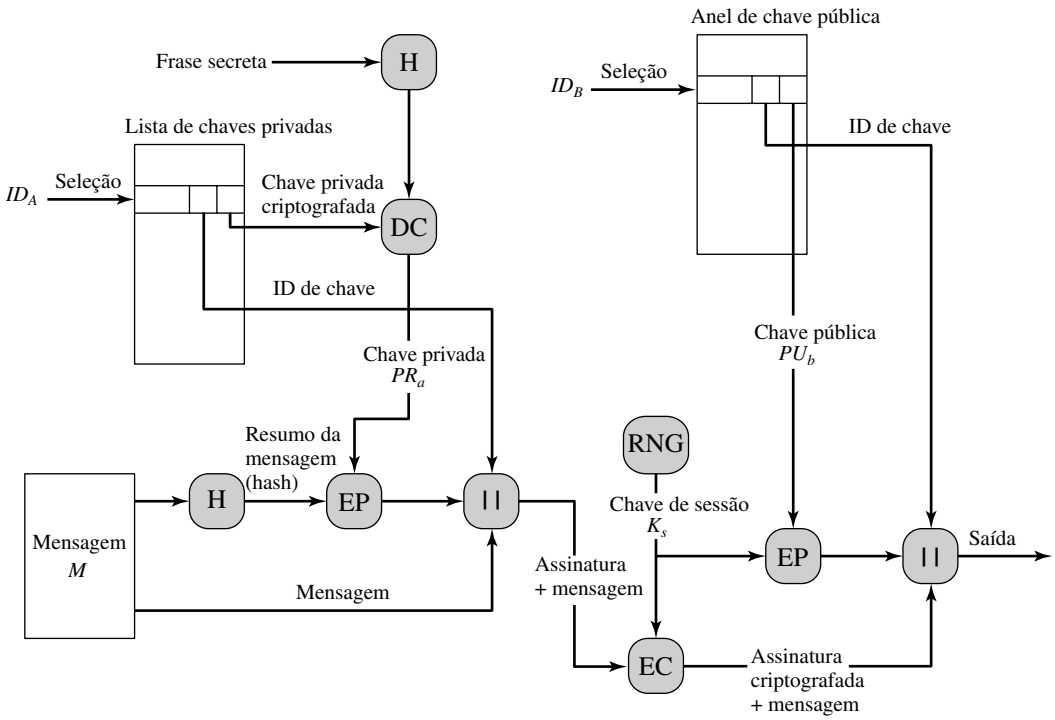


Figura 15.5 Geração de mensagem PGP (do Usuário A para Usuário B; sem compressão ou conversão radix 64).

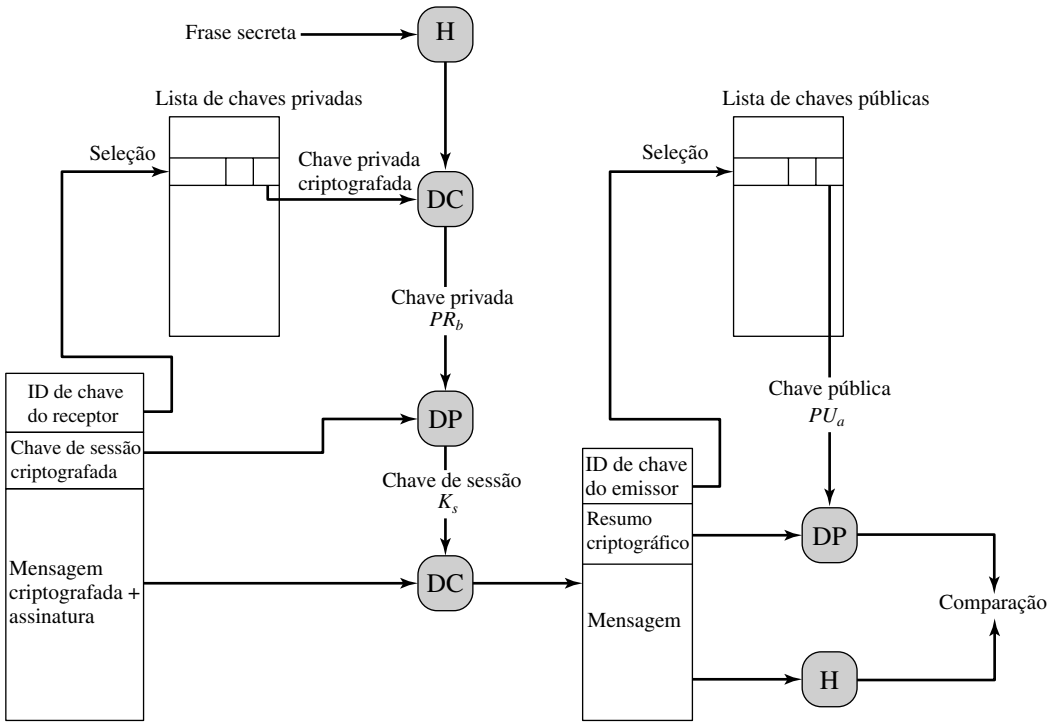


Figura 15.6 Recepção de mensagem do PGP (do Usuário A para o Usuário B; sem compressão ou conversão radix 64).

1. Decriptografar a mensagem
   a. O PGP recupera a chave privada do receptor a partir da lista de chaves privadas usando o campo de ID de chave no componente de chave de sessão da mensagem como um índice.
   b. O PGP pede a frase secreta do usuário para recuperar a chave privada não criptografada.
   c. O PGP então recupera a chave de sessão e decriptografa a mensagem.

2. Autenticar a mensagem
   a. O PGP recupera a chave pública do emissor a partir da lista de chaves públicas usando o campo de ID de chave no componente de chave de assinatura da mensagem como índice.
   b. O PGP recupera o hash da mensagem transmitida.
   c. O PGP calcula o hash da mensagem para a mensagem recebida e a compara ao hash da mensagem transmitida para autenticar.

Gerenciamento de chaves públicas

Como podemos constatar pela explicação até este ponto, o PGP contém um conjunto de funções e formatos inteligentes, eficientes e ajustados para oferecer um serviço eficaz de confidencialidade e autenticação. Para completar o sistema, uma última área precisa ser resolvida, a de gerenciamento de chaves públicas. A documentação do PGP capta a importância dessa área:

> A proteção de chaves públicas contra violação é o problema isolado mais difícil nas aplicações práticas de chave pública. Esse é o "calcanhar-de-aquiles" da criptografia de chave pública, e muita complexidade no software está ligada à solução desse problema.

O PGP oferece uma estrutura para solucionar esse problema, com várias opções sugeridas que podem ser usadas. Como o PGP tem como finalidade ser usado em vários ambientes formais e informais, nenhum esquema rígido de gerenciamento de chaves públicas está estabelecido, como veremos em nossa explicação sobre S/MIME, mais adiante neste capítulo.

Técnicas para o gerenciamento de chaves públicas A essência do problema é esta: o usuário A precisa montar uma lista de chaves públicas contendo as chaves públicas de outros usuários para interoperar com eles usando o PGP. Suponha que a lista de chaves de A contenha uma chave pública atribuída a B, mas que a chave, na verdade, pertença a C. Isso pode acontecer se, por exemplo, A obtivesse a chave de um sistema de BBS usado por B para postar a chave pública, mas que foi comprometido por C. O resultado é que agora existem duas ameaças. Primeiramente, C pode enviar mensagens para A e falsificar a assinatura de B, de modo que A aceitará a mensagem como se fosse proveniente de B. Segundo, qualquer mensagem criptografada de A para B poderá ser lida por C.

Diversas técnicas são possíveis para minimizar o risco de que a lista de chaves públicas de um usuário contenha chaves públicas falsas. Suponha que A queira obter uma chave pública confiável para B. A seguir estão algumas técnicas que poderiam ser usadas:

1. Consiga fisicamente a chave a partir de B. B poderia armazenar sua chave pública (PU_b) em um disquete e entregá-la a A. A poderia, então, carregar a chave em seu sistema a partir do disquete. Esse é um método muito seguro, mas possui limitações práticas óbvias.
2. Verifique uma chave por telefone. Se A puder reconhecer B no telefone, A poderia chamar B e pedir para que dite a chave, em formato radix 64, pelo telefone. Como uma alternativa mais prática, B pode transmitir sua chave em uma mensagem de e-mail para A. A pode fazer com que o PGP gere um resumo SHA-1 de 160 bits da chave e a apresente em formato hexadecimal; isso é conhecido como 'impressão digital' da chave. A poderia então chamar B e pedir que ele ditasse a impressão digital pelo telefone. Se as duas impressões digitais fossem iguais, a chave estaria verificada.
3. Obtenha a chave pública de B a partir de um D mutuamente confiável. Para essa finalidade, o que vai apresentar B, D, cria um certificado assinado. O certificado inclui a chave pública de B, a hora da criação da chave, e um período de validade para a chave. D gera um resumo SHA-1 desse certificado, criptografa-o com sua chave privada e anexa a assinatura ao certificado. Como somente D poderia ter criado a assinatura, ninguém mais pode criar uma chave pública falsa e fingir que ela foi assinada por D. O certificado assinado pode ser enviado diretamente a A por B ou D, ou pode ser postado em um BBS.
4. Obtenha a chave pública de B a partir de uma autoridade certificadora confiável. Novamente, um certificado de chave pública é criado e assinado pela autoridade. A pode então acessar a autoridade, fornecendo um nome de usuário e recebendo um certificado assinado.

Para os casos 3 e 4, A já precisaria ter uma cópia da chave pública do certificador e confiar que essa chave era válida. Por fim, fica a critério de A atribuir um nível de confiança a qualquer um que deve atuar como certificador.

O uso de confiança Embora o PGP não inclua qualquer especificação para estabelecer autoridades certificadoras ou para estabelecer confiança, ela oferece um meio conveniente para usar a confiança, associando confiança a chaves públicas e explorando a informação de confiança.

A estrutura básica é a seguinte. Cada entrada na lista de chaves públicas é um certificado de chave pública, conforme descrito na subseção anterior. Associado a cada uma dessas entradas existe um **campo de legitimidade de chave (*key legitimacy field*)**, que indica a extensão segundo a qual o PGP confiará que essa é uma chave pública válida para esse usuário; quanto mais alto o nível de confiança, mais forte o vínculo dessa ID de usuário com essa chave. Esse campo é calculado pelo PGP. Também associado à entrada estão zero ou mais assinaturas que o proprietário do anel de chave coletou, que assinam esse certificado. Por sua vez, cada assinatura possui associada a ela um **campo de confiança de assinatura (*signature trust field*)**, que indica o grau segundo o qual esse usuário do PGP confia no assinante para certificar as chaves públicas. O campo de legitimidade de chave é obtido a partir do conjunto de campos de confiança de assinatura na entrada. Finalmente, cada entrada define uma chave pública associada a um proprietário em particular, e um **campo de confiança no proprietário (*owner trust field*)** é incluído para indicar o grau em que se confia que essa chave pública assina outros certificados de chave pública; esse nível de confiança é atribuído pelo usuário. Podemos pensar em campos de confiança na assinatura como cópias em cache do campo de confiança do proprietário a partir de outra entrada.

Os três campos mencionados no parágrafo anterior estão contidos cada um em uma estrutura conhecida como byte de flag de confiança (*trust flag byte*). O conteúdo desse flag de confiança para cada um desses três usos aparece na Tabela 15.2. Suponha que estejamos lidando com a lista de chaves públicas do usuário A. Podemos descrever a operação do processamento de confiança da seguinte forma:

1. Quando A insere uma nova chave pública na lista de chaves públicas, o PGP precisa atribuir um valor ao flag de confiança que está associado ao proprietário dessa chave pública. Se o proprietário for A, e portanto essa chave pública também aparecer na lista de chaves privadas, então um valor de *confiança máxima* será automaticamente atribuído ao campo de confiança. Caso contrário, o PGP solicita a A sua avaliação da confiança a ser atribuída ao proprietário dessa chave, e A precisa definir o nível desejado. O usuário pode especificar que esse proprietário é desconhecido, não confiável, pouco confiável ou completamente confiável.
2. Quando a nova chave pública é inserida, uma ou mais assinaturas podem ser anexadas a ela. Mais assinaturas podem ser acrescentadas mais tarde. Quando uma assinatura é inserida na entrada, o PGP pesquisa a lista de chaves públicas para ver se o autor dessa assinatura está entre os proprietários de chave pública conhecidos. Se estiver, o valor de OWNERTRUST para esse proprietário é atribuído ao campo SIGTRUST para essa assinatura. Se não, um valor de *usuário desconhecido* é atribuído.
3. O valor do campo de legitimidade de chave é calculado com base nos campos de confiança de assinatura presentes nessa entrada. Se pelo menos uma assinatura tiver um valor de confiança de assinatura *máximo (ultimate)*, então o valor de legitimidade de chave é definido como completo. Caso contrário, o PGP calcula uma soma ponderada dos valores de confiança. Um peso de $1/X$ é dado a assinaturas que são sempre confiáveis e $1/Y$ a assinaturas que normalmente são confiáveis, onde X e Y são parâmetros configuráveis pelo usuário. Quando o total dos pesos dos certificadores de uma combinação de chave/ID do usuário alcançar 1, o vínculo será considerado confiável, e o valor de legitimidade da chave será definido como completo. Assim, na ausência de confiança máxima, é preciso haver pelo menos X assinaturas que sejam sempre confiáveis ou Y assinaturas que normalmente sejam confiáveis, ou alguma combinação.

Periodicamente, o PGP processa a lista de chaves públicas para obter consistência. Basicamente, esse é um processo de cima para baixo. Para cada campo OWNERTRUST, o PGP varre a lista em busca de todas as assinaturas feitas pelo proprietário e atualiza o campo SIGTRUST como sendo igual ao campo OWNERTRUST. Esse processo começa com chaves para as quais existe confiança máxima. Depois, todos os campos KEYLEGIT são calculados com base nas assinaturas anexadas.

A Figura 15.7 contém um exemplo do modo como a confiança na assinatura e a legitimidade da chave são relacionados.[2] A figura mostra a estrutura de uma lista de chaves públicas. O usuário adquiriu uma série de chaves públicas, algumas diretamente de seus proprietários e algumas de um terceiro, como um servidor de chaves.

O nó rotulado "Você" refere-se à entrada na lista de chaves públicas correspondente a esse usuário. Essa chave é legítima e o valor de OWNERTRUST é de confiança máxima. Cada outro nó na lista de chaves tem o valor de OWNERTRUST indefinido, a menos que algum outro valor seja atribuído pelo usuário. Neste exemplo, esse usuário especificou que ele sempre confia nos seguintes usuários para assinar outras chaves: D, E, F, L. Esse usuário confia parcialmente nos usuários A e B para assinar outras chaves.

Assim, o sombreamento, ou a falta de sombreamento dos nós da Figura 15.7, indica o nível de confiança atribuído por esse usuário. A estrutura em árvore indica quais chaves foram assinadas por quais outros usuários. Se uma chave for assinada por um usuário cuja chave também está na lista de chaves, a seta ligará a chave assinada ao signatário. Se a chave for assinada por um usuário cuja chave não está presente nesse anel de chave, a seta ligará a chave assinada a um ponto de interrogação, indicando que o signatário é desconhecido por esse usuário.

Vários pontos estão ilustrados nessa Figura 15.7.

2. Figura fornecida ao autor por Phil Zimmermann.

Tabela 15.2 Conteúdo do byte de flag de confiança

(a) Confiança atribuída ao proprietário da chave pública (aparece após o pacote de chave; definido pelo usuário)	**(b) Confiança atribuída ao par chave pública/ID de usuário (aparece após o pacote de ID do usuário; calculada pelo PGP)**	**(c) Confiança atribuída à assinatura (aparece após o pacote de assinatura; cópia em cache de OWNERTRUST para esse assinante)**
Campo OWNERTRUST – confiança indefinida – usuário desconhecido – normalmente não confiável para assinar outras chaves – normalmente confiável para assinar outras chaves – sempre confiável para assinar outras chaves – essa chave está presente na lista de chaves secretas (confiança máxima) Bit BUCKSTOP – marcado se essa chave aparecer na lista de chaves secretas	Campo KEYLEGIT – confiança desconhecida ou indefinida – proprietário da chave não confiável – confiança pequena no proprietário da chave – confiança completa no proprietário da chave Bit WARNONLY – marcado se o usuário só deseja ser avisado quando uma chave não totalmente validada for usada para criptografia	Campo SIGTRUST – confiança indefinida – usuário desconhecido – normalmente não confiável para assinar outras chaves – normalmente confiável para assinar outras chaves – sempre confiável para assinar outras chaves – essa chave está presente na lista de chaves secretas (confiança máxima) CONTIG – marcado se a assinatura levar um caminho de certificação confiável de volta ao proprietário da lista de chaves mais confiável.

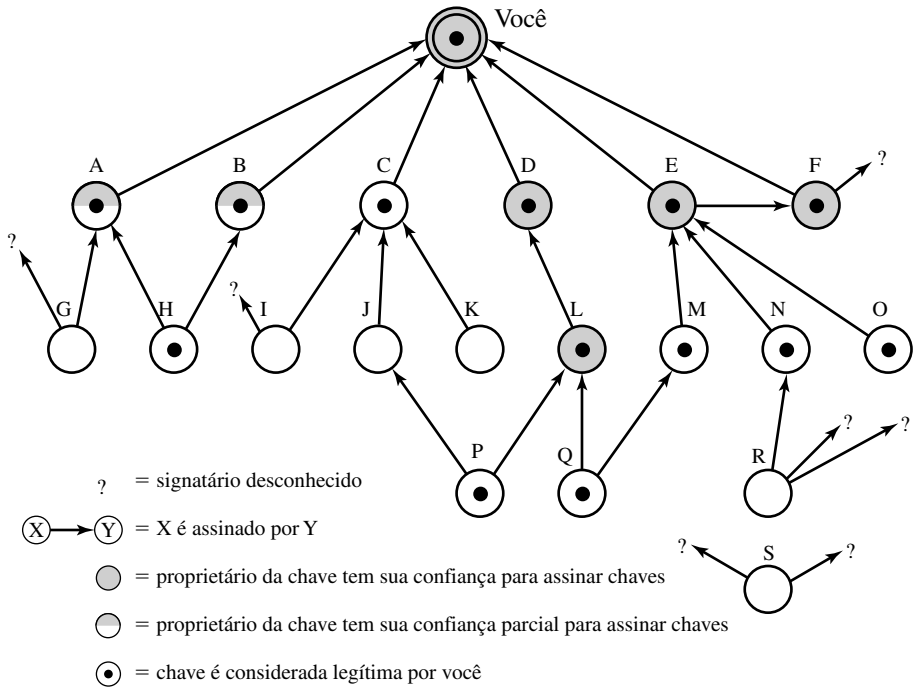


Figura 15.7 Exemplo do modelo de confiança do PGP.

1. Observe que todas as chaves cujos proprietários são total ou parcialmente considerados confiáveis por esse usuário e foram assinadas por este, com a exceção do nó L. A assinatura desse usuário nem sempre é necessária, como indica a presença do nó L, mas, na prática, a maioria dos usuários provavelmente assinará as chaves para a maioria dos proprietários em quem eles confiam. Assim, por exemplo, embora a chave de E já esteja assinada pelo certificador confiável F, o usuário escolheu assinar a chave de E diretamente.
2. Achamos que duas assinaturas parcialmente confiáveis são suficientes para certificar uma chave. Logo, a chave para o usuário H é considerada legítima pelo PGP porque é assinada por A e B, ambos parcialmente confiáveis.
3. Uma chave pode ser determinada como legítima porque é assinada por um signatário totalmente confiável ou dois parcialmente confiáveis, mas seu usuário pode não ser confiável para assinar outras chaves. Por exemplo, a chave de N é legítima porque é assinada por E, em quem esse usuário confia, mas N não é confiável para assinar outras chaves, pois esse usuário não atribuiu a N esse valor de confiança. Portanto, embora a chave de R seja assinada por N, o PGP não considera a chave de R legítima. Essa situação faz total sentido. Se você quiser enviar uma mensagem privada a algum indivíduo, não será necessário que você confie nesse indivíduo de alguma forma. Só será necessário que você esteja certo de possuir a chave pública correta para esse indivíduo.
4. A Figura 15.7 também mostra um exemplo de um nó 'órfão' S, com duas assinaturas desconhecidas. Essa chave pode ter sido adquirida de um servidor de chaves. O PGP não pode supor que essa chave é legítima simplesmente porque veio de um servidor com boa reputação. O usuário precisa declarar a chave legítima assinando-a ou dizendo ao PGP que deseja confiar totalmente em um dos signatários da chave.

Um último ponto: Já mencionamos que várias IDs de usuário podem ser associadas a uma única chave pública na lista de chaves públicas. Isso poderia ser porque uma pessoa trocou de nome ou foi apresentada via assinatura sob vários nomes, indicando diferentes endereços de e-mail para a mesma pessoa, por exemplo. Assim, podemos pensar em uma chave pública como a raiz de uma árvore. Uma chave pública possui uma série de IDs de usuário associados a ela, com uma série de assinaturas abaixo de cada ID de usuário. O vínculo de determinada ID de usuário com uma chave depende das assinaturas associadas a essa ID de usuário e a essa chave, enquanto o nível de confiança nessa chave (para uso na assinatura de outras chaves) é uma função de todas as assinaturas dependentes.

Revogando chaves públicas Um usuário pode querer revogar sua chave pública atual porque suspeita de comprometimento ou simplesmente para evitar o uso da mesma chave por muito tempo. Observe que um comprometimento exigiria que um oponente de alguma forma tivesse obtido uma cópia de sua chave privada não criptografada ou que o oponente tivesse obtido a chave privada da sua lista de chaves privadas e sua frase secreta.

A convenção para revogar uma chave pública é que o proprietário emita um certificado de revogação de chave, assinado pelo proprietário. Esse certificado tem a mesma forma de um certificado de assinatura normal, mas inclui um indicador de que a finalidade desse certificado é revogar o uso dessa chave pública. Observe que a chave privada correspondente precisa ser usada para assinar um certificado que revoga uma chave pública. O proprietário deve então tentar disseminar esse certificado ao máximo e o mais rapidamente possível, para permitir que correspondentes em potencial atualizem suas listas de chave pública.

Observe que um oponente que comprometeu a chave privada de um proprietário também pode emitir esse certificado. Porém, isso negaria ao oponente e também ao proprietário legítimo o uso da chave pública e, portanto, parece ser uma ameaça muito menos provável do que o uso malicioso de uma chave privada roubada.

15.2 S/MIME

O S/MIME (*Secure/Multipurpose Internet Mail Extension*) é um mecanismo de segurança para o padrão de formato de e-mail MIME da Internet, com base na tecnologia da RSA Data Security. Embora o PGP e S/MIME estejam a caminho de se tornar um padrão IETF, parece provável que o S/MIME emergirá como o padrão do setor para uso comercial e organizacional, enquanto o PGP continuará sendo a escolha para segurança de e-mail pessoal por muitos anos. O S/MIME é definido em diversos documentos, sendo os mais importantes as RFCs 3369, 3370, 3850 e 3851.

Para entender o S/MIME, primeiro precisamos ter um conhecimento geral do formato básico de e-mail que ele utiliza, a saber, MIME. Mas, para entender o significado do MIME, precisamos voltar ao padrão tradicional de formato de e-mail, RFC 822, que ainda é comumente utilizado. Por conseguinte, esta seção oferece primeiramente uma introdução a esses dois padrões anteriores, e depois prossegue para uma explicação do S/MIME.

RFC 822

A RFC 822 define um formato para mensagens de texto que são enviadas por meio de e-mail. Ele tem sido o padrão para mensagem de correio de texto baseada na Internet e continua sendo muito usado. No contexto da RFC 822, as mensagens são vistas como tendo um envelope e conteúdo. O envelope contém qualquer informação que seja necessária para conseguir transmissão e entrega. O conteúdo compõe o objeto a ser entregue ao destinatário. O padrão RFC 822 só se aplica ao conteúdo. Porém, o padrão do conteúdo inclui um conjunto de campos de cabeçalho que podem ser usados pelo sistema de correio para criar o envelope, e o padrão visa a facilitar a aquisição dessas informações pelos programas.

A estrutura geral de uma mensagem que esteja em conformidade com a RFC 822 é muito simples. Uma mensagem consiste em algum número de linhas de cabeçalho (*o cabeçalho*) seguidas por texto irrestrito (*o corpo*). O cabeçalho é separado do corpo por uma linha em branco. Em outras palavras, uma mensagem é texto ASCII, e todas as linhas até a primeira linha em branco são consideradas linhas de cabeçalho usadas pela parte do agente do usuário do sistema de correio.

Uma linha de cabeçalho normalmente consiste em uma palavra-chave, seguida por um sinal de dois pontos, seguido pelos argumentos da palavra-chave; o formato permite que uma linha longa seja desmembrada em várias linhas. As palavras-chave mais utilizadas são *From*¸ *To*, *Subject* e *Date*. Eis uma mensagem de exemplo:

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
Subject: A Sintaxe na RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Ola. Esta secao inicia o verdadeiro
corpo da mensagem, o qual eh separado
do cabecalho da mensagem por uma linha
em branco
```

Outro campo que normalmente é encontrado nos cabeçalhos da RFC 822 é *Message-ID*. Esse campo contém um identificador exclusivo associado a essa mensagem.

Multipurpose Internet Mail Extensions

O MIME é uma extensão da estrutura da RFC 822 que pretende resolver alguns dos problemas e limitações do uso do SMTP (Simple Mail Transfer Protocol) ou algum outro protocolo de transferência de correio e a RFC 822 para e-mail. [RODR02] lista as seguintes limitações do esquema SMTP/822:

1. O SMTP não pode transmitir arquivos executáveis ou outros objetos binários. Diversos esquemas são usados para converter arquivos binários em um formato de texto que possa ser usado pelos sistemas de correio

SMTP, incluindo o popular esquema UUencode/UUdecode do UNIX. Porém, nenhum desses é um padrão ou mesmo um padrão *de facto*.

2. O SMTP não pode transmitir dados de texto que incluam caracteres acentuados, pois estes são representados por códigos de 8 bits com valores a partir de 128 (decimal), e o SMTP é limitado ao ASCII de 7 bits.
3. Os servidores SMTP podem rejeitar a mensagem de e-mail acima de um certo tamanho.
4. Gateways SMTP que convertem entre ASCII e o código de caracteres EBCDIC não utilizam um conjunto consistente de mapeamentos, resultando em problemas de conversão.
5. Gateways SMTP para redes de e-mail X.400 não podem tratar de dados não textuais incluídos em mensagens X.400.
6. Algumas implementações SMTP não aderem completamente aos padrões SMTP definidos na RFC 821. Alguns problemas comuns incluem:

   - Exclusão, adição ou reordenação de retorno de carro (*carriage return*-CR) e alimentação de linha (*line-feed*-LF)
   - Truncagem ou quebra de linhas maiores do que 76 caracteres
   - Remoção de espaços em branco no final (caracteres de tabulação e espaço)
   - Complementação de linhas em uma mensagem para o mesmo tamanho
   - Conversão de caracteres de tabulação em vários caracteres de espaço

O MIME tem por finalidade resolver esses problemas de maneira compatível com as implementações RFC 822 existentes. A especificação é fornecida nas RFCs 2045 a 2049.

Visão geral A especificação MIME inclui os seguintes elementos:

1. Cinco novos campos de cabeçalho de mensagem são definidos, e estes podem ser incluídos em um cabeçalho RFC 822. Esses campos oferecem informações sobre o corpo da mensagem.
2. Diversos formatos de conteúdo são definidos, padronizando assim representações que suportam mensagens multimídia.
3. Codificações de transferência são definidas para permitir a conversão de qualquer formato de conteúdo em uma forma protegida contra alteração pelo sistema de correio.

Nesta subseção, apresentamos os cinco cabeçalhos de mensagem. As duas subseções seguintes tratam dos formatos de conteúdo e de codificações de transferência.

Os cinco campos de cabeçalho definidos no MIME são os seguintes:

- **MIME-Version:** Precisa ter o valor de parâmetro 1.0. Esse campo indica que a mensagem está em conformidade com as RFCs 2045 e 2046.
- **Content-Type:** Descreve os dados contidos no corpo com detalhes suficiente para que o agente do usuário receptor possa escolher um agente ou mecanismo apropriado para representar os dados ao usuário ou lidar de alguma outra forma com os dados de uma maneira apropriada.
- **Content-Transfer-Encoding:** Indica o tipo de transformação que foi usado para representar o corpo da mensagem de uma maneira que seja aceitável para transporte de correio.
- **Content-ID:** Usado para identificar entidades MIME exclusivamente em contextos múltiplos.
- **Content-Description:** Uma descrição textual do objeto com o corpo; isso é útil quando o objeto não é legível (por exemplo, dados de áudio).

Qualquer um ou todos esses campos podem aparecem em um cabeçalho RFC 822 normal. Uma implementação compatível precisa ter suporte para os campos MIME-Version, Content-Type e Content-Transfer-Encoding; os campos Content-ID e Content-Description são opcionais e podem ser ignorados pela implementação do destinatário.

Tipos de conteúdo MIME O núcleo da especificação MIME trata da definição de diversos tipos de conteúdo. Isso reflete a necessidade de fornecer maneiras padronizadas de lidar com uma grande variedade de representações de informação em um ambiente multimídia.

A Tabela 15.3 lista os tipos de conteúdo especificados na RFC 2046. Existem sete tipos de conteúdo principais diferentes e um total de 15 subtipos. Em geral, um tipo de conteúdo declara o tipo geral dos dados, e o subtipo especifica um formato em particular para esse tipo de dados.

Para o **tipo text** do corpo, nenhum software especial é exigido para obter o significado completo do texto, fora o suporte do conjunto de caracteres indicado. O subtipo primário é *texto sem formatação (plaintext)*, que é simplesmente uma string de caracteres ASCII ou caracteres ISO 8859. O subtipo *enriched* permite uma maior flexibilidade de formatação.

O **tipo multipart** indica que o corpo contém várias partes independentes. O campo de cabeçalho Content-Type inclui um parâmetro, chamado limite ou fronteira (*boundary*), que define o delimitador entre as partes do corpo. Esse

Tabela 15.3 Tipos de conteúdo MIME

Tipo	Subtipo	Descrição
Text	Plain	Texto não formatado; pode ser ASCII ou ISO 8859.
	Enriched	Oferece maior flexibilidade de formato.
Multipart	Mixed	As diferentes partes são independentes, mas devem ser transmitidas juntas. Elas devem ser apresentadas ao receptor na ordem em que aparecem na mensagem de e-mail.
	Parallel	Difere de Mixed apenas porque nenhuma ordem é definida para a entrega das partes ao receptor.
	Alternative	As diferentes partes são versões alternativas da mesma informação. Elas são ordenadas em ordem crescente de fidelidade ao original, e o sistema de e-mail do destinatário deverá exibir a 'melhor' versão ao usuário.
	Digest	Semelhante a Mixed, mas o tipo/subtipo padrão de cada parte é message/rfc822.
Message	rfc822	O corpo é, ele mesmo, uma mensagem encapsulada em conformidade com a RFC 822.
	Partial	Usado para permitir a fragmentação de grandes itens de e-mail, de uma maneira transparente ao destinatário.
	External-body	Contém um ponteiro para um objeto que existe em outro lugar.
Image	jpeg	A imagem está no formato JPEG, codificação JFIF.
	gif	A imagem está no formato GIF.
Video	mpeg	Formato MPEG.
Audio	Basic	Codificação ISDN lei μ em 8 bits e único canal, em uma taxa de amostragem de 8 kHz.
Application	PostScript	Adobe Postscript.
	octet-stream	Dados binários em geral consistindo em bytes de 8 bits.

limite não deverá aparecer em qualquer parte da mensagem. Cada limite começa em uma nova linha e consiste em dois hifens seguidos pelo valor do limite. O limite final, que indica o final da última parte, também possui um sufixo de dois hifens. Dentro de cada parte pode haver um cabeçalho MIME comum opcional.

Aqui está um exemplo simples de uma mensagem multipart, contendo duas partes, ambas consistindo em texto simples (retirado da RFC 2046):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To:  Ned Freed <ned@innosoft.com>
Subject: Mensagem exemplo
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="limite simples"

Este eh o preambulo. Deve ser ignorado, embora
Seja um bom lugar para programas de e-mail
incluirem uma nota explicativa para leitores
de e-mail que não sejam compativeis com MIME.
—limite simples
Este eh um texto ASCII sem formatacao implicitamente identificado.
Ele nao
termina com uma quebra de linha
—limite simples
Content-type: text/plain; charset=us-ascii
Este eh um texto ASCII sem formatacao explicitamente identificado.
Ele
TERMINA com uma quebra de linha
—limite simples
Esta eh a ultima parte. Tambem deve ser ignorada
```

Existem quatro subtipos do tipo multipart, todos tendo a mesma sintaxe geral. O **subtipo multipart/mixed** é usado quando existem várias partes de corpo independentes, que precisam ser combinadas em uma determinada ordem. Para o **subtipo multipart/parallel**, a ordem das partes não é significativa. Se o sistema do destinatário for apropriado, as várias partes podem ser apresentadas em paralelo. Por exemplo, uma parte de imagem ou texto poderia ser acompanhada por um comentário de voz reproduzido enquanto a figura ou texto é exibido.

Para o **subtipo multipart/alternative**, as diversas partes são diferentes representações da mesma informação. A seguir vemos um exemplo:

```
From:  Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Mensagem de texto formatado
MIME-Version: 1.0
Content-Type: multipart/alternative;boundary=limite42
—limite42
Content-Type: text/plain; charset=us-ascii
   ... versao de texto sem formatacao da mensagem entra aqui ....
—limite42
Content-Type: text/enriched
   .... versao text/enriched da RFC 1896 da mesma mensagem
entra aqui ...
—limite42—
```

Nesse subtipo, as partes do corpo são ordenadas em termos de preferência crescente. Para este exemplo, se o sistema do destinatário for capaz de exibir a mensagem no formato text/enriched, isso será feito; caso contrário, o formato de texto claro será usado.

O **subtipo multipart/digest** é usado quando cada uma das partes do corpo é interpretada como uma mensagem RFC 822 com cabeçalhos. Esse subtipo permite a construção de uma mensagem cujas partes são mensagens individuais. Por exemplo, o moderador de um grupo pode coletar mensagens de e-mail dos participantes, reunir essas mensagens e enviá-las em uma mensagem MIME encapsulada.

O **tipo message** oferece diversos recursos importantes em MIME. O **subtipo message/rfc822** indica que o corpo é uma mensagem inteira, incluindo cabeçalho e corpo. Apesar do nome desse subtipo, a mensagem encapsulada pode ser não apenas uma mensagem RFC 822 simples, mas também qualquer mensagem MIME.

O **subtipo message/partial** permite a fragmentação de uma mensagem grande em diversas partes, que precisam ser remontadas no destino. Para esse subtipo, três parâmetros são especificados no campo Content-Type: Message/Partial: uma ID comum a todos os fragmentos da mesma mensagem, um *número de seqüência* exclusivo de cada fragmento e o número *total* de fragmentos.

O **subtipo message/external-body** indica que os dados reais a serem transportados nessa mensagem não estão contidos no corpo. Em vez disso, o corpo contém as informações necessárias para acessar os dados. Assim como os outros tipos de mensagem, o subtipo message/external-body tem um cabeçalho externo e uma mensagem encapsulada com seu próprio cabeçalho. O único campo necessário no cabeçalho externo é o campo Content-Type, que identifica isso como um subtipo message/external-body. O cabeçalho interno é o cabeçalho da mensagem para a mensagem encapsulada. O campo Content-Type no cabeçalho externo precisa incluir um parâmetro de tipo de acesso, que indica o método de acesso, como FTP (File Transfer Protocol).

O **tipo application** refere-se a outros tipos de dados, normalmente dados binários não interpretados ou informações a serem processadas por um aplicativo baseada em e-mail.

Codificações de transferência MIME O outro componente importante da especificação MIME, além da especificação de tipo de conteúdo, é uma definição de codificações de transferência para corpos de mensagem. O objetivo é oferecer remessa confiável para a maior variedade de ambientes.

O padrão MIME define dois métodos de codificação de dados. O campo Content-Transfer-Encoding pode realmente assumir seis valores, conforme listados na Tabela 15.4. Porém, três desses valores (7bit, 8bit e binary) indicam que nenhuma codificação foi feita, mas oferecem alguma informação sobre a natureza dos dados. Para transferência SMTP, é seguro usar o formato 7bit. Os formatos 8bit e binary podem ser utilizáveis em outros contextos de transporte de correio. Outro valor de Content-Transfer-Encoding é x-token, que indica que algum outro esquema de codificação é utilizado, para o qual um nome deve ser fornecido. Esse poderia ser um esquema específico do fornecedor ou específico da aplicação. Os dois esquemas de codificação reais definidos são quoted-printable e base64. Dois esquemas são definidos para oferecer uma escolha entre uma técnica de transferência que é basicamente legível para o ser humano e outra que é segura para todos os tipos de dados de uma maneira razoavelmente compacta.

Tabela 15.4 Codificações de transferência MIME

7bit	Os dados são representados por linhas curtas de caracteres ASCII.
8bit	As linhas são curtas, mas pode haver caracteres não-ASCII (octetos com o bit mais significativo setado).
Binary	Não apenas os caracteres não-ASCII podem estar presentes, mas as linhas não são necessariamente curtas o suficiente para o transporte SMTP.
Quoted-printable	Codifica os dados de modo que, se os dados sendo codificados forem principalmente texto ASCII, a forma codificada dos dados permanecerá em grande parte reconhecível por seres humanos.
base64	Codifica dados pelo mapeamento de blocos de 6 bits de entrada para blocos de 8 bits de saída, todos caracteres ASCII imprimíveis.
x-token	Uma codificação especificada não padronizada.

A codificação de transferência **quoted-printable** é útil quando os dados consistem em grande parte de octetos que correspondem a caracteres ASCII imprimíveis. Basicamente, ela representa caracteres inseguros pela representação hexadecimal de seu código, e introduz quebras de linhas reversíveis (flexíveis) para limitar as linhas da mensagem a 76 caracteres.

A **codificação de transferência base64**, também conhecida como codificação radix 64, é comum para a codificação de dados binários arbitrários de modo que sejam invulneráveis ao processamento por programas de transporte de correio. Ela também é usada no PGP e descrita no Apêndice 15B.

Um exemplo de multipart A Figura 15.8, retirada da RFC 2045, é um esboço de uma mensagem multipart complexa. A mensagem tem cinco partes a serem exibidas serialmente: duas partes de texto claro introdutórias, uma mensagem multipart embutida, uma parte richtext e uma mensagem de texto encapsulada de fechamento, em um conjunto de caracteres não ASCII. A mensagem multipart embutida tem duas partes a serem exibidas em paralelo, uma imagem e um fragmento de áudio.

Forma canônica Um conceito importante em MIME e S/MIME é o da forma canônica. A forma canônica é um formato, apropriado ao tipo de conteúdo, que é padronizado para uso entre sistemas. Isso é contrário à forma nativa, que é um formato que pode ser peculiar a determinado sistema. A Tabela 15.5, da RFC 2049, deverá ajudar a esclarecer essa questão.

Funcionalidade do S/MIME

Em termos de funcionalidade geral, O S/MIME é muito semelhante ao PGP. Ambos oferecem a capacidade de assinar e/ou criptografar mensagens. Nesta subseção, resumimos rapidamente a capacidade do S/MIME. Depois, examinamos essa capacidade com mais detalhes, examinando os formatos de mensagem e a preparação da mensagem.

Funções O S/MIME oferece as seguintes funções:

- **Dados envelopados:** Consiste em conteúdo criptografado de qualquer tipo e as chaves de criptografia do conteúdo criptografado para um ou mais destinatários.
- **Dados assinados:** Uma assinatura digital é formada tomando-se o resumo da mensagem do conteúdo a ser assinado, e depois o criptografando com a chave privada do assinante. O conteúdo mais assinatura são então codificados usando a codificação base64. Uma mensagem de dados assinada só pode ser vista por um destinatário com capacidade S/MIME.
- **Dados assinados às claras:** Assim como com os dados assinados, uma assinatura digital do conteúdo é realizada. Porém, nesse caso, somente a assinatura digital é codificada usando base64. Como resultado, os destinatários sem capacidade S/MIME podem visualizar o conteúdo da mensagem, embora não possam verificar a assinatura.
- **Dados assinados e envelopados:** Pode haver aninhamento, de modo que dados criptografados possam ser assinados e dados assinados ou assinados às claras possam ser criptografados.

Algoritmos criptográficos A Tabela 15.6 resume os algoritmos criptográficos usados em S/MIME. O S/MIME usa a terminologia a seguir, retirada da RFC 2119 para especificar o nível do requisito:

- **Deve:** A definição é um requisito absoluto da especificação. Uma implementação precisa incluir esse recurso ou função para estar em conformidade com a especificação.
- **Deveria:** Pode haver motivos válidos em determinadas circunstâncias para ignorar esse recurso ou função, mas recomenda-se que uma implementação inclua o recurso ou a função.

```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Um exemplo de multiparte
Content-Type: multipart/mixed;
boundary=limite-unico-1

Este eh um preambulo de uma mensagem multiparte. Leitores de e-mail deveriam ignorar
isto. Se voce estah lendo este texto, deveria considerar trocar para um leitor de e-
mail que compreenda como mostrar adequadamente mensagens multipart.

—limite-unico-1

...Algum texto aparece aqui...
[Observe que a linha em branco precedente significa que não foram fornecidos campos
de cabecalho e que isto eh texto com conjunto de caracteres US ASCII. Poderia ter
sido feito de forma explicita como na próxima parte.]

—limite-unico-1

Content-type: text/plain; charset=US-ASCII
Isto poderia estar na parte anterior, mas serve para ilustrar a identificacao impli-
cita e explicita do tipo das partes no corpo da mensagem.

—limite-unico-1
Content-Type: multipart/parallel; boundary=limite-unico-2

—limite-unico-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64
... audio formato lei mu canal-unico 8000 Hz codificado em base64 vai aqui....

—limite-unico-2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

... imagem codificada em base 64 vai aqui ....

—limite-unico-2——

—limite-unico-1
Content-type: text/enriched

Este eh <bold><italic>richtext (texto formatado).</italic></bold> <smaller>como
definido na RFC 1896</smaller>

Nao eh <bigger><bigger>legal?</bigger></bigger>

—limite-unico-1
Content-Type: message/rfc822

From: (caixa postal em US-ASCII)
To: (endereco em US-ASCII)
Subject: (assunto em US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

... Texto adicional em ISO-8859-1 vai aqui ...

—limite-unico-1—
```

Figura 15.8 Exemplo de estrutura da mensagem MIME.

Tabela 15.5 Formas nativa e canônica

Forma nativa	O corpo a ser transmitido é criado no formato nativo do sistema. O conjunto de caracteres nativo é usado e, onde apropriado, as convenções de fim de linha locais também são usadas. O corpo pode ser um arquivo de texto em estilo UNIX, ou uma imagem bitmap da Sun, ou um arquivo indexado do VMS, ou dados de áudio em um formato dependente do sistema armazenado apenas na memória, ou qualquer outra coisa que corresponda ao modelo local para a representação de alguma forma de informação. Fundamentalmente, os dados são criados no formato 'nativo' que corresponde ao tipo especificado pelo tipo de mídia.
Forma canônica	O corpo inteiro, incluindo informações extras como tamanhos de registro e possivelmente informações de atributos de arquivo, é convertido em uma forma canônica universal. O tipo de mídia específico do corpo, além de seus atributos associados, dita a natureza da forma canônica usada. A conversão na forma canônica apropriada pode envolver a conversão do conjunto de caracteres, transformação de dados de áudio, compressão ou várias outras operações específicas aos diversos tipos de mídia. Porém, se a conversão do conjunto de caracteres for envolvida, deve-se ter o cuidado de entender a semântica do tipo de mídia, que pode ter fortes implicações para qualquer conversão de conjunto de caracteres (por exemplo, com relação a caracteres sintaticamente significativos em um subtipo de texto diferente de 'plain' – sem formatação).

Tabela 15.6 Algoritmos criptográficos usados no S/MIME

Função	Requisito
Criar uma mensagem a ser usada na formação de uma assinatura digital Criptografar resumo da mensagem para formar assinatura digital.	DEVE admitir SHA-1. O recebedor DEVERIA oferecer suporte a MDS para compatibilidade com versões anteriores. Agentes de envio e recebimento DEVEM oferecer suporte à DSS. Agentes de recebimento DEVERIAM oferecer suporte à criptografia RSA. Agentes de recebimento DEVERIAM oferecer suporte à verificação de assinaturas RSA com tamanhos de chave de 512 bits a 1.024 bits.
Criptografar chave de sessão para transmissão com mensagem.	Agentes de envio e recebimento DEVERIAM oferecer suporte a Diffie-Hellman. Agentes de envio e recebimento DEVEM oferecer suporte à criptografia RSA com tamanhos de chave de 512 bits a 1.024 bits.
Criptografar mensagem para transmissão com chave de sessão de uso único.	Agentes de envio e recebimento DEVEM oferecer suporte à criptografia com 3DES. Agentes de envio DEVERIAM oferecer suporte à criptografia com AES. Agentes de envio DEVERIAM oferecer suporte à criptografia com RC2/40.
Criar um código de autenticação de mensagem.	Agentes de recebimento DEVEM oferecer suporte a HMAC com SHA-1. Agentes de recebimento DEVERIAM oferecer suporte a HMAC com SHA-1.

O S/MIME incorpora três algoritmos de chave pública. O Digital Signature Standard (DSS) descrito no Capítulo 13 é o algoritmo preferido para assinatura digital. O S/MIME lista Diffie-Hellman como algoritmo preferido para criptografar chaves de sessão; de fato, O S/MIME usa uma variante do Diffie-Hellman que oferece criptografia/decriptografia, conhecida como ElGamal (ver Problema 10.6). Como alternativa, o RSA, descrito no Capítulo 9, pode ser usado para assinaturas e criptografia de chave de sessão. Esses são os mesmos algoritmos usados no PGP, e oferecem um alto nível de segurança. Para a função de hash usada para criar a assinatura digital, a especificação exige o SHA-

1 de 160 bits, mas recomenda suporte do receptor para o MD5 de 128 bits, por motivo de compatibilidade com versões mais antigas do S/MIME. Conforme explicamos no Capítulo 12, existe uma preocupação justificável quanto à segurança do MD5, de modo que o SHA-1 é claramente a alternativa preferida.

Para criptografia de mensagens, o 3DES (triple DES) com três chaves é recomendado, mas implementações compatíveis precisam admitir o RC2 com 40 bits. Esse último é um algoritmo de criptografia fraco, mas permite compatibilidade com os controles de exportação dos Estados Unidos.

A especificação S/MIME inclui uma explicação do procedimento para decidir qual algoritmo de criptografia de conteúdo deve ser usado. Basicamente, um agente de envio tem duas decisões a tomar. Primeiro, o agente de envio precisa determinar se o agente de recebimento é capaz de decriptografar usando determinado algoritmo de criptografia. Segundo, se o agente de recebimento só for capaz de aceitar conteúdo fracamente criptografado, o agente de envio precisará decidir se é aceitável enviar usando criptografia fraca. Para apoiar esse processo de decisão, um agente de envio pode informar seus recursos de decriptografia, em ordem de preferência, em qualquer mensagem enviada. Um agente de recebimento pode armazenar essa informação para uso futuro.

As regras a seguir, nessa ordem, deverão ser seguidas por um agente de envio:

1. Se o agente de envio tiver uma lista de recursos de decriptografia preferenciais de um destinatário, ele DEVERIA escolher o primeiro recurso (preferência mais alta) na lista que ele seja capaz de usar.
2. Se o agente de envio não tiver essa lista de recursos de um destinatário, mas tiver recebido uma ou mais mensagens do destinatário, então a mensagem sendo enviada DEVERIA usar o mesmo algoritmo de criptografia usado na última mensagem assinada e criptografada, recebida desse destinatário.
3. Se o agente de envio não tiver conhecimento sobre os recursos de decriptografia do destinatário e quiser correr o risco de o destinatário não ser capaz de decriptografar a mensagem, então o agente de envio DEVERIA usar 3DES.
4. Se o agente de envio não tiver conhecimento sobre os recursos de decriptografia do destinatário e não quiser correr o risco de o destinatário não ser capaz de decriptografar a mensagem, então o agente de envio DEVE usar RC2/40.

Se uma mensagem tiver de ser enviada a vários destinatários e um algoritmo de criptografia comum não puder ser selecionado para todos, então o agente de envio terá de enviar duas mensagens. Porém, nesse caso, é importante observar que a segurança da mensagem se torna vulnerável pela transmissão de uma cópia com menor segurança.

Mensagens S/MIME

O S/MIME utiliza diversos tipos de conteúdo MIME novos, que são mostrados na Tabela 15.7. Todos os novos tipos de aplicação utilizam a designação PKCS. Isso se refere a um conjunto de especificações de criptografia de chave pública emitidas pela RSA Laboratories e disponíveis para o esforço S/MIME.

Examinamos cada um desses por sua vez, depois de examinar os procedimentos gerais para preparação de mensagem S/MIME.

Tabela 15.7 Tipos de conteúdo S/MIME

Tipo	Subtipo	Parâmetro smime	Descrição
Multipart	Signed		Uma mensagem assinada às claras em duas partes: uma é a mensagem e a outra é a assinatura.
Application	pkcs 7-mime	signedData	Uma entidade S/MIME assinada.
	pkcs 7-mime	envelopedData	Uma entidade S/MIME criptografada.
	pkcs 7-mime	signedData degenerado	Uma entidade contendo apenas certificados de chave pública.
	pkcs 7-mime	CompressedData	Uma entidade S/MIME comprimida.
	pkcs 7-signature	signedData	O tipo de conteúdo da subparte de assinatura de uma mensagem multipart/signed.

Protegendo uma entidade MIME O S/MIME protege uma entidade MIME com uma assinatura, criptografia ou ambos. Uma entidade MIME pode ser uma mensagem inteira (exceto para os cabeçalhos RFC 822), ou, se o tipo de conteúdo MIME for multipart, então a entidade MIME é uma ou mais das subpartes da mensagem. A entidade MIME é preparada de acordo com as regras normais para preparação de mensagens MIME. Depois, a entidade MIME mais alguns dados relacionados a segurança, como identificadores de algoritmo e certificados, são processados pelo S/MIME para produzir o que é conhecido como um objeto PKCS. Um objeto PKCS é então tratado como conteúdo de mensagem e embrulhado no MIME (fornecido com cabeçalhos MIME apropriados). Esse processo deverá se tornar claro quando examinarmos os objetos específicos e fornecermos exemplos.

Em todos os casos, a mensagem a ser enviada é convertida na forma canônica. Em particular, para determinado tipo e subtipo, a forma canônica apropriada é usada para o conteúdo da mensagem. Para uma mensagem multipart, a forma canônica apropriada é usada para cada subparte.

O uso da codificação de transferência requer atenção especial. Para a maioria dos casos, o resultado de aplicar o algoritmo de segurança será produzir um objeto parcial ou totalmente representado por dados binários arbitrários. Este será, então, embrulhado em uma mensagem MIME externa e a codificação de transferência poderá ser aplicada nesse ponto, normalmente base64. Porém, no caso de uma mensagem assinada multipart, descrita com mais detalhes adiante, o conteúdo da mensagem em uma das subpartes permanece inalterado pelo processo de segurança. A menos que esse conteúdo seja 7bit, ele deverá ser codificado usando base64 ou quoted-printable, de modo que não haja perigo de alterar o conteúdo ao qual a assinatura foi aplicada.

Examinamos cada um dos tipos de conteúdo S/MIME.

EnvelopedData Um subtipo application/pkcs7-mime é usado para uma das quatro categorias de processamento S/MIME, cada uma com um parâmetro smime-type exclusivo. Em todos os casos, a entidade resultante, conhecida como um *objeto*, é representada em um formato conhecido como Basic Encoding Rules (BER), que é definido na ITU-T Recommendation X.209. O formato BER consiste em strings de octeto arbitrárias e, portanto, são dados binários. Esse objeto deve ser codificado por transferência com base64 na mensagem MIME externa. Primeiramente, examinamos envelopedData.

As etapas para preparar uma entidade MIME envelopedData são as seguintes:

1. Gere uma chave de sessão pseudo-aleatória para determinado algoritmo de criptografia simétrica (RC2/40 ou 3DES).
2. Para cada destinatário, criptografe a chave de sessão com a chave RSA pública do destinatário.
3. Para cada destinatário, prepare um bloco conhecido como RecipientInfo, que contém um identificador do certificado de chave pública do destinatário,[3] um identificador do algoritmo usado para criptografar a chave de sessão e a chave de sessão criptografada.
4. Criptografe o conteúdo da mensagem com a chave de sessão.

Os blocos RecipientInfo seguidos pelo conteúdo criptografado constituem o envelopedData. Essa informação é então codificada com base64. Uma mensagem de exemplo (excluindo os cabeçalhos RFC 822) é a seguinte:

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-
        data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
rfvbnj75.6tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

Para recuperar a mensagem criptografada, o destinatário remove primeiramente a codificação base64. Depois, a chave privada do destinatário é usada para recuperar a chave de sessão. Finalmente, o conteúdo da mensagem é decriptografado com a chave de sessão.

SignedData O smime-type signedData, na realidade, pode ser usado com um ou mais assinantes. Por clareza, restringimos nossa descrição ao caso de uma única assinatura digital. As etapas para preparar uma entidade MIME signedData são as seguintes:

1. Selecione um algoritmo de resumo de mensagem (SHA ou MD5).
2. Calcule o resumo da mensagem, ou função de hash, do conteúdo a ser assinado.
3. Criptografe o resumo da mensagem com a chave privada do assinante.
4. Prepare um bloco, conhecido como SignerInfo, que contém o certificado de chave pública do assinante, um identificador do algoritmo de resumo da mensagem, um identificador do algoritmo usado para criptografar o resumo da mensagem e o resumo da mensagem criptografada.

A entidade signedData consiste em uma série de blocos, incluindo um identificador do algoritmo de resumo da mensagem, a mensagem sendo assinada, e SignerInfo. A entidade signedData também pode incluir um conjunto de certificados de chave pública suficientes para constituir uma cadeia a partir de uma raiz reconhecida ou autoridade de certificação de alto nível até o assinante. Essa informação é então codificada com base64. Uma mensagem de exemplo (excluindo os cabeçalhos RFC 822) é a seguinte:

3. Esse é um certificado X.509, abordado mais adiante, nesta seção.

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
        name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

Para recuperar a mensagem assinada e verificar a assinatura, o destinatário remove primeiramente a codificação base64. Depois, a chave pública do assinante é usada para decriptografar o resumo da mensagem. O destinatário calcula o resumo da mensagem e o compara com o resumo da mensagem decriptografado para verificar a assinatura.

Assinatura às claras A assinatura às claras é obtida por meio do tipo de conteúdo multipart com um subtipo assinado. Como já dissemos, esse processo de assinatura não envolve a transformação da mensagem a ser assinada, de modo que a mensagem é enviada 'às claras'. Assim, os destinatários com capacidade MIME, mas não com capacidade S/MIME, serão capazes de ler a mensagem que chega.

Uma mensagem multipart/signed possui duas partes. A primeira parte pode ser qualquer tipo MIME, mas precisa ser preparada de modo que não seja alterada durante a transferência da origem ao destino. Isso significa que, se a primeira parte não for 7bit, então ela precisa ser codificada usando base64 ou quoted-printable. Depois, essa parte é processada da mesma maneira que signedData, mas nesse caso um objeto com formato signedData é criado, com um campo de conteúdo de mensagem vazio. Esse objeto é uma assinatura separada. Depois, ele é codificado usando base64, para se tornar a segunda parte da mensagem multipart/signed. Essa segunda parte tem um tipo de conteúdo MIME application e um subtipo pkcs7-signature. Aqui está uma mensagem de exemplo:

```
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary=limite42
—limite42
Content-Type: text/plain
This is a clear-signed message.
—limite42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
   —limite42—
```

O parâmetro *protocol* indica que essa é uma entidade assinada às claras em duas partes. O parâmetro *micalg* indica o tipo de resumo de mensagem utilizado. O receptor pode verificar a assinatura apanhando o resumo da mensagem da primeira parte e comparando-o com o resumo da mensagem recuperado da assinatura na segunda parte.

Solicitação de registro Normalmente, uma aplicação ou usuário solicitará de uma autoridade de certificação um certificado de chave pública. A entidade S/MIME application/pkcs10 é usada para transferir uma solicitação de certificação. A solicitação de certificação inclui o bloco certificationRequestInfo, seguido por um identificador do algoritmo de criptografia de chave pública, seguido pela assinatura do bloco certificationRequestInfo, feita por meio da chave privada do emissor. O bloco certificationRequestInfo inclui um nome do titular do certificado (a entidade cuja chave pública deve ser certificada) e uma representação de string de bits da chave pública do usuário.

Mensagem apenas com certificados Uma mensagem contendo apenas certificados ou uma lista de revogação de certificado (CRL) pode ser enviada em resposta a uma solicitação de registro. A mensagem é um tipo/subtipo application/pkcs7-mime com um parâmetro smime-type de degenerate. As etapas envolvidas são iguais àquelas para a criação de uma mensagem signedData, exceto que não existe conteúdo de mensagem e o campo signerInfo está vazio.

Processamento de certificado S/MIME

O S/MIME usa certificados de chave pública que estão de acordo com a versão 3 do X.509 (ver Capítulo 14). O esquema de gerenciamento de chaves usado pelo S/MIME é, de algumas maneiras, um híbrido entre uma hierarquia de cer-

tificação X.509 estrita e a teia de confiança do PGP. Assim como o modelo do PGP, gerenciadores e/ou usuários do S/MIME precisam configurar cada cliente com uma lista de chaves confiáveis e com listas de revogação de certificado. Ou seja, é local a responsabilidade por manter os certificados necessários para verificar as assinaturas que chegam e criptografar as mensagens que saem. Por outro lado, os certificados são assinados por autoridades de certificação.

Papel do agente do usuário Um usuário S/MIME possui diversas funções de gerenciamento de chaves a realizar:

- **Geração de chave:** O usuário de algum utilitário administrativo relacionado (por exemplo, um associado com gerenciamento de LAN) DEVE ser capaz de gerar pares de chaves Diffie-Hellman e DSS separadas e DEVERIA ser capaz de gerar pares de chave RSA. Cada par de chaves DEVE ser gerado a partir de uma boa fonte de entrada aleatória não determinística e ser protegido de modo seguro. Um agente do usuário DEVERIA gerar pares de chaves RSA com um tamanho no intervalo de 768 a 1.024 bits, e NÃO DEVE gerar com um tamanho menor que 512 bits.
- **Registro:** A chave pública de um usuário precisa ser registrada com uma autoridade de certificação a fim de receber um certificado de chave pública X.509.
- **Armazenamento e recuperação de certificado:** Um usuário exige acesso a uma lista local de certificados a fim de verificar as assinaturas que chegam e criptografar as mensagens que saem. Essa lista pode ser mantida pelo usuário ou por alguma entidade administrativa local em nome de diversos usuários.

Certificados VeriSign Existem várias empresas que oferecem serviços de Autoridade Certificadoras (CA – Certification Authority). Por exemplo, a Nortel criou uma solução de CA empresarial e pode oferecer suporte para S/MIME dentro de uma organização. Existem diversas CAs baseadas na Internet, incluindo VeriSign, GTE e o U.S. Postal Service. Dessas, a mais utilizada é o serviço de CA VeriSign, do qual oferecemos agora uma rápida descrição.

A VeriSign oferece um serviço de CA que busca ser compatível com S/MIME e diversas outras aplicações. A VeriSign emite certificados X.509 com o nome de produto VeriSign Digital ID. No início de 1998, mais de 35.000 sites Web comerciais estavam usando VeriSign Server Digital IDs e mais de um milhão de Digital IDs de consumidor tinham sido emitidas aos usuários de navegadores Netscape e Microsoft.

As informações contidas em uma Digital ID depende do tipo de Digital ID e de seu uso. No mínimo, cada Digital ID contém

- Chave pública do proprietário
- Nome ou pseudônimo do proprietário
- Data de expiração da Digital ID
- Número de série da Digital ID
- Nome da autoridade certificadora que emitiu a Digital ID
- Assinatura digital da autoridade certificadora que emitiu a Digital ID

Digital IDs também podem conter outras informações fornecidas pelo usuário, incluindo

- Endereço
- Endereço de e-mail
- Informações básicas de registrador (país, código postal, idade e sexo)

A VeriSign oferece três níveis, ou classes de segurança, para certificados de chave pública, conforme resumidos na Tabela 15.8. Um usuário solicita um certificado on-line no site Web da VeriSign ou em outros sites Web participantes. As solicitações Classe 1 e Classe 2 são processadas on-line e na maioria dos casos são aprovadas em apenas alguns segundos. Resumidamente, os seguintes procedimentos são usados:

- Para Digital IDs de Classe 1, o VeriSign confirma o endereço de e-mail do usuário enviando uma informação de escolha de PIN e Digital ID para o endereço de e-mail fornecido na aplicação.
- Para Digital IDs de Classe 2, a VeriSign verifica as informações na aplicação por meio de uma comparação automatizada com um banco de dados de consumidor, além de realizar toda a verificação associada a uma Digital ID de Classe 1. Finalmente, a confirmação é enviada ao endereço postal especificado, alertando o usuário de que uma Digital ID foi emitida em seu nome.
- Para Digital IDs de Classe 3, o VeriSign requer um nível mais alto de garantia de identidade. Um indivíduo precisa provar sua identidade fornecendo credenciais reconhecidas ou solicitando pessoalmente.

Serviços de segurança aperfeiçoados

No momento em que este livro estava sendo escrito, três serviços de segurança aperfeiçoados tinham sido propostos em um Internet draft. Os detalhes poderão mudar, e outros serviços podem ser acrescentados. Os três serviços são os seguintes:

Tabela 15.8 Classes de certificado de chave pública VeriSign

	Resumo de confirmação de identidade	Proteção de chave privada IA	Pedido de certificado e proteção de chave privada do assinante	Aplicações implementadas ou contempladas pelos usuários
Classe 1	Busca de nome e endereço de e-mail automatizada não ambíguasem ambiguidade de nome e endereço de e-mail	PCA: hardware confiável; CA: software confiável ou hardware confiável	Software de criptografia (protegido por PIN) recomendado, mas não obrigatório	Navegação na Web e uso acreditado de e-mail
Classe 2	O mesmo que a Classe 1, mais verificação automatizada de informação de cadastro mais verificação de endereço automatizada	PCA e CA: hardware confiável	Software de criptografia (protegido por PIN) obrigatório	E-mail individual e intra/entre empresas, inscrições on-line, substituição de senha e validação de software
Classe 3	O mesmo que a Classe 1, mais presença pessoal e documentos de ID mais verificação de ID automatizada da Classe 2 para indivíduos; registros comerciais (ou de registros públicos) para organizações	PCA e CA: hardware confiável	Software de criptografia (protegido por PIN) obrigatório; token de hardware recomendado, mas não obrigatório	E-banking, corp, acesso a banco de dados, personal banking, serviços on-line baseados em inscrição, serviços de integridade de conteúdo, servidor de e-commerce, validação de software; autenticação de Autoridades de Registro Locais; e criptografia forte para certos servidores

IA Issuing Authority
CA Certification Authority
PCA Autoridade de certificação primária pública do VeriSign
PIN Personal Identification Number
LRAA Local Registration Authority Administrator

- **Recibos assinados:** Um recibo assinado pode ser solicitado em um objeto SignedData. O retorno de um recibo assinado oferece prova de remessa ao originador de uma mensagem e permite que este demonstre a um terceiro que o destinatário recebeu a mensagem. Basicamente, o destinatário assina a mensagem original inteira mais a assinatura original (do emissor) e anexa a nova assinatura para formar uma nova mensagem S/MIME.
- **Rótulos de segurança:** Um rótulo de segurança pode estar incluído nos atributos autenticados de um objeto SignedData. Um rótulo de segurança é um conjunto de informações de segurança relativos à importância do conteúdo protegido pelo encapsulamento S/MIME. Os rótulos podem ser usados para controle de acesso, indicando quais usuários têm acesso permitido a um objeto. Outros usos incluem prioridade (secreta, confidencial, restrita etc.) ou baseado em função, descrevendo que tipo de pessoas podem visualizar as informações (por exemplo, a equipe de assistência médica do paciente, agentes de cobrança médica etc.).
- **Listas de correspondência seguras:** Quando um usuário envia uma mensagem a vários destinatários, uma certa quantidade de processamento por destinatário é exigida, incluindo o uso da chave pública de cada destinatário. O usuário pode ser dispensado desse trabalho empregando os serviços de um S/MIME Mail List Agent (MLA). Um MLA pode apanhar uma única mensagem que chega, realizar a criptografia específica para cada destinatário e encaminhar a mensagem. O originador de uma mensagem só precisa enviar a mensagem para o MLA, com a criptografia realizada usando a chave pública do MLA.

Sites Web recomendados

- **PGP Home Page:** site Web do PGP, pela PGP Corp., o principal fornecedor comercial de PGP.
- **International PGP Home Page:** criada para promover o uso mundial do PGP. Contém documentos e links interessantes.
- **MIT Distribution Site for PGP:** principal distribuidor do PGP freeware. Contém FAQs, outras informações e links para outros sites sobre PGP.
- **PGP Charter:** RFCs mais recentes e Internet drafts para Open Specification PGP.
- **S/MIME Charter:** RFCs e Internet drafts mais recentes para S/MIME.

15.3 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

assinatura separada
chave de sessão
confiança
e-mail
Multipurpose Internet Mail Extensions (MIME)
Pretty Good Privacy (PGP)
radix 64
S/MIME
ZIP

Perguntas de revisão

15.1 Quais são os cinco principais serviços fornecidos pelo PGP?
15.2 Qual é a utilidade de uma assinatura separada?
15.3 Por que o PGP gera uma assinatura antes de aplicar a compressão?
15.4 O que é conversão R64?
15.5 Por que a conversão R64 é útil para uma aplicação de e-mail?
15.6 Por que a função de segmentação e remontagem é necessária no PGP?
15.7 Como o PGP usa o conceito de confiança?
15.8 O que é a RFC 822?
15.9 O que é MIME?
15.10 O que é S/MIME?

Problemas

15.1 O PGP utiliza o modo de feedback cifrado (CFB) do CAST-128, enquanto a maioria das aplicações de criptografia simétrica (fora a criptografia de chave) utiliza o modo de encadeamento de cifra de bloco (CBC). Temos

$$\text{CBC:} \quad C_i = \mathrm{E}(K, [C_{i-1} \oplus P_i]); \quad P_i = C_{i-1} \oplus D(K, C_i)$$
$$CFB: \quad C_i = P_i \oplus E(K, C_{i-1}); \quad P_i = C_i \oplus E(K, C_{i-1})$$

Esses dois parecem oferecer a mesma segurança. Sugira um motivo pelo qual o PGP usa o modo CFB.

15.2 No esquema de PGP, qual é o número esperado de chaves de sessão geradas antes que uma chave já criada anteriormente seja produzida?

15.3 No PGP, qual é a probabilidade de que um usuário com N chaves públicas tenha pelo menos uma ID de chave duplicada?

15.4 Os primeiros 16 bits do resumo de mensagem em uma assinatura PGP são lidos às claras.

a. Até que ponto isso compromete a segurança do algoritmo de hash?
b. Até que ponto isso realmente realiza a função pretendida, ou seja, ajudar a determinar se a chave RSA correta foi usada para decriptografar o resumo?

15.5 Na Figura 15.4, cada entrada na lista de chaves públicas contém um campo de confiança do proprietário, que indica o grau de confiança associado ao proprietário dessa chave pública. Por que isso não é suficiente? Ou seja, se esse proprietário for confiável e essa é supostamente a chave pública do proprietário, por que não é um grau de confiança suficiente para permitir que o PGP use essa chave pública?

15.6 Considere a conversão radix 64 como uma forma de criptografia. Nesse caso, não existe chave. Mas suponha que um oponente soubesse apenas que alguma forma de algoritmo de substituição estivesse sendo usada para criptografar o texto em inglês e não soubesse que era R64. Que eficácia esse algoritmo teria contra criptoanálise?

15.7 Phil Zimmermann escolheu IDEA, 3DES com três chaves e CAST-128 como algoritmos de criptografia simétrica para o PGP. Forneça os motivos pelos quais cada um dos algoritmos de criptografia simétrica a seguir, descritos neste livro, são adequados ou não para o PGP: DES, 3DES com duas chaves e AES.

APÊNDICE 15A COMPRESSÃO DE DADOS USANDO ZIP

O PGP utiliza um pacote de compressão chamado ZIP, escrito por Jean-lup Gailly, Mark Adler e Richard Wales. O ZIP é um pacote freeware escrito em C, executado como um utilitário no UNIX e alguns outros sistemas. O ZIP é funcionalmente equivalente ao PKZIP, um pacote shareware muito utilizado em sistemas Windows, desenvolvido pela PKWARE, Inc. O algoritmo zip talvez seja a técnica de compressão mais utilizada entre plataformas; versões freeware e shareware estão disponíveis para Macintosh e outros sistemas, além dos sistemas Windows e UNIX.

O ZIP e algoritmos semelhantes surgiram da pesquisa de Jacob Ziv e Abraham Lempel. Em 1977, eles descreveram uma técnica baseada em um buffer de janela deslizante, que mantém o texto processado mais recentemente [ZIV77]. Esse algoritmo geralmente é conhecido como LZ77. Uma versão desse algoritmo é usada no esquema de compressão zip (PKZIP, gzip, zipit etc.).

O LZ77 e suas variantes exploram o fato de que palavras e frases dentro de um fluxo de texto (padrões de imagem no caso do GIF) provavelmente são repetidas. Quando ocorre uma repetição, a seqüência repetida pode ser substituída por um código curto. O programa de compressão varre essas repetições e desenvolve códigos no ato para substituir a seqüência repetida. Com o tempo, os códigos são reutilizados para capturar novas seqüências. O algoritmo precisa ser definido de modo que o programa de descompressão seja capaz de deduzir o mapeamento atual entre códigos e seqüências de dados de origem.

Antes de examinar os detalhes do LZ77, vejamos um exemplo simples.[4] Considere a frase sem sentido

the brown fox jumped over the brown foxy jumping frog

que possui 53 octetos = 424 bits de extensão. O algoritmo processa esse texto da esquerda para a direita. Inicialmente, cada caractere é mapeado para um padrão de 9 bits consistindo em um binário 1 seguido pela representação ASCII de 8 bits do caractere. À medida que o processamento continua, o algoritmo procura seqüências repetidas. Quando uma repetição é encontrada, o algoritmo continua varrendo até que a repetição termine. Em outras palavras, toda vez que ocorre uma repetição, o algoritmo inclui tantos caracteres quanto possível. A primeira seqüência desse tipo encontrada é **the brown fox**. Essa seqüência é substituída por um ponteiro para a seqüência anterior e o tamanho da seqüência. Nesse caso, a seqüência anterior de **the brown fox** ocorre 26 caracteres antes e o tamanho da seqüência é de 13 caracteres. Para este exemplo, considere duas opções de codificação; um ponteiro de 8 bits e um tamanho de 4 bits, ou um ponteiro de 12 bits e um tamanho de 6 bits; um cabeçalho de 2 bits indica qual opção é escolhida, com 00 indicando a primeira opção e 01 a segunda opção. Assim, a segunda ocorrência de **the brown fox** é codificada como $<00_b><26_d><13_d>$, ou 00 00011010 1101.

As outras partes da mensagem comprimida são a letra **y**; a seqüência $<00_b><27_d><5_d>$, que substitui a seqüência consistindo no caractere de espaço seguido por **jump**; e a seqüência de caracteres **ing frog**.

A Figura 15.9 ilustra o mapeamento de compressão. A mensagem comprimida consiste em 35 caracteres de 9 bits e dois códigos, para um total de $35 \times 9 + 2 \times 14 = 343$ bits. Isso se compara aos 424 bits da mensagem não comprimida, ocasionando uma razão de compressão de 1,24.

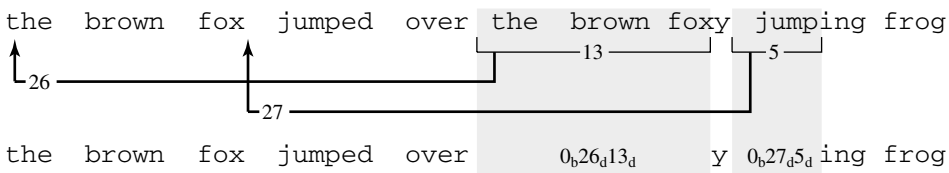


Figura 15.9 Exemplo do esquema LZ77.

Algoritmo de compressão

O algoritmo de compressão para o LZ77 e suas variantes utiliza dois buffers. Um **buffer de histórico deslizante** contém os últimos N caracteres da origem que foram processados e um **buffer look-ahead** contém os próximos L caracteres a serem processados (Figura 15.10a). O algoritmo tenta combinar dois ou mais caracteres do início do buffer look-ahead com uma string no buffer de histórico deslizante. Se nenhuma combinação for encontrada, o primeiro caractere no buffer look-ahead é enviado como um caractere de 9 bits e também é deslocado para a janela deslizante,

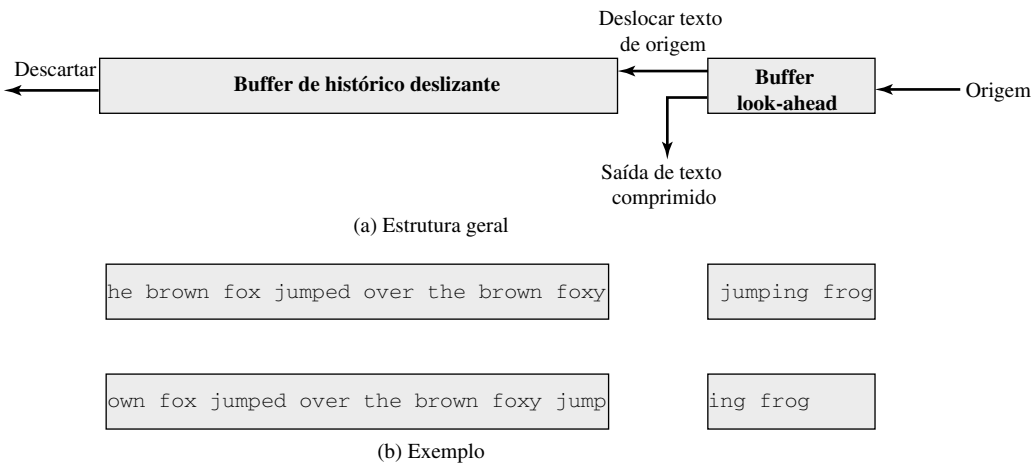


Figura 15.10 Esquema LZ77.

4. Baseado em um exemplo em [WEIS93].

com o caractere mais antigo na janela deslizante sendo deslocado para fora. Se for encontrada uma combinação, o algoritmo continua a varrer em busca da combinação mais longa. Depois, a string combinada é enviada como uma tripla (indicador, ponteiro, tamanho). Para uma string de *K* caracteres, os *K* caracteres mais antigos na janela deslizante são deslocados para fora, e os *K* caracteres da string codificada são deslocados para dentro da janela.

A Figura 15.10b mostra a operação desse esquema em nossa seqüência de exemplo. A ilustração considera uma janela deslizante de 39 caracteres e um buffer look-ahead de 13 caracteres. Na parte superior do exemplo, os primeiros 40 caracteres foram processados e a versão não comprimida dos 39 mais recentes desses caracteres está na janela deslizante. A origem restante está na janela look-ahead. O algoritmo de compressão determina a próxima combinação, desloca 5 caracteres do buffer look-ahead para a janela deslizante e envia o código dessa string para a saída. O estado do buffer após essas operações aparece na parte inferior do exemplo.

Embora o LZ77 seja eficaz e adapte-se à natureza da entrada atual, ele tem algumas desvantagens. O algoritmo usa uma janela finita para procurar combinações no texto anterior. Para um bloco de texto muito longo, em comparação com o tamanho da janela, muitas combinações em potencial são eliminadas. O tamanho da janela pode ser aumentado, mas isso impõe duas penalidades: (1) O tempo de processamento do algoritmo aumenta, pois ele precisa realizar uma comparação de strings em relação ao buffer look-ahead para cada posição na janela deslizante e (2) o campo <ponteiro> precisa ser maior para acomodar os saltos mais longos.

Algoritmo de descompressão

A descompressão do texto comprimido por LZ77 é simples. O algoritmo de descompressão precisa salvar os últimos *N* caracteres da saída descomprimida. Quando uma string codificada é encontrada, o algoritmo de descompressão utiliza os campos <ponteiro> e <tamanho> para substituir o código pela string de texto real.

APÊNDICE 15B CONVERSÃO RADIX64

Tanto o PGP quanto o S/MIME utilizam uma técnica de codificação conhecida como conversão radix64. Essa técnica relaciona a entrada binária arbitrária à saída de caracteres imprimíveis. A forma de codificação tem as seguintes características relevantes:

1. O intervalo da função é um conjunto de caracteres representável universalmente em todos os lugares, e não uma codificação binária específica desse conjunto de caracteres. Assim, os próprios caracteres podem ser codificados em qualquer forma necessária por um sistema específico. Por exemplo, o caractere 'E' é representado em um sistema baseado em ASCII como o hexadecimal 45, e em um sistema baseado em EBCDIC, como o hexadecimal C5.
2. O conjunto de caracteres consiste em 65 caracteres imprimíveis, um dos quais é usado para complementação. Com $2^6 = 64$ caracteres disponíveis, cada caractere pode ser usado para representar 6 bits de entrada.
3. Nenhum caractere de controle é incluído no conjunto. Assim, uma mensagem codificada em radix-64 pode atravessar sistemas de tratamento de correio que varrem o fluxo de dados em busca de caracteres de controle.
4. O caractere hífen ("−") não é usado. Esse caractere tem significado especial no formato RFC 822 e por isso deveria ser evitado.

A Tabela 15.9 mostra o relacionamento entre valores de entrada de 6 bits e caracteres. O conjunto de caracteres consiste em caracteres alfanuméricos mais "+" e "/". O caractere "=" é usado como caractere de complementação.

A Figura 15.11 ilustra o esquema de mapeamento simples. A entrada binária é processada em blocos de 3 octetos, ou 24 bits. Cada conjunto de 6 bits no bloco de 24 bits é mapeado para um caractere. Na figura, os caracteres são mostrados codificados como quantidades de 8 bits. Nesse caso típico, cada entrada de 24 bits é expandida para 32 bits de saída.

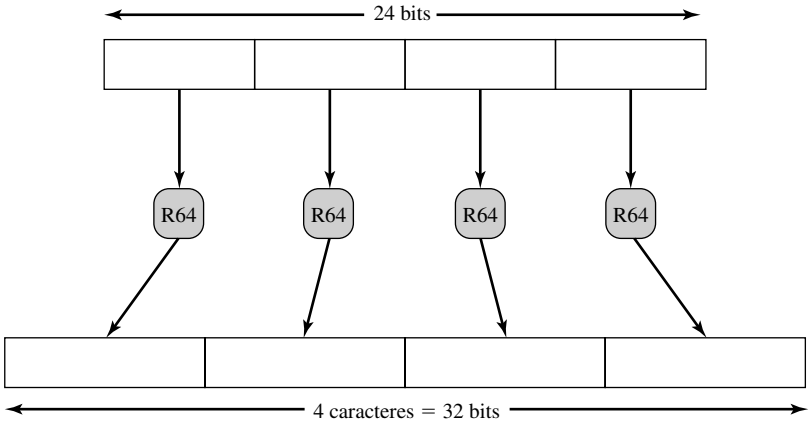


Figura 15.11 Codificação imprimível de dados binários no formato radix-64.

Tabela 15.9 Codificação radix-64

Valor de 6 bits	Codificação de caractere	Valor de 6 bits	Codificação de caractere	Valor de 6 bits	Codificação de caractere	Valor de 6 bits	Codificação de caractere
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(complemetação)	=

Por exemplo, considere a seqüência de texto bruta de 24 bits 00100011 01011100 10010001, que pode ser expressa em hexadecimal como 235C91. Organizamos essa entrada em blocos de 6 bits:

001000 110101 110010 010001

Os valores decimais de 6 bits extraídos são 8, 53, 50, 17. Pesquisando-os na Tabela 15.9, geramos a codificação radix64 como os seguintes caracteres: I1yR. Se esses caracteres forem armazenados no formato ASCII de 8 bits com o bit de paridade marcado como zero, teremos

01001001 00110001 01111001 01010010

Em hexadecimal, isso é 49317952. Para resumir,

Dados de entrada	
Representação binária	00100011 01011100 10010001
Representação hexadecimal	235C91
Codificação radix64 dos dados de entrada	
Representação de caractere	I1yR
Código ASCII (8 bits, paridade zero)	01001001 00110001 01111001 01010010
Representação hexadecimal	49317952

APÊNDICE 15C GERAÇÃO DE NÚMEROS ALEATÓRIOS PGP

O PGP utiliza um esquema complexo e poderoso para gerar números aleatórios e números pseudo-aleatórios, para diversas finalidades. O PGP gera números aleatórios a partir do conteúdo e temporização de toques de tecla do usuário, e números pseudo-aleatórios usando um algoritmo baseado naquele do ANSI X9.17. O PGP utiliza esses números para as seguintes finalidades:

- Números aleatórios verdadeiros:
  — usados para gerar pares de chaves RSA

— fornecem a semente inicial para o gerador de números pseudo-aleatórios
— fornecem entrada adicional durante geração de números pseudo-aleatórios

- Números pseudo-aleatórios:

  — usados para gerar chaves de sessão
  — usados para gerar vetores de inicialização (IVs) para uso com a chave de sessão na criptografia em modo CFB

Números aleatórios verdadeiros

O PGP mantém um buffer de 256 bytes de bits aleatórios. Toda vez que o PGP aguarda por um toque de tecla, ele registra a hora, em formato de 32 bits, em que começa a aguardar. Quando recebe o toque de tecla, ele registra a hora em que a tecla foi pressionada e o valor de 8 bits da tecla. A informação de hora e da tecla é usada para gerar uma chave, que, por sua vez, é usada para criptografar o valor atual do buffer de bits aleatórios.

Números pseudo-aleatórios

A geração de números pseudo-aleatórios utiliza uma semente de 24 octetos e produz uma chave de sessão de 16 octetos, um vetor de inicialização de 8 octetos e uma nova semente a ser usada para a próxima geração de números pseudo-aleatórios. O algoritmo é baseado no algoritmo X9.17 descrito no Capítulo 7 (ver Figura 7.14), mas usa CAST-128 em vez de 3DES para criptografia. O algoritmo usa as seguintes estruturas de dados:

1. Entrada
   — randseed.bin (24 octetos): Se esse arquivo estiver vazio, ele será preenchido com 24 octetos verdadeiramente aleatórios.
   — mensagem: A chave de sessão e o IV (vetor de Inicialização) que serão usados para criptografar uma mensagem são uma função dessa mensagem. Isso contribui ainda mais para a aleatoriedade da chave e do IV, e se um oponente já souber o conteúdo de texto claro da mensagem, não haverá necessidade aparente de obter a chave de sessão de uso único.

2. Saída

   — K (24 octetos): Os primeiros 16 octetos, K[0..15], contêm uma chave de sessão, e os oito últimos octetos, K[16..23], contêm um IV.
   — randseed.bin (24 octetos): Um novo valor de semente é colocado nesse arquivo.

3. Estruturas de dados internas

   — dtbuf (8 octetos): Os 4 primeiros octetos, dtbuf[0..3], são inicializados com o valor atual de data/hora. Esse buffer é equivalente à variável DT no algoritmo X12.17.
   — rkey (16 octetos): Chave de criptografia CAST-128 usada em todos os estágios do algoritmo.
   — rseed (8 octetos): Equivalente à variável V_i do X12.17.
   — rbuf (8 octetos): Um número pseudo-aleatório gerado pelo algoritmo. Esse buffer é equivalente à variável R_i do X12.17.
   — K′ (24 octetos): Buffer temporário para o novo valor de randseed.bin.

O algoritmo consiste em nove etapas, G1 a G9. A primeira e última etapas são etapas de ofuscamento, que servem para reduzir o valor de um arquivo randseed.bin obtido por um oponente. As etapas restantes são basicamente equivalentes às três iterações do algoritmo X12.17 e são ilustradas na Figura 15.12 (compare a Figura 7.14). Resumindo,

G1. [Pré-lavagem de semente anterior]

a. Copie randseed.bin para K[0..23].

b. Apanhe o hash da mensagem (esse já foi gerado se a mensagem estiver sendo assinada; se não, os primeiros 4K octetos da mensagem são usados). Use o resultado como chave, use um IV nulo e criptografe K no modo CFB; armazene o resultado de volta em K.

G2. [Defina a semente inicial]

a. Defina dtbuf[0..3] como a hora local com 32 bits. Defina dtbuf[4..7] como zeros. Copie rkey ← K[0..15]. Copie rseed ← K[16..23].

b. Criptografe o dtbug de 64 bits usando a rkey de 128 bits no modo ECB; armazene o resultado novamente em dtbuf.

G3. [Prepare-se para gerar octetos aleatórios] Defina rcount ← 0 e k ← 23. O loop das etapas G4-G7 será executado 24 vezes (k = 23...0), uma para cada octeto aleatório produzido e colocado em K. A variável rcount é o número de octetos aleatórios não usados em rbuf. Ele contará de 8 até 0 três vezes para gerar os 24 octetos.

G4. [Bytes disponíveis?] Se rcount = 0 vá para G5, senão, vá para G7. Etapas G5 e G6 realizam uma instância do algoritmo X12.17 para gerar um novo batch de oito octetos aleatórios.

G5. [Gere novos octetos aleatórios]

a. rseed ← rseed ⊕ dtbuf
b. rbuf ← E_{rkey}[rseed] no modo ECB

G6. [Gere próxima semente]

a. rseed ← rbuf ⊕ dtbuf
b. rseed ← E_{rkey}[rseed] no modo ECB
c. Defina rcount ← 8

G7. [Transfira um byte de cada vez de rbuf para K]

a. Defina rcount ← rcount - 1
b. Gere um byte aleatório verdadeiro b, e defina K[k] ß rbuf[rcount] ⊕ b

G8. [Pronto?] Se k = 0, vá para G9, senão, defina k ß k - 1 e vá para G4

G9. [Pós-lavagem da semente e retorno do resultado]

a. Gere mais 24 bytes pelo método das etapas G4-G7,mas não faça o XOR com um byte aleatório em G7. Coloque o resultado no buffer K'
b. Criptografe K' com a chave K[0..15] e IV K[16..23] no modo CFB; armazene o resultado em randseed.bin
c. Retorne K

Não deveria ser possível determinar a chave de sessão a partir dos 24 novos octetos gerados na etapa G9.a. Porém, para certificar-se de que o arquivo randseed.bin armazenado não oferece informações sobre a chave de sessão mais recente, os 24 novos octetos são criptografados e o resultado é armazenado como a nova semente.

Esse algoritmo complexo deverá oferecer números pseudo-aleatórios criptograficamente fortes.

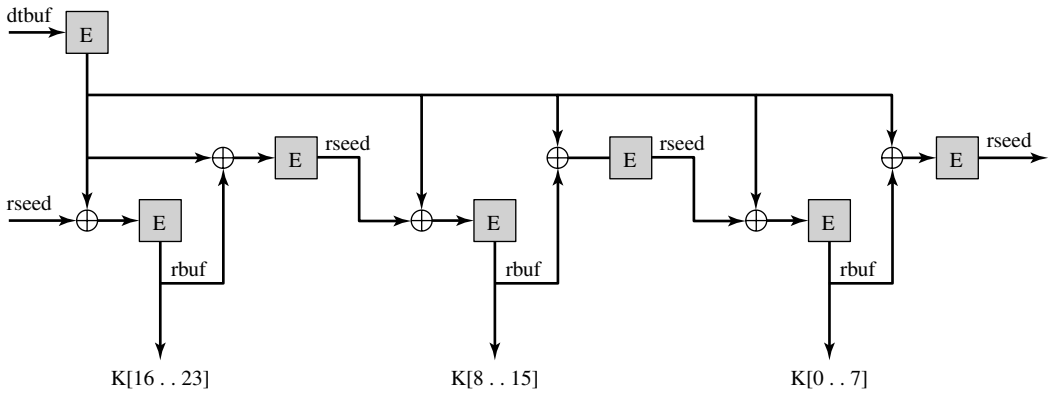


Figura 15.12 Chave de sessão PGP e geração do IV (etapas G2 a G8).

Segurança de IP

16.1 **Visão geral da segurança de IP**
16.2 **Arquitetura da segurança de IP**
16.3 **Cabeçalho de autenticação**
16.4 **Encapsulamento de segurança do payload**
16.5 **Combinando associações de segurança**
16.6 **Gerenciamento de chaves**
16.7 **Leitura e sites Web recomendados**
16.8 **Principais termos, perguntas de revisão e problemas**
Apêndice 16A Interligação de redes e protocolos de Internet

Se uma notícia secreta é divulgada por um espião antes da hora certa, ele precisa ser morto, juntamente com o homem a quem o segredo foi dito.

— *A arte da guerra*, Sun Tzu

PONTOS PRINCIPAIS

- A segurança de IP (IPSec) é uma capacidade que pode ser acrescentada a qualquer versão atual do Internet Protocol (IPv4 ou IPv6), por meio de cabeçalhos adicionais.
- O IPSec compreende três áreas funcionais: autenticação, confidencialidade e gerenciamento de chaves.
- A autenticação utiliza o código de autenticação de mensagens HMAC. A autenticação pode ser aplicada a todo o pacote de IP original (modo túnel) ou a todo o pacote exceto o cabeçalho de IP (modo transporte).
- A confidencialidade é fornecida por um formato de criptografia conhecido como encapsulamento de segurança do payload. Os modos túnel e transporte podem ser acomodados.
- O IPSec define diversas técnicas para gerenciamento de chaves.

A comunidade da Internet tem desenvolvido mecanismos de segurança específicos em diversas áreas de aplicação, incluindo e-mail (S/MIME, PGP), cliente/servidor (Kerberos), acesso à Web (Secure Sockets Layer) e outros. Porém, os usuários têm questões de segurança que transcendem as camadas de protocolos. Por exemplo, uma empresa pode possuir uma rede TCP/IP segura, privada, desativando as conexões com sites não confiáveis, criptografando pacotes que saem das instalações e autenticando pacotes que entram nas instalações. Implementando a segurança no nível do IP, uma organização pode garantir uma rede segura não apenas para aplicações que possuem mecanismos de segurança, mas também para as muitas aplicações que ignoram a segurança.

A segurança no nível do IP compreende três áreas funcionais: autenticação, confidencialidade e gerenciamento de chaves. O mecanismo de autenticação garante que um pacote recebido foi, realmente, transmitido pela parte identificada como origem no cabeçalho do pacote. Além disso, esse mecanismo garante que o pacote não foi alterado em trânsito. O recurso de confidencialidade permite que os nós que se comunicam criptografem mensagens para impedir interceptação por terceiros. O recurso de gerenciamento de chaves está relacionado à troca segura de chaves.

Começamos este capítulo com uma visão geral da segurança de IP (IPSec) e com uma introdução à arquitetura IPSec. Depois, examinamos cada uma das três áreas funcionais detalhadamente. O apêndice deste capítulo revisa os protocolos de Internet.

16.1 VISÃO GERAL DA SEGURANÇA DE IP

Em resposta a essas questões, o IAB (Internet Architecture Board) incluiu autenticação e criptografia como recursos de segurança necessários no IP da próxima geração, que tem sido anunciada como IPv6. Felizmente, essas capacidades de segurança foram projetadas para que possam ser usadas com o IPv4 atual e com o IPv6 futuro. Isso significa que os fornecedores podem começar a oferecer esses recursos agora, e muitos deles agora já possuem capacidade IPSec em seus produtos.

Aplicações do IPSec

O IPSec oferece a capacidade de proteger comunicações por uma LAN, por WANs privadas e públicas e pela Internet. Alguns exemplos de seu uso incluem o seguinte:

- **Conectividade segura do escritório pela Internet:** Uma empresa pode montar uma rede privada virtual segura pela Internet ou por uma WAN pública. Isso permite que uma empresa confie bastante na Internet e reduza sua necessidade de redes privadas, economizando custos e overhead de gerenciamento de rede.
- **Acesso remoto seguro pela Internet:** Um usuário final cujo sistema é equipado com protocolos de segurança de IP pode fazer uma ligação local para um provedor de serviços de Internet (ISP) e obter acesso seguro à rede de uma empresa. Isso reduz o custo das tarifas de ligações para funcionários que trabalham viajando.
- **Estabelecimento de conectividade de extranet e intranet com parceiros:** O IPSec pode ser usado para proteger a comunicação com outras organizações, garantindo a autenticação e a confidencialidade, além de fornecer um mecanismo de troca de chaves.
- **Aprimoramento da segurança do e-commerce:** Embora algumas aplicações de e-commerce na Web tenham protocolos de segurança embutidos, o uso do IPSec aumenta essa segurança.

A característica principal do IPSec que permite dar suporte a essas aplicações variadas é que ele pode criptografar e/ou autenticar todo o tráfego no nível de IP. Assim, todas as aplicações distribuídas, incluindo logon remoto, cliente/servidor, e-mail, transferência de arquivos, acesso à Web etc., podem ser protegidas.

A Figura 16.1 é um cenário típico do uso do IPSec. Uma organização mantém LANs em locais dispersos. O tráfego de IP não protegido é conduzido em cada LAN. Para o tráfego fora do site, através de algum tipo de WAN privada ou pública, são utilizados protocolos IPSec. Esses protocolos operam em dispositivos de rede, como um roteador ou firewall, que conectam cada LAN ao mundo exterior. O dispositivo de rede IPSec normalmente irá criptografará e compactará todo o tráfego que vai para a WAN e decriptografará e descompactará o tráfego que vem da WAN; essas operações são transparentes às estações de trabalho e servidores na LAN. A transmissão segura também é possível com usuários individuais que discam para a WAN. Essas estações de trabalho do usuário precisam implementar protocolos IPSec para fornecer segurança.

Benefícios do IPSec

[MARK97] lista os seguintes benefícios do IPSec:

- Quando o IPSec é implementado em um firewall ou roteador, ele oferece segurança forte, que pode ser aplicada a todo o tráfego cruzando o perímetro. O tráfego dentro de uma empresa ou grupo de trabalho não gera o overhead do processamento relacionado à segurança.
- O IPSec em um firewall é resistente ao *bypass* se todo o tráfego vindo de fora tiver de usar IP, e o firewall for o único meio de entrada da Internet para a organização.

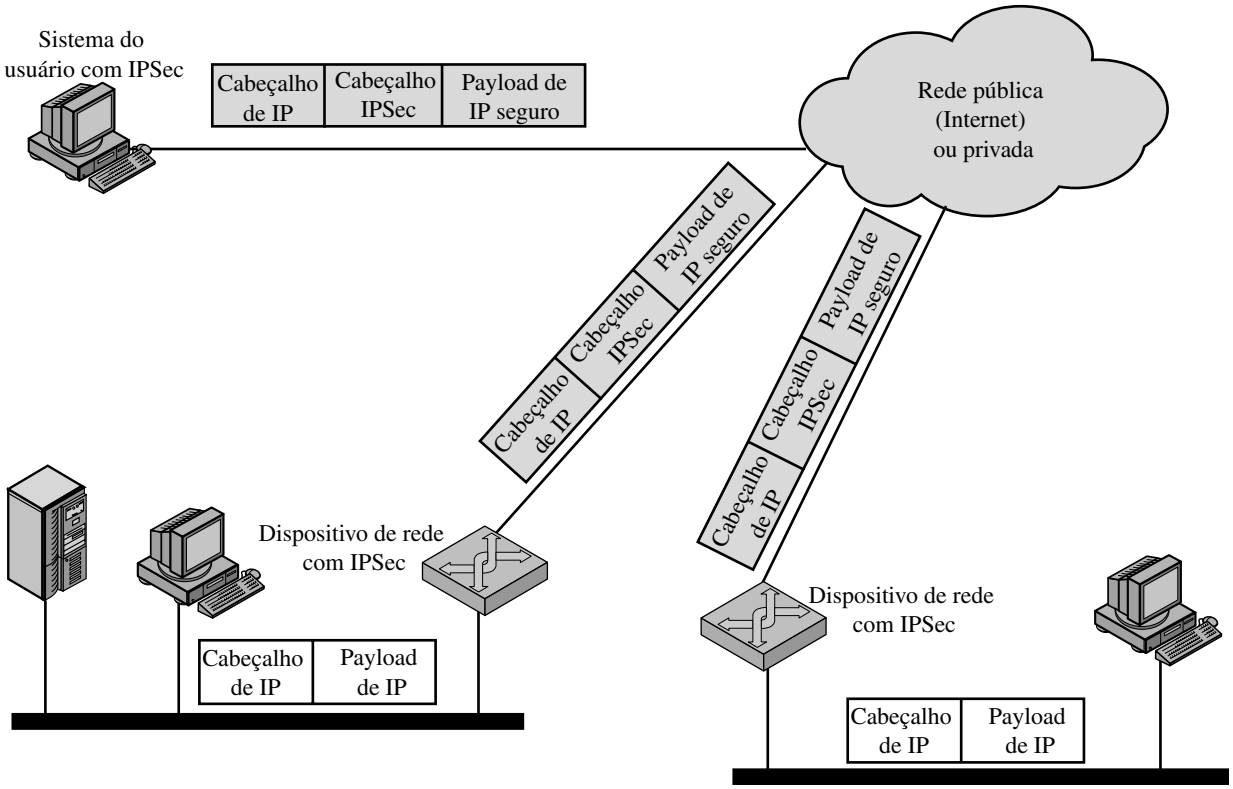


Figura 16.1 Um cenário de segurança de IP.

- O IPSec está abaixo da camada de transporte (TCP, UDP) e, por isso, é transparente às aplicações. Não há necessidade de mudar o software em um sistema do usuário ou servidor quando o IPSec for implementado no firewall ou roteador. Mesmo que o IPSec seja implementado nos sistemas finais, o software da camada superior, incluindo as aplicações, não é afetado.
- O IPSec pode ser transparente aos usuários finais. Não há necessidade de treinar usuários em mecanismos de segurança, emitir chaves para cada usuário, ou revogar esse material quando os usuários deixarem a organização.
- O IPSec pode oferecer segurança a usuários individuais, se for necessário. Isso é útil para trabalhadores externos e para configurar uma sub-rede virtual segura dentro de uma organização para aplicações importantes.

Aplicações de roteamento

Além de dar suporte a usuários finais e proteger sistemas e redes, o IPSec pode desempenhar um papel vital na arquitetura de roteamento exigida para interligação de redes. [HUIT98] lista os seguintes exemplos do uso do IPSec. O IPSec pode garantir que

- Um anúncio de roteador (um novo roteador anuncia sua presença) vem de um roteador autorizado.
- Um anúncio de vizinho (um roteador procura estabelecer ou manter um relacionamento de vizinhança com um roteador em outro domínio de roteamento) vem de um roteador autorizado.
- Uma mensagem de redirecionamento vem de um roteador ao qual o pacote inicial foi enviado.
- Uma atualização de roteamento não foi falsificada.

Sem essas medidas de segurança, um oponente pode interromper as comunicações ou desviar parte do tráfego. Protocolos de roteamento como OSPF devem ser executados sobre as associações de segurança entre roteadores definidos pelo IPSec.

16.2 ARQUITETURA DA SEGURANÇA DE IP

A especificação IPSec tornou-se bastante complexa. Para se ter uma idéia da arquitetura geral, começamos com uma visão dos documentos que definem o IPSec. Depois, explicamos os serviços IPSec e apresentamos o conceito de associação de segurança.

Documentos IPSec

A especificação IPSec consiste em diversos documentos. Os mais importantes deles, emitidos em novembro de 1998, são as RFCs 2401, 2402, 2406 e 2408:

- RFC 2401: Uma visão geral de uma arquitetura de segurança
- RFC 2402: Descrição de uma extensão de autenticação de pacote para IPv4 e IPv6
- RFC 2406: Descrição de uma extensão de criptografia de pacote para IPv4 e IPv6
- RFC 2408: Especificação das capacidades de gerenciamento de chaves

O suporte a esses recursos é obrigatório para IPv6 e opcional para IPv4. Nos dois casos, as medidas de segurança são implementadas como cabeçalhos de extensão que vêm após o cabeçalho de IP principal. O cabeçalho de extensão para autenticação é conhecido como cabeçalho Authentication, que para a criptografia é conhecido como cabeçalho de *encapsulamento de segurança do payload* (ESP).

Além dessas quatro RFCs, diversos rascunhos adicionais foram publicados pelo IP Security Protocol Working Group estabelecido pelo IETF. Os documentos estão divididos em sete grupos, conforme representados na Figura 16.2 (RFC 2401):

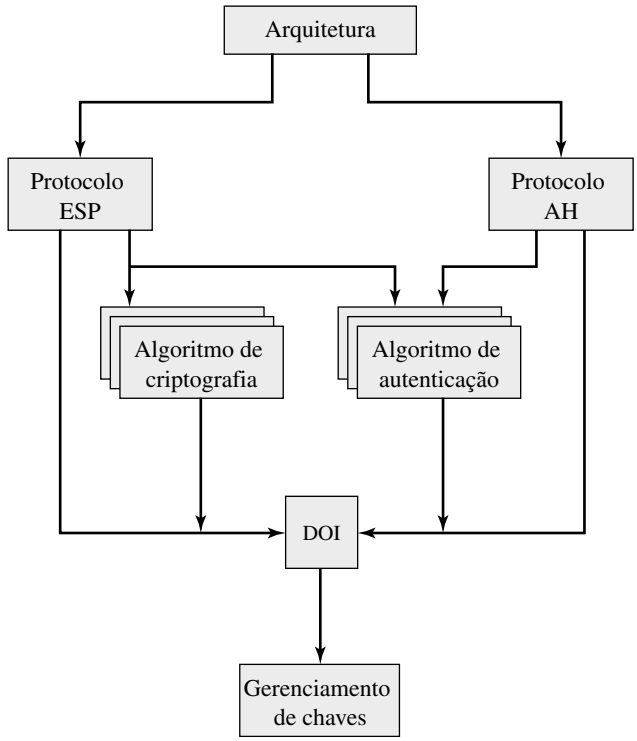


Figura 16.2 Visão geral do documento IPSec.

- **Arquitetura:** Abrange os conceitos gerais, requisitos de segurança, definições e mecanismos definindo a tecnologia IPSec.
- **Encapsulamento de segurança do payload (Encapsulating Security Payload) — ESP):** Abrange o formato de pacote e questões gerais relacionadas ao uso do ESP para criptografia de pacote e, opcionalmente, autenticação.
- **Autenticação do cabeçalho (Authentication Header — AH):** Abrange o formato de pacote e questões gerais relacionadas ao uso do AH para autenticação de pacotes.
- **Algoritmo de criptografia:** Um conjunto de documentos que descrevem como diversos algoritmos de criptografia são usados para ESP.
- **Algoritmo de autenticação:** Um conjunto de documentos que descrevem como vários algoritmos de autenticação são usados para AH e para a opção de autenticação do ESP.
- **Gerenciamento de chaves:** Documentos que descrevem esquemas de gerenciamento de chaves.
- **Domínio de interpretação (DOI):** Contém valores necessários para os outros documentos se relacionarem entre si. Estes incluem identificadores para algoritmos aprovados de criptografia e autenticação, além de parâmetros operacionais, como tempo de vida da chave.

Serviços IPSec

O IPSec oferece serviços de segurança na camada de IP permitindo que um sistema selecione protocolos de segurança exigidos, determine o(s) algoritmo(s) a ser(em) usado(s) para o(s) serviço(s) e coloque no lugar quaisquer chaves criptográficas exigidas para oferecer os serviços solicitados. Dois protocolos são usados para oferecer segurança: um protocolo de autenticação designado pelo cabeçalho do protocolo, autenticação do cabeçalho (AH) e um protocolo combinado de criptografia/autenticação, designado pelo formato do pacote para esse protocolo, encapsulamento de segurança do payload (ESP). Os serviços são

- Controle de acesso
- Integridade sem conexão
- Autenticação da origem de dados
- Rejeição de pacotes repetidos (uma forma de integridade de seqüência parcial)
- Confidencialidade (criptografia)
- Confidencialidade limitada de fluxo de tráfego

A Tabela 16.1 mostra quais serviços são fornecidos pelos protocolos AH e ESP. Para ESP, existem dois casos: com e sem a opção de autenticação. Tanto AH quanto ESP são veículos para controle de acesso baseados na distribuição de chaves criptográficas e no gerenciamento de fluxos de tráfego relativos a esses protocolos de segurança.

Tabela 16.1 Serviços IPSec

	AH	ESP (apenas criptografia)	ESP (criptografia mais autenticação)
Controle de acesso	✔	✔	✔
Integridade sem conexão	✔		✔
Autenticação da origem de dados	✔		✔
Rejeição de pacotes repetidos	✔	✔	✔
Confidencialidade		✔	✔
Confidencialidade limitada do fluxo de tráfego		✔	✔

Associações de segurança

Um conceito importante que aparece nos mecanismos de autenticação e confidencialidade para IP é a associação de segurança (SA – Security Association). Uma associação é um relacionamento unidirecional entre o emissor e um receptor, que fornece serviços de segurança ao tráfego transportado nele. Se um relacionamento entre pares for necessário, para a troca segura bidirecional, então duas associações de segurança serão exigidas. Os serviços de segurança são concedidos a uma SA para o uso de AH ou ESP, mas não ambos.

Uma associação de segurança é identificada exclusivamente por três parâmetros:

- **Índice de parâmetros de segurança (Security Parameters Index — SPI):** Uma seqüência de bits atribuída a essa SA com significado apenas local. O SPI é transportado nos cabeçalhos AH e ESP para permitir que o sistema receptor selecione a SA sob a qual um pacote recebido será processado.
- **Endereço de IP de destino:** Atualmente, apenas endereços unicast são permitidos; esse é o endereço da extremidade de destino da SA, que pode ser um sistema de usuário final ou um sistema de rede, como um firewall ou roteador.
- **Identificador de protocolo de segurança (Security Protocol Identifier):** Isso indica se a associação é uma associação de segurança AH ou ESP.

Logo, em qualquer pacote de IP,[1] a associação de segurança é identificada exclusivamente pelo endereço de destino no cabeçalho IPv4 ou IPv6 e pelo SPI no cabeçalho de extensão incluído (AH ou ESP).

Parâmetros da SA Em cada implementação IPSec, existe um banco de dados de associação de segurança mínimo,[2] que define os parâmetros associados a cada SA. Uma associação de segurança normalmente é definida pelos seguintes parâmetros:

1. Neste capítulo, o termo pacote IP refere-se a um datagrama IPv4 ou a um pacote IPv6.
2. Mínimo no sentido de que a funcionalidade fornecida por um banco de dados de associação de segurança precisa estar presente em qualquer implementação IPSec, mas o modo como essa funcionalidade é fornecida fica a critério do implementador.

- **Contador de números de seqüências:** Um valor de 32 bits usado para gerar o campo de número de seqüência nos cabeçalhos AH ou ESP, descritos na Seção 16.3 (exigido para todas as implementações).
- **Estouro do contador de seqüência:** Um flag indicando se o estouro do contador de números de seqüência deverá gerar um evento auditável e impedir outra transmissão de pacotes nessa SA (exigido para todas as implementações).
- **Janela anti-repetição:** Usada para determinar se um pacote AH ou ESP que chega é uma replicação, descrita na Seção 16.3 (exigido para todas as implementações).
- **Informação de AH:** Algoritmo de autenticação, chaves, tempos de vida de chave e parâmetros relacionados sendo usados com AH (exigido para implementações de AH).
- **Informações de ESP:** Algoritmo de criptografia e autenticação, chaves, valores de inicialização, tempos de vida da chave e parâmetros relacionados sendo usados com ESP (exigido para implementações de ESP).
- **Tempo de vida da associação de segurança:** Um intervalo de tempo ou contagem de bytes após a qual uma SA precisa ser substituída por uma nova SA (e novo SPI) ou encerrada, mais uma indicação de quais dessas ações deverão ocorrer (exigido para todas as implementações).
- **Modo do protocolo IPSec:** Túnel, transporte ou curinga (exigido para todas as implementações). Esses modos são explicados mais adiante nesta seção.
- **Caminho MTU:** Qualquer unidade de transmissão máxima do caminho (comprimento máximo de um pacote que pode ser transmitido sem fragmentação) e variáveis de envelhecimento (exigidas para todas as implementações).

O mecanismo de gerenciamento de chaves usado para distribuir chaves está ligado aos mecanismos de autenticação e privacidade apenas por meio dos índices de parâmetros de segurança. Logo, a autenticação e a privacidade foram especificadas independentemente de qualquer mecanismo específico de gerenciamento de chaves.

Seletores de SA O IPSec oferece ao usuário uma flexibilidade considerável no modo como os serviços IPSec são aplicados ao tráfego IP. Conforme veremos mais adiante, as SAs podem ser combinadas de diversas maneiras para gerar a configuração de usuário desejada. Além disso, o IPSec oferece um alto grau de granularidade na discriminação entre o tráfego que recebe proteção IPSec e o tráfego que tem permissão para contornar o IPSec, no caso anterior relativo ao tráfego de IP para SAs específicas.

O meio pelo qual o tráfego de IP está relacionado a SAs específicas (ou nenhuma SA, caso o tráfego tenha permissão para evitar o IPSec) é o banco de dados de políticas de segurança (Security Policy Database — SPD). Em sua forma mais simples, um SPD contém entradas, cada uma das quais define um subconjunto do tráfego de IP, apontando para uma SA para esse tráfego. Em ambientes mais complexos, podem existir várias entradas que potencialmente se relacionam a uma única SA ou a várias SAs associadas a uma única entrada no SPD. O leitor poderá consultar os documentos IPSec relevantes para obter uma explicação completa.

Cada entrada no SPD é definida por um conjunto de valores de campo de protocolo IP e de camadas superiores, chamados de *seletores*. Na verdade, esses seletores são usados para filtrar o tráfego de saída a fim de mapeá-lo para determinada SA. O processamento de saída obedece à seguinte seqüência geral para cada pacote de IP:

1. Comparar os valores dos campos apropriados no pacote (os campos seletores) com o SPD para encontrar uma entrada do SPD correspondente, que apontará para zero ou mais SAs.
2. Determinar a SA, se houver, para esses pacote e seu SPI associado.
3. Fazer o processamento IPSec necessário (ou seja, processamento de AH ou ESP).

Os seletores a seguir determinam uma entrada do SPD:

- **Endereço de IP de destino:** Este pode ser um único endereço de IP, uma lista enumerada ou intervalo de endereços, ou um endereço curinga (máscara). Os dois últimos são exigidos para oferecer suporte a mais de um sistema de destino compartilhando a mesma SA (por exemplo, por trás de um firewall).
- **Endereço de IP de origem:** Este pode ser um único endereço de IP, uma lista enumerada ou intervalo de endereços, ou um endereço de curinga (máscara). Os dois últimos são exigidos para oferecer suporte a mais de um sistema de origem compartilhando a mesma SA (por exemplo, por trás de um firewall).
- **UserID:** Um identificador de usuário do sistema operacional. Esse não é um campo no IP ou em cabeçalhos da camada superior, mas estará disponível se o IPSec estiver sendo executado no mesmo sistema operacional do usuário.
- **Nível de importância dos dados:** Usado para sistemas que fornecem segurança do fluxo de informações (por exemplo, secreta ou não confidencial).
- **Protocolo da camada de transporte:** Obtido a partir do campo Protocolo do IPv4 ou Próximo Cabeçalho do IPv6. Esse pode ser um número de protocolo individual, uma lista de números de protocolo ou um intervalo de números de protocolo.
- **Portas de origem e destino:** Estas podem ser valores de porta TCP ou UDP individuais, uma lista enumerada de portas ou uma porta curinga.

Modos transporte e túnel

Tanto AH quanto ESP admitem dois modos de uso: modo transporte e túnel. A operação desses dois modos é mais bem entendida no contexto de uma descrição de AH e ESP, que são explicados nas Seções 16.3 e 16.4, respectivamente. Aqui, oferecemos uma breve visão geral.

Modo transporte O modo transporte oferece proteção principalmente para os protocolos da camada superior, isto é, a proteção no modo transporte se estende para o payload de um pacote de IP. Exemplos incluem um segmento TCP ou UDP ou um pacote ICMP, todos operando diretamente acima do IP em uma pilha de protocolo do host. Normalmente, o modo transporte é usado para a comunicação de ponta a ponta entre dois hosts (por exemplo, um cliente e um servidor, ou duas estações de trabalho). Quando um host executa AH ou ESP sobre IPv4, o payload são os dados que normalmente vêm após o cabeçalho de IP. Para o IPv6, o payload são os dados que normalmente vêm após o cabeçalho IP e quaisquer extensões do cabeçalho IPv6 que estejam presentes, com a possível exceção das opções de destino do cabeçalho, que podem ser incluídas na proteção.

O ESP no modo transporte criptografa e, opcionalmente, autentica o payload de IP mas não o cabeçalho de IP. AH no modo transporte autentica o payload de IP e partes selecionadas do cabeçalho de IP.

Modo túnel O modo túnel oferece proteção a todo o pacote de IP. Para conseguir isso, depois que os campos AH ou ESP são acrescentados ao pacote de IP, o pacote inteiro mais os campos de segurança são tratados como o payload do novo pacote de IP 'externo', com um novo cabeçalho de IP externo. Todo o pacote original, ou interno, viaja por um 'túnel' de um ponto de uma rede de IP para outro; nenhum roteador ao longo do caminho é capaz de examinar o cabeçalho de IP interno. Como o pacote original é encapsulado, o novo pacote maior pode ter endereços de origem e destino totalmente diferentes, aumentando a segurança. O modo túnel é usado quando uma ou ambas as extremidades de uma SA são um gateway de segurança, como um firewall ou roteador que implementa o IPSec. Com o modo túnel, diversos hosts nas redes por trás de firewalls podem se empenhar em comunicações seguras sem implementar o IPSec. Os pacotes desprotegidos gerados por tais hosts são colocados em túneis para redes externas por SAs no modo túnel, preparadas por software de IPSec no firewall ou roteador seguro no limite da rede local.

Aqui está um exemplo de como o IPSec no modo túnel opera. O host A em uma rede gera um pacote de IP com o endereço de destino do host B em outra rede. Esse pacote é roteado do host de origem para um firewall ou roteador seguro na borda da rede de A. O firewall filtra todos os pacotes que saem para determinar a necessidade de processamento do IPSec. Se esse pacote de A para B exigir o IPSec, o firewall realizará o processamento do IPSec e encapsulará o pacote com um cabeçalho de IP externo. O endereço de IP de origem desse pacote de IP externo é esse firewall, e o endereço de destino pode ser um firewall que forma o limite para a rede local B. Esse pacote agora é roteado para o firewall de B, com roteadores intermediários examinando apenas o cabeçalho de IP externo. No firewall de B, o cabeçalho de IP externo é removido, e o pacote inteiro é entregue a B.

O ESP no modo túnel criptografa e, opcionalmente, autentica todo o pacote de IP interno, incluindo o cabeçalho de IP interno. AH no modo túnel autentica todo o pacote de IP interno e partes selecionadas do cabeçalho de IP externo.

A Tabela 16.2 Resume a funcionalidade do modo transporte e túnel.

Tabela 16.2 Funcionalidade do modo túnel e do modo transporte

	SA do modo túnel	SA do modo transporte
AH	Autentica todo o pacote de IP interno (cabeçalho interno mais payload de IP) mais partes selecionadas do cabeçalho de IP externo e cabeçalhos de extensão IPv6 externos.	Autentica o payload de IP e partes selecionadas do cabeçalho de IP e cabeçalhos de extensão IPv6.
ESP	Criptografa todo o pacote de IP interno.	Criptografa o payload de IP e quaisquer cabeçalhos de extensão IPv6 após o cabeçalho ESP.
ESP com autenticação	Criptografa todo o pacote de IP interno. Autentica o pacote de IP interno.	Criptografa o payload de IP e quaisquer cabeçalhos de extensão IPv6 após o cabeçalho ESP. Autentica o payload de IP, mas não o cabeçalho de IP.

16.3 CABEÇALHO DE AUTENTICAÇÃO

O cabeçalho de autenticação oferece suporte para integridade de dados e autenticação dos pacotes de IP. O recurso de integridade de dados garante que a modificação não detectada do conteúdo de um pacote em trânsito não seja possível. O recurso de autenticação permite que um sistema final ou dispositivo de rede autentique o usuário ou a aplicação e filtre o tráfego adequadamente; isso também impede ataques de falsificação de endereço, observados na Internet atualmente. O AH também protege contra o ataque por repetição descrito mais adiante nesta seção.

A autenticação baseia-se no uso de um código de autenticação de mensagem (MAC — Message Authentication Code), conforme descrito no Capítulo 11, e por isso as duas partes precisam compartilhar uma chave secreta.

O cabeçalho de autenticação consiste dos seguintes campos (Figura 16.3):

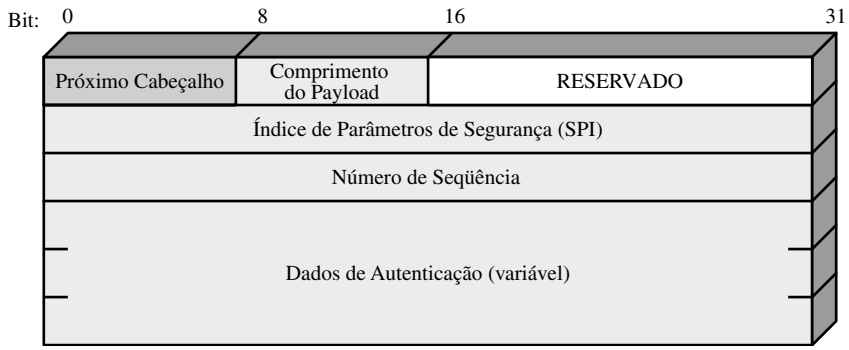


Figura 16.3 Cabeçalho de autenticação IPSec.

- **Próximo cabeçalho (8 bits):** Identifica o tipo de cabeçalho imediatamente seguinte a este cabeçalho.
- **Comprimento do payload (8 bits):** Comprimento do cabeçalho de autenticação em palavras de 32 bits menos 2. Por exemplo, o comprimento padrão do campo de dados de autenticação é de 96 bits, ou três palavras de 32 bits. Com um cabeçalho fixo de três palavras, existe um total de seis palavras no cabeçalho, e o campo de comprimento do payload tem um valor 4.
- **Reservado (16 bits):** Para uso futuro.
- **Índice de parâmetros de segurança (32 bits):** Identifica uma associação de segurança.
- **Número de seqüência (32 bits):** Um valor de contador aumentando monotonicamente, explicado mais adiante.
- **Dados de autenticação (variável):** Um campo de comprimento variável (precisa ser um número inteiro de palavras de 32 bits) que contém o valor de verificação de integridade (ICV — Integrity Check Value), ou MAC, para esse pacote, explicado mais adiante.

Serviço anti-repetição

Um ataque por repetição é aquele em que um atacante obtém uma cópia de um pacote autenticado e mais tarde o transmite para o destino pretendido. O recebimento duplicado de pacotes de IP autenticados pode perturbar o serviço de alguma maneira ou pode ter alguma outra conseqüência indesejada. O campo Número de Seqüência é projetado para evitar esses ataques. Primeiramente, explicamos a geração de números de seqüência pelo emissor e depois examinamos como eles são processado pelo destinatário.

Quando uma nova SA é estabelecida, o **emissor** inicializa um contador de números de seqüência em 0. Toda vez que um pacote for enviado para essa SA, o emissor incrementará o contador e colocará o valor no campo de Número de Seqüência. Assim, o primeiro valor a ser usado será 1. Se a anti-repetição estiver ativada (o padrão), o emissor não deverá permitir que o número de seqüência para ciclo passe de $2^{32} - 1$ de volta a zero. Caso contrário, haveria vários pacotes válidos com o mesmo número de seqüência. Se o limite de $2^{32} - 1$ for alcançado, o emissor deverá terminar essa SA e negociar uma nova SA com uma nova chave.

Como o IP é um serviço sem conexão, não confiável, o protocolo não garante que os pacotes serão entregues na ordem e não garante que todos os pacotes serão entregues. Portanto, o documento de autenticação IPSec dita que o receptor deverá implementar uma janela de comprimento W, com um padrão de $W = 64$. A borda direita da janela representa o número de seqüência mais alto, N, até aqui recebido para um pacote válido. Para qualquer pacote com um número de seqüência no intervalo de $N - W + 1$ até N que tenha sido recebido corretamente (ou seja, devidamente autenticado), o slot correspondente na janela é marcado (Figura 16.4). O processamento de entrada quando um pacote é recebido prossegue da seguinte maneira:

1. Se o pacote recebido estiver dentro da janela e for novo, o MAC será verificado. Se o pacote for autenticado, o slot correspondente na janela será marcado.
2. Se o pacote recebido estiver à direita da janela e for novo, o MAC será verificado. Se o pacote for autenticado, a janela será avançada de modo que esse número de seqüência seja a borda direita da janela, e o slot correspondente na janela será marcado.
3. Se o pacote recebido estiver à esquerda da janela, ou se a autenticação falhar, o pacote será descartado; esse é um evento auditável.

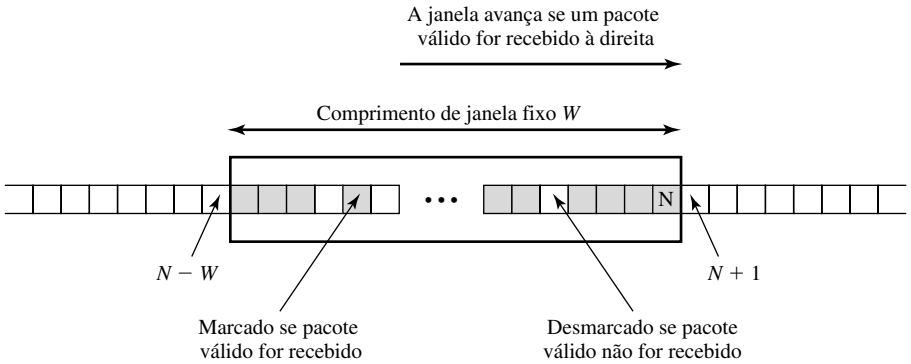


Figura 16.4 Mecanismo anti-repetição.

Valor de verificação de integridade

O campo Dados de Autenticação mantém um valor chamado valor de verificação de integridade (ICV —Integrity Check Value). O ICV é um código de autenticação de mensagem ou uma versão truncada de um código produzido por um algoritmo MAC. A especificação atual dita que uma implementação compatível precisa ter suporte para

- HMAC-MD5-96
- HMAC-SHA-1-96

Ambos utilizam o algoritmo HMAC, o primeiro com o código de hash MD5 e o segundo com o código de hash SHA-1 (todos esses algoritmos são descritos no Capítulo 12). Nos dois casos, o valor HMAC completo é calculado, mas depois truncado usando os primeiros 96 bits, que é o comprimento padrão para o campo Dados de Autenticação.

O MAC é calculado sobre

- Os campos do cabeçalho de IP que não mudam em trânsito (imutáveis) ou que são previsíveis no valor de chegada na extremidade para a SA AH. Os campos que podem mudar em trânsito e cujo valor na chegada são imprevisíveis são definidos como zero para fins de cálculo na origem e no destino.
- O cabeçalho AH sem levar em conta o campo Dados de Autenticação. O campo Dados de Autenticação é definido como zero, para fins de cálculo, na origem e no destino.
- Os dados completos do protocolo de nível superior, que são considerados imutáveis em trânsito (por exemplo, um segmento TCP ou um pacote de IP interno no modo túnel).

Para o IPv4, exemplos de campos imutáveis são Comprimento de Cabeçalho Internet e Endereço de Origem. Um exemplo de um campo mutável, mas previsível, é o Endereço de Destino (com roteamento de origem livre ou estrito). Alguns exemplos de campos mutáveis que são zerados antes do cálculo do ICV são os campos Tempo de Vida e Checksum de Cabeçalho. Observe que os campos de endereço de origem e destino estão protegidos, de modo que a falsificação de endereço é impedida.

Para o IPv6, os exemplos no cabeçalho básico são Versão (imutável), Endereço de Destino (mutável, mas previsível) e Etiqueta de Fluxo (mutável e zerado para cálculo).

Modos transporte e túnel

A Figura 16.5 mostra duas maneiras segundo as quais o serviço de autenticação IPSec pode ser usado. Em um caso, a autenticação é fornecida diretamente entre um servidor e estações de trabalho clientes; a estação de trabalho pode estar na mesma rede que o servidor ou em uma rede externa. Enquanto a estação de trabalho e o servidor compartilharem uma chave secreta protegida, o processo de autenticação será seguro. Esse caso usa um SA no modo transporte. No outro caso, uma estação de trabalho remota autentica-se no firewall corporativo, seja para acesso à rede interna inteira ou porque o servidor solicitado não admite o recurso de autenticação. Esse caso usa uma SA no modo túnel.

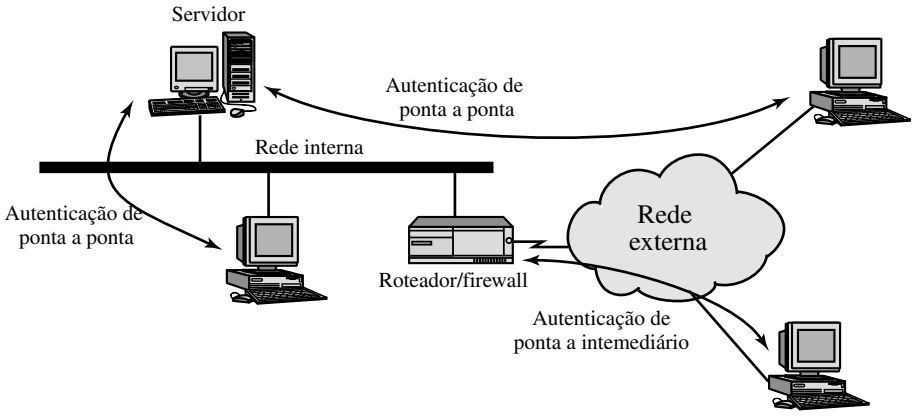


Figura 16.5 Autenticação de ponta a ponta *versus* autenticação de ponta a intermediário.

Nesta subseção examinamos o escopo da autenticação fornecido pelo AH e o local do cabeçalho de autenticação para os dois modos. As considerações são um pouco diferentes para IPv4 e IPv6. A Figura 16.6a mostra pacotes IPv4 e IPv6 típicos. Nesse caso, o payload de IP é um segmento TCP; ele também poderia ser uma unidade de dados para qualquer outro protocolo que usa IP, como UDP Ou ICMP.

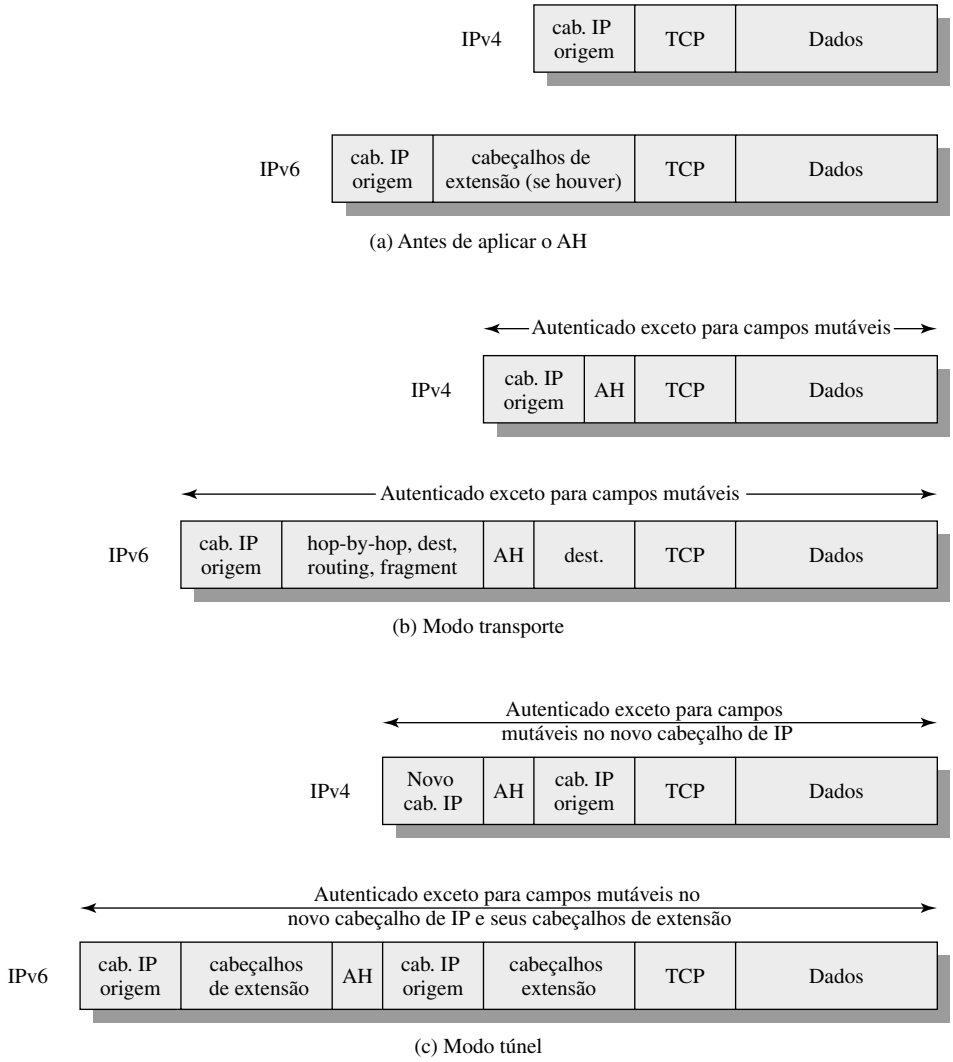


Figura 16.6 Escopo da autenticação AH.

Para o **AH no modo transporte** usando IPv4, o AH é inserido após o cabeçalho de IP original e antes do payload de IP (por exemplo, um segmento TCP); isso pode ser visto na parte superior da Figura 16.6b. A autenticação abrange o pacote inteiro, excluindo campos mutáveis no cabeçalho IPv4 que são definidos como zero para o cálculo do MAC.

No contexto do IPv6, o AH é visto como um payload de ponta a ponta; ou seja, ele não é examinado ou processado pelos roteadores intermediários. Portanto, o AH aparece após o cabeçalho básico do IPv6 e os cabeçalhos de extensão de roteamento, de fragmento e de salto a salto. O cabeçalho de extensão das opções de destino poderia aparecer antes ou depois do cabeçalho AH, dependendo da semântica desejada. Novamente, a autenticação abrange o pacote inteiro, excluindo campos mutáveis que são definidos como zero para cálculo do MAC.

Para o **AH no modo túnel**, o pacote IP original inteiro é autenticado, e o AH é inserido entre o cabeçalho de IP original e um novo cabeçalho de IP externo (Figura 16.6c). O cabeçalho de IP interno transporta os últimos endereços de origem e destino, enquanto um cabeçalho de IP externo pode conter diferentes endereços de IP (por exemplo, endereços de firewall ou outros gateways de segurança).

Com o modo túnel, todo o pacote de IP interno, incluindo o cabeçalho de IP interno inteiro, é protegido pelo AH. O cabeçalho de IP externo (e, no caso do IPv6, os cabeçalhos de extensão IP externos) é protegido, exceto para os campos mutáveis e imprevisíveis.

16.4 ENCAPSULAMENTO DE SEGURANÇA DO PAYLOAD

O encapsulamento de segurança do payload oferece serviços de confidencialidade, incluindo confidencialidade de conteúdo da mensagem e a confidencialidade limitada de fluxo de tráfego. Como um recurso opcional, o ESP também pode oferecer um serviço de autenticação.

Formato ESP

A Figura 16.7 mostra o formato do pacote ESP. Ele contém os seguintes campos:

- **Índice de Parâmetros de Segurança (32 bits):** Identifica uma associação de segurança.
- **Número de Seqüência (32 bits):** Um valor de contador que cresce monotonicamente; este oferece uma função anti-repetição, conforme explicado para o AH.
- **Dados de Payload (variável):** Este é um segmento do nível de transporte (modo transporte) ou pacote de IP (modo túnel) protegido por criptografia.
- **Complementação (0-255 bytes):** A finalidade deste campo é explicada mais adiante.
- **Comprimento da Complementação (8 bits):** Indica o número de bytes de complementação imediatamente anteriores a este campo.
- **Próximo Cabeçalho (8 bits):** Identifica o tipo de dados contido no campo de dados de payload, identificando o primeiro cabeçalho nesse payload (por exemplo, um cabeçalho de extensão no IPv6, ou um protocolo de camada superior como TCP).
- **Dados de autenticação (variável):** Um campo de comprimento variável (precisa ser um número inteiro de palavras de 32 bits) que contém o valor de verificação de integridade calculado sobre o pacote ESP, exceto o campo Autenticação de Dados.

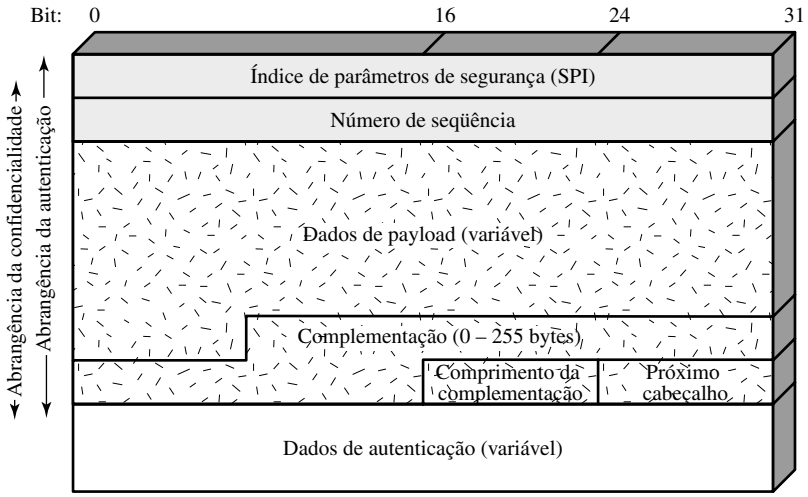


Figura 16.7 Formato ESP do IPSec.

Algoritmos de criptografia e autenticação

Os campos Dados de Payload, Complementação, Comprimento da Complementação e Próximo Cabeçalho são criptografados pelo serviço ESP. Se o algoritmo usado para criptografar o payload exigir dados de sincronismo criptográfico, como um vetor de inicialização (IV), então esses dados poderão ser carregados explicitamente no início do campo Dados do Payload. Se incluído, um IV normalmente não é criptografado, embora freqüentemente seja referenciado como fazendo parte do texto cifrado.

A especificação atual dita que uma implementação compatível precisa oferecer suporte a DES no modo de encadeamento de cifra de bloco (descrito no Capítulo 3). Diversos outros algoritmos receberam identificadores no documento DOI e, portanto, podem ser facilmente usados para criptografia. Entre eles estão:

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

Muitos desses algoritmos são descritos no Capítulo 6.

Assim como AH, o ESP admite o uso de um MAC com um comprimento-padrão de 96 bits. Além disso, assim como no AH, a especificação atual dita que uma implementação compatível precisa suportar HMAC-MD5-96 e HMAC-SHA-1-96.

Complementação

O campo de complementação tem várias finalidades:

- Se o algoritmo de criptografia exigir que o texto claro seja um múltiplo de algum número de bytes (por exemplo, o múltiplo de um único bloco para uma cifra de bloco), o campo de complementação será usado para expandir o texto claro (formado pelos campos Dados do Payload, Complementação, Comprimento da Complementação e Próximo Cabeçalho) para o comprimento exigido.
- O formato ESP exige que os campos Comprimento da Complementação e Próximo Cabeçalho sejam alinhados à direita dentro de uma palavra de 32 bits. De modo equivalente, o texto cifrado precisa ser um múltiplo inteiro de 32 bits. O campo Complementação é usado para garantir esse alinhamento.
- Uma complementação adicional pode ser acrescentada para oferecer confidencialidade parcial do fluxo de tráfego, ocultando o comprimento real do payload.

Modos túnel e transporte

A Figura 16.8 mostra duas maneiras como o serviço ESP do IPSec pode ser utilizado. Na parte superior da figura, a criptografia (e opcionalmente a autenticação) é obtida diretamente entre dois hosts. A Figura 16.8b mostra como a operação no modo túnel pode ser usada para configurar uma *rede privada virtual* (VPN — Virtual Private Network). Neste exemplo, uma organização possui quatro redes privadas interconectadas pela Internet. Os hosts nas redes internas utilizam a Internet para transporte de dados, mas não interagem com outros hosts baseados na Internet. Ao terminar os túneis no gateway de segurança de cada rede interna, a configuração permite que os hosts não tenham de implementar as funções de segurança. A primeira técnica tem o suporte de uma SA no modo transporte, enquanto a última técnica utiliza uma SA no modo túnel.

Nesta seção, examinamos o escopo do ESP para os dois modos. As considerações são um pouco diferentes para IPv4 e IPv6. Assim como em nossa explicação do escopo do AH, usaremos os formatos de pacote da Figura 16.6a como ponto de partida.

ESP no modo transporte O ESP no modo transporte é usado para criptografar e, opcionalmente, autenticar os dados carregados pelo IP (por exemplo, um segmento TCP), como mostra a Figura 16.9a. Para esse modo usando IPv4, o cabeçalho ESP é inserido no pacote IP imediatamente antes do cabeçalho da camada de transporte (por exemplo, TCP. UDP, ICMP) e um trailer ESP (campos Complementação, Comprimento da Complementação e Próximo Cabeçalho) é colocado após o pacote de IP; se a autenticação for selecionada, o campo Autenticação de Dados do ESP será acrescentado após o trailer ESP. O segmento inteiro em nível de transporte mais o trailer ESP serão criptografados. A autenticação abrange todo o texto cifrado mais o cabeçalho ESP.

No contexto do IPv6, o ESP é visto como um payload de ponta a ponta, isto é, ele não é examinado ou processado por roteadores intermediários. Portanto, o cabeçalho ESP aparece após o cabeçalho básico IPv6 e os cabeçalhos de extensão salto a salto, roteamento e fragmento. O cabeçalho de extensão das opções de destino aparece antes ou

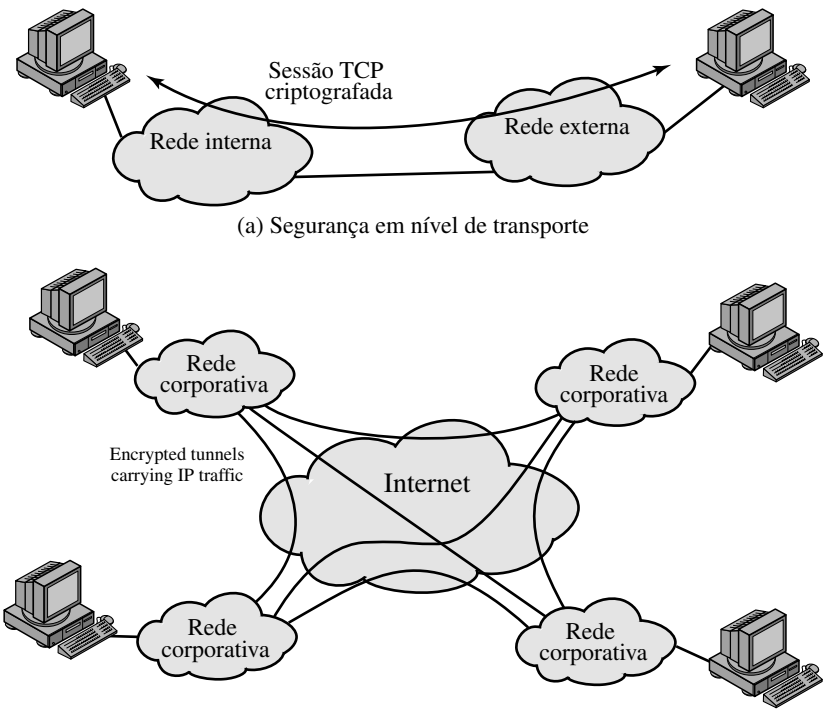


Figura 16.8 Criptografia no modo transporte *versus* no modo túnel.

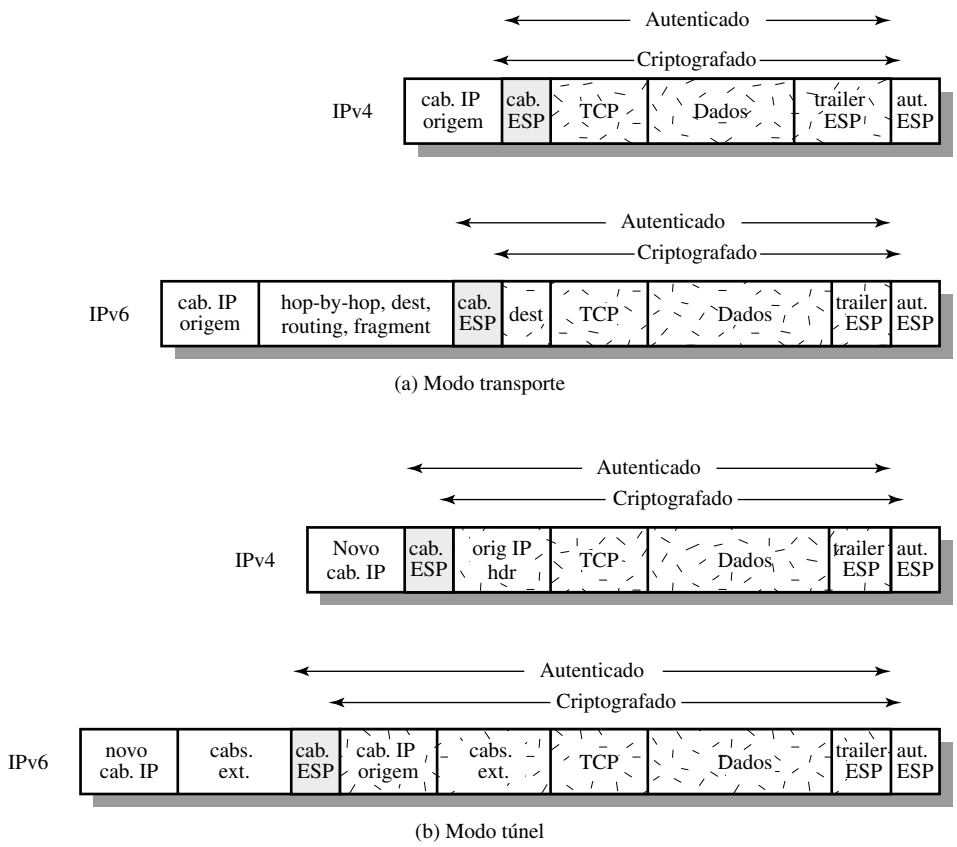


Figura 16.9 Escopo da criptografia e autenticação ESP.

depois do cabeçalho ESP, dependendo da semântica desejada. Para o IPv6, a criptografia aborda o segmento inteiro em nível de transporte mais o trailer ESP mais o cabeçalho de extensão das opções de destino, se este último ocorrer após o cabeçalho ESP. Novamente, a autenticação abrange o texto cifrado mais o cabeçalho ESP.

A operação no modo transporte pode ser resumida da seguinte maneira:

1. Na origem, o bloco de dados consistindo no trailer ESP mais o segmento inteiro da camada de transporte é criptografado e o texto claro desse bloco é substituído por seu texto cifrado para formar o pacote de IP para transmissão. A autenticação será acrescentada se essa opção for selecionada.
2. O pacote é, então, roteado para o destino. Cada roteador intermediário precisa examinar e processar o cabeçalho de IP mais quaisquer cabeçalhos de extensão de IP em texto claro, mas não precisa examinar o texto cifrado.
3. O nó de destino examina e processa o cabeçalho de IP mais quaisquer cabeçalhos de extensão de IP de texto claro. Depois, com base no SPI do cabeçalho ESP, o nó de destino decriptografa o restante do pacote para recuperar o segmento da camada de transporte em texto claro.

A operação no modo de transporte oferece confidencialidade para qualquer aplicação que a utilize, evitando a necessidade de implementar a confidencialidade em cada aplicação individual. Esse modo de operação também é razoavelmente eficiente, acrescentando pouco ao comprimento total do pacote de IP. Uma desvantagem desse modo é que é possível realizar a análise de tráfego nos pacotes transmitidos.

ESP modo túnel O ESP modo túnel é usado para criptografar um pacote de IP inteiro (Figura 16.9b). Para esse modo, o cabeçalho ESP é prefixado para o pacote e, então, o pacote mais o trailer ESP são criptografados. Esse método pode ser usado para dificultar a análise de tráfego.

Como o cabeçalho de IP contém o endereço de destino e possivelmente as diretivas do roteamento da origem e informações de opção salto a salto, não é possível simplesmente transmitir o pacote de IP criptografado prefixado pelo cabeçalho ESP. Roteadores intermediários seriam incapazes de processar esse pacote. Portanto, é necessário encapsular o bloco inteiro (cabeçalho ESP mais texto cifrado mais Autenticação de Dados, se estiver presente) com um novo cabeçalho de IP que conterá informações suficientes para o roteamento, mas não para a análise de tráfego.

Enquanto o modo transporte é satisfatório para proteger conexões entre hosts que suportam o recurso ESP, o modo túnel é útil em uma configuração que inclui um firewall ou outro tipo de gateway de segurança que protege uma rede confiável contra redes externas. Neste último caso, a criptografia só ocorre entre um host externo e o gateway de segurança, ou entre dois gateways de segurança. Isso alivia os hosts na rede interna da carga de processamento da criptografia e simplifica a tarefa de distribuição de chave, uma vez que reduz o número de chaves necessárias. Além disso, impede a análise de tráfego baseada no destino final.

Considere um caso em que um host externo deseja se comunicar com um host em uma rede interna, protegida por um firewall, e em que o ESP é implementado no host externo e nos firewalls. As etapas a seguir ocorrem para a transferência de um segmento da camada de transporte a partir do host externo para o host interno:

1. A origem prepara um pacote de IP interno com um endereço de destino do host interno de destino. Esse pacote é prefixado por um cabeçalho ESP; então o pacote e o trailer ESP são criptografados e dados de autenticação podem ser acrescentados. O bloco resultante é encapsulado com um novo cabeçalho de IP (cabeçalho básico mais extensões opcionais, como opções de roteamento e salto a salto para IPv6), cujo endereço de destino é o firewall; isso forma o pacote de IP externo.
2. O pacote externo é roteado para o firewall de destino. Cada roteador intermediário precisa examinar e processar o cabeçalho de IP externo mais quaisquer cabeçalhos de extensão de IP externos, mas não precisa examinar o texto cifrado.
3. O firewall de destino examina e processa o cabeçalho de IP externo mais quaisquer cabeçalhos de extensão de IP externos. Então, com base no SPI no cabeçalho ESP, o nó de destino decriptografa o restante do pacote para recuperar o texto claro do pacote de IP interno. Esse pacote é, então, transmitido na rede interna.
4. O pacote interno é roteado por zero ou mais roteadores na rede interna até o host de destino.

16.5 COMBINANDO ASSOCIAÇÕES DE SEGURANÇA

Uma SA individual pode implementar o AH ou o protocolo ESP, mas não ambos. Algumas vezes, determinado fluxo de tráfego exigirá os serviços fornecidos por AH e ESP. Além disso, determinado fluxo de tráfego pode exigir serviços IPSec entre hosts e, para esse mesmo fluxo, serviços diferentes entre gateways de segurança, como firewalls. Em todos esses casos, várias SAs precisam ser empregadas para que o mesmo fluxo de tráfego alcance os serviços IPSec desejados. O termo *combinação de associação de segurança* (*security association bundle*) refere-se a uma seqüência de SAs através das quais o tráfego precisa ser processado para fornecer o conjunto desejado de serviços IPSec. As SAs combinadas podem terminar em diferentes extremidades ou nas mesmas extremidades.

As associações de segurança podem ser combinadas de duas maneiras:

- **Adjacência de transporte:** Refere-se à aplicação de mais de um protocolo de segurança para o mesmo pacote de IP, sem utilizar o túnel. Essa técnica para combinar AH e ESP permite apenas um nível de combinação; mais aninhamento não gera benefício adicional, pois o processamento é realizado em uma instância IPSec: o destino (final).
- **Túnel com iteração:** Refere-se à aplicação de várias camadas de protocolos de segurança efetuadas por meio do túnel de IP. Essa técnica permite vários níveis de aninhamento, pois cada túnel pode originar ou terminar em um ponto IPSec diferente ao longo do caminho.

As duas técnicas podem ser combinadas, por exemplo, fazendo-se com que uma SA de transporte entre os hosts trafegue parte do caminho por uma SA de túnel entre gateways de segurança.

Uma questão interessante que surge quando se considera combinações de SA é a ordem em que a autenticação e a criptografia podem ser aplicadas entre determinado par de extremidades e as formas de fazer isso. Examinamos essa questão em seguida. Depois, examinamos combinações de SAs que envolvam pelo menos um túnel.

Autenticação mais confidencialidade

Criptografia e autenticação podem ser combinados a fim de transmitir um pacote de IP que tenha tanto confidencialidade quanto autenticação entre os hosts. Examinamos várias técnicas.

ESP com opção de autenticação Essa técnica é ilustrada na Figura 16.9. O usuário primeiramente aplica o ESP aos dados a serem protegidos e depois anexa a autenticação do campo de dados. Na realidade, existem dois subcasos:

- **ESP no modo transporte:** Autenticação e criptografia aplicadas ao payload de IP entregue ao host, mas o cabeçalho de IP não é protegido.
- **ESP no modo túnel:** A autenticação se aplica ao pacote de IP inteiro entregue ao endereço de destino de IP externo (por exemplo, um firewall), e a autenticação é realizada nesse destino. Todo o pacote de IP interno é protegido pelo mecanismo de privacidade, para entrega ao destino de IP interno.

Para os dois casos, a autenticação se aplica ao texto cifrado, em vez do texto claro.

Adjacência de transporte Outra maneira de aplicar a autenticação após a criptografia é usar duas SAs de transporte em conjunto, com a interna sendo uma SA ESP e a externa sendo uma SA AH. Nesse caso, o ESP é usado sem sua opção de autenticação. Como a SA interna é uma SA de transporte, a criptografia é aplicada ao payload de IP. O pacote resultante consiste em um cabeçalho de IP (e, possivelmente, extensões de cabeçalho IPv6) seguido por um ESP. AH é então aplicado no modo transporte, de modo que a autenticação abrange o ESP mais o cabeçalho de IP original (e extensões) exceto para campos mutáveis. A vantagem dessa técnica em relação a simplesmente usar uma única SA ESP com a opção de autenticação ESP é que a autenticação abrange mais campos, incluindo os endereços de IP de origem e destino. A desvantagem é o overhead de duas SAs *versus* uma SA.

Combinação de transporte-túnel O uso da autenticação antes da criptografia pode ser preferível por vários motivos. Primeiramente, como a autenticação dos dados é protegida pela criptografia, é impossível que alguém intercepte a mensagem e altere a autenticação dos dados sem ser detectado. Segundo, pode ser desejável armazenar as informações de autenticação com a mensagem no destino, para consulta posterior. É mais conveniente fazer isso se a autenticação das informações se aplicar à mensagem não criptografada; caso contrário, a mensagem teria de ser recriptografada para verificar a autenticação das informações.

Uma técnica para a aplicação da autenticação antes da criptografia entre dois hosts é usar uma combinação consistindo de uma SA de transporte do AH interno e uma SA de túnel do ESP externo. Nesse caso, a autenticação é aplicada ao payload de IP mais o cabeçalho de IP (e extensões), exceto para os campos mutáveis. O pacote de IP resultante é então processado no modo túnel pelo ESP; o resultado é que todo o pacote interno autenticado é criptografado e um novo cabeçalho de IP externo (e extensões) é acrescentado.

Combinações básicas de associações de segurança

O documento de arquitetura do IPSec lista quatro exemplos de combinações das SAs que precisam ser suportados por hosts IPSec compatíveis (por exemplo, estação de trabalho, servidor) ou gateways de segurança (por exemplo, firewall, roteador). Estes são ilustrados na Figura 16.10. A parte inferior de cada caso na figura representa a conectividade física dos elementos; a parte superior representa a conectividade lógica por meio de uma ou mais SAs aninhadas. Cada SA pode ser AH ou ESP. Para SAs host para host, o modo pode ser transporte ou túnel; caso contrário, o modo precisa ser túnel.

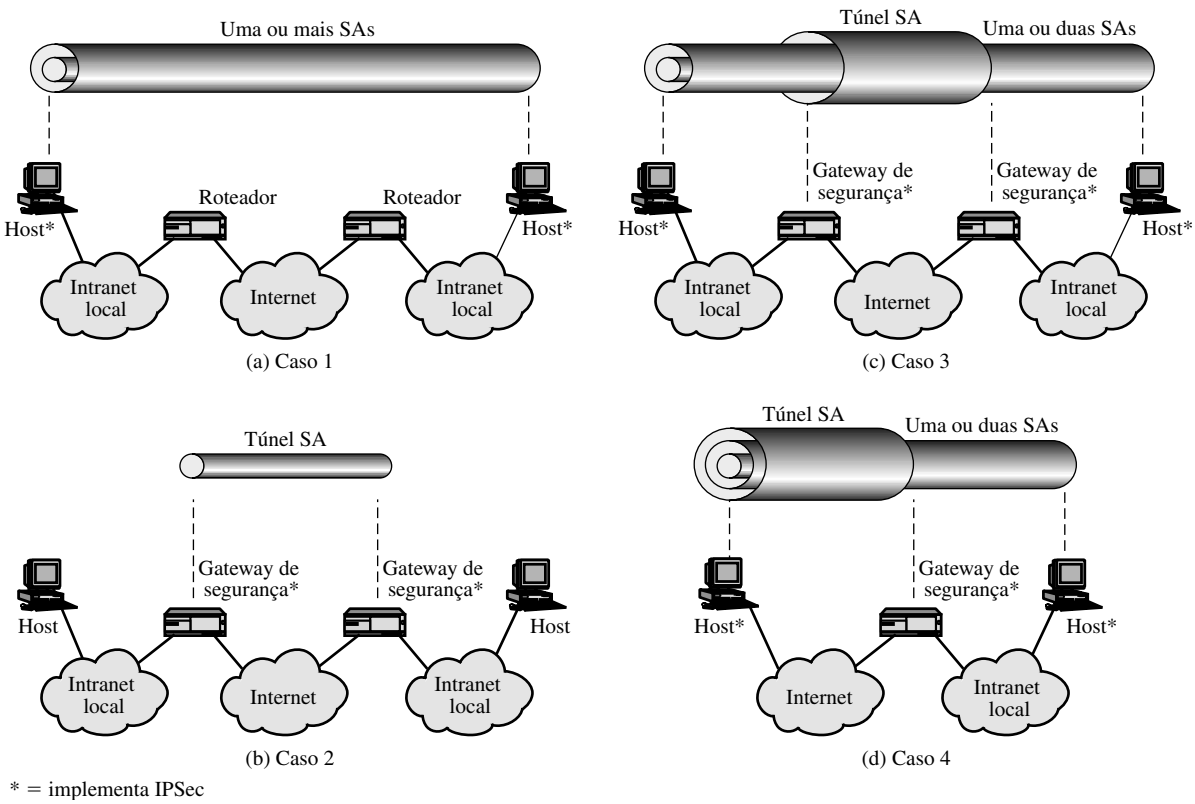


Figura 16.10 Combinações básicas das associações de segurança (SAs).

No **Caso 1**, toda a segurança é fornecida entre os sistemas finais que implementam o IPSec. Para dois sistemas finais quaisquer se comunicarem por meio de uma SA, eles precisam compartilhar as chaves secretas apropriadas. Entre as combinações possíveis:

a. AH no modo transporte
b. ESP no modo transporte
c. ESP seguido por AH no modo transporte (uma SA ESP dentro de uma SA AH)
d. Qualquer um de a, b ou c dentro de um AH ou ESP no modo túnel

Já explicamos como essas diversas combinações podem ser usadas para suportar autenticação, criptografia, autenticação antes da criptografia e autenticação após a criptografia.

Para o **Caso 2**, a segurança é fornecida apenas entre gateways (roteadores, firewalls etc.) e nenhum host implementa o IPSec. Esse caso ilustra o suporte simples para rede privada virtual. O documento de arquitetura de segurança especifica que somente um único túnel SA é necessário para esse caso. O túnel poderia admitir AH, ESP ou ESP com a opção de autenticação. Os túneis aninhados não são exigidos, porque os serviços IPSec se aplicam a todo o pacote interno.

O **Caso 3** é baseado no Caso 2, acrescentando a segurança de ponta a ponta. As mesmas combinações explicadas para os casos 1 e 2 são permitidas aqui. O túnel gateway para gateway oferece autenticação ou confidencialidade ou ambos para todo o tráfego entre os sistemas finais. Quando o túnel gateway para gateway é ESP, ele também oferece uma forma limitada de confidencialidade de tráfego. Os hosts individuais podem implementar quaisquer serviços IPSec adicionais exigidos para determinadas aplicações ou determinados usuários por meio das SAs de ponta a ponta.

O **Caso 4** oferece suporte para um host remoto que usa a Internet para alcançar o firewall de uma organização e depois obter acesso a algum servidor ou estação de trabalho por trás do firewall. Somente o modo túnel é exigido entre o host remoto e o firewall. Assim como no Caso 1, uma ou duas SAs podem ser usadas entre o host remoto e o host local.

16.6 GERENCIAMENTO DE CHAVES

A parte de gerenciamento de chaves do IPSec envolve a determinação e a distribuição de chaves secretas. Um requisito típico é o uso de quatro chaves para a comunicação entre duas aplicações: pares de transmissão e recepção tanto para AH quanto para ESP. O documento de arquitetura do IPSec exige suporte para dois tipos de gerenciamento de chaves:

- **Manual:** Um administrador do sistema configura manualmente cada sistema com suas próprias chaves e com as chaves de outros sistemas em comunicação. Isso é prático para ambientes pequenos, relativamente estáticos.
- **Automatizado:** Um sistema automatizado permite a criação de chaves para SAS sob demanda e facilita o uso de chaves em um grande sistema distribuído com uma configuração que não é constante.

O protocolo de gerenciamento de chaves automatizado padrão para IPSec é conhecido como ISAKMP/Oakley, e consiste nos seguintes elementos:

- **Protocolo de determinação de chave Oakley:** Oakley é um protocolo de troca de chaves baseado no algoritmo Diffie-Hellman, mas oferecendo segurança adicional. O Oakley é genérico no sentido de que não dita formatos específicos.
- **Protocolo de gerenciamento de chaves e de associações de segurança na Internet (ISAKMP — Internet Security Association and Key Management Protocol):** O ISAKMP oferece uma estrutura para o gerenciamento de chaves pela Internet e oferece o suporte a um protocolo específico, incluindo formatos, para a negociação de atributos de segurança.

O ISAKMP por si só não dita um algoritmo de troca de chaves específico; em vez disso, o ISAKMP consiste de um conjunto de tipos de mensagens que permite o uso de uma série de algoritmos de troca de chaves. O Oakley é o algoritmo de troca de chaves específico exigido para uso com a versão final do ISAKMP.

Vamos começar com uma visão geral do Oakley e depois examinaremos o ISAKMP.

Protocolo de determinação de chaves Oakley

O Oakley é um aprimoramento do algoritmo de troca de chaves Diffie-Hellman. Lembre-se de que o Diffie-Hellman envolve a seguinte interação entre os usuários A e B. Existe acordo prévio sobre dois parâmetros globais: q, um número primo grande; e α, uma raiz primitiva de q. A seleciona um inteiro aleatório X_A e sua chave privada, e transmite para B sua chave pública $Y_A = \alpha^{X_A} \bmod q$. De modo semelhante, B seleciona um inteiro aleatório X_B como sua chave privada e transmite para A sua chave pública $Y_B = \alpha^{X_B} \bmod q$. Cada lado pode agora calcular a chave de sessão secreta:

$$K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q = \alpha^{X_A X_B} \bmod q$$

O algoritmo Diffie-Hellman possui dois recursos atraentes:

- Chaves secretas são criadas apenas quando necessárias. Não há necessidade de armazenar chaves secretas por um longo período de tempo, expondo-as a maior vulnerabilidade.
- A troca não exige infra-estrutura pré-existente, além de um acordo sobre os parâmetros globais.

Porém, existem vários pontos fracos no Diffie-Hellman, conforme indicado em [HUIT98]:

- Ele não oferece qualquer informação sobre as identidades das partes.
- Ele está sujeito a um ataque de homem no meio (*man-in-the-middle*), em que um terceiro C se faz passar por B enquanto se comunica com A e se faz passar por A enquanto se comunica com B. Tanto A quanto B acabam trocando uma chave com C, que pode então interceptar o tráfego e passá-lo adiante. O ataque de homem do meio prossegue da seguinte maneira:

  1. B envia sua chave pública Y_B em uma mensagem endereçada para A (ver Figura 10.8).
  2. O inimigo (E) intercepta essa mensagem. E salva a chave pública de B e envia uma mensagem para A, que tem a ID de usuário de B, mas a chave pública de E, Y_E. Essa mensagem é enviada de modo que apareça como se fosse enviada pelo sistema host de B. A recebe a mensagem de E e armazena a chave pública de E com a ID de usuário de B. Semelhantemente, E envia uma mensagem a B com a chave pública de E, fingindo ter vindo de A.
  3. B calcula uma chave secreta K_1 com base na chave privada de B e Y_E. A calcula uma chave secreta K_2 com base na chave secreta de A e Y_E. E calcula K_1 usando a chave secreta de E X_E e Y_B e calcula K_2 usando X_E e Y_A.
  4. De agora em diante, E é capaz de repassar mensagens de A para B e de B para A, alterando apropriadamente sua cifra no caminho, de modo que nem A nem B saberão que compartilham sua comunicação com E.

- Ele é computacionalmente intensivo. Como resultado, é vulnerável a um ataque de obstrução (*clogging attack*), em que um oponente solicita uma grande quantidade de chaves. A vítima gasta recursos de computação consideráveis realizando exponenciação modular inútil, em vez de trabalho real.

O Oakley foi projetado para preservar as vantagens do Diffie-Hellman ao mesmo tempo que combate seus pontos fracos.

Características do Oakley O algoritmo Oakley possui cinco características importantes:

1. Ele emprega um mecanismo conhecido como cookies para impedir ataques de obstrução.
2. Ele permite que as duas partes negociem um grupo; isso, basicamente, especifica os parâmetros globais da troca de chaves Diffie-Hellman.
3. Ele usa nonces para impedir os ataques de replicação.
4. Ele permite a troca de valores de chave pública Diffie-Hellman.
4. Ele autentica a troca Diffie-Hellman para impedir ataques de homem do meio.

Já explicamos o Diffie-Hellman. Vejamos o restante desses elementos, um por vez. Primeiro, considere o problema dos ataques de obstrução. Nesse ataque, um oponente forja o endereço de origem de um usuário legítimo e envia uma chave Diffie-Hellman à vítima. A vítima realiza uma exponenciação modular para calcular a chave secreta. Mensagens repetidas desse tipo podem *obstruir* o sistema da vítima com trabalho inútil. A **troca de cookies** requer que cada lado envie um número pseudo-aleatório, o cookie, na mensagem inicial, que o outro lado confirma. Essa confirmação precisa ser repetida na primeira mensagem de troca de chaves Diffie-Hellman. Se o endereço de origem foi forjado, o oponente não recebe resposta. Assim, um oponente só pode forçar um usuário a gerar confirmações e não a realizar o cálculo Diffie-Hellman.

O ISAKMP exige que a geração de cookies satisfaça três requisitos básicos:

1. O cookie precisa depender das partes específicas. Isso impede que um atacante obtenha um cookie usando um endereço de IP real e porta UDP, e depois o utilize para inundar a vítima com solicitações de endereços de IP ou portas escolhidas aleatoriamente.
2. Não deverá ser possível a alguém diferente da entidade emissora gerar cookies que sejam aceitos por essa entidade. Isso implica que a entidade emissora usará informações secretas locais na geração e subseqüente verificação de um cookie. Não deverá ser possível deduzir essa informação secreta a partir de qualquer cookie em particular. O objetivo desse requisito é que a entidade emissora não precise salvar cópias de seus cookies, que são então mais vulneráveis a serem descobertos, mas possa verificar uma confirmação de cookie que chega, quando for necessário.
3. Os métodos de geração e verificação de cookies precisam ser rápidos para impedir ataques que pretendam sabotar recursos do processador.

O método recomendado para criar o cookie é realizar um hash rápido (por exemplo, MD5) sobre os endereços de origem e destino de IP, as portas UDP de origem e destino, e um valor secreto gerados localmente.

O Oakley admite o uso de diferentes **grupos** para a troca de chaves Diffie-Hellman. Cada grupo inclui a definição dos dois parâmetros globais e da identidade do algoritmo. A especificação atual inclui os seguintes grupos:

- Exponenciação modular com um módulo de 768 bits
  $q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149686)$
  $\alpha = 2$
- Exponenciação modular com um módulo de 1.024 bits
  $q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129093)$
  $\alpha = 2$
- Exponenciação modular com um módulo de 1.536 bits
  —Parâmetros a serem determinados
- Grupo de curva elíptica sobre 2^{155}
  —Gerador (hexadecimal): X = 7B, Y = 1C8
  —Parâmetros da curva elíptica (hexadecimal): A = 0, Y = 7338F
- Grupo de curva elíptica sobre 2^{185}
  —Gerador (hexadecimal): X = 18, Y = D
  —Parâmetros da curva elíptica (hexadecimal): A = 0, Y = 1EE9

Os três primeiros grupos são o algoritmo Diffie-Hellman clássico usando exponenciação modular. Os dois últimos grupos utilizam a curva elíptica análoga ao Diffie-Hellman, que foi descrito no Capítulo 10.

O Oakley emprega **nonces** para impedir ataques de replicação. Cada nonce é um número pseudo-aleatório gerado localmente. Os nonces aparecem em respostas e são criptografados durante certas partes da troca para assegurar seu uso.

Esses diferentes métodos de autenticação podem ser usados com Oakley:

- **Assinaturas digitais:** A troca é autenticada assinando-se um hash que pode ser obtido mutuamente; cada parte criptografa o hash com sua chave privada. O hash é gerado sobre parâmetros importantes, como IDs de usuário e nonces.

I → R: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notação:

I	=	Iniciador
R	=	Respondedor
CKY_I, CKY_R	=	iniciador, cookies do respondedor
OK_KEYX	=	tipo de mensagem de troca de chaves
GRP	=	Name of Diffie-Hellman group for this exchange
g^x,g^y	=	chave pública do iniciador, respondedor; g^{xy} = chave de sessão dessa troca
EHAO, EHAS	=	criptografia, funções de autenticação de hash, oferecida e selecionada
NIDP	=	indica que a criptografia não é usada para o restante dessa mensagem
ID_I, ID_R	=	identificador para iniciador, respondedor
N_I, N_R	=	nonce aleatório fornecido pelo iniciador, respondedor para essa troca
$S_{KI}[X]$, $S_{KR}[X]$	=	indica a assinatura sobre X usando a chave privada (chave de assinatura) do iniciador, respondedor

Figura 16.11 Exemplo de troca agressiva de chave Oakley.

- **Criptografia de chave pública:** A troca é autenticada criptografando-se parâmetros como IDs e nonces com a chave privada do emissor.
- **Criptografia de chave simétrica:** Uma chave obtida por algum mecanismo *out-of-band* (por canal alternativo) pode ser usada para autenticar a troca por criptografia simétrica dos parâmetros da troca.

Exemplo de troca Oakley A especificação Oakley inclui diversos exemplos de trocas que são permitidas sob o protocolo. Para dar uma idéia do Oakley, apresentamos um exemplo, denominado troca de chaves agressiva na especificação, uma vez que apenas três mensagens são trocadas.

A Figura 16.11 mostra o protocolo de troca de chaves agressiva. No primeiro passo, o iniciador (I) transmite um cookie, o grupo a ser usado, e a chave pública Diffie-Hellman de I para essa troca. Também indica a criptografia de chave pública oferecida, hash e algoritmos de autenticação a serem usados nessa troca. Também incluídos nessa mensagem estão os identificadores de I e o respondedor (R) e o nonce de I para essa troca. Finalmente, I acrescenta uma assinatura usando a chave privada de I que assina os dois identificadores, o nonce, o grupo, a chave pública Diffie-Hellman e os algoritmos oferecidos.

Quando R recebe a mensagem, R verifica a assinatura usando a chave de assinatura pública de I. R confirma a mensagem ecoando de volta o cookie, o identificador e o nonce, além do grupo. R também inclui na mensagem um cookie, a chave pública Diffie-Hellman, os algoritmos selecionados (que precisam estar entre os algoritmos oferecidos), o identificador de R e o nonce de R para essa troca. Finalmente, R anexa uma assinatura usando a chave privada de R que assina os dois identificadores, os dois nonces, o grupo, as chaves públicas Diffie-Hellman e os algoritmos selecionados.

Quando I recebe a segunda mensagem, I verifica a assinatura usando a chave pública de R. Os valores de nonce na mensagem garantem que essa não é uma repetição de uma mensagem antiga. Para completar a troca, I precisa enviar uma mensagem de volta a R para verificar se I recebeu a chave pública de R.

ISAKMP

O ISAKMP define procedimentos e formatos de pacote para estabelecer, negociar, modificar e excluir associações de segurança. Como parte do estabelecimento da SA, o ISAKMP define payloads para a troca de geração de chaves e autenticação de dados. Esses formatos de payload oferecem uma estrutura consistente independente do protocolo de troca de chaves específico, algoritmo de criptografia e mecanismo de autenticação.

Formato de cabeçalho ISAKMP Uma mensagem ISAKMP consiste em um cabeçalho ISAKMP seguido por um ou mais payloads. Tudo isso é transmitido via um protocolo de transporte. A especificação dita que as implementações precisam oferecer suporte ao uso de UDP para o protocolo de transporte.

A Figura 16.12a mostra o formato de cabeçalho para uma mensagem ISAKMP. Ele consiste nos seguintes campos:

- **Cookie iniciador (64 bits):** Cookie da entidade que iniciou o estabelecimento da SA, notificação da SA ou exclusão da SA.
- **Cookie respondedor (64 bits):** Cookie da entidade respondedora; nulo na primeira mensagem do iniciador.
- **Próximo payload (8 bits):** Indica o tipo do primeiro payload na mensagem; payloads são explicados na próxima subseção.
- **Versão principal (4 bits):** Indica a versão principal do ISAKMP em uso.

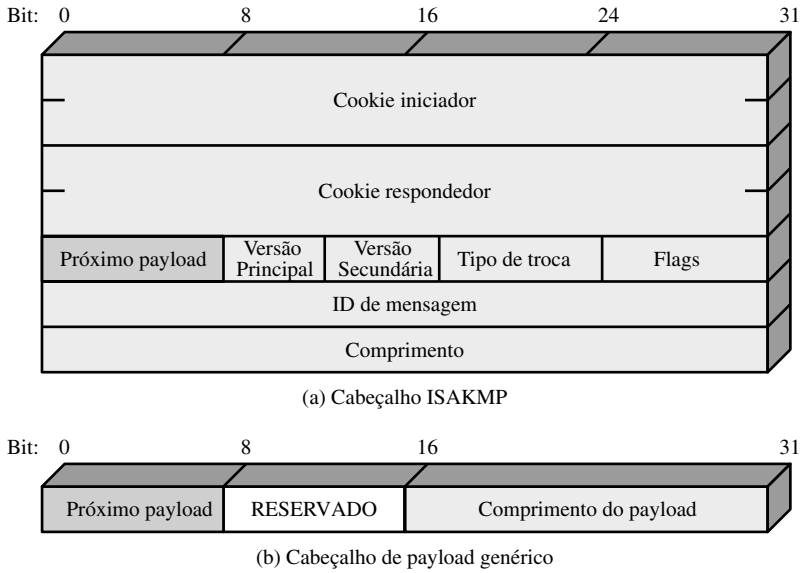


Figura 16.12 Formatos ISAKMP.

- **Versão secundária (4 bits):** Indica a versão secundária em uso.
- **Tipo de troca (8 bits):** Indica o tipo da troca; estas são explicadas mais adiante nesta seção.
- **Flags (8 bits):** Indica opções específicas definidas para essa troca ISAKMP. Há dois bits definidos até agora: o bit de criptografia é definido se todos os payloads após o cabeçalho forem criptografados usando o algoritmo de criptografia para esta SA. O bit Commit é usado para garantir que o material criptografado não seja recebido antes da conclusão do estabelecimento da SA.
- **ID de mensagem (32 bits):** ID exclusiva para esta mensagem.
- **Comprimento (32 bits):** Comprimento da mensagem total (cabeçalho mais todos os payloads) em octetos.

Tipos de payload ISAKMP Todos os payloads ISAKMP começam com o mesmo cabeçalho de payload genérico mostrado na Figura 16.12b. O campo de próximo payload tem um valor 0 se este for o último payload na mensagem; caso contrário, seu valor é o tipo do próximo payload. O campo de comprimento do payload indica o comprimento em octetos desse payload, incluindo o cabeçalho genérico de payload.

A Tabela 16.3 resume os tipos de payload definidos para ISAKMP, e lista os campos, ou parâmetros, que fazem parte de cada payload. O **payload SA** é usado para começar o estabelecimento de uma SA. Nesse payload, o parâmetro Domínio de Interpretação identifica o DOI sob o qual a negociação está ocorrendo. O DOI IPSec é um exemplo, mas o ISAKMP pode ser usado em outros contextos. O parâmetro Situação define a política de segurança para essa negociação; basicamente, os níveis de segurança exigidos para criptografia e confidencialidade são especificados (por exemplo, nível de importância, compartimento de segurança).

O **payload Transformação** define uma transformação de segurança a ser usada para proteger o canal de comunicações para o protocolo designado. O parâmetro Nº Transformação serve para identificar esse payload em particular de modo que o respondedor possa usá-lo para indicar a aceitação dessa transformação. Os campos ID de Transformação e Atributos identificam uma transformação específica (por exemplo, 3DES em ESP, HMAC-SHA-1-96 para AH) com seus atributos associados (por exemplo, comprimento do hash).

O **payload Troca de Chaves** pode ser usado para diversas técnicas de troca de chaves, incluindo Oakley, Diffie-Hellman e a troca de chaves baseada em RSA, usada pelo PGP. O campo de dados de troca de chaves contém os dados exigidos para gerar uma chave de sessão e depende do algoritmo de troca de chaves utilizado.

O **payload Identificação** é usado para determinar a identidade das partes em comunicação e pode ser usado para determinar a autenticidade da informação. Normalmente, o campo de dados de ID conterá um endereço IPv4 ou IPv6.

O **payload Certificado** transfere um certificado de chave pública. O campo codificação do certificado indica o tipo de certificado ou informações relacionadas ao certificado, que podem incluir o seguinte:

- Certificado X.509 envelopado com PKCS #7
- Certificado PGP
- Chave assinada do DNS
- Certificado X.509 – assinatura
- Certificado X.509 – troca de chaves

Tabela 16.3 Tipos de payload ISAKMP

Tipo	Parâmetros	Descrição
Associação de segurança (SA)	Domínio de interpretação, situação	Usado para negociar atributos de segurança e indicar o DOI e a situação sob os quais a negociação está ocorrendo.
Proposta (P)	Nº da proposta, ID de protocolo, comprimento do SPI, nº de transformações, SPI	Usado durante a negociação da SA; indica o protocolo a ser usado e o número de transformações.
Transformação (T)	Nº transformação, ID de transformação, atributos de SA	Usado durante a negociação da SA; indica a transformação e os atributos de SA relacionados.
Troca de chaves (KE)	Dados de troca de chaves	Admite uma série de técnicas de troca de chaves.
Identificação (ID)	Tipo de ID, dados de ID	Usado para trocar informações de identificação.
Certificado (CERT)	Codificação do certificado, dados de certificado	Usado para transportar certificados e outras informações relacionadas.
Solicitação de certificado (CR)	Nº de tipos de certificado, tipos de certificado, nº autoridade de certificação, autoridades de certificação	Usado para solicitar certificados; indica os tipos de certificados solicitados e as autoridades de certificação aceitáveis.
Hash (HASH)	Dados de hash	Contém dados gerados por uma função de hash.
Assinatura (SIG)	Dados de assinatura	Contém dados gerados por uma função de assinatura digital.
Nonce (NONCE)	Dados de nonce	Contém um nonce.
Notificação (N)	DOI, ID de protocolo, tamanho de SPI, tipo de mensagem de notificação, SPI, dados de notificação	Usado para transmitir dados de notificação, como uma condição de erro.
Exclusão (D)	DOI, ID protocolo, tamanho de SPI, nº de SPIs, SPI (um ou mais)	Indica uma SA que não é mais válida.

- Tokens Kerberos
- Lista de revogação de certificados (CRL)
- Lista de revogação de autoridades (ARL)
- Certificado SPKI

A qualquer momento em uma troca ISAKMP, o emissor poderá incluir um payload **Solicitação de Certificado** para solicitar o certificado da outra entidade da comunicação. O payload pode listar mais de um tipo de certificado aceito e mais de uma autoridade certificadora aceita.

O **payload Hash** contém dados gerados por uma função de hash sobre alguma parte da mensagem e/ou estado ISAKMP. Esse payload é usado para verificar a integridade dos dados em uma mensagem ou para autenticar entidades em negociação.

O **payload Assinatura** contém dados gerados por uma função de assinatura digital sobre alguma parte da mensagem e/ou estado ISAKMP. Esse payload é usado para verificar a integridade dos dados em uma mensagem e pode ser usado para serviços de irretratabilidade.

O **payload Nonce** contém dados aleatórios usados para garantir a atividade durante uma troca e proteção contra ataques por repetição.

O **payload Notificação** contém qualquer erro ou informação de estado associadas a essa SA ou a essa negociação de SA. As mensagens de erro ISAKMP a seguir foram definidas:

Invalid Payload Type Invalid Protocol ID Invalid Cert Encoding DOI Not Supported Invalid SPI Invalid Certificate Situation Not Supported Invalid Transform ID Bad Cert Request Syntax Invalid Cookie Attributes Not Supported Invalid Cert Authority Invalid Major Version No Proposal Chosen Invalid Hash Information Invalid Minor Version Bad Proposal Syntax Authentication Failed Invalid Exchange Type Payload Malformed Invalid Signature Invalid Flags Invalid Key Information Address Notification Invalid Message ID

A única mensagem de status ISAKMP definida até agora é Conectado (Connected). Além dessas notificações ISAKMP, notificações específicas do DOI são usadas. Para o IPSec, as seguintes mensagens de status adicionais são definidas:

- **Duração-Respondedor (Responder-Lifetime):** Comunica o tempo de duração da SA escolhida pelo respondedor.
- **Repetição-Status(Replay-Status):** Usada para confirmação positiva da escolha do respondedor em relação a se ele efetuará ou não a detecção anti-repetição.
- **Inicial-Contato (Initial-Contact):** Informa o outro lado que esta é a primeira SA sendo definida com o sistema remoto. O receptor dessa notificação poderia, então, excluir quaisquer SAs para o sistema emissor sob a suposição de que o sistema emissor foi reinicializado e não tem mais acesso a essas SAs.

O **payload Exclusão** indica uma ou mais SAs que o emissor excluiu de seu banco de dados e que, portanto, não são mais válidas.

Trocas ISAKMP O ISAKMP oferece uma estrutura para troca de mensagens, com os tipos de payload servindo como elementos básicos. A especificação identifica os cinco tipos-padrão de troca que devem ser suportados; estes são resumidos na Tabela 16.4. Na tabela, SA refere-se a um payload SA com payloads de protocolo e transformação associados.

A **troca básica** permite que a troca de chaves e o material de autenticação sejam transmitidos juntos. Isso reduz o número de trocas à custa de não oferecer proteção de identidade. As duas primeiras mensagens oferecem cookies e definem uma SA com protocolo e transformações combinados; os dois lados utilizam um nonce para garantia contra ataques por repetição. As duas últimas mensagens trocam o material relativo à chave e IDs de usuário, com um mecanismo de autenticação usado para autenticar chaves, identidades e os nonces das duas primeiras mensagens.

A **troca de proteção de identidade** expande a troca básica para proteger as identidades dos usuários. As duas primeiras mensagens definem a SA. As duas mensagens seguintes realizam a troca de chaves, com nonces para proteção contra repetição. Quando a chave de sessão tiver sido calculada, as duas partes trocarão as mensagens criptografadas que contêm informações de autenticação, como assinaturas digitais e, opcionalmente, validam as chaves públicas.

A **troca apenas de autenticação** é usada para realizar a autenticação mútua, sem uma troca de chaves. As duas primeiras mensagens definem a SA. Além disso, o respondedor usa a segunda mensagem para transportar sua ID e usa a autenticação para proteger a mensagem. O iniciador envia a terceira mensagem para transmitir sua ID autenticada.

A **troca agressiva** minimiza o número de trocas à custa de não oferecer proteção de identidade. Na primeira mensagem, o iniciador propõe uma SA com protocolos e opções de transformação oferecidos. O iniciador também inicia a troca de chaves e oferece sua ID. Na segunda mensagem, o respondedor indica sua aceitação da SA com um protocolo e transformação determinados, completa a troca de chaves e autentica as informações transmitidas. Na terceira mensagem, o iniciador transmite um resultado de autenticação que abrange as informações anteriores, criptografadas usando a chave de sessão secreta compartilhada.

A **troca informativa** é usada para a transmissão em única direção das informações para gerenciamento de SAs.

Tabela 16.4 Tipos de troca ISAKMP

Troca	Nota
(a) Troca básica	
(1) **I → R**: SA; NONCE	Inicia negociação ISAKMP-SA
(2) **R → I**: SA; NONCE	SA básica acordada
(3) **I → R**: KE; ID1; AUTH	Chave gerada; identidade do iniciador verificada pelo respondedor
(4) **R → I**: KE; IDR; AUTH	Identidade do respondedor verificada pelo iniciador; chave gerada; SA definida
(b) Troca de proteção de identidade	
(1) **I → R**: SA	Inicia negociação ISAKMP-SA
(2) **R → I**: SA	SA básica acordada
(3) **I → R**: KE; NONCE	Chave gerada
(4) **R → I**: KE; NONCE	Chave gerada
(5)* **I → R**: ID1; AUTH	Identidade do iniciador verificada pelo respondedor
(5)* **R → I**: IDR; AUTH	Identidade do respondedor verificada pelo iniciador; SA definida
(c) Troca apenas de autenticação	
(1) **I → R**: SA; NONCE	Inicia negociação ISAKMP-SA
(2) **R → I**: SA; NONCE; IDR; AUTH	SA básica acordada; identidade do respondedor verificada pelo iniciador
(3) **I → R**: ID1; AUTH	Identidade do iniciador verificada pelo respondedor; SA estabelecida
(d) Troca agressiva	
(1) **I → R**: SA; KE; NONCE; ID1	Inicia negociação ISAKMP-SA e troca de chaves
(2) **R → I**: SA; KE; NONCE; IDR; AUTH	Identidade do iniciador verificada pelo respondedor; chave gerada; SA básica acordada
(3)* **I → R**: AUTH	Identidade do respondedor verificada pelo iniciador; SA definida
(e) Troca informativa	
(1)* **I → R**: N/D	Erro ou notificação de status, ou exclusão

Legenda:
I = iniciador
R = respondedor
* = significa criptografia do payload após o cabeçalho ISAKMP
AUTH = mecanismo de autenticação utilizado

16.7 LEITURA E SITES WEB RECOMENDADOS

IPv6 e IPv4 são explicados com mais detalhes em [STAL04]. [CHEN98] oferece uma boa explicação sobre o projeto do IPSec. [FRAN01] e [DORA99] são explicações mais abrangentes do IPSec.

CHEN98 Cheng, P. et al. "A security architecture for the internet protocol." *IBM Systems Journal*, número 1, 1998.
DORA03 Doraswamy, N. e Harkins, D. *IPSec*. Upper Saddle River: Prentice Hall, 2003.
FRAN01 Frankel, S. *Demystifying the IPSec puzzle*. Boston: Artech House, 2001.
STAL04 Stallings, W. *Computer networking with Internet protocols and technology*. Upper Saddle River: Prentice Hall, 2004.

Site Web recomendado:

- **NIST IPSEC Project:** Contém artigos, apresentações e implementações de referência.

16.8 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

associação de segurança
ataques por repetição
autenticação do cabeçalho (AH)
encapsulamento de segurança do payload (ESP)
Internet Security Association and Key Management Protocol (ISAKMP)
IP Security (IPSec)
IPv4
IPv6
modo transporte
modo túnel
protocolo de determinação de chave Oakley
serviço anti-repetição

Perguntas de revisão

16.1 Dê exemplos de aplicações do IPSec.
16.2 Que serviços são fornecidos pelo IPSec?
16.3 Que parâmetros identificam uma SA e que parâmetros caracterizam a natureza de uma SA em particular?
16.4 Qual é a diferença entre o modo transporte e o modo túnel?
16.5 O que é um ataque por repetição?
16.6 Por que o ESP inclui um campo de complementação?
16.7 Quais são as técnicas básicas para as SAs combinadas?
16.8 Quais são as funções do protocolo de determinação de chave Oakley e o ISAKMP no IPSec?

Problemas

16.1 Na explicação do processamento do AH, foi mencionado que nem todos os campos em um cabeçalho de IP são incluídos no cálculo do MAC.
- **a.** Para cada um dos campos no cabeçalho IPv4, indique se o campo é imutável, mutável mas previsível, ou mutável (zerado antes do cálculo do ICV).
- **b.** Faça o mesmo para o cabeçalho IPv6.
- **c.** Faça o mesmo para os cabeçalhos de extensão do IPv6.

Em cada caso, justifique sua decisão para cada campo.

16.2 Quando o modo túnel é usado, um novo cabeçalho de IP externo é construído. Para IPv4 e IPv6, indique o relacionamento de cada campo de cabeçalho de IP externo e de cada cabeçalho de extensão no pacote externo com o campo ou com o cabeçalho de extensão correspondente do pacote de IP interno. Isto é, indique quais valores externos são obtidos de valores internos e quais são construídos independentemente dos valores internos.

16.3 A autenticação e a criptografia de ponta a ponta são desejadas entre dois hosts. Desenhe figuras semelhantes às figuras 16.6 e 16.9 que mostrem
- **a.** A adjacência de transporte, com a criptografia aplicada antes da autenticação.
- **b.** Uma SA de transporte agrupada dentro de um túnel SA, com criptografia aplicada antes da autenticação.
- **c.** Uma SA de transporte agrupada dentro de um túnel SA, com a autenticação aplicada antes da criptografia.

16.4 O documento da arquitetura IPSec declara que, quando duas SAs no modo transporte são agrupadas para permitir protocolos AH e ESP no mesmo fluxo de ponta a ponta, somente uma ordenação de protocolos de segurança parece apropriada: realizar o protocolo ESP antes de realizar o protocolo AH. Por que essa técnica é recomendada, em vez de autenticação antes da criptografia?

16.5
- **a.** Qual dos tipos de troca ISAKMP (Tabela 16.4) corresponde à troca de chaves Oakley agressiva (Figura 16.11)?
- **b.** Para a troca de chaves agressiva Oakley, indique quais parâmetros em cada mensagem vão em quais tipos de payload ISAKMP.

APÊNDICE 16A INTERLIGAÇÃO DE REDES E PROTOCOLOS DA INTERNET

Este apêndice oferece uma visão geral dos protocolos de interligação de redes. Começamos com um resumo do papel de um protocolo oferecendo interligação de redes. Depois, os dois protocolos de interligação de redes principais, IPv4 e IPv6, são apresentados.

O papel de um protocolo de interligação de redes

Um protocolo de Internet (IP) oferece a funcionalidade de interconectar sistemas finais por várias redes. Para essa finalidade, o IP é implementado em cada sistema final e nos roteadores, que são dispositivos que oferecem conexão entre as redes. Dados de nível mais alto em um sistema final de origem são encapsulados em uma unidade de protocolo de dados (Protocol Data Unit — PDU) de IP para transmissão. Essa PDU é então transmitida por uma ou mais redes e roteadores de conexão para alcançar o sistema final de destino.

O roteador precisa ser capaz de lidar com uma série de diferenças entre as redes, incluindo

- **Esquemas de endereçamento:** As redes podem usar diferentes esquemas para atribuir endereços aos dispositivos. Por exemplo, uma LAN IEEE 802 utiliza endereços binários de 16 ou 48 bits para cada dispositivo conectado; uma rede de comutação de pacotes pública X.25 utiliza endereços decimais de 12 dígitos (codificados como 4 bits por dígito para um endereço de 48 bits). Alguma forma de endereçamento de rede global precisa ser fornecida, além de um serviço de referência.
- **Tamanhos máximos de pacote:** Os pacotes de uma rede podem ter de ser divididos em pedaços menores para serem transmitidos por outra rede, em um processo conhecido como **fragmentação**. Por exemplo, a Ethernet impõe um comprimento de pacote máximo de 1.500 bytes; já um comprimento máximo de 1.000 bytes é comum em redes X.25. Um pacote transmitido em um sistema Ethernet e recebido por um roteador para retransmissão para uma rede X.25 pode ter de ser fragmentado em dois pacotes menores pelo roteador.
- **Interfaces:** As interfaces de hardware e software para várias redes diferem. O conceito de um roteador precisa ser independente dessas diferenças.
- **Confiabilidade:** Diversos serviços de rede podem oferecer qualquer coisa, desde um circuito virtual confiável ponta a ponta até um serviço não confiável. A operação dos roteadores não deve depender de uma suposição de confiabilidade da rede.

A operação de um roteador, como indica a Figura 16.13, depende de um protocolo de interligação de redes. Neste exemplo, o Internet Protocol (IP) do conjunto de protocolos TCP/IP realiza essa função. O IP precisa ser implementado em todos os sistemas de todas as redes, bem como nos roteadores. Além disso, cada sistema final precisa ter, em camadas acima do protocolo IP, protocolos compatíveis para que possam se comunicar com sucesso. Os roteadores intermediários só precisam ir até o IP.

Considere a transferência de um bloco de dados do sistema final *X* para o sistema final *Y* na Figura 16.13. A camada de IP em *X* recebe blocos de dados a serem enviados a *Y* a partir do TCP em X. A camada de IP anexa um cabeçalho que especifica o endereço global de Internet de *Y*. Esse endereço está em duas partes: identificador de rede

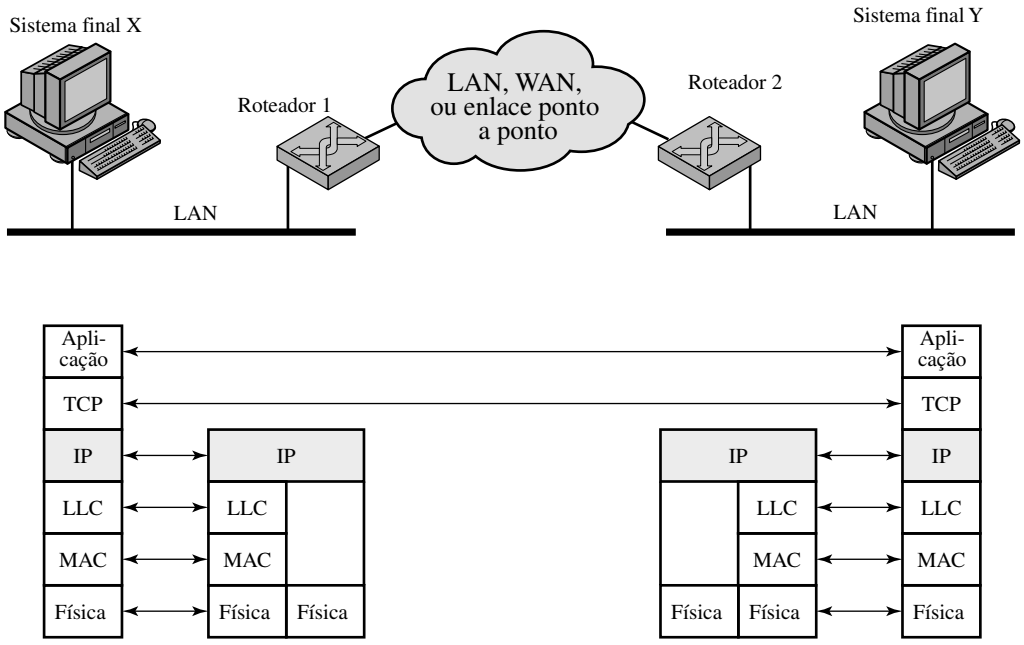


Figura 16.13 Configuração para exemplo de TCP/IP.

e identificador de sistema final. Vamos nos referir a esse bloco como pacote de IP. Em seguida, o IP reconhece o destino (*Y*) em uma outra sub-rede. Assim, o primeiro passo é enviar o pacote a um roteador, neste caso, o roteador 1. Para conseguir isso, o IP entrega sua unidade de dados, descendo, para o LLC com as informações de endereçamento apropriadas. O LLC cria uma PDU LLC, que é entregue para baixo para a camada MAC. A camada MAC constrói um pacote MAC cujo cabeçalho contém o endereço do roteador 1.

Em seguida, o pacote atravessa a LAN até o roteador 1. O roteador remove o pacote e os cabeçalhos e trailers LLC, e analisa o cabeçalho de IP para determinar o destino final dos dados, neste caso, *Y*. O roteador precisa agora tomar uma decisão de roteamento. Existem duas possibilidades:

1. O sistema final de destino *Y* está conectado diretamente a uma das sub-redes às quais o roteador está conectado.
2. Para alcançar o destino, é preciso atravessar um ou mais roteadores adicionais.

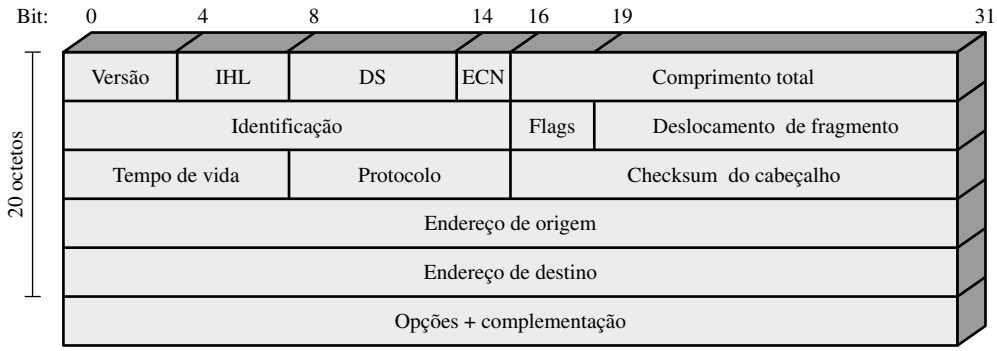
Neste exemplo, o pacote precisa ser roteado pelo roteador 2 antes de alcançar o destino. Assim, o roteador 1 passa o pacote de IP ao roteador 2 por meio da rede intermediária. Para essa finalidade, os protocolos dessa rede são utilizados. Por exemplo, se a rede intermediária for uma rede X.25, a unidade de dados de IP é quebrada em um pacote X.25 com informações de endereçamento apropriadas para alcançar o roteador 2. Quando esse pacote chega ao roteador 2, o cabeçalho do pacote é removido. O roteador determina que esse pacote de IP é destinado a *Y*, que está conectado diretamente à sub-rede à qual o roteador está conectado. O roteador, portanto, cria um pacote com um endereço de destino de *Y* e o envia para a LAN. Os dados finalmente chegam a *Y*, onde o pacote, LLC e os cabeçalhos de Internet e trailers podem ser removidos.

Esse serviço oferecido pelo IP é um serviço não confiável, isto é, o IP não garante que todos os dados sejam entregues ou que os dados que forem entregues chegarão na ordem correta. É de responsabilidade da próxima camada mais alta, neste caso, TCP, recuperar-se de quaisquer erros que ocorram. Essa técnica oferece muita flexibilidade. Como a entrega não é garantida, não existe requisito de confiabilidade em particular sobre qualquer uma das sub-redes. Assim, o protocolo funcionará com qualquer combinação de tipos de sub-rede. Como a seqüência de entrega não é garantida, pacotes sucessivos podem seguir caminhos diferentes pelas redes interligadas. Isso permite que o protocolo reaja ao congestionamento e falha na interligação alternando as rotas.

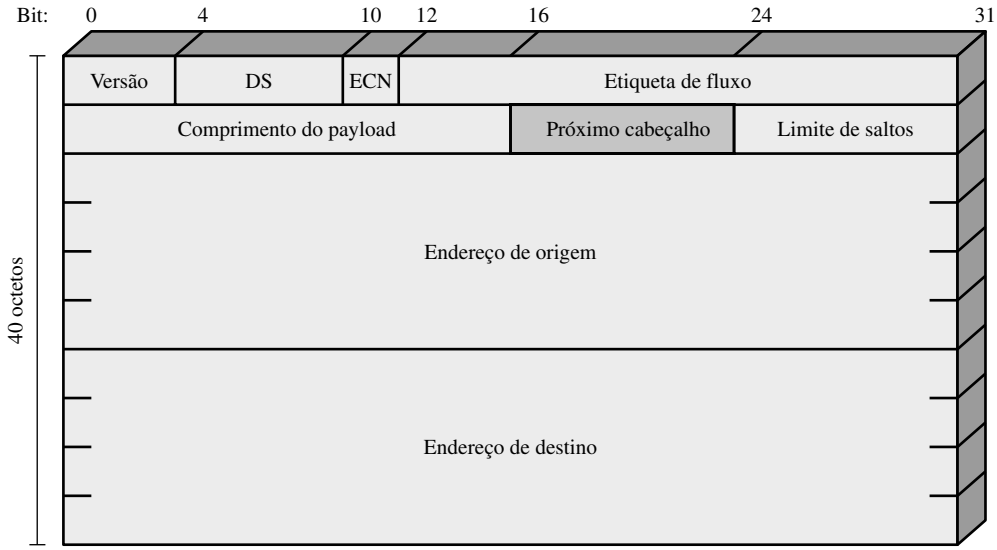
IPv4

Durante décadas, a pedra angular da arquitetura de protocolos TCP/IP tem sido o Internet Protocol (IP) versão 4. A Figura 16.14a mostra o formato do cabeçalho de IP, que tem um mínimo de 20 octetos, ou 160 bits. Os campos são os seguintes:

- **Versão (Version) (4bits):** Indica o número da versão, para permitir a evolução do protocolo; o valor é 4.
- **Comprimento do cabeçalho de Internet (Internet Header Length — IHL) (4 bits):** Comprimento do cabeçalho em palavras de 32 bits. O valor mínimo é de cinco, para um comprimento mínimo de cabeçalho de 20 octetos.
- **DS/ECN (8 bits):** Antes da introdução dos serviços diferenciados, esse campo era conhecido como o campo **Tipo de Serviço (Type of Service — TOS)** e especificava parâmetros de confiabilidade, precedência, atraso e processamento. Essa interpretação agora foi substituída. Os primeiros 6 bits do campo TOS agora são conhecidos como o campo DS (*Differentiated Services*). Os 2 bits restantes são reservados para um campo ECN (Explicit Congestion Notification).
- **Comprimento Total (Total Length) (16 bits):** Comprimento total do pacote IP, em octetos.
- **Identificação (Identification) (16 bits):** Um número de seqüência que, juntamente com o endereço de origem, endereço de destino e protocolo do usuário, serve para identificar um pacote exclusivamente. Assim, esse número deverá ser exclusivo para o endereço de origem do pacote, endereço de destino e protocolo do usuário para o período durante o qual o pacote permanecerá na Internet.
- **Flags (3 bits):** Somente dois dos bits estão atualmente definidos. Quando um pacote é fragmentado, o bit Mais indica se esse é o último fragmento no pacote original. O bit Não Fragmentar proíbe a fragmentação quando marcado. Esse bit pode ser útil se for conhecido que o destino não tem a capacidade de remontar fragmentos. Porém, se esse bit for marcado, o pacote será descartado se ultrapassar o comprimento máximo de uma sub-rede na rota. Portanto, se o bit for marcado, pode ser aconselhável usar o roteamento de origem para evitar sub-redes com um comprimento máximo de pacote pequeno.
- **Deslocamento de Fragmento (Fragment Offset) (13 bits):** Indica em que posição no pacote original esse fragmento aparece, medido em unidades de 64 bits. Isso implica que os fragmentos que não sejam o último precisam conter um campo de dados com um tamanho que seja um múltiplo de 64 bits .
- **Tempo de Vida (Time to Live)(8 bits):** Especifica por quanto tempo, em segundos, um pacote tem permissão para permanecer na Internet. Cada roteador que processa um pacote precisa diminuir o TTL em pelo menos um, de modo que o TTL é semelhante a um contador de saltos.



(a) Cabeçalho IPv4



(b) Cabeçalho IPv6

DS = Campo de serviços diferenciados
ECN = Campo de notificação explícita de congestionamento

Nota: Os campos de 8 bits DS/ECN eram conhecidos anteriormente como campo Tipo de Serviço no cabeçalho IPv4 e campo Classe de Tráfego no cabeçalho IPv6.

Figura 16.14 Cabeçalhos IP.

- **Protocolo (Protocol) (8 bits):** Indica o próximo protocolo da camada superior, o qual deve receber o campo de dados no destino; assim, esse campo identifica o tipo do próximo cabeçalho no pacote após o cabeçalho de IP.
- **Soma de verificação do cabeçalho (Header Checksum) (16 bits):** Um código de detecção de erro aplicado apenas ao cabeçalho. Como alguns campos do cabeçalho podem mudar durante o trânsito (por exemplo, tempo de vida, campos relacionados à segmentação), este é verificado novamente e recalculado em cada roteador. O campo de soma de verificação é uma adição, em complemento de 1, de todas as palavras de 16 bits no cabeçalho. Para fins de cálculo, o campo de soma de verificação é inicializado com um valor zero.
- **Endereço de origem (Source Address) (32 bits):** Codificado para permitir que uma alocação variável de bits especifique a rede e o sistema final conectado à rede especificada (7 e 24 bits, 14 e 16 bits, ou 21 e 8 bits).
- **Endereço de destino (Destination Address) (32 bits):** As mesmas características do endereço de origem.
- **Opções (Options) (variável):** Codifica as opções solicitadas pelo usuário de origem; estas podem incluir rótulo de segurança, roteamento de origem, registro de rota e carimbo de tempo.
- **Complementação (Padding) (variável):** Usado para garantir que o cabeçalho do pacote é um múltiplo de 32 bits em comprimento.

IPv6

Em 1995, a Internet Engineering Task Force (IETF), que desenvolve padrões de protocolo para a Internet, emitiu uma especificação para um IP da próxima geração, conhecido então como IPng. Essa especificação foi transformada

em um padrão em 1996, conhecido como IPv6. O IPv6 oferece diversos aprimoramentos funcionais em relação ao IP existente (conhecido como IPv4), projetado para acomodar as velocidades mais altas das redes de hoje e a mistura de fluxos de dados, incluindo gráfico e vídeo, que estão se tornando mais predominantes. Porém, o motivo principal por trás do desenvolvimento do novo protocolo foi a necessidade de mais endereços.

O IPv4 utiliza um endereço de 32 bits para especificar uma origem ou um destino. Com o crescimento explosivo da Internet e das redes privadas conectadas à Internet, esse comprimento de endereço tornou-se insuficiente para acomodar todos os sistemas que precisam de endereços. Como mostra a Figura 16.14b, o IPv6 inclui campos de endereço de origem e destino de 128 bits. No final, é esperado que todas as instalações usando TCP/IP migrem do IP atual para o IPv6, mas esse processo levará muitos anos, senão décadas.

Cabeçalho IPv6 O cabeçalho IPv6 possui um comprimento fixo de 40 octetos, consistindo nos seguintes campos (Figura 16.14b):

- **Versão (Version) (4 bits):** Número de versão do Internet Protocol; o valor é 6.
- **DS/ECN (8 bits):** Antes da introdução dos serviços diferenciados, esse campo era conhecido como **Classe de Tráfego (Traffic Class)** e era reservado para uso por nós de origem e/ou roteadores de encaminhamento, para identificar e distinguir entre diferentes classes ou prioridades de pacotes IPv6. Os seis primeiros bits do campo Classe de Tráfego agora são conhecidos como campo DS (Differentiated Services)). Os 2 bits restantes são reservados para um campo ECN (Explicit Congestion Notification).
- **Etiqueta de Fluxo (Flow Label)(20 bits):** Pode ser usado por um host para marcar os pacotes para os quais tratamento especial está sendo solicitado pelos roteadores dentro de uma rede. A marcação de fluxo pode auxiliar na reserva de recursos e no processamento de tráfego em tempo real.
- **Comprimento do payload (Payload Length) (16 bits):** Comprimento do restante do pacote IPv6 após o cabeçalho, em octetos. Em outras palavras, este é o comprimento total de todos os cabeçalhos de extensão mais a PDU do nível de transporte.
- **Próximo Cabeçalho (Next Header)(8 bits):** Identifica o tipo de cabeçalho imediatamente após o cabeçalho IPv6; este será um cabeçalho de extensão IPv6 ou um cabeçalho de camada superior, como TCP ou UDP.
- **Limite de saltos (Hop Limit)(8 bits):** O número restante de saltos permitidos para esse pacote. O limite de saltos é definido com algum valor máximo desejado pela origem e decrementado em 1 a cada nó que encaminha o pacote. O pacote será descartado se o Limite de saltos for decrementado até zero.
- **Endereço de Origem (Source Address)(128 bits):** O endereço do emissor do pacote.
- **Endereço de Destino (Destination Address)(128 bits):** O endereço do destinatário desejado do pacote. Este pode não ser realmente o destino final desejado, se um cabeçalho de extensão Roteamento (Routing) estiver presente, conforme explicamos adiante.

Embora o cabeçalho IPv6 seja maior que a parte obrigatória do cabeçalho IPv4 (40 octetos contra 20 octetos), ele contém menos campos (8 contra 12). Assim, os roteadores têm menos processamento a executar por cabeçalho, o que deverá agilizar o roteamento.

Cabeçalhos de extensão do IPv6 Um pacote IPv6 inclui o cabeçalho IPv6, que acabamos de explicar, e zero ou mais cabeçalhos de extensão. Fora do IPSec, os seguintes cabeçalhos de extensão foram definidos:

- **Cabeçalho de Opções Salto a Salto (Hop-by-Hop Options Header):** Define opções especiais que exigem processamento salto a salto
- **Cabeçalho de Roteamento (Routing Header):** Fornece roteamento estendido, semelhante ao roteamento de origem do IPv4
- **Cabeçalho de Fragmento (Fragment Header):** Contém informações de fragmentação e remontagem
- **Cabeçalho de Autenticação (Authentication Header):** Fornece integridade e autenticação de pacote
- **Cabeçalho de Encapsulamento de Segurança do Payload (Encapsulating Security Payload Header):** Fornece privacidade
- **Cabeçalho de opções de destino (Destination Options Header):** Contém informações opcionais a serem examinadas pelo nó de destino

O padrão IPv6 recomenda que, quando vários cabeçalhos de extensão forem usados, os cabeçalhos IPv6 apareçam na seguinte ordem:

1. Cabeçalho IPv6: obrigatório, sempre precisa aparecer primeiro
2. Cabeçalho de Opções Salto a Salto
3. Cabeçalho de Opções de Destino: para opções a serem processadas pelo primeiro destino que aparece no campo Endereço de Destino do IPv6 mais destinos subseqüentes listados no Cabeçalho de Roteamento

4. Cabeçalho de Roteamento
5. Cabeçalho de Fragmento
6. Cabeçalho de Autenticação
7. Cabeçalho de Encapsulamento de Segurança do Payload
8. Cabeçalho de Opções de Destino: para opções a serem processadas apenas pelo destino final do pacote

A Figura 16.15 mostra um exemplo do pacote IPv6 que inclui uma instância de cada cabeçalho que não seja relativo à segurança. Observe que o cabeçalho IPv6 e cada cabeçalho de extensão inclui o campo Próximo Cabeçalho. Esse campo identifica o tipo do cabeçalho imediatamente seguinte. Se o próximo cabeçalho for um cabeçalho de extensão, então esse campo conterá o identificador de tipo desse cabeçalho. Caso contrário, esse campo conterá o identificador de protocolo do protocolo da camada superior usando IPv6 (normalmente, um protocolo em nível de transporte), usando os mesmos valores do campo de Protocolo do IPv4. Na figura, o protocolo da camada superior é TCP, assim os dados da camada superior transportados pelo pacote IPv6 consistem em um cabeçalho TCP seguido por um bloco de dados da aplicação.

O **Cabeçalho de Opções Salto a Salto** transporta informações opcionais que, se presentes, precisam ser examinadas por cada roteador ao longo do caminho. O cabeçalho consiste nos seguintes campos:

- **Próximo Cabeçalho (8 bits):** Identifica o tipo de cabeçalho imediatamente após este cabeçalho.
- **Comprimento da Extensão do Cabeçalho (8 bits):** Comprimento deste cabeçalho em unidades de 64 bits, não incluindo os primeiros 64 bits.
- **Opções:** Contém uma ou mais opções. Cada opção consiste em três subcampos: uma etiqueta, indicando o tipo de opção, um comprimento e um valor.

Apenas uma opção até aqui foi definida: a opção Payload Jumbo, usada para enviar pacotes IPv6 com payloads maiores que $2^{16} - 1 = 65.535$ octetos. O campo de dados dessa opção tem 32 bits de extensão e oferece o comprimento do pacote em octetos, excluindo o cabeçalho IPv6. Para tais pacotes, o campo de comprimento do payload no cabeçalho IPv6 precisa ser definido como zero, e é preciso que não haja Cabeçalho de Fragmento. Com essa opção, o IPv6 suporta tamanhos de pacote com até mais de 4 bilhões de octetos. Isso facilita a transmissão de grandes pacotes de vídeo e permite que o IPv6 faça o melhor uso da capacidade disponível por qualquer meio de transmissão.

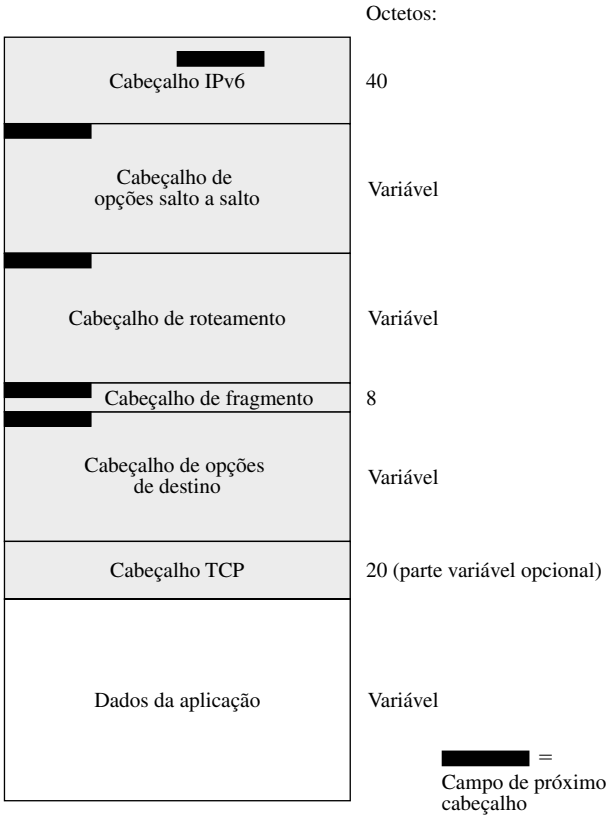


Figura 16.15 Pacote IPv6 com cabeçalhos de extensão (contendo um segmento TCP).

O **Cabeçalho de Roteamento** contém uma lista de um ou mais nós intermediários a serem visitados no caminho para o destino de um pacote. Todos os cabeçalhos de roteamento começam com um bloco de 32 bits consistindo em quatro campos de 8 bits, seguidos por dados de roteamento específicos para um determinado tipo de roteamento. Os quatro campos de 8 bits são Próximo Cabeçalho, Comprimento da Extensão do Cabeçalho e

- **Tipo de Roteamento:** Identifica determinada variante de Cabeçalho de Roteamento. Se um roteador não reconhecer o valor de Tipo de Roteamento, ele terá de descartar o pacote.
- **Segmentos Restantes:** Número de nós intermediários explicitamente listados ainda a serem visitados antes de alcançar o destino final.

Além dessa definição geral do cabeçalho, a especificação do IPv6 define o cabeçalho de Roteamento Tipo 0. Quando estiver usando o Cabeçalho de Roteamento Tipo 0, o nó de origem não coloca o endereço de destino final no cabeçalho IPv6. Em vez disso, esse endereço é o último endereço listado no Cabeçalho de Roteamento, e o cabeçalho IPv6 contém o endereço de destino do primeiro roteador desejado no caminho. O Cabeçalho de Roteamento não será examinado até que o pacote alcance o nó identificado no cabeçalho IPv6. Neste ponto, o conteúdo dos cabeçalhos IPv6 e de roteamento são atualizados e o pacote é encaminhado. A atualização consiste em colocar o próximo endereço a ser visitado no cabeçalho IPv6 e decrementar o campo Segmentos Restantes no Cabeçalho de Roteamento.

O IPv6 exige que um nó IPv6 reverta as rotas em um pacote recebido contendo um Cabeçalho de Roteamento, para retornar um pacote ao emissor.

O **Cabeçalho de Fragmento** é usado por uma origem quando a fragmentação é exigida. No IPv6, a fragmentação só pode ser realizada pelos nós de origem, não pelos roteadores ao longo do caminho de entrega de um pacote. Para tirar total vantagem do ambiente de interligação de redes, um nó precisa realizar um algoritmo de descoberta de caminho que permita descobrir a menor unidade de transmissão máxima (MTU — Maximum Transmission Unit) suportada por qualquer sub-rede no caminho. Em outras palavras, o algoritmo de descoberta de caminho permite que um nó 'aprenda' a MTU da sub-rede de 'gargalo' no caminho. Com esse conhecimento, o nó de origem será fragmentado, conforme exigido, para cada endereço de destino fornecido. Caso contrário, a origem precisará limitar todos os pacotes em 1.280 octetos, que é a MTU mínima que precisa ser suportada por cada sub-rede.

Além do campo Próximo Cabeçalho, o Cabeçalho de Fragmento inclui os seguintes campos:

- **Deslocamento de Fragmento (13 bits):** Indica em que posição no pacote original esse fragmento aparece. Este deslocamento é medido em unidades de 64 bits. Isso implica que os fragmentos (exceto o último fragmento) precisam conter um campo de dados que tenha um tamanho que seja um múltiplo de 64 bits.
- **Res (2 bits):** Reservado para uso futuro.
- **Flag M (1 bit):** 1 = mais fragmentos; 0 = último fragmento
- **Identificação (32 bits):** Serve para identificar exclusivamente o pacote original. O identificador precisa ser exclusivo para o endereço de origem e o endereço de destino do pacote pelo tempo durante o qual o pacote permanecerá na Internet. Todos os fragmentos com o mesmo identificador, endereço de origem e endereço de destino são remontados para formar o pacote original.

O **Cabeçalho de Opções de Destino** transporta informações opcionais que, se presentes, são examinadas apenas pelo nó de destino do pacote. O formato desse cabeçalho é o mesmo usado pelo Cabeçalho de Opções Salto a Salto.

Segurança na Web

17.1 **Considerações sobre segurança na Web**

17.2 **Secure Socket Layer e Transport Layer Security**

17.3 **Secure Electronic Transaction**

17.4 **Leitura e sites Web recomendados**

17.5 **Principais termos, perguntas de revisão e problemas**

Use sua mente.
Acorde para a realidade.

— Da canção "I've got you under my skin", de Cole Porter

PONTOS PRINCIPAIS

- O Secure socket layer (SSL) oferece serviços de segurança entre TCP e aplicações que usam TCP. A versão-padrão da Internet é chamada de Transport Layer Service (TLS).
- O SSL/TLS oferece confidencialidade usando criptografia simétrica e integridade de mensagens usando um código de autenticação de mensagens.
- O SSL/TLS inclui mecanismos de protocolo para permitir que dois usuários TCP determinem os mecanismos e serviços de segurança que eles usarão.
- O Secure Electronic Transaction (SET) é uma especificação aberta de criptografia e segurança, projetada para proteger transações com cartão de crédito na Internet.

Praticamente todas as empresas, a maioria dos órgãos do governo e muitos indivíduos agora possuem sites Web. O número de indivíduos e empresas com acesso à Internet está aumentando rapidamente e todos eles têm navegadores Web gráficos. Como resultado, as empresas demonstram grande interesse em utilizar a Web para comércio eletrônico. Mas a realidade é que a Internet e a Web são extremamente vulneráveis a riscos de vários tipos. À medida que as empresas percebem essa realidade, a demanda por serviços Web seguros aumenta.

O assunto de segurança na Web é muito amplo e pode facilmente ocupar um livro inteiro (vários são recomendados no final deste capítulo). Neste capítulo, começamos com uma explicação sobre os requisitos gerais para segurança na Web e depois focalizamos dois esquemas padronizados que estão se tornando cada vez mais importantes como parte do comércio na Web: SSL/TLS e SET.

17.1 CONSIDERAÇÕES SOBRE SEGURANÇA NA WEB

A World Wide Web é fundamentalmente uma aplicação cliente/servidor trabalhando sobre a Internet e intranets TCP/IP. Dessa maneira, as ferramentas de segurança e as técnicas explicadas até aqui neste livro são relevantes para a questão de segurança na Web. Mas, conforme indicado em [GARF97], a Web apresenta novos desafios geralmente não apreciados no contexto da segurança de computador e de rede.

- A Internet tem duas direções. Ao contrário dos ambientes de publicação tradicionais, até mesmo sistemas de publicação eletrônica envolvendo teletexto, resposta de voz, ou fax back, a Web é vulnerável a ataques aos servidores Web pela Internet.
- A Web está cada vez mais servindo como uma vitrine com alta visibilidade para as informações corporativas e de produtos e como plataforma para transações comerciais. Reputações podem ser prejudicadas e dinheiro pode ser perdido se os servidores Web forem subvertidos.
- Embora os navegadores Web sejam muito fáceis de usar, os servidores Web sejam relativamente fáceis de configurar e gerenciar e o conteúdo da Web esteja cada vez mais fácil de desenvolver, o software que dá suporte a tudo isso é extraordinariamente complexo. Esse software complexo pode ocultar muitas falhas de segurança em potencial. A história recente da Web está repleta de exemplos de sistemas novos e atualizados, devidamente instalados, que são vulneráveis a uma série de ataques à segurança.
- Um servidor Web pode ser explorado como uma plataforma de lançamento para o sistema de computadores de uma empresa ou de um órgão governamental. Quando o servidor Web é contaminado, um atacante pode ser capaz de obter acesso a dados e sistemas que não fazem parte da Web, mas que estão em máquinas conectadas localmente ao servidor.
- Os usuários ocasionais e não treinados (em questões de segurança) são clientes comuns de serviços baseados na Web. Esses usuários não estão necessariamente cientes dos riscos de segurança que existem e não possuem as ferramentas ou o conhecimento para tomar contramedidas eficazes.

Ameaças à segurança na Web

A Tabela 17.1 oferece um resumo dos tipos de ameaças à segurança enfrentadas no uso da Web. Uma maneira de agrupar essas ameaças é em termos de ataques passivos e ativos. Ataques passivos incluem acesso não autorizado ao tráfego de rede entre navegador e servidor e obtenção de acesso a informações em um site Web que deveria ser restrito. Ataques ativos incluem simulação de outro usuário, alteração de mensagens em trânsito entre cliente e servidor e alteração de informações em um site Web.

Outra maneira de classificar as ameaças de segurança na Web é em termos do local da ameaça: servidor Web, navegador Web e tráfego de rede entre navegador e servidor. As questões de segurança de servidor e navegador entram na categoria de segurança de sistema de computador; a Parte Quatro deste livro focaliza a questão da segurança de sistema em geral, mas também se aplica à segurança de sistemas Web. As questões de segurança de tráfego entram na categoria de segurança da rede e são focalizadas neste capítulo.

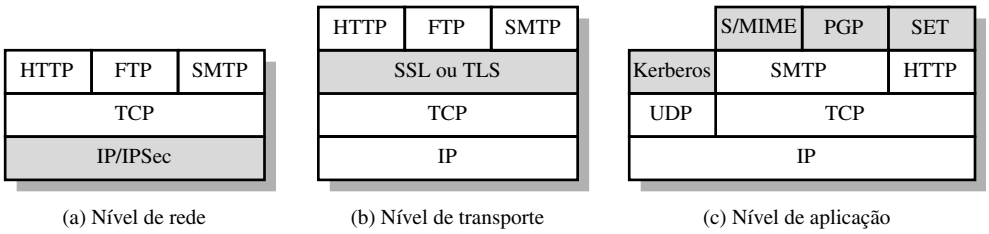
Técnicas de segurança de tráfego na Web

Diversas técnicas para oferecer segurança na Web são possíveis. As diversas técnicas que têm sido consideradas são semelhantes nos serviços que oferecem e, até certo ponto, nos mecanismos que utilizam, mas diferem com relação ao seu escopo de aplicabilidade e seu local relativo dentro da pilha de protocolos TCP/IP.

A Figura 17.1 ilustra essa diferença. Um modo de oferecer a segurança na Web é usar IP Security (Figura 17.1a). A vantagem de usar IPSec é que ele é transparente para os usuários finais e aplicações e oferece uma solução de uso geral. Além disso, o IPSec inclui uma capacidade de filtragem pela qual somente o tráfego selecionado precisa ser submetido à etapa adicional do processamento IPSec.

Tabela 17.1 Comparação de ameaças na Web [RUBI97].

	Ameaças	Conseqüências	Contramedidas
Integridade	• Modificação de dados do usuário • Navegador cavalo-de-tróia • Modificação de memória • Modificação de mensagens em trânsito	• Perda de informações • Comprometimento da máquina • Vulnerabilidade a todas as outras ameaças Somas de verificação (checksums) criptográficas	Somas de verificação (checksums) criptográficas
Confidencialidade	• Registro não autorizado de tráfego na rede • Roubo de informações do servidor • Roubo de dados do cliente • Informações sobre configurações de rede • Informações sobre qual cliente fala com o servidor	• Perda de informações • Perda de privacidade	Criptografia, proxies Web
Negação de serviço	• Encerramento de threads do usuário • Inundação da máquina com solicitações falsas • Preenchimento da capacidade total do disco ou da memória • Isolamento da máquina por ataques de DNS	• Interrupção • Incômodo • Impede que o usuário realize o trabalho	Difícil de impedir
Autenticação	• Simulação de usuários legítimos • Falsificação de dados	• Falsificação da identidade do usuário • Crença de que informações falsas são válidas	Técnicas criptográficas



(a) Nível de rede (b) Nível de transporte (c) Nível de aplicação

Figura 17.1 Local relativo das instalações de segurança na pilha de protocolos TCP/IP.

Outra solução de uso relativamente geral é implementar a segurança logo acima do TCP (Figura 17.1b). O exemplo principal dessa técnica é o Secure Sockets Layer (SSL) e seu padrão de Internet correspondente, conhecido como Transport Layer Security (TLS). Nesse nível, existem duas opções de implementação. Para a generalidade total, o SSL (ou TLS) pode ser fornecido como parte do conjunto de protocolos básico e, portanto, ser transparente às aplicações. Como alternativa, o SSL pode ser embutido em pacotes específicos. Por exemplo, os navegadores Netscape e Microsoft Explorer vêm equipados com SSL, e a maioria dos servidores Web tem implementado o protocolo.

Os serviços de segurança específicos de aplicação são embutidos na aplicação específica. A Figura 17.1c mostra exemplos dessa arquitetura. A vantagem dessa abordagem é que o serviço pode ser ajustado às necessidades específicas de determinada aplicação. No contexto da segurança na Web, um exemplo importante dessa técnica é o Secure Electronic Transaction (SET).[1]

O restante deste capítulo é dedicado a uma explicação sobre SSL/TLS e SET.

1. A Figura 17.1c mostra o SET em cima do HTTP; essa é uma implementação comum. Em algumas implementações, o SET utiliza o TCP diretamente.

17.2 SECURE SOCKET LAYER E TRANSPORT LAYER SECURITY

A Netscape criou o SSL. A versão 3 do protocolo foi projetada com revisão pública e do setor e foi publicada como um *draft* na Internet. Posteriormente, quando se chegou a um consenso para submeter o protocolo a padronização na Internet, um grupo de trabalho de TLS foi formado dentro do IETF para desenvolver um padrão comum. Essa primeira versão publicada do TLS pode ser vista como basicamente um SSLv3.1 e é muito próximo e compatível com o SSLv3.

A maior parte desta seção é dedicada a uma explicação do SSLv3. Ao final da seção, são descritas as principais diferenças entre SSLv3 e TLS.

Arquitetura SSL

O SSL é projetado para utilizar TCP para oferecer um serviço seguro confiável de ponta a ponta. O SSL não é um protocolo isolado, mas duas camadas de protocolo, conforme ilustra a Figura 17.2.

Figura 17.2 Pilha de protocolos SSL.

O Protocolo de Registro (Record Protocol) SSL oferece serviços básicos de segurança para vários protocolos de camada superior. Em particular, o Hypertext Transfer Protocol (HTTP), que oferece o serviço de transferência para interação cliente/servidor Web, pode operar em cima do SSL. Três protocolos de camada superior são definidos como parte do SSL: o Protocolo de Estabelecimento de Sessão (Handshake Protocol), o Protocolo de Mudança de Especificação de Cifra (Change Cipher Spec Protocol) e o Protocolo de Alerta (Alert Protocol). Esses protocolos específicos do SSL são usados no gerenciamento de trocas SSL e serão examinados mais adiante nesta seção.

Dois conceitos importantes do SSL são a sessão SSL e a conexão SSL, definidos na especificação da seguinte maneira:

- **Conexão:** Uma conexão é um transporte (na definição do modelo de camadas OSI) que oferece um tipo adequado de serviço. Para o SSL, essas conexões são relacionamentos peer-to-peer. As conexões são transientes. Cada conexão está associada a uma sessão.
- **Sessão:** Uma sessão SSL é uma associação entre um cliente e um servidor. As sessões são criadas pelo Protocolo de Estabelecimento de Sessão. As sessões definem um conjunto de parâmetros de segurança criptográficos que podem ser compartilhados entre várias conexões. As sessões são usadas para evitar a trabalhosa negociação de novos parâmetros de segurança para cada conexão.

Entre qualquer conjunto de duas partes (aplicações como HTTP no cliente e servidor), pode haver várias conexões seguras. Em teoria, também pode haver várias sessões simultâneas entre as partes, mas esse recurso não é usado na prática.

Na realidade, existem diversos estados associados a cada sessão. Quando uma sessão é estabelecida, existe um estado operacional para leitura e escrita (ou seja, recepção e envio). Além disso, durante o Protocolo de Estabelecimento de Sessão, são criados estados de leitura e escrita pendentes. Na conclusão bem-sucedida desse protocolo, os estados pendentes se tornam os estados atuais.

Um estado de sessão é definido pelos seguintes parâmetros (definições extraídas da especificação SSL):

- **Identificador de sessão (session identifier):** Uma seqüência arbitrária de bytes escolhida pelo servidor para identificar um estado de sessão ativo ou que pode ser retomado.
- **Certificado da parte (peer certificate):** Um certificado X509.v3 da parte. Esse elemento do estado pode ser nulo.

- **Método de compactação (compression method):** O algoritmo usado para compactar dados antes da criptografia.
- **Especificação de cifra (cipher spec):** Especifica o algoritmo de criptografia de dados em massa (sem criptografia, AES etc.) e um algoritmo de hash (como MD5 ou SHA-1) usado para o cálculo do MAC. Ela também define as propriedades criptográficas, como hash_size.
- **Segredo mestre (master secret):** Segredo de 48 bytes compartilhado entre o cliente e o servidor.
- **É retomável (is resumable):** Um flag indicando se a sessão pode ser usada para iniciar novas conexões.

Um estado de conexão é definido pelos parâmetros a seguir:

- **Aleatório de servidor e aleatório de cliente:** Seqüências de bytes escolhidas pelo servidor e cliente para cada conexão.
- **Segredo MAC de escrita do servidor:** A chave secreta usada nas operações MAC em dados enviados pelo servidor.
- **Segredo MAC de escrita do cliente:** A chave secreta usada em operações MAC em dados enviados pelo cliente.
- **Chave de escrita do servidor:** A chave de criptografia convencional para dados criptografados pelo servidor e decriptografados pelo cliente.
- **Chave de escrita do cliente:** A chave de criptografia convencional para dados criptografados pelo cliente e decriptografados pelo servidor.
- **Vetores de inicialização:** Quando uma cifra de bloco no modo CBC é usada, um vetor de inicialização (IV) é mantido para cada chave. Esse campo é inicializado primeiro pelo Protocolo de Estabelecimento de Sessão do SSL. Depois disso, o bloco de texto cifrado final de cada registro é preservado para uso como o IV com o registro seguinte.
- **Números de seqüência:** Cada parte mantém números de seqüência separados para mensagens transmitidas e recebidas para cada conexão. Quando uma parte envia ou recebe uma mensagem de mudança de especificação de cifra, o número de seqüência apropriado é definido como zero. Números de seqüência não podem exceder $2^{64} - 1$.

Protocolo de registro SSL

O Protocolo de Registro SSL oferece dois serviços para conexões SSL:

- **Confidencialidade:** O Protocolo de Estabelecimento de Sessão define uma chave secreta compartilhada usada para criptografia convencional dos dados a serem transportados pelo SSL.
- **Integridade da mensagem:** o Protocolo de Estabelecimento de Sessão também define uma chave secreta compartilhada usada para formar um código de autenticação de mensagem (MAC).

A Figura 17.3 indica a operação geral do protocolo de registro SSL. O protocolo de registro toma uma mensagem da aplicação a ser transmitida, fragmenta os dados em blocos gerenciáveis, comprime opcionalmente os dados, aplica um MAC, criptografa, acrescenta um cabeçalho e transmite a unidade resultante em um segmento TCP. Os dados recebidos são decriptografados, verificados, descompactados e remontados e, então, entregues a usuários de nível mais alto.

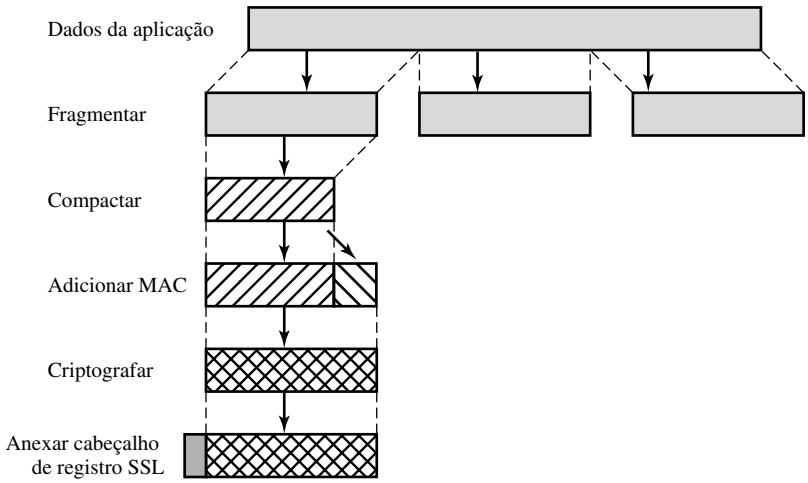


Figura 17.3 Operação do protocolo de registro SSL.

A primeira etapa é a **fragmentação**. Cada mensagem de camada superior é fragmentada em blocos de 2^{14} bytes (16384 bytes) ou menos. Em seguida, a **compactação** é aplicada opcionalmente. A compactação precisa ser sem perdas e não pode aumentar o tamanho do conteúdo em mais que 1.024 bytes.[2] No SSLv3 (além da versão atual do TLS), nenhum algoritmo de compactação é especificado, de modo que o algoritmo de compactação padrão é nulo (sem compactação).

A etapa seguinte no processamento é calcular um **código de autenticação de mensagens** sobre os dados compactados. Para essa finalidade, é usada uma chave secreta compartilhada. O cálculo é definido como

```
hash(MAC_write_secret || pad_2 ||
   hash(MAC_write_secret || pad_1 || seq_num ||
   SSLCompressed.type ||
                SSLCompressed.length || SSLCompressed.fragment))
```

onde

||	= concatenação
MAC_write_secret	= chave secreta compartilhada
hash	= algoritmo de hash criptográfico; ou MD5 ou SHA-1
pad_1	= o byte 0x36 (0011 0110) repetido 48 vezes (384 bits) para MD5 e 40 vezes (320 bits) para SHA-1
pad_2	= o byte 0x5C (0101 1100) repetido 48 vezes para MD5 e 40 vezes para SHA-1
seq_num	= o número de seqüência para essa mensagem
SSLCompressed.type	= o protocolo de nível mais alto usado para processar esse fragmento
SSLCompressed.length	= o tamanho do fragmento compactado
SSLCompressed.fragment	= o fragmento compactado (se a compactação não for usada, o fragmento de texto claro)

Observe que isso é muito semelhante ao algoritmo HMAC definido no Capítulo 12. A diferença é que os dois complementos são concatenados no SSLv3 e passam por um XOR no HMAC. O algoritmo MAC do SSLv3 é baseado no *draft* de Internet original para HMAC, que usava concatenação. A versão final do HMAC, definida na RFC 2104, utiliza o XOR.

Em seguida, a mensagem compactada mais o MAC são criptografados usando a criptografia simétrica. A criptografia não pode aumentar o tamanho do conteúdo em mais que 1.024 bytes, de modo que o tamanho total não pode exceder $2^{14} + 2.048$. Os algoritmos de criptografia a seguir são permitidos:

Cifra de bloco		Cifra de fluxo	
Algoritmo	**Tamanho da chave**	**Algoritmo**	**Tamanho da chave**
AES	128. 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza pode ser usado em um esquema de criptografia de smartcard.

Para a criptografia de fluxo, a mensagem compactada mais o MAC são criptografados. Observe que o MAC é calculado antes que a criptografia ocorra e que o MAC é então criptografado juntamente com o texto claro ou texto claro compactado.

Para a criptografia de bloco, a complementação pode ser acrescentada após o MAC antes da criptografia. A complementação está na forma de uma quantidade de bytes de complementação seguidos por uma indicação do comprimento da complementação de um byte. A quantidade total de complementação é a menor quantidade de modo que o tamanho total dos dados a serem criptografados (texto claro mais MAC mais preenchimento) seja um múltiplo do tamanho de bloco da cifra. Um exemplo é um texto claro (ou texto comprimido, se a compressão for usada) de 58

2. Naturalmente, espera-se que a compactação reduza ao invés de expandir os dados. Porém, para blocos muito curtos, é possível, devido a convenções de formatação, que o algoritmo de compactação resulte em uma saída que seja maior que a entrada.

bytes, com um MAC de 20 bytes (usando SHA-1), que é criptografado usando um tamanho de bloco de 8 bytes (por exemplo, DES). Com o byte padding.length, isso gera um total de 79 bytes. Para tornar o total um múltiplo inteiro de 8, um byte de preenchimento é acrescentado.

A última etapa do processamento do Protocolo de Registro SSL é anexar um cabeçalho no início, consistindo nos seguintes campos:

- **Tipo de conteúdo (content type)(8 bits):** O protocolo da camada mais alta, usado para processar o fragmento transportado.
- **Versão principal (major version)(8 bits):** Indica a versão principal do SSL em uso. Para SSLv3, o valor é 3.
- **Versão secundária (minor version)(8 bits):** Indica a versão secundária em uso. Para SSLv3, o valor é 0.
- **Tamanho compactado (compressed length) (16 bits):** O tamanho em bytes do fragmento de texto claro (ou fragmento compactado, se a compactação for usada). O valor máximo é $2^{14} + 2.048$.

Os tipos de conteúdo definidos são change_cipher_spec, alert, handshake e application_data. Os três primeiros são os protocolos específicos do SSL, explicados a seguir. Observe que nenhuma distinção é feita entre as diversas aplicações (por exemplo, HTTP) que poderiam usar SSL; o conteúdo dos dados criados por tais aplicações não é visível para o SSL.

A Figura 17.4 ilustra o formato do registro SSL.

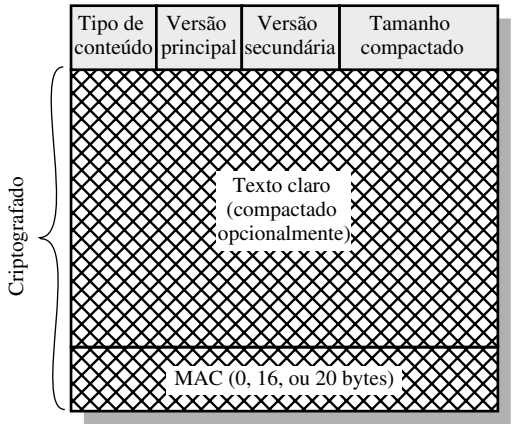


Figura 17.4 Formato do registro SSL.

Protocolo de mudança de especificação de cifra

O Protocolo de Mudança de Especificação de Cifra é um dos três protocolos específicos do SSL que utilizam o Protocolo de Registro SSL, e é o mais simples. Esse protocolo é composto por uma única mensagem (Figura 17.5a), que consiste em um único byte com o valor 1. A única finalidade dessa mensagem é fazer com que o estado pendente seja copiado para o estado atual, o que atualiza o conjunto de cifras a ser usado nessa conexão.

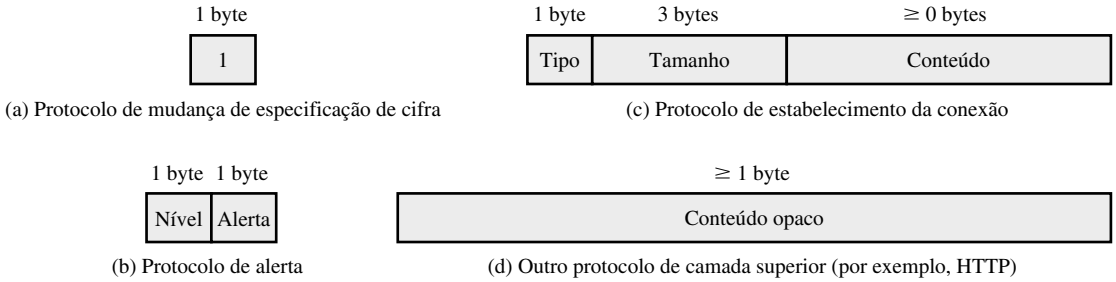


Figura 17.5 Payload do Protocolo de Registro SSL.

Protocolo de alerta

O Protocolo de Alerta é usado para transmitir alertas relacionados ao SSL para as partes envolvidas. Assim como outras aplicações que usam SSL, as mensagens de alerta são compactadas e criptografadas, conforme especificado pelo estado atual.

Cada mensagem nesse protocolo consiste em dois bytes (Figura 17.5b). O primeiro byte tem o valor warning(1) ou fatal(2) para transportar o grau de gravidade da mensagem. Se o nível for fatal, o SSL encerrará imediatamente a conexão. Outras conexões na mesma sessão podem continuar, mas nenhuma conexão nova nessa sessão pode ser estabelecida. O segundo byte contém um código que indica o alerta específico. Primeiramente, listamos os alertas que são sempre fatais (definições da especificação SSL).

- **unexpected_message:** uma mensagem não apropriada foi recebida.
- **bad_record_mac:** um MAC incorreto foi recebido.
- **decompression_failure:** a função de descompactação recebeu uma entrada imprópria (por exemplo, incapaz de descompactar ou descompactar para mais do que o tamanho máximo permitido).
- **handshake_failure:** o emissor foi incapaz de negociar um conjunto aceitável de parâmetros de segurança dadas as opções disponíveis.
- **illegal_parameter:** um campo em uma mensagem de estabelecimento de sessão estava fora de intervalo ou inconsistente com outros campos.

O restante dos alertas são os seguintes:

- **close_notify:** notifica o destinatário de que o emissor não enviará mais mensagens nessa conexão. Cada parte precisa enviar um alerta close_notify antes de fechar o lado de escrita de uma conexão.
- **no_certificate:** pode ser enviado em resposta a uma solicitação de certificado se nenhum certificado apropriado estiver disponível.
- **bad_certificate:** um certificado recebido foi adulterado (por exemplo, continha uma assinatura que não foi validada corretamente).
- **unsupported_certificate:** o tipo do certificado recebido não é admitido.
- **certificate_revoked:** um certificado foi revogado por seu assinante.
- **certificate_expired:** um certificado expirou.
- **certificate_unknown:** algum outro problema não especificado surgiu no processamento do certificado, tornando-o inaceitável.

Protocolo de estabelecimento de sessão

A parte mais complexa do SSL é Protocolo de Estabelecimento de Sessão. Esse protocolo permite que o servidor e o cliente autentiquem um ao outro e negociem um algoritmo de criptografia e de MAC e chaves criptográficas a serem usadas para proteger dados enviados em um registro SSL. O Protocolo de Estabelecimento de Sessão é usado antes que quaisquer dados de aplicação sejam transmitidos.

Esse protocolo é composto por uma série de mensagens trocadas entre cliente e servidor. Todas elas têm o formato mostrado na Figura 17.5c. Cada mensagem tem três campos:

- **Tipo (1 byte):** Indica uma de 10 mensagens. A Tabela 17.2 lista os tipos de mensagem definidos.
- **Tamanho (3 bytes):** O tamanho da mensagem em bytes.
- **Conteúdo (≥ 0 bytes):** Os parâmetros associados a essa mensagem; estes estão listados na Tabela 17.2.

Tabela 17.2 Tipos de mensagem do Protocolo de Estabelecimento de Sessão SSL

Tipo de mensagem	Parâmetros
hello_request	Nulo
client_hello	versão, aleatório, ID de sessão, conjunto de cifras, método de compactação
server_hello	versão, aleatório, ID de sessão, conjunto de cifras, método de compactação
certificate	cadeia de certificados X.509v3
server_key_exchange	parâmetros, assinatura
certificate_request	tipo, autoridades
server_done	Nulo
certificate_verify	Assinatura
client_key_exchange	parâmetros, assinatura
finished	valor de hash

A Figura 17.6 mostra a troca inicial necessária para estabelecer uma conexão lógica entre cliente e servidor. A troca pode ser vista como tendo quatro fases.

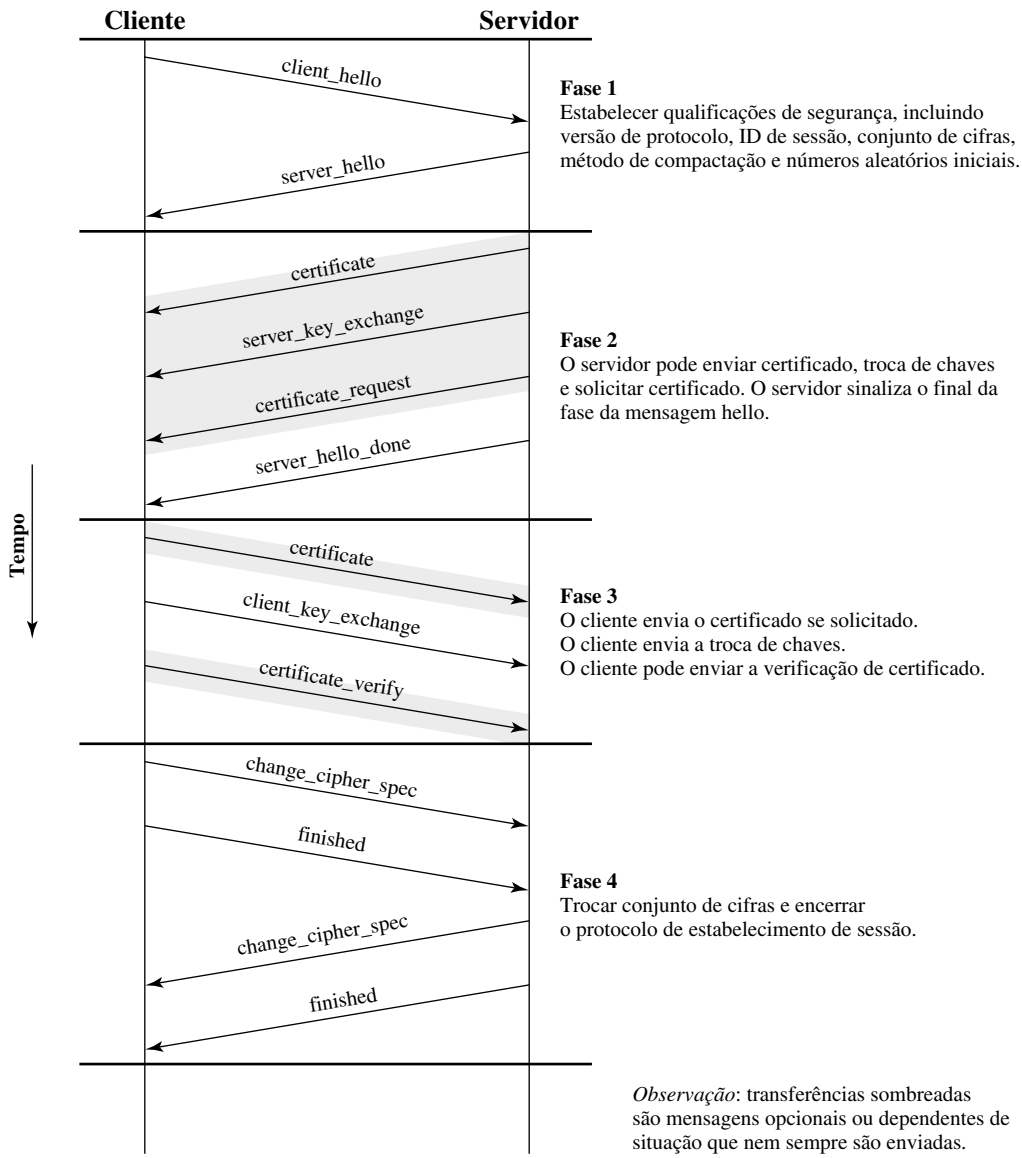


Figura 17.6 Ação do Protocolo de Estabelecimento de Sessão.

Fase 1. Estabelecer capacidades de segurança Esta fase é usada para iniciar uma conexão lógica e estabelecer as capacidades de segurança que serão associadas a ela. A troca é iniciada pelo cliente, que envia uma mensagem client_hello com os seguintes parâmetros:

- **Versão (version):** A versão SSL mais alta entendida pelo cliente.
- **Aleatório (random):** Uma estrutura aleatória gerada pelo cliente, consistindo em um carimbo de tempo de 32 bits e 28 bytes gerados por um gerador de números aleatórios seguro. Esses valores servem como nonces e são usados durante a troca de chaves para impedir ataques por repetição.
- **ID de sessão (session ID):** Um identificador de sessão de tamanho variável. Um valor diferente de zero indica que o cliente deseja atualizar os parâmetros de uma conexão existente ou criar uma nova conexão nesta sessão. Um valor zero indica que o cliente deseja estabelecer uma nova conexão em uma nova sessão.
- **Conjunto de cifras (CipherSuite):** Uma lista que contém as combinações de algoritmos criptográficos admitidos pelo cliente, em ordem decrescente de preferência. Cada elemento da lista (cada conjunto de cifras) define um algoritmo de troca de chaves e uma especificação de cifra; estas serão explicadas posteriormente.
- **Método de compactação (compression method):** Uma lista dos métodos de compactação que o cliente admite.

Depois de enviar a mensagem client_hello, o cliente espera pela mensagem server_hello, que contém os mesmos parâmetros da mensagem client_hello. Para a mensagem server_hello, as convenções a seguir se aplicam. O campo Versão contém a menor versão sugerida pelo cliente e a maior admitida pelo servidor. O campo Aleatório é gerado

pelo servidor e é independente do campo Aleatório do cliente. Se o campo ID de sessão do cliente for diferente de zero, o mesmo valor será usado pelo servidor; caso contrário, o campo ID de Sessão do servidor conterá o valor de uma nova sessão. O campo Conjunto de Cifras contém o único conjunto de cifras selecionado pelo servidor a partir daqueles propostos pelo cliente. O campo Método de Compactação contém o método de compactação selecionado pelo servidor a partir daqueles propostos pelo cliente.

O primeiro elemento do parâmetro Conjunto de Cifras é o método da troca de chaves (ou seja, o meio pelo qual as chaves criptográficas para a criptografia convencional e MAC são trocadas). Os seguintes métodos de troca de chaves são admitidos:

- **RSA:** A chave secreta é criptografada com a chave pública RSA do receptor. Um certificado de chave pública para a chave do receptor precisa ser disponibilizado.
- **Diffie-Hellman fixo:** Essa é uma troca de chaves Diffie-Hellman em que o certificado do servidor contém os parâmetros públicos Diffie-Hellman assinados pela autoridade certificadora (CA). Isto é, o certificado de chave pública contém os parâmetros de chave pública Diffie-Hellman. O cliente oferece esses parâmetros de chave pública Diffie-Hellman ou em um certificado, se a autenticação do cliente for exigida, ou em uma mensagem de troca de chaves. Esse método resulta em uma chave secreta fixa entre duas partes, com base no cálculo Diffie-Hellman usando as chaves públicas fixas.
- **Diffie-Hellman efêmero:** Essa técnica é usada para criar chaves secretas efêmeras (temporárias, de uso único). Nesse caso, as chaves públicas Diffie-Hellman são trocadas, assinadas usando a chave RSA ou DSS privada do emissor. O receptor pode usar a chave pública correspondente para verificar a assinatura. Os certificados são usados para autenticar as chaves públicas. Esta parece ser a mais segura das três opções Diffie-Hellman, pois resulta em uma chave temporária e autenticada.
- **Diffie-Hellman anônimo:** O algoritmo Diffie-Hellman básico é usado, sem autenticação. Ou seja, cada lado envia seus parâmetros Diffie-Hellman públicos para o outro, sem autenticação. Essa técnica é vulnerável a ataques de homem no meio (man-in-the-middle), em que o atacante realiza Diffie-Hellman anônimo com ambas as partes.
- **Fortezza:** A técnica definida para o esquema Fortezza.

Após a definição de um método de troca de chaves, existe a Especificação de Cifra (CipherSpec), que inclui os seguintes campos:

- **Algoritmo de cifra (CipherAlgorithm):** qualquer um dos algoritmos mencionados anteriormente: RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza
- **Algoritmo MAC (MACAlgorithm):** MD5 ou SHA-1
- **Tipo de cifra (CipherType):** fluxo ou bloco
- **É exportável (IsExportable):** Verdadeiro ou falso
- **Tamanho de hash (HashSize):** 0, 16 (para MD5), ou 20 (para SHA-1) bytes
- **Material da chave (Key Material):** uma seqüência de bytes que contém dados usados na geração das chaves de escrita
- **Tamanho do IV (IV size):** o tamanho do vetor de inicialização para a criptografia CBC (Cipher Block Chaining)

Fase 2. Autenticação de servidor e troca de chaves O servidor inicia essa fase enviando seus certificados, caso precise ser autenticado; a mensagem contém um ou uma cadeia de certificados X.509. A **mensagem de certificado (certificate message)** é exigida para qualquer método e troca de chaves que tenham sido acordados, exceto Diffie-Hellman anônimo. Observe que, se o Diffie-Hellman fixo for usado, essa mensagem de certificado funcionará como a mensagem de troca de chaves do servidor, pois contém os parâmetros Diffie-Hellman públicos do servidor.

Em seguida, uma mensagem **server_key_exchange** pode ser enviada, se for necessário. Ela não é exigida em dois casos: (1) O servidor enviou um certificado com parâmetros Diffie-Hellman fixos ou (2) a troca de chaves RSA será usada. A mensagem server_key_exchange é necessária para o seguinte:

- **Diffie-Hellman anônimo:** o conteúdo da mensagem consiste nos dois valores Diffie-Hellman globais (um número primo e uma raiz primitiva desse número) mais a chave Diffie-Hellman pública do servidor (ver Figura 10.7).
- **Diffie-Hellman efêmero:** o conteúdo da mensagem inclui os três parâmetros Diffie-Hellman fornecidos para Diffie-Hellman anônimo, mais uma assinatura desses parâmetros.
- **Troca de chaves RSA, em que o servidor está usando RSA, mas tem uma chave RSA apenas de assinatura:** por conseguinte, o cliente não pode simplesmente enviar uma chave secreta criptografada com a chave pública do servidor. Em vez disso, o servidor precisa criar um par temporário de chaves pública/privada RSA e usar a mensagem server_key_exchange para enviar a chave pública. O conteúdo da mensagem inclui os dois parâmetros da chave pública RSA temporária (expoente e módulo, ver Figura 9.5) mais uma assinatura desses parâmetros.
- **Fortezza**

Mais alguns detalhes sobre as assinaturas são garantidos. Como sempre, uma assinatura é criada tomando-se o hash de uma mensagem e criptografando-o com a chave privada do emissor. Nesse caso, o hash é definido como:

```
hash(ClientHello.random || ServerHello.random || ServerParams)
```

Assim, o hash abrange não apenas os parâmetros Diffie-Hellman ou RSA, mas também os dois nonces das mensagens hello iniciais. Esse procedimento fornece garantias contra ataques por repetição e falsificação de identidade. No caso de uma assinatura DSS, o hash é realizado usando-se o algoritmo SHA-1. No caso de uma assinatura RSA, tanto MD5 quanto um hash SHA-1 são calculados, e a concatenação dos dois hashes (36 bytes) é criptografada com a chave privada do servidor.

Em seguida, um servidor não anônimo (servidor não usando Diffie-Hellman anônimo) pode solicitar um certificado do cliente. A mensagem **certificate_request** inclui dois parâmetros: certificate_type e certificate_authorities. A mensagem certificate_type indica o algoritmo de chave pública e seu uso:

- RSA, somente assinatura
- DSS, somente assinatura
- RSA para Diffie-Hellman fixo; nesse caso, a assinatura só é usada para autenticação, enviando um certificado assinado com RSA
- DSS para Diffie-Hellman fixo; novamente, usado apenas para autenticação
- RSA para Diffie-Hellman efêmero
- DSS para Diffie-Hellman efêmero
- Fortezza

O segundo parâmetro na mensagem certificate_request é uma lista dos nomes distintos de autoridades certificadoras aceitáveis.

A mensagem final na Fase 2, e que é sempre exigida, é a **mensagem server_done**, que é enviada pelo servidor para indicar o final do hello do servidor e mensagens associadas. Depois de enviar essa mensagem, o servidor esperará pela resposta de um cliente. Essa mensagem não possui parâmetros.

Fase 3. Autenticação do cliente e troca de chaves Ao receber a mensagem server_done, o cliente deve verificar se o servidor forneceu um certificado válido, caso exigido, e verificar se os parâmetros server_hello são aceitáveis. Se tudo estiver satisfatório, o cliente envia uma ou mais mensagens de volta ao servidor.

Se o servidor tiver solicitado um certificado, o cliente iniciará essa fase enviando uma **mensagem certificate**. Se nenhum certificado apropriado estiver disponível, o cliente enviará um alerta no_certificate.

Em seguida vem a **mensagem client_key_exchange**, que precisa ser enviada nessa fase. O conteúdo da mensagem depende do tipo de troca de chaves, da seguinte forma:

- **RSA:** O cliente gera um *pré-segredo-mestre* (pre-master secret) de 48 bytes e o criptografa com a chave pública do certificado do servidor ou chave RSA temporária de uma mensagem server_key_exchange. Seu uso para calcular um *segredo-mestre* é explicado mais adiante.
- **Diffie-Hellman efêmero ou anônimo:** Os parâmetros Diffie-Hellman públicos do cliente são enviados.
- **Diffie-Hellman fixo:** Os parâmetros Diffie-Hellman públicos do cliente foram enviados em uma mensagem certificate, de modo que o conteúdo dessa mensagem é nulo.
- **Fortezza:** Os parâmetros Fortezza do cliente são enviados.

Finalmente, nessa fase, o cliente pode enviar uma mensagem certificate_verify para oferecer verificação explícita de um certificado do cliente. Essa mensagem só é enviada após qualquer certificado do cliente que tenha capacidade de assinatura (isto é, todos os certificados exceto aqueles contendo parâmetros Diffie-Hellman fixos). Essa mensagem assina um código de hash com base nas mensagens anteriores, definido da seguinte forma:

```
CertificateVerify.signature.md5_hash
   MD5(master_secret || pad_2 || MD5(handshake_messages ||
       master_secret || pad_1));
Certificate.signature.sha_hash
   SHA(master_secret || pad_2 || SHA(handshake_messages ||
       master_secret || pad_1));
```

onde pad_1 e pad_2 são os valores definidos anteriormente para o MAC, handshake_messages refere-se a todas as mensagens Protocolo de Estabelecimento de Sessão enviadas ou recebidas a partir de client_hello mas não incluindo essa mensagem, e master_secret é o segredo calculado cuja construção é explicada mais adiante nesta seção. Se a chave privada do usuário for DSS, então ela é usada para criptografar o hash SHA-1. Se a chave privada do usuário for RSA, ela é usada para criptografar a concatenação dos hashes MD5 e SHA-1. De qualquer forma, a finalidade é

verificar se o cliente possui a chave privada para o certificado do cliente. Mesmo que alguém estivesse usando o certificado do cliente sem autorização, ele não conseguiria enviar essa mensagem.

Fase 4. Término Esta fase completa a configuração de uma conexão segura. O cliente envia uma **mensagem change_cipher_spec** e copia o CipherSpec pendente para a CipherSpec atual. Observe que essa mensagem não é considerada parte do Protocolo de Estabelecimento de Sessão, mas sim enviada usando o Protocolo de Mudança de Especificação de Cifra. O cliente, então, envia imediatamente uma **finished message (mensagem de conclusão)** sob os novos algoritmos, chaves e segredos. A mensagem de conclusão verifica se os processos de troca de chaves e autenticação foram bem-sucedidos. O conteúdo da mensagem de conclusão é a concatenação de dois valores de hash:

```
MD5(master_secret || pad2 || MD5(handshake_messages ||
    Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
    Sender || master_secret || pad1))
```

onde Sender é um código que identifica que o emissor é o cliente e handshake_messages são todos os dados de todas as mensagens de estabelecimento de sessão até o momento, não incluindo esta mensagem.

Em resposta a essas duas mensagens, o servidor envia sua própria mensagem change_cipher_spec, transfere o CipherSpec pendente para a atual, e envia sua finished message. Nesse ponto, o estabelecimento da conexão está completo e cliente e servidor podem começar a trocar dados da camada de aplicação.

Cálculos criptográficos

Dois outros itens são interessantes: a criação de uma chave-mestra compartilhada por meio da troca de chaves e a geração de parâmetros criptográficos do segredo-mestre.

Criação de segredo-mestre O segredo-mestre (master secret) compartilhado é um valor de 48 bytes (384 bits) de uso único gerado para esta sessão por meio da troca de chaves segura. A criação é feita em dois estágios. Primeiro, um pre_master_secret é trocado. Segundo, o master_secret é calculado pelas duas partes. Para a troca de pre_master_secret, existem duas possibilidades:

- **RSA:** Um pre_master_secret de 48 bytes é gerado pelo cliente, criptografado com a chave RSA pública do servidor, e enviado ao servidor. O servidor decriptografa o texto cifrado usando sua chave privada para recuperar o pre_master_secret.
- **Diffie-Hellman:** Cliente e servidor geram uma chave pública Diffie-Hellman. Depois que estas forem trocadas, cada lado realizará o cálculo Diffie-Hellman para criar o pre_master_secret compartilhado.

Os dois lados agora calculam o master_secret da seguinte forma:

```
master_secret = MD5(pre_master_secret || SHA('A' ||
                    pre_master_secret ||ClientHello.random ||
                    ServerHello.random)) ||
                MD5(pre_master_secret || SHA('BB' ||
                    pre_master_secret || ClientHello.random ||
                    ServerHello.random)) ||
                MD5(pre_master_secret || SHA('CCC' ||
                    pre_master_secret || ClientHello.random ||
                    ServerHello.random))
```

onde ClientHello.random e ServerHello.random são os dois valores de nonce trocados nas mensagens hello iniciais.

Geração de parâmetros criptográficos O CipherSpecs requer um segredo MAC de escrita do cliente, um segredo MAC de escrita do servidor, uma chave de escrita do cliente, uma chave de escrita do servidor, um IV de escrita do cliente e um IV de escrita do servidor, que são gerados a partir do segredo-mestre, nesta ordem. Esses parâmetros são gerados a partir do segredo-mestre, fazendo o hashing desse valor e gerando uma seqüência de bytes de tamanho suficiente para todos os parâmetros necessários.

A geração dos parâmetros relativos a chaves e segredos a partir do segredo-mestre usa o mesmo formato utilizado na geração do segredo-mestre a partir do pré-segredo-mestre:

```
key_block = MD5(master_secret || SHA('A' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA('BB' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
```

```
MD5(master_secret || SHA('CCC' || master_secret ||
    ServerHello.random || ClientHello.random)) ||...
```

até que uma saída suficiente tenha sido gerada. O resultado dessa estrutura algorítmica é uma função pseudo-aleatória. Podemos pensar no master_secret como o valor de semente pseudo-aleatória para a função. Os números aleatórios de cliente e servidor podem ser vistos como salt values (valores de sal) para complicar a criptoanálise (veja no Capítulo 18 uma explicação sobre o uso dos salt values).

Transport Layer Security

O TLS é uma iniciativa de padronização do IETF cujo objetivo é produzir um padrão de Internet que seja uma versão do SSL. O TLS é definido como um Proposed Internet Standard (padrão proposto para a Internet) na RFC 2246, a qual é muito semelhante à SSLv3. Nesta seção, destacamos as diferenças.

Número de versão O formato do registro do TLS é o mesmo formato do registro do SSL (Figura 17.4), e os campos no cabeçalho têm os mesmos significados. A única diferença está nos valores de versão. Para a versão atual do TLS, a versão principal é 3 e a versão secundária é 1.

Message Authentication Code Existem duas diferenças entre os esquemas de MAC do SSLv3 e do TLS: o algoritmo real e o escopo do cálculo do MAC. O TLS utiliza o algoritmo HMAC definido na RFC 2104. Lembre-se, do Capítulo 12, de que o HMAC é definido da seguinte forma:

$$\text{HMAC}K(M) = \text{H}[(K^+ \oplus \text{opad})||\text{H}[K^+ \oplus \text{ipad})||M]]$$

onde

H	= função de hash embutida (para TLS, MD5 ou SHA-1)
M	= entrada de mensagem para HMAC
K^+	= chave secreta complementada com zeros à esquerda, de modo que o resultado é igual ao tamanho do bloco do código de hash (para MD5 e SHA-1, tamanho de bloco = 512 bits)
ipad	= 00110110 (36 em hexadecimal) repetido 64 vezes (512 bits)
opad	= 01011100 (5C em hexadecimal) repetido 64 vezes (512 bits)

O SSLv3 usa o mesmo algoritmo, exceto pelo fato de que os bytes de complementação são concatenados com a chave secreta em vez de realizar o XOR desses valores com a chave secreta complementada até o tamanho do bloco. O nível de segurança deverá ser praticamente o mesmo nos dois casos.

Para o TLS, o cálculo do MAC compreende os campos indicados na expressão a seguir:

```
HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
          TLSCompressed.version || TLSCompressed.length ||
          TLSCompressed.fragment)
```

O cálculo do MAC abrange todos os campos cobertos pelo cálculo do SSLv3, mais o campo TLSCompressed.version, que é a versão do protocolo que está sendo empregada.

Função pseudo-aleatória O TLS utiliza uma função pseudo-aleatória referenciada como PRF (Pseudo Random Function) para expandir segredos em blocos de dados para fins de geração ou validação de chave. O objetivo é utilizar um valor de segredo compartilhado relativamente pequeno, mas para gerar blocos de dados maiores de um modo que seja seguro contra os tipos de ataques feitos a funções de hash e MACs. O PRF é baseado na seguinte função de expansão de dados (Figura 17.7):

```
P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
                       HMAC_hash(secret, A(2) || seed) ||
                       HMAC_hash(secret, A(3) || seed) || ...
```

onde A() é definido como

$$\text{A}(0) = \text{semente}$$
$$\text{A}(i) = \text{HMAC_hash}(\text{segredo}, \text{A}(i-1))$$

A função de expansão de dados utiliza o algoritmo HMAC, seja com MD5 ou SHA-1 como função de hash básica. Como podemos ver, P_hash pode ser repetido tantas vezes quantas forem necessárias para produzir a quantidade de dados exigida. Por exemplo, se P_SHA-1 fosse usado para gerar 64 bytes de dados, ele teria de ser repetido quatro vezes, produzindo 80 bytes de dados, dos quais os últimos 16 seriam descartados. Nesse caso, P_MD5 também teria de ser repetido quatro vezes, produzindo exatamente 64 bytes de dados. Observe que cada repetição envolve duas execuções do HMAC, cada uma por sua vez envolvendo duas execuções do algoritmo de hash embutido.

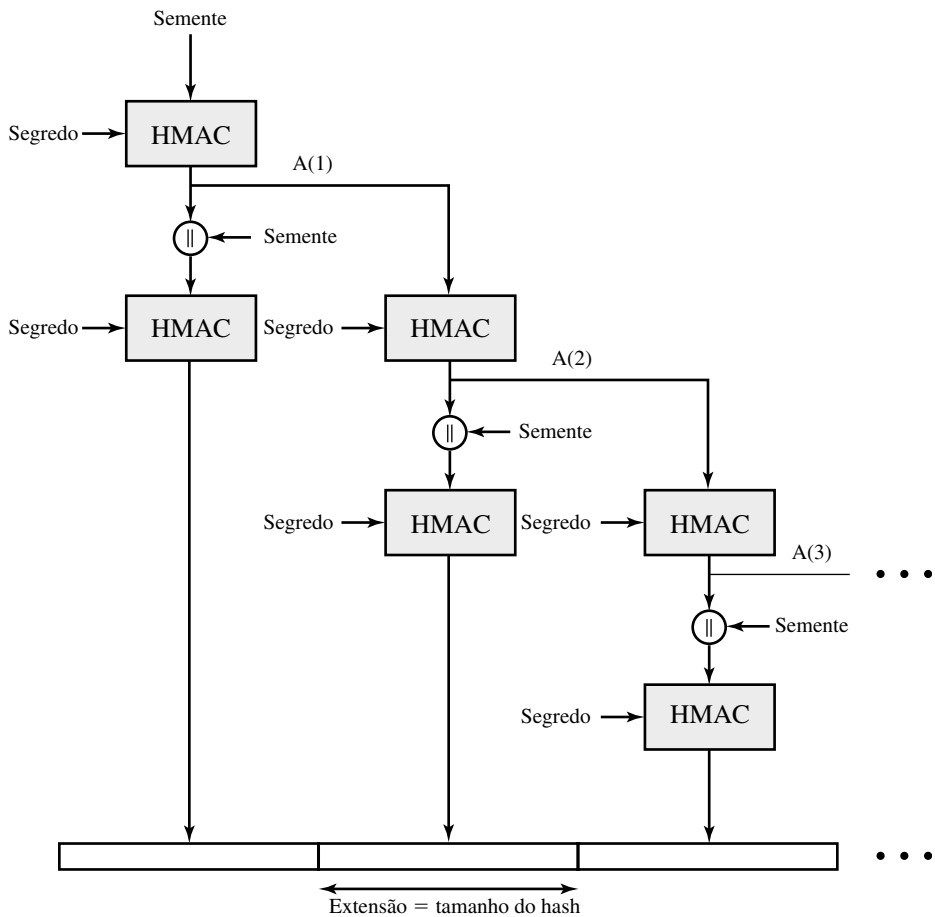


Figura 17.7 Função P_hash(segredo,semente) do TLS.

Para tornar a PRF tão segura quanto possível são utilizados dois algoritmos de hash, de modo a garantir a segurança da PRF se, pelo menos, um dos algoritmos permanecer seguro. A PRF é definida como

```
PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕
                           P_SHA-1(S2, label || seed)
```

Essa função tem como entrada um valor secreto, um rótulo de identificação e um valor de semente. Ela produz uma saída de tamanho arbitrário. A saída é criada dividindo-se o valor secreto em duas metades (S1 e S2) e realizando-se o P_hash em cada metade, usando MD5 em uma metade e SHA-1 na outra. Os dois resultados passam por um XOR para produzir a saída; para essa finalidade, P_MD5 geralmente terá de ser repetido mais vezes do que P_SHA-1 para produzir uma quantidade igual de dados para a entrada na função XOR.

Códigos de alerta O TLS admite todos os códigos de alerta definidos na SSLv3, com exceção de no_certificate. Diversos códigos adicionais são definidos no TLS; destes, os seguintes são sempre fatais:

- **decryption_failed:** um texto cifrado de uma maneira inválida; ou ele não era um múltiplo par do tamanho do bloco ou seus valores de complementação, quando verificados, estavam incorretos.
- **record_overflow:** um registro do TLS foi recebido com um payload (texto cifrado) cujo tamanho ultrapassa 2^{14} + 2.048 bytes, ou o texto cifrado foi decriptografado para um tamanho maior que 2^{14} + 1.024 bytes.
- **unknown_ca:** uma cadeia de certificados válida ou cadeia parcial foi recebida, mas o certificado não foi aceito porque o certificado da CA não pôde ser localizado ou não correspondeu a nenhuma CA conhecida e confiável.
- **access_denied:** um certificado válido foi recebido, mas quando a conectividade de acesso foi aplicada, o emissor decidiu não prosseguir com a negociação.
- **decode_error:** uma mensagem não pôde ser decodificada, pois um campo estava fora do seu intervalo especificado ou o tamanho da mensagem foi incorreto.

- **export_restriction:** foi detectada uma negociação que não estava em conformidade com as restrições de exportação sobre o tamanho da chave.
- **protocol_version:** a versão do protocolo que o cliente tentou negociar é reconhecida, mas não é aceita.
- **insufficient_security:** retornado no lugar de handshake_failure quando uma negociação falhou especificamente porque o servidor exige cifras mais seguras do que aquelas admitidas pelo cliente.
- **internal_error:** um erro interno não relacionado à outra parte ou à exatidão do protocolo torna impossível continuar.

O restante dos novos alertas inclui o seguinte:

- **decrypt_error:** uma operação criptográfica de estabelecimento de sessão falhou, o que inclui ser incapaz de verificar uma assinatura, decriptografar uma troca de chaves ou validar uma mensagem acabada.
- **user_canceled:** este estabelecimento de conexão está sendo cancelado por algum motivo não relacionado a uma falha de protocolo.
- **no_renegotiation:** enviado por um cliente em resposta a uma solicitação hello ou pelo servidor em resposta a um hello do cliente após o estabelecimento inicial da sessão. Uma mensagem dessas normalmente resultaria em renegociação, mas esse alerta indica que o emissor não é capaz de renegociar. Essa mensagem é sempre uma advertência.

Conjuntos de cifras Existem várias diferenças pequenas entre os conjuntos de cifras disponíveis no SSLv3 e no TLS:

- **Troca de chaves:** O TLS admite todas as técnicas de troca de chaves do SSLv3, com exceção do Fortezza.
- **Algoritmos de criptografia simétrica:** o TLS inclui todos os algoritmos de criptografia simétrica encontrados no SSLv3, com exceção do Fortezza.

Tipos de certificado do cliente O TLS define os seguintes tipos de certificado a serem solicitados em uma mensagem certificate_request: rsa_sign, dss_sign, rsa_fixed_dh e dss_fixed_dh. Todos esses são definidos no SSLv3. Além disso, o SSLv3 inclui rsa_ephemeral_dh, dss_ephemeral_dh e fortezza_kea. O Diffie-Hellman efêmero envolve a assinatura dos parâmetros Diffie-Hellman com RSA ou DSS; para o TLS, os tipos rsa_sign e dss_sign são usados para essa função; um tipo de assinatura separado não é necessário para assinar parâmetros Diffie-Hellman. O TLS não inclui o esquema Fortezza.

Mensagens certificate_verify e finished Na mensagem certificate_verify, os hashes MD5 e SHA-1 são calculados apenas sobre handshake_messages. Lembre-se de que, para o SSLv3, o cálculo de hash também inclui o segredo-mestre e complementações. Esses campos extras não representam segurança adicional.

Assim como a finished message (mensagem de conclusão) no SSLv3, a finished message no TLS é um hash com base no master_secret compartilhado, as mensagens de estabelecimento de sessão (handshake_messages) anteriores, e um rótulo que identifica o cliente ou servidor. O cálculo é um pouco diferente. Para o TLS, temos

```
PRF(master_secret, finished_label, MD5(handshake_messages)||
    SHA-1(handshake_messages))
```

onde finished_label é a string “client finished” para o cliente e “server finished” para o servidor.

Cálculos criptográficos O pre_master_secret para o TLS é calculado da mesma maneira que no SSLv3. Assim como no SSLv3, o master_secret no TLS é calculado como uma função de hash do pre_master_secret e os dois números aleatórios de hello. A forma do cálculo do TLS é diferente da usada no SSLv3, e é definida da seguinte forma:

```
master_secret = PRF(pre_master_secret, “master secret”,
                ClientHello.random || ServerHello.random)
```

O algoritmo é executado até que 48 bytes de saída pseudo-aleatória sejam produzidos. O cálculo dos parâmetros relativos a chaves e segredos (chaves secretas MAC, chaves de criptografia de sessão e IVs) é definido assim:

```
key_block = PRF(master_secret, “key expansion”,
            SecurityParameters.server_random ||
            SecurityParameters.client_random)
```

até que uma saída suficiente tenha sido gerada. Assim como no SSLv3, key_block é uma função de master_secret e os números aleatórios de cliente e servidor, mas, para o TLS, o algoritmo real é diferente.

Complementação No SSL, a complementação acrescentada antes da criptografia de dados do usuário é a quantidade mínima exigida para que o tamanho total dos dados a serem criptografados seja um múltiplo do tamanho de

bloco da cifra. No TLS, a complementação pode ser qualquer quantidade que resulte em um total que seja um múltiplo do tamanho de bloco da cifra, até um máximo de 255 bytes. Por exemplo, se o texto claro (ou texto compactado, se for usada compactação) mais MAC mais bytes do tamanho do preenchimento tiver 79 bytes de extensão, então o tamanho do preenchimento, em bytes, pode ser 1, 9, 17 e assim por diante, até 249. Um tamanho de complementação variável pode ser usado para frustrar ataques com base em uma análise dos tamanhos das mensagens trocadas.

17.3 SECURE ELECTRONIC TRANSACTION

O SET é uma especificação aberta de criptografia e segurança, criada para proteger transações envolvendo cartões de crédito pela Internet. A versão atual, SETv1, surgiu de um pedido de padrões de segurança pela MasterCard e Visa em fevereiro de 1996. Diversas empresas estiveram envolvidas no desenvolvimento da especificação inicial, incluindo IBM, Microsoft, Netscape, RSA, Terisa e Verisign. A partir de 1996 ocorreram diversos testes de conceito e por volta de 1998 a primeira onda de produtos compatíveis com o SET estava disponível.

O SET não é, por si só, um sistema de pagamentos. Em vez disso, é um conjunto de protocolos e formatos de segurança que permite que os usuários empreguem a infra-estrutura de pagamento por cartão de crédito existente em uma rede aberta, como a Internet, de modo seguro. Basicamente, o SET oferece três serviços:

- Um canal de comunicações seguro entre todas as partes envolvidas em uma transação
- Confiança no uso de certificados digitais X.509v3
- Privacidade, pois as informações só estão disponíveis às partes em uma transação quando e onde forem necessárias

O SET é uma especificação complexa, definida em três livros emitidos em maio de 1997:

- **Livro 1:** Descrição do negócio (80 páginas)
- **Livro 2:** Guia do programador (629 páginas)
- **Livro 3:** Definição formal do protocolo (262 páginas)

Isso compreende um total de 971 páginas de especificação. Em contraste, a especificação do SSLv3 tem 63 páginas e a especificação do TLS tem 80 páginas. Por conseguinte, nesta seção, fornecemos apenas um resumo dessa especificação de muitas facetas.

Visão geral do SET

Uma boa maneira de começar nossa explicação do SET é examinar os requisitos de negócio do SET, seus principais recursos e os participantes das transações SET.

Requisitos O Livro 1 da especificação SET lista os seguintes requisitos de negócio para o processamento de pagamento seguro com cartões de crédito pela Internet e outras redes:

- **Fornecer confidencialidade das informações de pagamento e de pedido:** É necessário garantir aos proprietários do cartão que essas informações são seguras e acessíveis apenas pelo destinatário pretendido. A confidencialidade também reduz o risco de fraude por uma das partes da transação ou por terceiros maliciosos. O SET utiliza criptografia para fornecer confidencialidade.
- **Garantir a integridade de todos os dados transmitidos:** Isto é, garantir que nenhuma mudança no conteúdo ocorra durante a transmissão de mensagens SET. Assinaturas digitais são usadas para fornecer integridade.
- **Fornecer autenticação de que o proprietário do cartão é um usuário legítimo de uma conta de cartão de crédito:** Um mecanismo que vincula o proprietário do cartão a um número de conta específico reduz a incidência de fraudes e o custo geral de processamento do pagamento. Assinaturas e certificados digitais são usados para verificar se um proprietário de cartão é um usuário legítimo de uma conta válida.
- **Oferecer autenticação de que um comerciante pode aceitar transações de cartão de crédito por meio de seu relacionamento com uma instituição financeira:** Esse é um complemento do requisito anterior. Os proprietários de cartão precisam ser capazes de identificar os comerciantes com quem podem realizar transações seguras. Novamente, são usados assinaturas e certificados digitais.
- **Garantir o uso das melhores práticas de segurança e das melhores técnicas de projeto de sistemas para proteger todas as partes legítimas em uma transação de comércio eletrônico:** O SET é uma especificação bem testada, com base em algoritmos criptográficos e protocolos altamente seguros.
- **Criar um protocolo que não dependa dos mecanismos de segurança de transporte nem impeça seu uso:** O SET pode operar com segurança sobre uma pilha TCP/IP 'bruta'. Porém, o SET não interfere no uso de outros mecanismos de segurança, como IPSec e SSL/TLS.
- **Facilitar e encorajar a interoperabilidade entre provedores de software e rede:** Os protocolos e formatos SET são independentes da plataforma de hardware, sistema operacional e software Web.

Principais recursos do SET

Para atender os requisitos esboçados acima, o SET incorpora os seguintes recursos:

- **Confidencialidade de informações:** Informações da conta do proprietário do cartão e do pagamento são protegidas enquanto trafegam pela rede. Um recurso interessante e importante do SET é que ele impede que o comerciante descubra o número de cartão de crédito do proprietário do cartão; isso só é fornecido ao banco emissor. A criptografia convencional pelo DES é usada para fornecer confidencialidade.
- **Integridade dos dados:** As informações de pagamento enviada dos proprietários de cartão para os comerciantes incluem informações do pedido, dados pessoais e instruções de pagamento. O SET garante que esse conteúdo de mensagem não seja alterado em trânsito. Assinaturas digitais RSA, incluindo códigos de hash SHA-1, oferecem integridade de mensagem. Certas mensagens também são protegidas por HMAC usando SHA-1.
- **Autenticação de conta do proprietário do cartão:** O SET permite que os comerciantes verifiquem se um proprietário de cartão é um usuário legítimo de um número de conta de cartão válido. O SET utiliza certificados digitais X.509v3 com assinaturas RSA para essa finalidade.
- **Autenticação do comerciante:** O SET permite que os proprietários de cartão verifiquem se um comerciante tem um relacionamento com uma instituição financeira, permitindo que ele aceite cartões de pagamentos. O SET utiliza certificados digitais X.509v3 com assinaturas RSA para essa finalidade.

Observe que, diferentemente de IPSec e SSL/TLS, o SET oferece apenas uma escolha para cada algoritmo criptográfico. Isso faz sentido, pois o SET é uma aplicação isolada, com um conjunto de requisitos exclusivo, enquanto o IPSec e o SSL/TLS buscam oferecer suporte a uma grande variedade de aplicações.

Participantes do SET A Figura 17.8 indica os participantes do sistema SET, que incluem os seguintes:

- **Proprietário do cartão:** No ambiente eletrônico, consumidores e compradores corporativos interagem com comerciantes a partir de computadores pessoais pela Internet. Um proprietário de cartão é um proprietário autorizado de um cartão de pagamentos (por exemplo, MasterCard, Visa) emitido por um emissor.
- **Comerciante:** Um comerciante é uma pessoa ou organização que possui bens ou serviços para vender ao proprietário do cartão. Normalmente, esses bens e serviços são oferecidos por meio de um site Web ou por e-mail. Um comerciante que aceita cartões de pagamentos precisa ter um relacionamento com um acquirer.
- **Emissor:** A instituição financeira, como um banco, que oferece um cartão de pagamentos ao proprietário do cartão. Normalmente, as contas são solicitadas e abertas por correio ou pessoalmente. Em última instância, é o emissor é o responsável pelo pagamento do débito do proprietário do cartão.
- **Acquirer:** Instituição financeira que estabelece uma conta com um comerciante e processa autorizações de cartão de pagamentos e pagamentos. Os comerciantes normalmente aceitarão mais de uma bandeira de cartão de crédito, mas não desejam lidar com as várias redes de cartão bancário ou com os vários emissores individuais. O acquirer oferece autenticação ao comerciante, indicando que determinada conta de cartão está ativa e que a compra proposta não ultrapassa o limite de crédito. O acquirer também oferece transferência eletrônica de pagamentos à conta do comerciante. Posteriormente, o acquirer é reembolsado pelo emissor, por meio de algum tipo de rede de pagamentos para transferência eletrônica de fundos.
- **Gateway de pagamento:** Função operada pelo acquirer ou por um terceiro designado que processa mensagens de pagamento do comerciante. O gateway de pagamento realiza a interface entre o SET e as redes de pagamento de cartão bancário existentes, para funções de autorização e pagamento. O comerciante troca mensagens SET com o gateway de pagamento pela Internet, enquanto o gateway de pagamento possui alguma conexão direta ou de rede com o sistema de processamento financeiro do acquirer.
- **Autoridade certificadora (CA):** Entidade confiável para emitir certificados de chave pública X.509v3 para proprietários de cartão, comerciantes e gateways de pagamento. O sucesso do SET dependerá da existência de uma infra-estrutura de CA disponível para essa finalidade. Conforme explicamos nos capítulos anteriores, é utilizada uma hierarquia de CAs, de modo que os participantes não precisam ser certificados diretamente por uma autoridade-raiz.

Agora, descrevemos rapidamente a seqüência de eventos exigidos para uma transação. Depois, veremos alguns dos detalhes criptográficos.

1. **O cliente abre uma conta.** O cliente obtém uma conta de cartão de crédito, por exemplo, MasterCard ou Visa, com um banco que aceita pagamentos eletrônicos e SET.
2. **O cliente recebe um certificado.** Após a verificação apropriada da identidade, o cliente recebe um certificado digital X.509v3, o qual é assinado pelo banco. O certificado atesta a chave pública RSA do cliente e sua data de expiração. Ele também estabelece uma correlação, garantida pelo banco, entre o par de chaves do cliente e seu cartão de crédito.

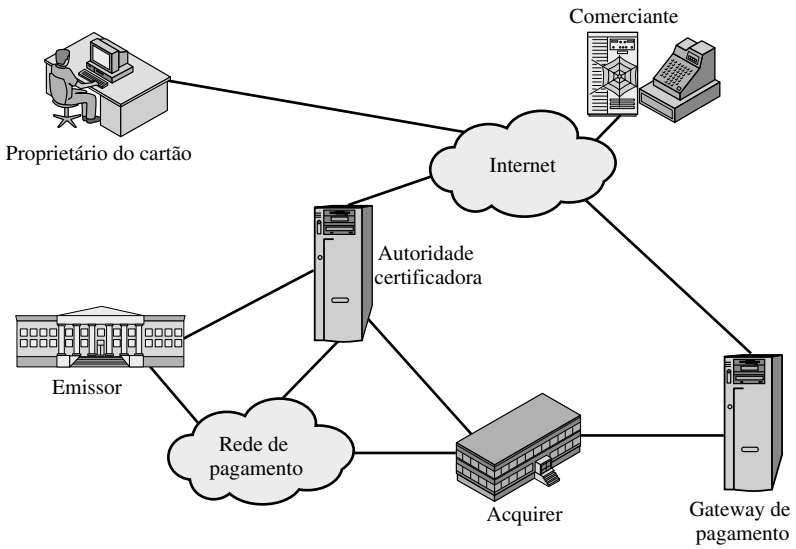


Figura 17.8 Componentes do comércio eletrônico seguro.

3. **Os comerciantes têm seus próprios certificados.** Um comerciante que aceita uma certa bandeira de cartão precisa possuir dois certificados para as duas chaves públicas pertencentes ao comerciante: uma para assinar mensagens e uma para troca de chaves. O comerciante também precisa de uma cópia do certificado de chave pública do gateway de pagamento.
4. **O cliente faz um pedido.** Esse é um processo envolvendo o cliente que, em primeiro lugar, navega pelo site Web do comerciante para selecionar itens e saber seus preços. O cliente então envia uma lista dos itens a serem comprados para o comerciante que, em retorno, fornece um formulário de pedido, o qual contém a lista de itens, seus preços, um preço total e um número do pedido.
5. **O comerciante é verificado.** Além do formulário de pedido, o comerciante envia uma cópia de seu certificado, para que o cliente possa verificar se está lidando com uma loja válida.
6. **O pedido e o pagamento são enviados.** O cliente envia o pedido e as informações de pagamento ao comerciante, juntamente com o certificado do cliente. O pedido confirma a compra dos itens no formulário de pedido. O pagamento contém detalhes do cartão de crédito. As informações de pagamento são criptografadas de modo que não possam ser lidas pelo comerciante. O certificado do cliente permite que o comerciante confirme a identidade do cliente.
7. **O comerciante solicita autorização de pagamento.** O comerciante envia as informações de pagamento ao gateway de pagamento, solicitando confirmação de que o crédito disponível do cliente é suficiente para essa compra.
8. **O comerciante confirma o pedido.** O comerciante envia a confirmação do pedido ao cliente.
9. **O comerciante fornece os bens ou serviços.** O comerciante envia os bens ou fornece o serviço ao cliente.
10. **O comerciante solicita o pagamento.** Essa solicitação é enviada ao gateway de pagamento, que trata de todo o processamento do pagamento.

Assinatura dual

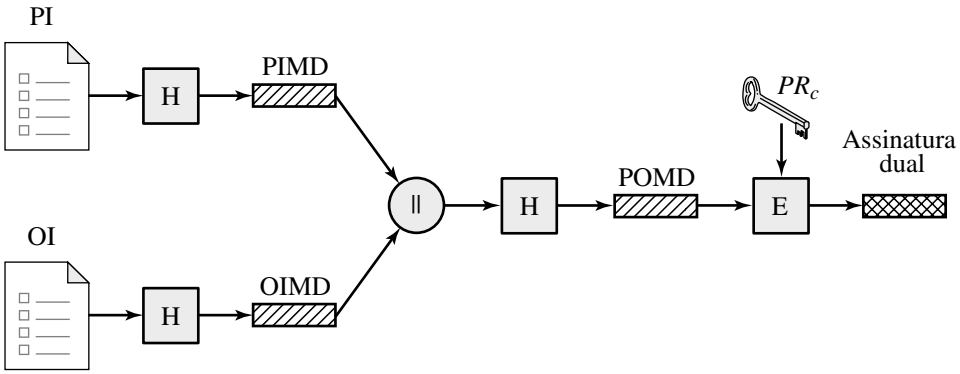
Antes de examinarmos os detalhes do protocolo SET, vamos explicar uma inovação importante apresentada no SET: a assinatura dual (dual signature). A finalidade da assinatura dual é vincular duas mensagens destinadas a dois destinatários diferentes. Nesse caso, o cliente deseja enviar as informações do pedido (OI) para o comerciante e a informação de pagamento (PI) para o banco. O comerciante não precisa saber o número do cartão de crédito do cliente, e o banco não precisa saber os detalhes do pedido do cliente. O cliente recebe proteção extra em termos de privacidade mantendo esses dois itens separados. Porém, os dois itens precisam ser vinculados de modo que possam ser usados para resolver disputas, se for preciso. O vínculo é necessário para que o cliente possa provar que esse pagamento é para esse pedido e não para algum outro bem ou serviço.

Para perceber a necessidade do vínculo, suponha que os clientes enviem duas mensagens ao comerciante: uma OI assinada e uma PI assinada, e o comerciante passe a PI para o banco. Se o comerciante puder captar outra OI desse cliente, ele pode afirmar que essa OI acompanha a PI, em vez da OI original. O vínculo evita isso.

A Figura 17.9 mostra o uso de uma assinatura dual para atender ao requisito do parágrafo anterior. O cliente apanha o hash (usando SHA-1) da PI e o hash da OI. Esses dois hashes são então concatenados e é calculado o hash

do resultado. Finalmente, o cliente criptografa o hash final com sua chave privada para assinatura, criando a assinatura dual. A operação pode ser resumida como

$$DS = E(PR_c, [H(H(PI)||H(OI)])$$



PI = Informações de pagamento
OI = Informações de pedido
H = Função de hash (SHA-1)
|| = Concatenação
PIMD = Resumo da mensagem de PI
OIMD = Resumo da mensagem de OI
POMD = Resumo da mensagem do pedido de pagamento
E = Criptografia (RSA)
PR_c = Chave de assinatura privada do cliente

Figura 17.9 Construção da assinatura dual.

onde PR_c é a chave privada do cliente para assinatura. Agora, suponha que o comerciante possua a assinatura dual (DS) da OI e do resumo da mensagem da PI (PIMD). O comerciante também tem a chave pública do cliente, obtida do certificado do cliente. Então, o comerciante pode calcular os valores

$$H(PIMD||H[OI]); D(PUc, DS)$$

onde PU_c é a chave pública do cliente para assinatura. Se esses dois valores forem iguais, então o comerciante confirmou a assinatura. De modo semelhante, se o banco possuir DS, PI, o resumo da mensagem da OI (OIMD) e a chave pública do cliente, então o banco calculará

$$H(H[OI]||OIMD); D(PU_c, DS)$$

Novamente, se esses dois valores forem iguais, então o banco confirmou a assinatura. Resumindo,

1. O comerciante recebe a OI e confirma a assinatura.
2. O banco recebe a PI e confirma a assinatura.
3. O cliente vinculou a OI à PI e pode comprovar essa relação.

Por exemplo, suponha que o comerciante queira mudar a OI nessa transação para tirar algum proveito. Assim, ele teria que encontrar outra OI cujo hash fosse igual ao hash da OI atual (OIMD). Com SHA-1, isso é considerado inviável. Assim, o comerciante não pode vincular outra OI a essa PI.

Processamento de pagamento

A Tabela 17.3 lista os tipos de transação admitidos pelo SET. A seguir, vemos alguns detalhes nas seguintes transações:

- Solicitação de compra
- Autorização de pagamento
- Captação de pagamento

Solicitação de compra Antes que a transação de solicitação de compra se inicie, o proprietário do cartão já concluiu a navegação, seleção e pedido. O final dessa fase preliminar ocorre quando o comerciante envia um formulário de pedido concluído ao cliente. Todo esse processamento anterior ocorre sem o uso do SET.

A transação de solicitação de compra consiste em quatro mensagens: iniciar solicitação, iniciar resposta, solicitação de compra e resposta de compra.

Tabela 17.3 Tipos de transação SET

Registro do proprietário do cartão	Os proprietários de cartão precisam ser registrados em uma CA antes que possam enviar mensagens SET aos comerciantes.
Registro do comerciante	Os comerciantes precisam ser registrados em uma CA antes que possam trocar mensagens SET com clientes e gateways de pagamento.
Solicitação de compra	Mensagem do cliente a comerciante, contendo a OI para o comerciante e a PI para o banco.
Autorização de pagamento	Troca entre comerciante e gateway de pagamento para autorizar determinado valor para uma compra em determinada conta de cartão de crédito.
Captação de pagamento	Permite que o comerciante solicite o pagamento do gateway de pagamento.
Consulta e status de certificado	Se a CA for incapaz de concluir o processamento de uma solicitação de certificado rapidamente, ela enviará uma resposta ao proprietário do cartão ou comerciante, indicando que o solicitante deverá verificar novamente mais tarde. O proprietário do cartão ou comerciante envia a mensagem de consulta de certificado para determinar o status da solicitação de certificado e receber o certificado se a solicitação tiver sido aprovada.
Consulta de compra	Permite que o proprietário do cartão verifique o status do processamento de um pedido depois que a resposta da compra tiver sido recebida. Observe que essa mensagem não inclui informações como o status de bens que não estejam em estoque, mas sim o status da autorização, captação e processamento de crédito.
Cancelamento da autorização	Permite que um comerciante corrija solicitações de autorização anteriores. Se o pedido não for completado, o comerciante cancela a autorização inteira. Se parte do pedido não for completada (como quando os bens não estão em estoque), o comerciante cancela parte do valor da autorização.
Cancelamento de captação	Permite que um comerciante corrija erros nas solicitações de captação, como valores de transação que foram inseridos incorretamente por um funcionário.
Crédito	Permite que um comerciante emita um crédito para a conta de um proprietário de cartão, por exmplo quando os bens tenham sido devolvidos ou danificados durante a entrega. Observe que a mensagem de crédito do SET sempre é iniciada pelo comerciante, e não pelo proprietário do cartão. Toda a comunicação entre o proprietário do cartão e o comerciante que resulta no processamento de um crédito acontece fora do SET.
Cancelamento de crédito	Permite que um comerciante corrija um crédito solicitado anteriormente.
Solicitação de certificado de gateway de pagamento	Permite que um comerciante consulte o gateway de pagamento e receba uma cópia dos certificados atuais de troca de chaves e de assinatura do gateway.
Administração em lote	Permite que um comerciante comunique informações ao gateway de pagamento com relação a lotes do comerciante.
Mensagem de erro	Indica que um respondedor rejeitou uma mensagem porque há falha nos testes de verificação de formato ou de conteúdo.

Para enviar mensagens SET ao comerciante, o proprietário do cartão precisa ter uma cópia dos certificados do comerciante e do gateway de pagamento. O cliente solicita os certificados na mensagem **Iniciar Solicitação** (Initiate Request), enviada ao comerciante. Essa mensagem inclui a bandeira do cartão de crédito que o cliente está usando. A mensagem também inclui uma ID atribuída a esse par de solicitação/resposta pelo cliente e um nonce usado para garantir o correto posicionamento em relação ao tempo.

O comerciante gera uma resposta e a assina com sua chave de assinatura privada. A resposta inclui o nonce do cliente, outro nonce para o cliente retornar na próxima mensagem e uma ID de transação para essa transação de compra. Além da resposta assinada, a mensagem **Iniciar Resposta** (Initiate Response) inclui o certificado de assinatura do comerciante e o certificado de troca de chaves do gateway de pagamento.

O proprietário do cartão confirma os certificados do comerciante e do gateway por meio de respectivas assinaturas nos seus certificados feitas pela CA e depois cria a OI e a PI. A ID da transação atribuída pelo comerciante é colocada tanto na OI quanto na PI. A OI não contém dados explícitos do pedido, como o número e o preço dos itens. Em vez disso, ela contém uma referência do pedido gerada na troca entre comerciante e cliente durante a fase de compra, antes da primeira mensagem SET. Em seguida, o proprietário do cartão prepara a mensagem **Solicitação de Compra** (Purchase Request) (Figura 17.10). Para essa finalidade, o proprietário do cartão gera uma chave de criptografia simétrica de uso único, K_s. A mensagem inclui o seguinte:

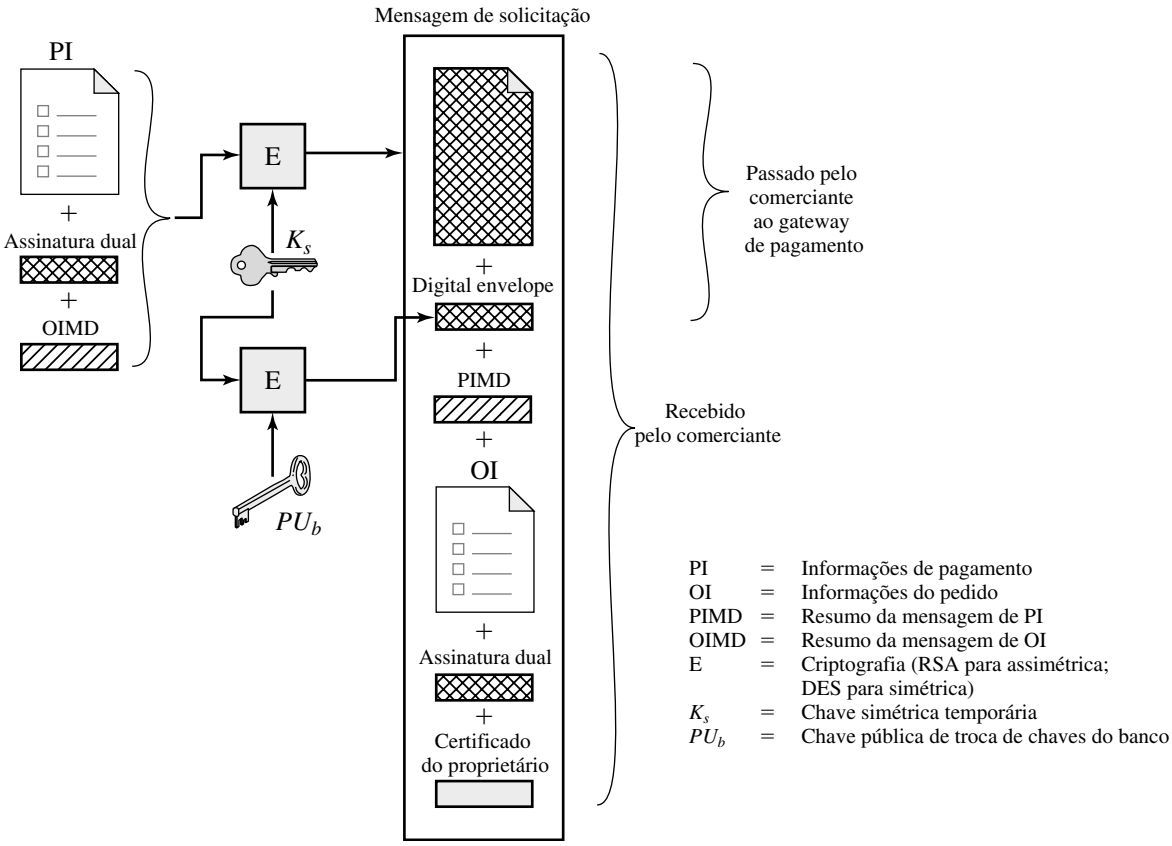


Figura 17.10 O proprietário do cartão envia a solicitação de compra.

1. **Informações relacionadas à compra.** Essas informações serão encaminhadas ao gateway de pagamento pelo comerciante e contêm:
   - PI
   - Assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente
   - O resumo da mensagem de OI (OIMD)

   O OIMD é necessário para o gateway de pagamento verificar a assinatura dual, conforme explicado anteriormente. Todos esses itens são criptografados com K_s. O item final é
   - O envelope digital. Este é formado criptografando-se K_s com a chave pública de troca de chaves do gateway de pagamento. Ele é chamado de envelope digital porque esse envelope precisa ser aberto (decriptografado) antes que os outros itens listados anteriormente possam ser lidos. O valor de K_s não se torna disponível ao comerciante. Portanto, o comerciante não pode ler qualquer uma dessas informações relacionadas ao pagamento.

2. **Informações relacionadas ao pedido.** Essas informações são necessárias para o comerciante e são formadas pela
   - OI
   - Assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente

– O resumo da mensagem de PI (PIMD) O PIMD é necessário para que o comerciante verifique a assinatura dual. Observe que a OI é enviada às claras.

3. **Certificado do proprietário do cartão.** Este contém a chave pública de assinatura do proprietário do cartão. Ele é necessário para o comerciante e para o gateway de pagamento.

Quando o comerciante recebe a mensagem de solicitação de compra, ele realiza as seguintes ações (Figura 17.11):

1. Confirma os certificados do proprietário do cartão por meio das respectivas assinaturas realizadas pela CA.
2. Confirma a assinatura dual usando a chave pública de assinatura do cliente. Isso garante que o pedido não foi alterado em trânsito e que foi assinado usando a chave privada de assinatura do proprietário do cartão.
3. Processa o pedido e encaminha as informações de pagamento ao gateway de pagamento para autorização (descrito mais adiante).
4. Envia uma resposta de compra ao proprietário do cartão.

A mensagem de **Resposta de Compra** inclui um bloco de resposta que confirma o pedido e referencia o número da transação correspondente. Esse bloco é assinado pelo comerciante usando sua chave privada de assinatura. O bloco e sua assinatura são enviados ao cliente juntamente com o certificado de assinatura do comerciante.

Quando o programa do proprietário do cartão recebe a mensagem de resposta de compra, ele verifica o certificado do comerciante e depois verifica a assinatura no bloco de resposta. Finalmente, ele toma alguma atitude com base na resposta, como exibir uma mensagem ao usuário ou atualizar um banco de dados com o status do pedido.

Autorização de pagamento Durante o processamento de um pedido de um proprietário de cartão, o comerciante consegue a autorização da transação com o gateway de pagamento. A autorização de pagamento garante que a transação foi aprovada pelo emissor e garante que o comerciante receberá o pagamento; o comerciante pode, portanto, fornecer os serviços ou bens ao cliente. A troca de autorização de pagamento consiste em duas mensagens: **Solicitação de Autorização** (Authorization Request) e **Resposta de Autorização** (Authorization Response).

O comerciante envia uma mensagem de **Solicitação de Autorização** ao gateway de pagamento, formada por:

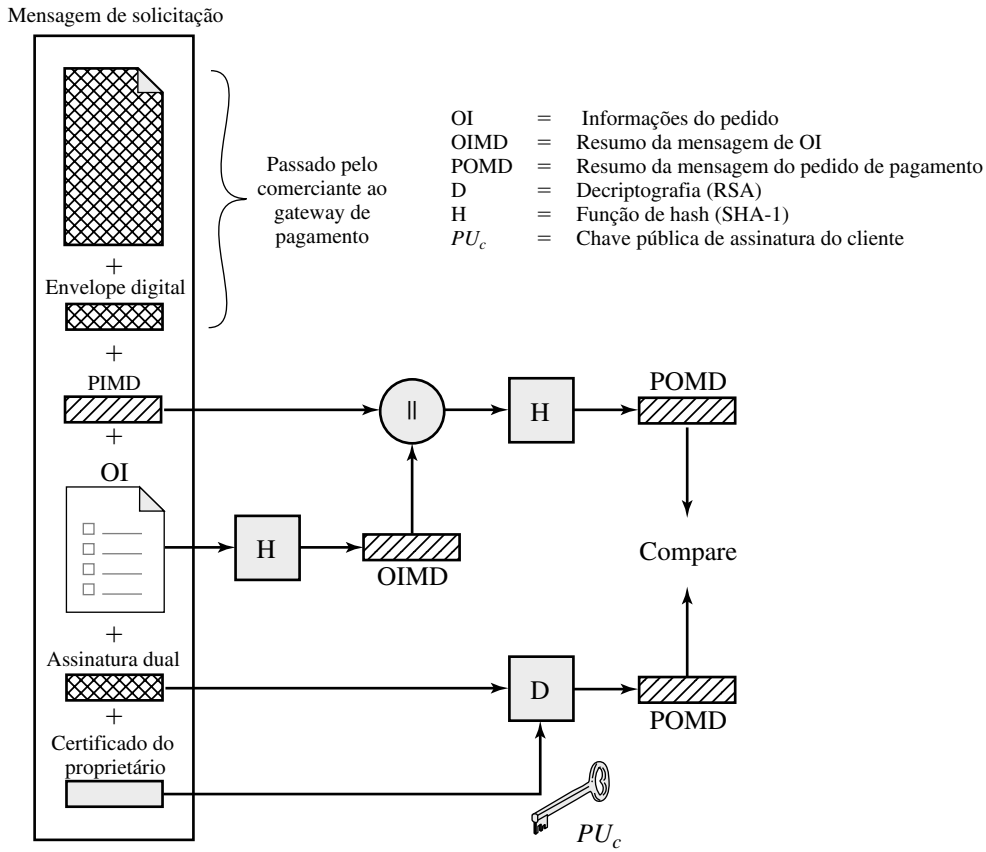


Figura 17.11 Comerciante verifica a solicitação de compra do cliente.

1. **Informações relacionadas à compra**. Essas informações foram obtidas do cliente e consistem em
   - A PI
   - A assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente
   - O resumo da mensagem de OI (OIMD)
   - O envelope digital
2. **Informações relacionadas à autorização.** Essas informações são geradas pelo comerciante e consistem em
   - Um bloco de autorização que inclui a ID da transação, assinado com a chave privada de assinatura do comerciante e criptografado com uma chave simétrica de uso único pelo comerciante.
   - Um envelope digital formado pela criptografia da chave de uso único usando a chave pública de troca de chaves do gateway de pagamento.
3. **Certificados.** O comerciante inclui o certificado de chave de assinatura do proprietário do cartão (usado para verificar a assinatura dual), o certificado de chave de assinatura do comerciante (usado para verificar a assinatura do comerciante) e o certificado de troca de chaves do comerciante (necessário na resposta do gateway de pagamento).

O gateway de pagamento realiza as seguintes tarefas:

1. Verifica todos os certificados
2. Decriptografa o envelope digital do bloco de autorização para obter a chave simétrica e depois decriptografa o bloco de autorização
3. Verifica a assinatura do comerciante no bloco de autorização
4. Decriptografa o envelope digital do bloco de pagamento para obter a chave simétrica e depois decriptografa o bloco de pagamento
5. Verifica a assinatura dual no bloco de pagamento
6. Verifica se a ID da transação recebida do comerciante combina com a da PI recebida (indiretamente) do cliente
7. Solicita e recebe uma autorização do emissor

Tendo obtido autorização do emissor, o gateway de pagamento retorna uma mensagem de **Resposta de Autorização** (Authorization Response) ao comerciante. Ela inclui os seguintes elementos:

1. **Informações relacionadas à autorização.** Inclui um bloco de autorização, assinado com a chave privada de assinatura do gateway e criptografado com uma chave simétrica de uso único, gerada pelo gateway. Também inclui um envelope digital que contém a chave de uso único criptografada com a chave pública de troca de chaves do comerciante.
2. **Informações do token de captação.** Essas informações serão usadas para efetuar o pagamento posteriormente. Esse bloco tem o mesmo formato de (1), a saber, um token de captação assinado, criptografado, juntamente com um envelope digital. Esse token não é processado pelo comerciante. Em vez disso, ele precisa ser retornado, como se encontra, com uma solicitação de pagamento.
3. **Certificado.** O certificado de chave de assinatura do gateway.

Com a autorização do gateway, o comerciante pode fornecer os bens ou serviços ao cliente.

Captação de pagamento Para obter o pagamento, o comerciante inicia com o gateway de pagamento uma transação de captura de pagamento, formada por uma mensagem de solicitação de captura e uma mensagem de resposta de captura.

Para a mensagem de **Solicitação de Captação** (Capture Request), o comerciante gera, assina e criptografa um bloco de solicitação de captação, que inclui o valor do pagamento e a ID da transação. A mensagem também inclui o token de captação criptografado recebido anteriormente (na Resposta de Autorização) para essa transação, além da chave de assinatura do comerciante e dos certificados de troca de chaves.

Quando o gateway de pagamento recebe a mensagem de solicitação de captação, ele decriptografa e verifica o bloco de solicitação de captação e decriptografa e verifica o bloco do token de captação. Depois, ele verifica a consistência entre a solicitação e o token de captação. Depois, cria uma solicitação de compensação que é enviada ao emissor pela rede de pagamento privada. Essa solicitação faz com que sejam transferidos fundos para a conta do comerciante.

O gateway, então, notifica o pagamento ao comerciante em uma mensagem de **Resposta de Captação** (Capture Response). A mensagem inclui um bloco de resposta de captação que o gateway assina e criptografa. A mensagem também inclui o certificado de chave de assinatura do gateway. O software do comerciante armazena a resposta de captação a ser usada para reconciliação com o pagamento recebido do acquirer.

17.4 LEITURA E SITES WEB RECOMENDADOS

[RESC01] é um boa explicação detalhada sobre SSL e TLS.

A visão geral mais detalhada do SET está no Livro 1 da especificação, disponível no site Web sobre SET da MasterCard. Outra excelente visão geral é [MACG97]. [DREW99] também é uma boa fonte.

DREW99 Drew, G. *Using SET for secure electronic commerce*. Upper Saddle River: Prentice Hall, 1999.
MACG97 Macgregor, R.; Ezvan, C.; Liguori, L. e Han, J. *Secure electronic transactions: credit card payment on the Web in theory and practice*. IBM RedBook SG24-4978-00, 1997. Disponível em **www.redbooks.ibm.com**.
RESC01 Rescorla, E. *SSL and TLS: designing and building secure systems*. Reading: Addison-Wesley, 2001.

Sites Web recomendados:

- **Netscape's SSL Page:** Contém a especificação SSL.
- **Transport Layer Security Charter:** RFCs e drafts de Internet mais recentes para TLS.
- **OpenSSL Project:** Um projeto para desenvolver software SSL e TLS de fonte aberta. O site inclui documentos e links.

17.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

acquirer
assinatura dual
autoridade certificadora (CA)
comerciante
emissor
gateway de pagamento
proprietário do cartão
Secure Electronic Transaction (SET)
Secure Socket Layer (SSL)
Transport Layer Security (TLS)

Perguntas de revisão

17.1 Quais são as vantagens de cada uma das três abordagens mostradas na Figura 17.1?
17.2 Que protocolos estão incorporados ao SSL?
17.3 Qual é a diferença entre uma conexão SSL e uma sessão SSL?
17.4 Liste e defina resumidamente os parâmetros que definem um estado de sessão SSL.
17.5 Liste e defina resumidamente os parâmetros que definem uma conexão de sessão SSL.
17.6 Que serviços são fornecidos pelo Protocolo de Registro SSL?
17.7 Que etapas estão envolvidas na transmissão do Protocolo de Registro SSL?
17.8 Liste e defina resumidamente as principais categorias de participantes do SET.
17.9 O que é uma assinatura dual e qual é sua finalidade?

Problemas

17.1 No SSL e TLS, por que existe um Protocolo de Mudança de Especificação de Cifra separado, em vez de incluir uma mensagem change_cipher_spec no Protocolo de Estabelecimento de Sessão?

17.2 Considere as seguintes ameaças à segurança na Web e descreva como cada uma é impedida por um recurso específico do SSL.

a. Ataque criptoanalítico por força bruta: Um teste completo de todas as chaves possíveis para um algoritmo de criptografia convencional.

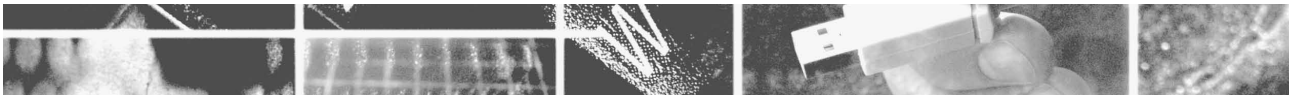
b. Ataque de dicionário com texto claro conhecido: Muitas mensagens terão texto claro previsível, como o comando GET do HTTP. Um atacante cria um dicionário contendo cada criptografia possível da mensagem de texto claro conhecido. Quando uma mensagem criptografada é interceptada, o atacante apanha a parte contendo o texto claro conhecido criptografado e pesquisa o texto cifrado no dicionário. O texto cifrado deverá ser igual a uma entrada criptografada com a mesma chave secreta. Se for igual a mais de uma, cada uma delas pode ser experimentada em relação ao texto cifrado completo para determinar a entrada correta. Esse ataque é especialmente eficaz quanto a tamanhos de chave pequenos (por exemplo, 40 bits).

c. Ataque por repetição: Mensagens de estabelecimento de conexão SSL anteriores são repetidas.

d. Ataque de homem no meio (man-in-the-middle): Um atacante se interpõe durante a troca de chave, atuando como cliente perante o servidor e como servidor perante o cliente.

e. Sniffing de senha: As senhas em HTTP ou outro tráfego de aplicação são interceptadas sem autorização.
f. Falsificação do IP (IP spoofing): Usa endereços de IP forjados para enganar um host para aceitar dados falsos.
g. Seqüestro de IP (IP hijacking): Uma conexão ativa, autenticada, entre dois hosts é interrompida e o atacante toma o lugar de um dos hosts.
h. Inundação de SYN (SYN flooding): Um atacante envia mensagens SYN do TCP para solicitar uma conexão, mas não responde à mensagem final para estabelecer a conexão totalmente. O módulo TCP atacado normalmente deixa a conexão 'meio aberta' por alguns minutos. As mensagens SYN repetidas podem congestionar os módulos TCP.

17.3 Com base no que você aprendeu neste capítulo, é possível no SSL que o receptor reordene blocos de registro SSL que chegam fora de ordem? Em caso positivo, explique como isso pode ser feito. Em caso negativo, por que não?



PARTE QUATRO

Segurança do sistema

Segurança é uma preocupação das organizações com ativos controlados por sistemas de computador. Acessando ou alterando dados, um atacante pode roubar ativos tangíveis ou levar uma organização a tomar ações que não seriam tomadas de outra maneira. Meramente examinando os dados, um atacante pode obter uma vantagem competitiva, sem que o proprietário dos dados saiba.

— *Computers at risk: safe computing in the information age,*
National Research Council, 1991

Os desenvolvedores de software seguro não podem adotar as diversas medidas probabilísticas de qualidade que os desenvolvedores de outros tipos de software geralmente podem. Para muitas aplicações, é bastante razoável tolerar uma falha que raramente é exposta, e assumir que isso tenha ocorrido uma vez não aumenta a probabilidade de que ocorrerá novamente. Também é razoável supor que falhas logicamente independentes serão estatisticamente independentes e não acontecerão em conjunto. Ao contrário, uma vulnerabilidade de segurança, uma vez descoberta, será rapidamente disseminada entre uma comunidade de atacantes e poderá ser explorada regularmente até ser corrigida.

— *Computers at risk: safe computing in the information age*,
National Research Council, GAO/OSI-94-2, novembro de 1993

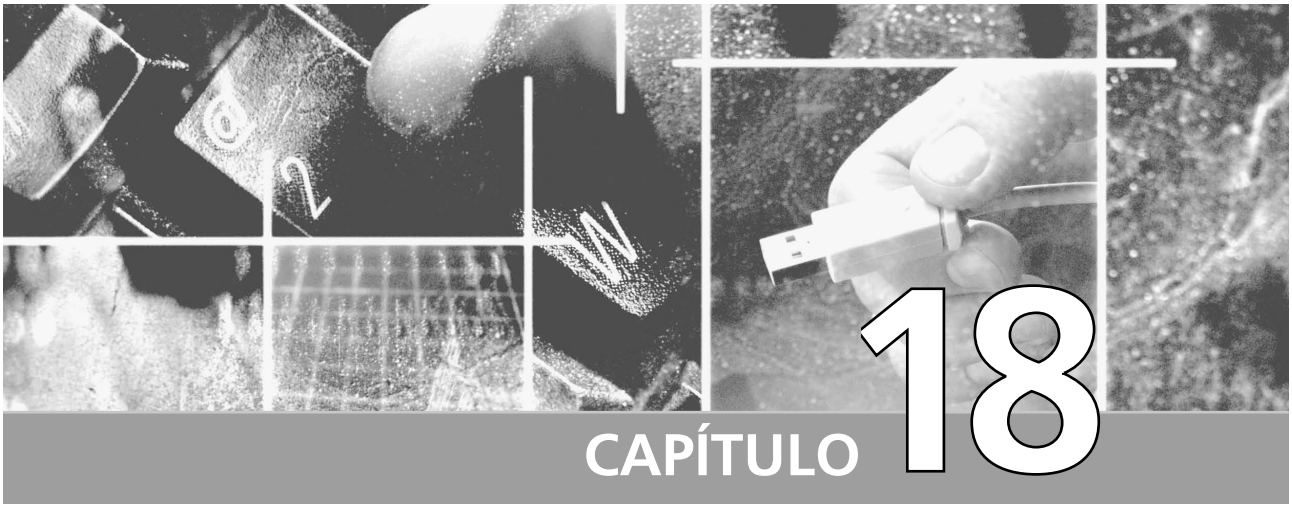
A Parte Quatro examina os assuntos de segurança no nível do sistema, incluindo a ameaça e contramedidas para intrusos e vírus, o uso de firewalls e sistemas confiáveis.

MAPA DA PARTE QUATRO

Capítulo 18: Intrusos
O Capítulo 18 examina uma variedade de ameaças a serviços e ao acesso a informações apresentadas por hackers que exploram vulnerabilidades nos sistemas de computação baseados em rede. O capítulo começa com uma explicação dos tipos de ataques que podem ser efetuados por usuários não autorizados, ou intrusos, e analisa diversas técnicas para prevenção e detecção. Este capítulo também aborda a questão relacionada ao gerenciamento de senhas.

Capítulo 19: Software malicioso
O Capítulo 19 examina ameaças de software aos sistemas, com ênfase especial em vírus e vermes (worms). O capítulo começa com um estudo dos vários tipos de softwares maliciosos, com uma visão mais detalhada da natureza de vírus e vermes. O capítulo, em seguida, examina as contramedidas. Finalmente, o capítulo trata dos ataques distribuídos de negação de serviço.

Capítulo 20: Firewalls
Uma técnica-padrão para a proteção dos ativos de computadores locais contra ameaças externas é o uso de um firewall. O Capítulo 20 explica os princípios do projeto de firewall e examina técnicas específicas. Este capítulo também aborda a questão relacionada a sistemas confiáveis.



CAPÍTULO 18

Intrusos

18.1 **Intrusos**
18.2 **Detecção de intrusão**
18.3 **Gerenciamento de senhas**
18.4 **Leitura e sites Web recomendados**
18.5 **Principais termos, perguntas de revisão e problemas**
Apêndice 18A O mito da probabilidade de base

Eles concordaram que Graham deveria definir o teste para Charles Mabledene. Nada mais, nada menos do que Dragon deveria obter o código de Stern. Se ele tivesse a 'entrada' em Utting, que afirmou ter, isso deveria ser possível, e somente a lealdade a Central de Moscou impediria isso. Se ele tivesse a chave para o código, provaria sua lealdade à Central de Londres sem sombra de dúvida.

— *Talking to strange men*, Ruth Rendell

PONTOS PRINCIPAIS

- A intrusão não autorizada em um sistema ou rede de computadores é uma das ameaças mais sérias à segurança de computadores.
- Sistemas de detecção de intrusão têm sido desenvolvidos para fornecer um aviso inicial de intrusão, para que a ação defensiva possa ser tomada para impedir ou minimizar danos.
- A detecção de intrusão envolve a detecção de padrões incomuns de atividade ou padrões de atividade sabidamente relacionados a intrusões.
- Um elemento importante da prevenção à intrusão é o gerenciamento de senhas, com o objetivo de impedir que usuários não autorizados tenham acesso às senhas de outros.

Um problema de segurança significativo para sistemas em rede é a intrusão hostil ou, no mínimo, indesejada por usuários ou software. A intrusão feita pelo usuário pode ter a forma de um logon não autorizado em uma máquina ou, no caso de um usuário autorizado, a obtenção de privilégios ou realização de ações além das autorizadas. A intrusão por software pode ter a forma de um vírus, verme ou cavalo-de-tróia.

Todos esses ataques relacionam-se à segurança de rede, pois a entrada no sistema pode ser alcançada por meio de uma rede. Porém, esses ataques não estão confinados a ataques baseados em rede. Um usuário com acesso a um terminal local pode tentar a intrusão sem usar uma rede intermediária. Um vírus ou cavalo-de-tróia pode ser introduzido em um sistema por meio de um disquete. Somente o verme é um fenômeno exclusivamente de rede. Assim, a violação de sistemas é uma área na qual as preocupações relativas à segurança de rede e à segurança de computadores se sobrepõem.

Como o foco deste livro é a segurança de rede, não tentamos realizar uma análise abrangente dos ataques ou das contramedidas de segurança relacionados à intrusão de sistemas. Em vez disso, nesta parte do livro, apresentamos uma ampla visão dessas preocupações.

Este capítulo envolve o assunto de intrusos. Primeiro, examinamos a natureza do ataque e, depois, vemos as estratégias planejadas para prevenção e, se isso falhar, detecção. Em seguida, examinamos o tópico relacionado ao gerenciamento de senhas.

18.1 INTRUSOS

Uma das duas ameaças à segurança que ganha mais publicidade é a do intruso (a outra são os vírus), geralmente conhecido como hacker ou cracker. Em um estudo preliminar importante sobre a intrusão, Anderson [ANDE80] identificou três classes de intrusos:

- **Mascarado (masquerader):** Um indivíduo que não está autorizado a usar o computador e que penetra nos controles de acesso de um sistema para explorar a conta de um usuário legítimo
- **Infrator (misfeasor):** Um usuário legítimo que acessa dados, programas ou recursos para os quais não tem autorização de acesso, ou que está autorizado para tal acesso, mas faz mau uso de seus privilégios
- **Usuário clandestino (clandestine user):** Um indivíduo que se apodera do controle de supervisor do sistema e utiliza esse controle para escapar de auditorias e controles de acesso ou para suprimir provas parciais de auditoria

O mascarado provavelmente é alguém de fora, o infrator geralmente é de dentro e o usuário clandestino pode ser de dentro ou de fora da empresa.

A extensão dos ataques de intruso varia desde não perigosos até sérios. No extremo benigno da escala, existem muitas pessoas que simplesmente querem explorar a interligação de redes e ver o que há lá fora. No extremo sério estão indivíduos que tentam ler dados privilegiados, realizar modificações não autorizadas nos dados, ou interromper o sistema.

A ameaça de intruso tem obtido bastante publicidade, particularmente devido ao famoso incidente 'Wily Hacker' de 1986-1987, documentado por Cliff Stoll [STOL88, 89]. Em 1990, houve uma repressão em nível nacional nos EUA aos hackers ilegais de computadores, com prisões, ações criminais, um julgamento dramático, várias alegações de culpa e o confisco de quantidades enormes de dados e equipamentos de computador [STER92]. Muitas pessoas acreditaram que o problema havia sido controlado.

Na verdade, o problema não foi controlado. Para citar um exemplo, um grupo no Bell Labs [BELL92, BELL93] informou ataques persistentes e freqüentes via Internet a seu complexo de computadores por um longo período e a partir de diversas origens. Na época desses relatórios, o grupo Bell passou pelo seguinte:

- Tentativas de copiar o arquivo de senha (explicado mais adiante) com uma freqüência superior a uma vez a cada dois dias
- Solicitações suspeitas de chamada de procedimento remoto (RPC) com uma freqüência superior a uma vez por semana
- Tentativas de conexão a máquinas de 'isca' inexistentes pelo menos a cada duas semanas

Os intrusos benignos podem ser toleráveis, embora consumam recursos e possam tornar lento o desempenho para usuários legítimos. Porém, não há uma maneira de saber antecipadamente se um intruso será benigno ou maligno. Conseqüentemente, até mesmo para sistemas sem recursos particularmente importantes existe uma motivação para controlar esse problema.

Um exemplo que ilustra bem a ameaça ocorreu na Texas A&M University [SAFF93]. Em agosto de 1992, o centro de computação foi notificado de que uma de suas máquinas estava sendo usada para atacar computadores em outro local via Internet. Monitorando a atividade, o pessoal do centro de computação descobriu que havia vários intrusos externos envolvidos, que estavam executando rotinas para quebra de senhas em vários computadores (o local é composto por um total de 12.000 máquinas interconectadas). O centro desconectou as máquinas afetadas, fechou as brechas de segurança conhecidas e retomou a operação normal. Alguns dias depois, um dos gerentes de sistemas locais detectou que o ataque do intruso tinha sido retomado. Acontece que o ataque foi muito mais sofisticado do que se acre-

ditava inicialmente. Foram encontrados arquivos contendo centenas de senhas captadas, incluindo algumas em servidores importantes e supostamente seguros. Além disso, uma máquina local tinha sido configurada como um informativo para hackers, o qual era usado para entrar em contato com outros hackers e discutir técnicas e progressos obtidos.

Uma análise desse ataque revelou que havia na realidade dois níveis de hackers. Os de alto nível eram usuários sofisticados com um conhecimento profundo da tecnologia; os de baixo nível eram os 'soldados rasos' que simplesmente usavam os programas de cracking fornecidos, com pouco conhecimento de como eles funcionavam. Esse trabalho de equipe combinou as duas armas mais importantes no arsenal do intruso: conhecimento sofisticado de como invadir e um desejo de gastar inúmeras horas 'girando maçanetas' para sondar pontos fracos.

Um dos resultados da crescente consciência do problema de intrusão tem sido o estabelecimento de diversas equipes de resposta a emergências de computador (CERTs — Computer Emergency Response Teams). Esses empreendimentos cooperativos reúnem informações sobre vulnerabilidades do sistema que são disseminadas para gerentes de sistemas. Infelizmente, os hackers também podem obter acesso aos relatórios do CERT. No incidente da Texas A&M, uma análise posterior mostrou que os hackers tinham desenvolvido programas para testar as máquinas atacadas em relação a praticamente todas as vulnerabilidades que tinham sido anunciadas pelo CERT. Se somente uma máquina tivesse deixado de responder prontamente a um aviso do CERT, ela estaria amplamente aberta a tais ataques.

Além de executar programas de quebra de senhas, os intrusos tentaram modificar software de login para permitir a captação de senhas de usuários efetuando login nos sistemas. Isso tornou possível montar uma coletânea impressionante de senhas descobertas, que foram disponibilizadas no quadro de avisos instalado em uma das máquinas da própria vítima.

Nesta seção, examinamos as técnicas usadas para a intrusão. Depois, examinamos as maneiras de detectar a intrusão. Finalmente, vemos as abordagens de prevenção baseadas em senhas.

Técnicas de intrusão

O objetivo do intruso é obter acesso a um sistema ou aumentar a gama de privilégios acessíveis em um sistema. Geralmente, isso exige que o intruso adquira informações que deveriam ter sido protegidas. Em alguns casos, essas informações estão na forma de uma senha de usuário. Com o conhecimento da senha de alguns outros usuários, um intruso pode efetuar o login em um sistema e exercer todos os privilégios concedidos ao usuário legítimo.

Normalmente, um sistema precisa manter um arquivo que associa uma senha a cada usuário autorizado. Se esse arquivo for armazenado sem proteção, então é uma questão de fácil solução obter acesso a ele e descobrir as senhas registradas. O arquivo de senhas pode ser protegido de duas maneiras:

- **Função unidirecional:** O sistema armazena somente o valor de uma função calculado a partir da senha do usuário. Quando o usuário apresenta uma senha, o sistema transforma essa senha e a compara ao valor armazenado. Na prática, o sistema normalmente realiza uma transformação unidirecional (não reversível), em que a senha é usada para gerar uma chave para uma função unidirecional, a partir da qual uma saída de tamanho fixo é produzida.
- **Controle de acesso:** O acesso ao arquivo de senha é limitado a somente uma ou a muito poucas contas.

Se uma ou ambas as contramedidas forem utilizadas, algum esforço será necessário para que um intruso em potencial descubra as senhas. Com base em um estudo da literatura e entrevistas com diversos crackers de senha, [ALVA90] informa as seguintes técnicas para descobrir senhas:

1. Tentar senhas-padrão usadas com contas padrão, que são entregues com o sistema. Muitos administradores não se preocupam em alterar esses padrões.
2. Tentar exaustivamente todas as senhas curtas (aquelas de um a três caracteres).
3. Tentar palavras de um sistema de dicionário on-line, ou de uma lista de senhas prováveis. Alguns exemplos de senhas prováveis estão disponíveis em informativos para hackers.
4. Coletar informações sobre usuários, como seus nomes completos, os nomes de seus cônjuges e filhos, fotos e livros em seu escritório que estejam relacionados a hobbies.
5. Tentar os números de telefone dos usuários, números de documentos e números de sala.
6. Tentar todos os números de placas de automóveis legítimos do estado.
7. Usar um cavalo-de-tróia (descrito na Seção 18.2) para contornar as restrições ao acesso.
8. Realizar escuta clandestina na linha entre um usuário remoto e o sistema host.

Os seis primeiros métodos são modos diversos de adivinhar uma senha. Se um intruso tiver de testar se a adivinhação está correta tentando efetuar um login, esse será um modo de ataque tedioso e facilmente impedido. Por exemplo, um sistema pode simplesmente rejeitar qualquer login após três tentativas de senha, exigindo assim que o intruso se reconecte ao host para tentar novamente. Sob essas circunstâncias, não é prático tentar mais do que algumas poucas senhas. Porém, o intruso provavelmente não tentará esses métodos primitivos. Por exemplo, se um intruso puder obter acesso com um baixo nível de privilégios a um arquivo de senhas criptografado, então a estratégia seria

captar esse arquivo e depois usar o mecanismo de criptografia desse sistema em particular sem pressa até que seja descoberta uma senha válida que ofereça mais privilégios.

Ataques de adivinhação são viáveis, e na realidade altamente eficazes, quando uma grande quantidade de adivinhações pode ser tentada automaticamente e cada tentativa testada sem que o processo de adivinhação seja detectável. Mais adiante, neste capítulo, teremos muito a dizer sobre como impedir ataques de adivinhação.

O sétimo método de ataque listado anteriormente, o cavalo-de-tróia, pode ser particularmente difícil de impedir. Um exemplo de um programa que evita controles de acesso foi citado em [ALVA90]. Um usuário com baixos privilégios produziu um programa de jogo e convidou o operador do sistema a usá-lo em seu tempo livre. O programa realmente executava um jogo, mas em segundo plano ele também continha código para copiar o arquivo de senha, que estava decriptografado, mas com acesso protegido, para um arquivo de usuário. Como o programa de jogo estava sendo executado sob o modo de usuário com altos privilégios, ele foi capaz de obter acesso ao arquivo de senhas.

O oitavo ataque listado, escuta clandestina da linha, é uma questão de segurança física. Ele pode ser impedido com técnicas de criptografia do link, explicadas na Seção 7.1.

Outras técnicas de intrusão não exigem descoberta de uma senha. Os intrusos podem obter acesso a um sistema explorando ataques como estouros de buffer em um programa executado com certos privilégios. O aumento de privilégios também pode ser feito dessa maneira.

Agora, passamos para uma explicação das duas contramedidas principais: detecção e prevenção. A detecção trata da descoberta de um ataque, seja antes ou depois do seu sucesso. A prevenção é um objetivo desafiador de segurança, e uma batalha árdua e contínua. A dificuldade vem do fato de que o defensor precisa tentar impedir todos os ataques possíveis, enquanto o atacante está livre para tentar encontrar o elo mais fraco na cadeia de defesa e atacar nesse ponto.

18.2 DETECÇÃO DE INTRUSÃO

Inevitavelmente, o melhor sistema de prevenção de intrusão falhará. A segunda linha de defesa de um sistema é a detecção de intrusão e esse tem sido o foco de muitas pesquisas nos anos recentes. Esse interesse é motivado por uma série de considerações, incluindo as seguintes:

1. Se uma intrusão for detectada com rapidez suficiente, o intruso poderá ser identificado e expulso do sistema antes que seja feito qualquer dano ou quaisquer dados sejam comprometidos. Mesmo que a detecção não seja suficientemente oportuna para impedir o intruso, quanto mais cedo a intrusão for detectada, menor a quantidade de danos e mais rapidamente a recuperação poderá ser obtida.
2. Um sistema de detecção de intrusão eficaz pode servir como um elemento desencorajador, atuando para impedir intrusões.
3. A detecção de intrusão permite a coleta de informações sobre as técnicas de intrusão, o que pode ser usado para fortalecer a estrutura de prevenção de intrusão.

A detecção de intrusão é baseada na suposição de que o comportamento do intruso difere daquele de um usuário legítimo de maneiras que podem ser quantificadas. Naturalmente, não podemos esperar que haverá uma distinção clara, exata, entre um ataque de um intruso e o uso normal dos recursos por um usuário autorizado. Em vez disso, devemos esperar que haja alguma sobreposição.

A Figura 18.1 sugere, em termos bastante abstratos, a natureza da tarefa que o projetista de um sistema de detecção de intrusão enfrenta. Embora o comportamento típico de um intruso seja diferente do comportamento típico de um usuário autorizado, existe uma sobreposição desses comportamentos. Assim, uma interpretação mais abrangente em relação ao comportamento do intruso, a qual irá captar mais intrusos, também levará a uma série de 'falsos positivos', ou usuários autorizados identificados como intrusos. Por outro lado, uma tentativa de limitar falsos positivos por uma interpretação rígida do comportamento do intruso levará a um aumento nos falsos negativos, ou intrusos não identificados como intrusos. Assim, existe um equilíbrio quanto à escolha na arte de detecção de intrusos.

No estudo de Anderson [ANDE80], foi postulado que alguém poderia, com razoável confiabilidade, distinguir entre um mascarado e um usuário legítimo. Padrões de comportamento do usuário legítimo podem ser estabelecidos observando-se o histórico, e um desvio significativo desses padrões poderá ser detectado. Anderson sugere que a tarefa de detectar um usuário infrator (usuário legítimo trabalhando de modo não autorizado) é mais difícil, pois a distinção entre comportamento anormal e normal pode ser pequena. Anderson concluiu que essas violações seriam indetectáveis unicamente pela busca do comportamento anômalo. Porém, o comportamento do infrator poderia, apesar disso, ser detectável pela definição inteligente da classe de condições que sugerem o uso não autorizado. Finalmente, a detecção do usuário clandestino estava além do escopo das técnicas puramente automatizadas. Esas observações, feitas em 1980, continuam sendo verdadeiras hoje.

[PORR92] identifica as seguintes técnicas de detecção de intrusão:

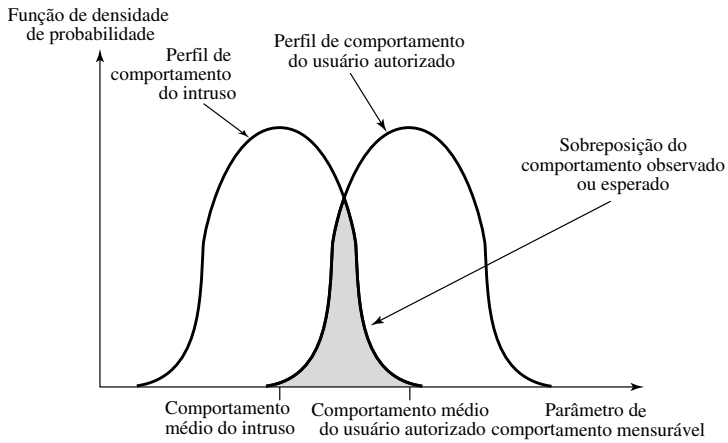


Figura 18.1 Perfis de comportamento de intrusos e usuários autorizados.

1. **Detecção estatística de anomalia:** Envolve a coleta de dados relacionados ao comportamento de usuários legítimos por um período de tempo. Depois, testes estatísticos são aplicados ao comportamento observado para determinar com um alto nível de confiabilidade se esse comportamento não é um comportamento legítimo do usuário.
   a. Detecção de limiar: Esta técnica envolve a definição de limiares, independentes do usuário, para a freqüência de ocorrência de vários eventos.
   b. Baseada em perfil: Um perfil da atividade de cada usuário é desenvolvido e usado para detectar mudanças no comportamento das contas individuais.
2. **Detecção baseada em regras:** Envolve uma tentativa de definir um conjunto de regras que podem ser usadas para decidir se determinado comportamento é o de um intruso.
   a. Detecção de anomalia: Regras são desenvolvidas para detectar o desvio dos padrões de uso anteriores.
   b. Identificação de penetração: Uma abordagem por sistema especialista que procura por comportamento suspeito.

Em resumo, as técnicas estatísticas tentam definir o comportamento normal, ou esperado, enquanto as técnicas baseadas em regras tentam definir o comportamento apropriado.

Em termos dos tipos de atacantes listados anteriormente, a detecção estatística de anomalia é eficaz contra mascarados, que provavelmente não imitam os padrões de comportamento das contas de que se apropriam. Por outro lado, essas técnicas podem ser incapazes de lidar com infratores. Para esses ataques, as técnicas baseadas em regras podem ser capazes de reconhecer eventos e seqüências que, no contexto, revelam a penetração. Na prática, um sistema pode exibir uma combinação de ambas as técnicas para tornar-se eficaz contra uma grande variedade de ataques.

Registros de auditoria

Uma ferramenta fundamental para detecção de intrusão é o registro de auditoria. Algum registro da atividade contínua realizada pelos usuários precisa ser mantido como entrada de um sistema de detecção de intrusão. Basicamente, dois planos são usados:

- **Registros de auditoria nativos:** praticamente todos os sistemas operacionais multiusuário incluem software de contabilidade que coleta informações sobre a atividade do usuário. A vantagem de usar essas informações é que nenhum software de coleta adicional é necessário. A desvantagem é que os registros de auditoria nativos podem não conter as informações necessárias ou podem não contê-las em um formato conveniente.
- **Registros de auditoria específicos para detecção:** um recurso de coleta pode ser implementado para gerar registros de auditoria contendo apenas as informações exigidas pelo sistema de detecção de intrusão. Uma vantagem dessa técnica é que ela pode ser feita independentemente do fornecedor e transportada para vários sistemas. A desvantagem é o overhead extra envolvido pelo fato de haver, na verdade, dois pacotes de contabilização sendo executados na máquina.

Um bom exemplo de registro de auditoria específico para detecção é aquele desenvolvido por Dorothy Denning [DENN87]. Cada registro de auditoria contém os seguintes campos:

- **Sujeito:** iniciadores das ações. Alguém que executa uma ação (o sujeito da ação) normalmente é um usuário de terminal, mas também pode ser um processo atuando em nome de usuários ou grupos de usuários. Toda a atividade surge por meio de comandos emitidos pelos sujeitos. Os sujeitos podem ser agrupados em diferentes classes de acesso, e essas classes podem se sobrepor.

- **Ação:** Operação realizada pelo sujeito em ou com um objeto; por exemplo, login, leitura, realização de E/S (entrada/saída), execução.
- **Objeto:** Receptores da ação. Alguns exemplos incluem arquivos, programas, mensagens, registros, terminais, impressoras e estruturas criadas pelo usuário ou programa. Quando um sujeito é o destinatário de uma ação, como um e-mail, então esse sujeito é considerado um objeto. Os objetos podem ser agrupados por tipo. A granularidade do objeto pode variar por tipo de objeto e por ambiente. Por exemplo, ações de banco de dados podem ser auditadas para o banco de dados como um todo ou em nível de registro.
- **Condição de exceção:** Indica qual condição de exceção, se houver, foi gerada no retorno.
- **Uso de recurso:** Uma lista de elementos quantitativos em que cada elemento indica a quantidade usada de algum recurso (por exemplo, número de linhas impressas ou exibidas, número de registros lidos ou escritos, tempo do processador, unidades de E/S utilizadas, tempo decorrido na sessão).
- **Carimbo de tempo:** Carimbo de data e hora exclusivo, identificando quando a ação ocorreu.

A maioria das operações do usuário é composta de uma série de ações elementares. Por exemplo, uma cópia de arquivo envolve a execução do comando do usuário, que inclui realizar validação de acesso e preparar uma cópia, mais a leitura de um arquivo, mais a escrita em outro arquivo. Considere o comando

```
COPY GAME.EXE TO <Library>GAME.EXE
```

dado por Smith para copiar um arquivo executável GAME do diretório atual para o diretório <Library>. Os registros de auditoria a seguir podem ser gerados:

Smith	executar	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	ler	<Smith>GAME.EXE	0	REGISTROS = 0	11058721679
Smith	executar	<Library>COPY.EXE	violação-escrita	REGISTROS = 0	11058721680

Nesse caso, a cópia é abortada porque Smith não tem permissão de escrita em <Library>.

A decomposição de uma operação do usuário em ações elementares tem três vantagens:

1. Uma vez que objetos são as entidades que devem ser protegidas em um sistema, o uso de ações elementares permite uma auditoria de todo o comportamento afetando um objeto. Assim, o sistema pode detectar tentativas de subversão dos controles de acesso (observando uma anormalidade no número de condições de exceção retornadas) e pode detectar subversões bem-sucedidas observando uma anormalidade no conjunto de objetos acessíveis ao sujeito.
2. Registros de auditoria de ação isolada em um objeto isolado simplificam o modelo e a implementação.
3. Por causa da estrutura simples e uniforme dos registros de auditoria específicos da detecção, pode ser relativamente fácil obter essas informações, ou pelo menos parte delas, por um mapeamento direto dos registros de auditoria nativos existentes para os registros de auditoria específicos da detecção.

Detecção estatística de anomalia

Conforme mencionamos, as técnicas de detecção estatística de anomalia estão em duas categorias amplas: detecção de limiar e sistemas baseados em perfil. A detecção de limiar envolve a contagem do número de ocorrências de um tipo de evento específico ao longo de um intervalo de tempo. Se a contagem ultrapassar o que é considerado um número razoável de se esperar, então a intrusão é presumida.

A análise de limiar, por si só, é um detector primitivo e ineficaz até mesmo de ataques moderadamente sofisticados. Tanto o limiar quanto o intervalo de tempo precisam ser determinados. Devido à variabilidade entre usuários, esses limiares provavelmente geram muitos falsos positivos ou muitos falsos negativos. Porém, detectores simples de limiar podem ser úteis em conjunto com técnicas mais sofisticadas.

A detecção de anomalia baseada em perfil focaliza a caracterização do comportamento passado dos usuários individuais ou grupos relacionados de usuários, e depois a detecção de desvios significativos. Um perfil pode ser formado por um conjunto de parâmetros, de modo que o desvio em apenas um único parâmetro pode não ser suficiente por si só para sinalizar um alerta.

O fundamento dessa técnica é uma análise dos registros de auditoria. Os registros de auditoria oferecem entrada para a função de detecção de intrusão de duas maneiras. Primeiro, o projetista precisa decidir sobre diversas métricas quantitativas que podem ser usadas para medir o comportamento do usuário. Uma análise dos registros de auditoria por um período de tempo pode ser usada para determinar o perfil de atividade do usuário comum. Assim, os registros de auditoria servem para definir o comportamento típico. Segundo, os registros de auditoria atuais são a entrada usada para detectar a intrusão. Ou seja, o modelo de detecção de intrusão analisa os registros de auditoria que chegam para determinar o desvio do comportamento médio.

Exemplos de métricas úteis para a detecção de intrusão baseada em perfil são os seguintes:

- **Contador:** Um inteiro não negativo que pode ser incrementado, mas não decrementado até que seja reiniciado por uma ação de gerenciamento. Geralmente, uma contagem de certos tipos de evento é mantida por um determinado período de tempo. Exemplos incluem o número de logins de um único usuário durante uma hora, o número de vezes que determinado comando é executado durante uma única sessão do usuário, e o número de falhas de senha durante um minuto.
- **Medidor:** Um inteiro não negativo que pode ser incrementado ou decrementado. Normalmente, um medidor é usado para medir o valor atual de alguma entidade. Exemplos incluem o número de conexões lógicas atribuídas a uma aplicação do usuário e o número de mensagens de saída enfileiradas para um processo do usuário.
- **Temporizador de intervalo:** O período de tempo entre dois eventos relacionados. Um exemplo é o período de tempo entre logins sucessivos em uma conta.
- **Utilização de recursos:** Quantidade de recursos consumidos durante um período especificado. Exemplos incluem o número de páginas impressas durante uma sessão do usuário e o tempo total consumido pela execução de um programa.

Dadas essas métricas gerais, diversos testes podem ser realizados para determinar se a atividade atual se encaixa dentro dos limites aceitáveis. [DENN87] lista as seguintes técnicas que podem ser usadas:

- Média e desvio padrão
- Multivariado
- Processo de Markov
- Séries de tempo
- Operacional

O teste estatístico mais simples é medir a **média** e o **desvio padrão** de um parâmetro durante algum período histórico. Isso oferece um reflexo do comportamento médio e sua variabilidade. O uso da média e do desvio padrão se aplica a uma grande variedade de contadores, temporizadores e medições de recursos. Mas essas medidas, por si só, normalmente são muito primitivas para fins de detecção de intrusão.

Um modelo **multivariado** baseia-se em correlações entre duas ou mais variáveis. O comportamento do intruso pode ser caracterizado com maior confiabilidade considerando-se essas correlações (por exemplo, tempo de processador e uso de recursos, ou freqüência de login e tempo gasto na sessão).

Um modelo do **processo de Markov** é usado para estabelecer probabilidades de transição entre vários estados. Como exemplo, esse modelo poderia ser usado para examinar transições entre certos comandos.

Um modelo de **série temporal** focaliza intervalos de tempo, observando seqüências de eventos que acontecem muito rapidamente ou muito lentamente. Diversos testes estatísticos podem ser aplicados para caracterizar a temporização anormal.

Finalmente, um **modelo operacional** baseia-se no julgamento daquilo que é considerado anormal, em vez de uma análise automatizada de registros passados de auditoria. Geralmente, limites fixos são definidos e suspeita-se de intrusão por uma observação daquilo que está fora dos limites. Esse tipo de técnica funciona melhor quando o comportamento do intruso pode ser deduzido a partir de certos tipos de atividades. Por exemplo, uma grande quantidade de tentativas de login em um curto período sugere uma tentativa de intrusão.

Como exemplo do uso dessas métricas e modelos, a Tabela 18.1 mostra diversas medidas consideradas ou testadas para o sistema de detecção de intrusão (IDES) do Stanford Research Institute (SRI) [DENN87, JAVI91, LUNT88].

A principal vantagem do uso de perfis estatísticos é que um conhecimento prévio das falhas de segurança não é exigido. O programa detector aprende o que é o comportamento 'normal' e depois procura desvios. A técnica não é baseada em características e vulnerabilidades dependentes do sistema. Assim, ela deverá ser prontamente portável entre uma série de sistemas.

Detecção de intrusão baseada em regras

Técnicas baseadas em regras detectam a intrusão observando eventos no sistema e aplicando um conjunto de regras que levam a uma decisão com relação a se determinado padrão de atividade é ou não suspeito. Em termos muito gerais, podemos caracterizar todas as técnicas como focalizando ou detectando anomalias ou a identificação de penetração, embora haja alguma sobreposição dessas abordagens.

A **detecção de anomalia baseada em regras** é semelhante à detecção estatística de anomalia em termos de abordagem utilizada e de pontos fortes. Com a técnica baseada em regras, registros de auditoria históricos são analisados para identificar padrões de uso e gerar automaticamente regras que descrevem esses padrões. As regras podem representar padrões de comportamento de usuários, programas, privilégios, intervalos de tempo, terminais etc. O comportamento atual é então observado, e cada transação é comparada com o conjunto de regras para determinar se está em conformidade com qualquer padrão de comportamento observado anteriormente.

Assim como a detecção estatística de anomalia, a detecção de anomalia baseada em regras não exige conhecimento das vulnerabilidades de segurança do sistema. Em vez disso, o esquema baseia-se na observação do comportamento

Tabela 18.1 Medidas que podem ser usadas para detecção de intrusão

Medida	Modelo	Tipo de intrusão detectada
Atividade de login e sessão		
Freqüência de login por dia e hora	Média e desvio padrão	Intrusos provavelmente poderão efetuar login durante horas livres.
Freqüência de login em diferentes locais	Média e desvio padrão	Intrusos podem efetuar login a partir de um local que um usuário em particular nunca ou quase nunca usa.
Tempo desde o último login	Operacional	Invasão em uma conta 'morta'.
Tempo decorrido por sessão	Média e desvio padrão	Desvios significativos podem indicar um mascarado.
Quantidade de saída para um local	Média e desvio padrão	Quantidades excessivas de dados transmitidos para locais remotos podem significar vazamento de dados sensíveis.
Utilização de recursos da sessão	Média e desvio padrão	Níveis incomuns de processador ou E/S podem sinalizar um intruso.
Falhas de senha no login	Operacional	Tentativa de entrada por adivinhação de senha.
Falhas de login a partir de terminais especificados	Operacional	Tentativa de invasão
Comando ou atividade de execução de programa		
Freqüência de execução	Média e desvio padrão	Pode detectar intrusos, que provavelmente usam comandos diferentes, ou uma penetração bem-sucedida por um usuário legítimo, que obteve acesso a comandos privilegiados.
Utilização de recursos do programa	Média e desvio padrão	Um valor anormal pode sugerir injeção de um vírus ou cavalo-de-tróia, que acarreta efeitos colaterais que aumentam a utilização de E/S ou processador.
Negações de execução	Modelo operacional	Pode detectar tentativa de penetração por usuário individual que busca privilégios mais altos.
Atividade de acesso a arquivo		
Freqüência de leitura, escrita, criação, exclusão	Média e desvio padrão	Anormalidades de acesso de leitura e escrita para usuários individuais pode significar mascaramento ou navegação.
Registros lidos, escritos	Média e desvio padrão	Anormalidade pode significa uma tentativa de obter dados importantes por inferência e agregação.
Contador de falhas de leitura, escrita, criação, exclusão	Operacional	Pode detectar usuários que tentam persistentemente acessar arquivos não autorizados.

passado e, na realidade, supondo que o futuro será como o passado. Para que essa técnica seja eficaz, um grande banco de dados de regras será necessário. Por exemplo, um esquema descrito em [VACC89] contém algo entre 10^4 e 10^6 regras.

A **identificação de penetração baseada em regras** usa uma técnica muito diferente para a detecção de intrusão baseada na tecnologia de um sistema especialista. Um recurso importante desse tipo de sistema é o uso de regras para identificar penetrações conhecidas ou penetrações que exploram pontos fracos conhecidos. Regras também podem ser definidas para identificar o comportamento suspeito, até mesmo quando o comportamento está dentro dos limites estabelecidos pelos padrões de uso. Normalmente, as regras usadas nesses sistemas são específicas para a máquina e sistema operacional.

Além disso, essas regras são geradas por 'especialistas' e não por meio de uma análise automatizada dos registros de auditoria. O procedimento normal é entrevistar administradores de sistemas e analistas de segurança para coletar um conjunto de cenários de penetração conhecidos e eventos importantes que ameaçam a segurança do sistema de destino.[1] Assim, o ponto forte da técnica depende da habilidade dos envolvidos na criação das regras.

Um exemplo simples do tipo de regras que podem ser usadas pode ser encontrado no NIDX, um sistema pioneiro que usava regras heurísticas que poderiam ser utilizadas para atribuir graus de suspeita às atividades [BAUE88].

1. Essas entrevistas podem até ser estendidas a crackers ativos ou não, que compartilharão sua experiência se forem pagos [FREE93].

Alguns exemplos de heurísticas são os seguintes:

1. Usuários não devem ler arquivos nos diretórios pessoais de outros usuários.
2. Usuários não devem escrever nos arquivos de outros usuários.
3. Usuários que efetuam login após o horário normal acessam freqüentemente os mesmos arquivos usados anteriormente.
4. Usuários geralmente não abrem dispositivos de disco diretamente, mas fazem uso de utilitários de nível mais alto do sistema operacional.
5. Usuários não devem estar logados mais de uma vez no mesmo sistema.
6. Usuários não fazem cópias dos programas do sistema.

O esquema de identificação de penetração usado no IDES representa a estratégia seguida. Os registros de auditoria são examinados à medida que são gerados e comparados com a base de regras. Se forem enquadrados em alguma regra, então a *avaliação de suspeita* do usuário aumenta. Se forem enquadrados em um número suficiente de regras, então a avaliação passará de um limiar, o que resulta no relato de uma anomalia.

A técnica do IDES é baseada em um exame dos registros de auditoria. Um ponto fraco desse plano é sua falta de flexibilidade. Para determinado cenário de penetração, pode haver diversas seqüências alternativas de registro de auditoria que podem ser produzidas, cada uma variando ligeiramente ou de modo sutil. Pode ser difícil localizar exatamente todas essas variações em regras explícitas. Outro método é desenvolver um modelo de nível mais alto independente dos registros de auditoria específicos. Um exemplo disso é um modelo de transição de estado conhecido como USTAT [ILGU93]. O USTAT lida com ações gerais em vez de ações detalhadas específicas, registradas pelo mecanismo de auditoria do UNIX. O USTAT é implementado no sistema SunOS que oferece registros de auditoria de 239 eventos. Desses, somente 28 são usados por um pré-processador, que os mapeia para 10 ações gerais (Tabela 18.2). Usando apenas essas ações e os parâmetros invocados a cada ação, um diagrama de transição de estado é desenvolvido para caracterizar a atividade suspeita. Como diversos eventos auditáveis diferentes estão mapeados para um número menor de ações, o processo de criação de regras é simples. Além disso, o modelo de diagrama de transição de estado é facilmente modificado para acomodar comportamentos de intrusão recentemente descobertos.

O mito da probabilidade de base

Para que tenha uso prático, um sistema de detecção de intrusão deve detectar uma porcentagem substancial de intrusões, enquanto mantém a taxa de alarmes falsos em um nível aceitável. Se apenas uma porcentagem moderada das intrusões reais for detectada, o sistema oferece um falso sentido de segurança. Por outro lado, se o sistema freqüentemente disparar um alerta quando não houver intrusão (um alarme falso), então os gerentes do sistema começarão a ignorar os alarmes, ou muito tempo será gasto analisando os alarmes falsos.

Tabela 18.2 Ações USTAT versus tipos de evento do SunOS

Ação USTAT	Tipo de evento do SunOS
Ler	open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt
Escrever	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct
Criar	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Apagar	rmdir, unlink
Executar	exec, execve
Sair	exit
Modificar_proprietário	chown, fchown
Modificar_permissão	chmod, fchmod
Renomear	rename
Hardlink	link

Infelizmente, devido à natureza das probabilidades envolvidas, é muito difícil atender o padrão de alta taxa de detecções com uma baixa taxa de alarmes falsos. Em geral, se os números reais de intrusões forem baixos em comparação com o número de usos legítimos de um sistema, então a taxa de alarmes falsos será alta, a menos que o teste seja extremamente discriminatório. Um estudo dos sistemas de detecção de intrusão existentes, informado em [AXEL00], indicou que os sistemas atuais não contornaram o problema do mito da probabilidade de base. Veja, no Apêndice 18A, uma rápida apresentação da matemática desse problema.

Detecção de intrusão distribuída

Até recentemente, o trabalho sobre sistemas de detecção de intrusão focalizava as estruturas isoladas de um único sistema. Uma organização típica, porém, precisa defender um conjunto distribuído de hosts que compõem uma LAN ou inter-rede. Embora seja possível montar uma defesa usando sistemas de detecção de intrusão isolados em cada host, uma defesa mais eficaz pode ser obtida pela coordenação e cooperação entre sistemas de detecção de intrusão na rede.

Porras aponta os seguintes problemas principais no projeto do sistema de detecção de intrusão distribuído [PORR92]:

- Um sistema de detecção de intrusão distribuído pode ter de lidar com diferentes formatos de registro de auditoria. Em um ambiente heterogêneo, sistemas diferentes empregarão diversos sistemas de coleta de auditoria nativos e, se estiverem usando a detecção de intrusão, podem empregar formatos variados para registros de auditoria relacionados à segurança.
- Um ou mais nós na rede servirão como pontos de coleta e análise dos dados dos sistemas na rede. Assim, ou dados brutos de auditoria ou dados resumidos devem ser transmitidos pela rede. Portanto, existe um requisito para garantir a integridade e a confidencialidade desses dados. A integridade é exigida para impedir que um intruso mascare suas atividades alterando as informações de auditoria transmitida. A confidencialidade é exigida porque as informações de auditoria transmitidas podem ser valiosas.
- Pode ser usada uma arquitetura centralizada ou descentralizada. Com uma arquitetura centralizada, existe um ponto de coleta e análise central único de todos os dados de auditoria. Isso facilita a tarefa de correlacionar os relatórios que chegam, mas cria um gargalo em potencial e um ponto único de falha. Com a arquitetura descentralizada, existe mais de um centro de análise, mas estes precisam coordenar suas atividades e a troca de informações.

Um bom exemplo de um sistema de detecção de intrusão distribuído é aquele desenvolvido na University of California, em Davis [HEBE92, SNAP91]. A Figura 18.2 mostra a arquitetura geral, que consiste em três componentes principais:

- **Módulo agente do host:** Um módulo de coleta de auditoria operando em segundo plano em um sistema monitorado. Sua finalidade é coletar dados sobre eventos relacionados à segurança no host e transmiti-los para o gerenciador central.
- **Módulo agente monitor da LAN:** Opera da mesma maneira que um módulo agente do host, exceto pelo fato de que analisa o tráfego da LAN e informa os resultados ao gerenciador central.
- **Módulo gerenciador central:** Recebe relatórios do monitor da LAN e agentes do host e processa e correlaciona esses relatórios para detectar a intrusão.

O esquema foi projetado para ser independente de qualquer sistema operacional ou implementação de sistema de auditoria. A Figura 18.3 [SNAP91] mostra a técnica geral utilizada. O agente capta cada registro de auditoria produzido pelo sistema de coleta de auditoria nativo. Um filtro que retém apenas os registros que são de interesse à segurança é aplicado. Esses registros são então reformatados em um formato padronizado conhecido como registro de auditoria do host (HAR – Host Audit Record). Em seguida, um módulo de lógica guiado por modelo analisa os registros em busca de atividades suspeitas. No nível mais baixo, o agente procura eventos notáveis que são de interesse, independentemente de quaisquer eventos passados. Alguns exemplos incluem falha no acesso a arquivos, o acesso a

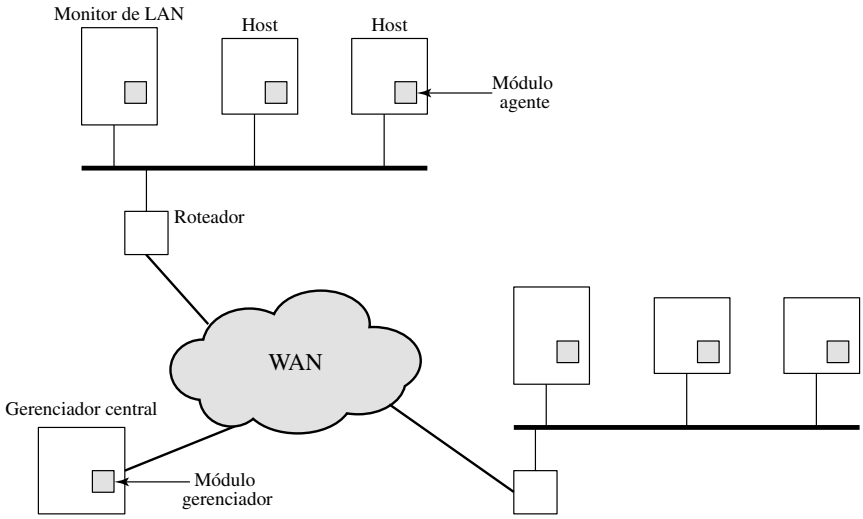


Figura 18.2 Arquitetura para detecção de intrusão distribuída.

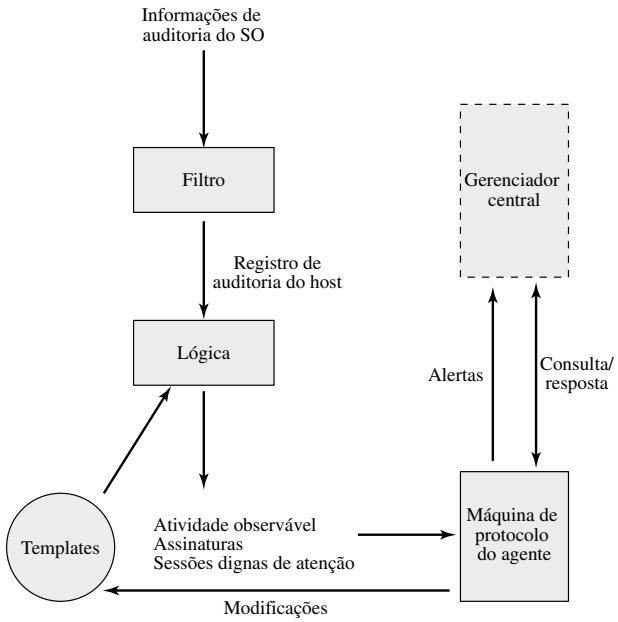


Figura 18.3 Arquitetura do agente.

arquivos do sistema e a alteração de arquivos de controle de acesso. No próximo nível mais alto, o agente procura seqüências de eventos, como padrões de ataque conhecidos (assinaturas). Finalmente, o agente procura um comportamento anômalo de um usuário individual baseado em um perfil histórico desse usuário, como o número de programas executados, número de arquivos acessados e coisas desse tipo.

Quando é detectada uma atividade suspeita, um alerta é enviado ao gerenciador central. O gerenciador central inclui um sistema especialista que pode tirar conclusões a partir dos dados recebidos. O gerenciador também pode consultar sistemas individuais para obter cópias de HARs para correlacionar com aquelas de outros agentes.

O agente monitor da LAN também fornece informações para o gerenciador central. O agente monitor da LAN examina as conexões host-host, serviços utilizados e volume de tráfego. Ele procura eventos significativos, como mudanças súbitas na carga da rede, o uso de serviços relacionados à segurança e atividades de rede como *rlogin*.

A arquitetura representada nas Figuras 18.2 e 18.3 é totalmente genérica e flexível. Ela oferece a base para uma abordagem independente da máquina, que pode se expandir a partir da detecção de intrusão independente para um sistema capaz de correlacionar a atividade de diversos locais e redes, para detectar a atividade suspeita que, de outra maneira, permaneceria não detectada.

Honeypots

Uma inovação relativamente recente na tecnologia de detecção de intrusão é o honeypot. Os honeypots são sistemas de armadilha, planejados para atrair um atacante em potencial para longe dos sistemas críticos. Os honeypots são projetados para

- desviar um atacante do acesso a sistemas críticos
- coletar informações sobre a atividade do atacante
- encorajar o atacante a permanecer no sistema por tempo suficiente para que os administradores respondam

Esses sistemas são recheados com informações inventadas, criadas para que pareçam ser valiosas, mas que um usuário legítimo do sistema não acessaria. Assim, qualquer acesso ao honeypot é suspeito. O sistema é preparado com monitores sensíveis e registradores de eventos que detectam esses acessos e coletam informações sobre as atividades do atacante. Como qualquer ataque contra o honeypot parece ser bem-sucedido, os administradores têm tempo para mobilizar, registrar e rastrear o atacante sem sequer expor sistemas em produção.

Os esforços iniciais envolviam em um único computador de honeypot com endereços de IP preparados para atrair os hackers. A pesquisa mais recente focalizou a criação de redes honeypot inteiras, que simulam uma empresa, possivelmente com tráfego e dados reais ou simulados. Quando os hackers estão dentro da rede, os administradores podem observar seu comportamento detalhadamente e delinear defesas.

Formato da troca de detecção de intrusão

Para facilitar o desenvolvimento de sistemas de detecção de intrusão distribuídos que possam funcionar em uma grande variedade de plataformas e ambientes, são necessários padrões para oferecer suporte à interoperabilidade. Esses

padrões são o foco do Intrusion Detection Working Group do IETF. A finalidade do grupo de trabalho é definir formatos de dados e procedimentos de troca para o compartilhamento de informações interessantes aos sistemas de detecção e resposta à intrusão e aos sistemas de gerenciamento que podem ter de interagir com eles. Os resultados desse grupo de trabalho incluem o seguinte:

1. Um documento de requisitos, que descreve os requisitos funcionais de alto nível para a comunicação entre os sistemas de detecção de intrusão e os requisitos para a comunicação entre sistemas de detecção de intrusão e com sistemas de gerenciamento, incluindo a razão para esses requisitos. Cenários serão usados para ilustrar esses requisitos.
2. Uma especificação de linguagem de intrusão comum, que descreve formatos de dados que satisfazem os requisitos.
3. Um documento de estrutura, que identifica os protocolos existentes mais utilizados para a comunicação entre os sistemas de detecção de intrusão, e descreve como os formatos dos dados inventados se relacionam a eles.

No momento em que este livro estava sendo escrito, todos esses documentos estavam sendo esboçados para a Internet.

18.3 GERENCIAMENTO DE SENHAS

Proteção de senhas

A primeira linha de defesa contra intrusos é o sistema de senhas. Praticamente todos os sistemas multiusuário exigem que um usuário forneça não apenas um nome ou identificador (ID), mas também uma senha. A senha serve para autenticar a ID do indivíduo efetuando o login no sistema. Por sua vez, a ID oferece segurança das seguintes maneiras:

- A ID determina se o usuário está autorizado a obter acesso a um sistema. Em alguns sistemas, apenas os que já têm uma ID armazenada no sistema têm permissão de acesso.
- A ID determina os privilégios concedidos ao usuário. Alguns usuários podem ter privilégio de supervisor ou 'superusuário' que lhes permita ler arquivos e realizar funções especialmente protegidas pelo sistema operacional. Alguns sistemas possuem contas anônimas ou de convidado, e os usuários dessas contas têm privilégios mais limitados do que outros.
- A ID é usada no que é chamado de controle de acesso discricionário. Por exemplo, listando as IDs dos outros usuários, um usuário pode conceder permissão para que eles leiam arquivos pertencentes a esses usuários.

A vulnerabilidade das senhas Para entender a natureza da ameaça aos sistemas baseados em senha, vamos considerar um esquema que é amplamente utilizado no UNIX, em que as senhas nunca são armazenadas às claras. Em vez disso, o procedimento a seguir é empregado (Figura 18.4a). Cada usuário seleciona uma senha de até oito caracteres de extensão entre um conjunto de letras, números e alguns símbolos. Esta é convertida em um valor de 56 bits (usando ASCII com 7 bits) que serve como chave de entrada para uma rotina de criptografia. A rotina de criptografia, conhecida como crypt(3), é baseada no DES. O algoritmo DES é modificado usando um valor de 'sal' de 12 bits. Normalmente, esse valor está relacionado ao momento em que a senha é atribuída ao usuário. O algoritmo DES modificado é exercitado com uma entrada de dados consistindo em um bloco de 64 bits de zeros. A saída do algoritmo, então, serve como entrada para uma segunda criptografia. Esse processo é repetido por um total de 25 criptografias. A saída de 64 bits resultante é então traduzida para uma seqüência de 11 caracteres. O hash da senha é então armazenado, juntamente com uma cópia em texto claro do valor de sal, no arquivo de senha para a ID correspondente de usuário. Esse método mostrou ser seguro contra diversos ataques criptoanalíticos [WAGN00].

O sal tem três finalidades:

- Ele impede que senhas duplicadas sejam visíveis no arquivo de senhas. Mesmo que dois usuários escolham a mesma senha, estas serão atribuídas em momentos diferentes. Logo, as senhas 'estendidas' dos dois usuários serão diferentes.
- Ele efetivamente aumenta o comprimento da senha sem exigir que o usuário se lembre de dois caracteres adicionais. Logo, o número de senhas possíveis é aumentado por um fator de 4.096, aumentando a dificuldade de adivinhar uma senha.
- Ele impede o uso de uma implementação em hardware do DES, que minimizaria a dificuldade de um ataque de adivinhação por força bruta.

Quando um usuário tenta efetuar o logon em um sistema UNIX, ele fornece uma ID e uma senha. O sistema operacional usa a ID para indexar o arquivo de senhas e apanhar o sal em texto claro e a senha criptografada. O sal e a senha fornecida pelo usuário são usados como entrada para a rotina de criptografia. Se o resultado for igual ao valor armazenado, a senha será aceita.

A rotina de criptografia é projetada para desencorajar ataques de adivinhação. As implementações de software do DES são lentas em comparação com as versões de hardware, e o uso de 25 iterações multiplica o tempo exigido por 25.

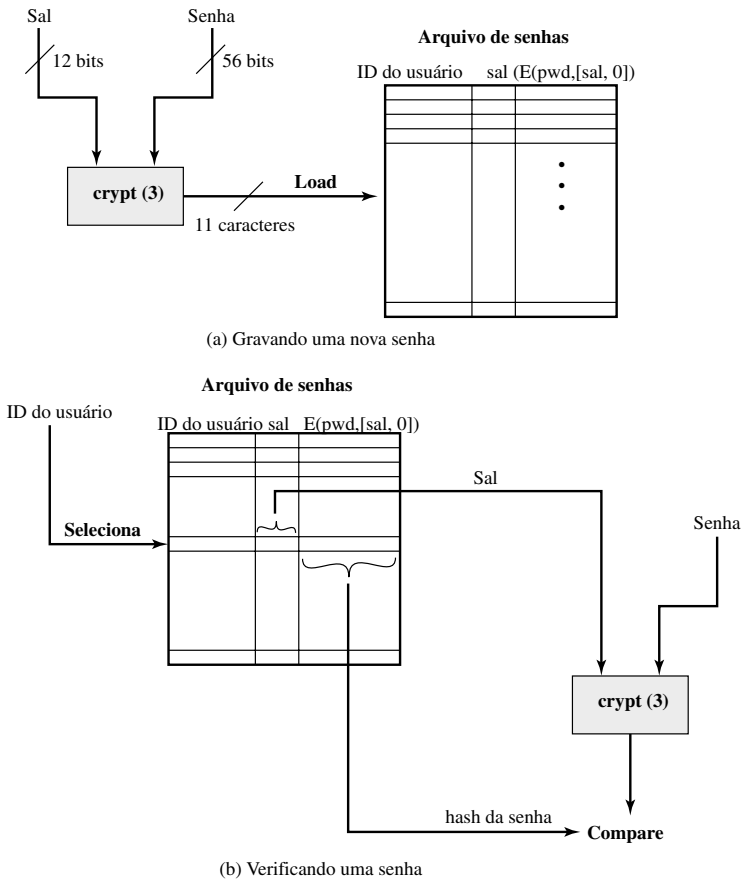


Figura 18.4 Esquema de senha do UNIX.

Porém, desde o projeto original desse algoritmo, duas mudanças ocorreram. Primeiro, as implementações mais recentes do próprio algoritmo resultaram em ganhos de velocidade. Por exemplo, o verme de Internet descrito no Capítulo 19 foi capaz de realizar adivinhação on-line de algumas centenas de senhas em um tempo relativamente curto, usando um algoritmo de criptografia mais eficiente do que o padrão armazenado nos sistemas UNIX sob ataque. Segundo, o desempenho do hardware continua a aumentar, de modo que qualquer algoritmo de software é executado mais rapidamente.

Assim, existem duas ameaças a um esquema de senha no UNIX. Primeiro, um usuário pode obter acesso a uma máquina usando uma conta de convidado ou por algum outro meio e depois executar um programa de adivinhação de senhas, chamado quebrador de senhas (password cracker), na máquina. O atacante deverá ser capaz de verificar centenas e talvez milhares de senhas possíveis com pouco consumo de recursos. Além disso, se um oponente for capaz de obter uma cópia do arquivo de senhas, então um programa quebrador poderá ser executado em outra máquina à vontade. Isso permite que o oponente percorra muitos milhares de senhas possíveis em um período razoável.

Como exemplo, um cracker de senhas foi relatado na Internet em agosto de 1993 [MADS93]. Usando um computador paralelo da Thinking Machines Corporation, foi alcançado um desempenho de 1.560 criptografias por segundo por unidade de vetor. Com quatro unidades de vetor por nó de processamento (uma configuração-padrão), chegou-se a 800 mil criptografias por segundo em uma máquina de 128 nós (que é um tamanho modesto) e 6,4 milhões de criptografias por segundo em uma máquina de 1.024 nós.

Até mesmo essas taxas de adivinhação estupendas ainda não tornam viável para um atacante usar uma técnica burra de força bruta ao tentar todas as combinações possíveis de caracteres para descobrir uma senha. Em vez disso, os crackers de senhas contam com o fato de que algumas pessoas escolhem senhas de fácil adivinhação.

Alguns usuários, quando podem escolher sua própria senha, escolhem uma que é absurdamente curta. Os resultados de um estudo na Purdue University aparecem na Tabela 18.3. O estudo observou escolhas de mudança de senhas em 54 máquinas, representando aproximadamente 7.000 contas de usuário. Quase 3 por cento das senhas tinham 3 caracteres ou menos. Um atacante poderia iniciar o ataque testando exaustivamente todas as senhas possíveis de tamanho 3 ou menor. Um remédio simples é que o sistema rejeite qualquer escolha de senha com menos do que, digamos, seis caracteres, ou ainda que exija que todas as senhas tenham exatamente oito caracteres de extensão. A maioria dos usuários não reclamaria dessa restrição.

Tabela 18.3 Tamanhos de senha observados [SPAF92a]

Tamanho	Número	Fração do total
1	55	0,004
2	87	0,006
3	212	0,02
4	449	0,03
5	1260	0,09
6	3035	0,22
7	2917	0,21
8	5772	0,42
Total	13787	1,0

O tamanho da senha é apenas parte do problema. Muitas pessoas, quando têm permissão para escolher sua própria senha, escolhem uma senha que pode ser descoberta, como seu próprio nome, o nome de sua rua, uma palavra comum do dicionário etc. Isso torna o trabalho de quebra de senhas muito simples. O cracker de senhas simplesmente precisa testar o arquivo de senhas em relação a listas de senhas prováveis. Como muitas pessoas usam senhas de fácil descoberta, essa estratégia deve ser bem-sucedida em praticamente todos os sistemas.

Uma demonstração da eficácia da adivinhação é relatada em [KLEI90]. A partir de diversas fontes, o autor coletou arquivos de senhas do UNIX, contendo quase 14.000 senhas criptografadas. O resultado, que o autor corretamente caracteriza como assustador, aparece na Tabela 18.4. Resumindo, quase um quarto das senhas foram descobertas. A estratégia a seguir foi utilizada:

1. Experimentar o nome do usuário, iniciais, nome de conta e outras informações pessoais relevantes. Ao todo, 130 permutações diferentes para cada usuário foram testadas.
2. Experimentar palavras de vários dicionários. O autor compilou um dicionário de mais de 60 mil palavras, incluindo o dicionário on-line no próprio sistema, e diversas outras listas, conforme mostrado.
3. Experimentar diversas permutações sobre as palavras da etapa 2. Isso incluiu tornar a primeira letra maiúscula ou um caractere de controle, tornar a palavra inteira maiúscula, inverter a palavra, alterar a letra 'o' pelo dígito 'zero' etc. Essas permutações acrescentaram mais um milhão de outras palavras à lista.
4. Experimentar diversas permutações de maiúsculas e minúsculas sobre as palavras da etapa 2 que não foram consideradas na etapa 3. Isso acrescentou quase 2 milhões de palavras adicionais à lista.

Assim, o teste envolveu perto de 3 milhões de palavras. Usando a implementação da Thinking Machines listada anteriormente, o tempo para criptografar todas essas palavras para todos os valores de sal possíveis é menor do que uma hora. Tenha em mente que essa pesquisa completa poderia produzir uma taxa de sucesso de aproximadamente 25 por cento, sendo que até mesmo um único acerto pode ser suficiente para obter uma grande variedade de privilégios em um sistema.

Controle de acesso Uma maneira de impedir um ataque de senha é negar o acesso do oponente ao arquivo de senhas. Se a parte da senha criptografada do arquivo for acessível apenas por um usuário privilegiado, então o oponente não poderá lê-la sem já conhecer a senha de um usuário privilegiado. [SPAF92a] aponta várias falhas nessa estratégia:

- Muitos sistemas, incluindo a maioria dos sistemas UNIX, são suscetíveis a invasões inesperadas. Uma vez que um atacante tenha obtido acesso por algum meio, ele poderá querer obter uma coleção de senhas a fim de usar diferentes contas para diferentes sessões de login, para diminuir o risco de detecção. Ou então um usuário com uma conta pode desejar a conta de outro usuário para acessar dados privilegiados ou sabotar o sistema.
- Um acidente de proteção poderia tornar o arquivo de senhas legível, comprometendo assim todas as contas.
- Alguns dos usuários possuem contas em outras máquinas em outros domínios de proteção, e eles usam a mesma senha. Assim, se as senhas puderem ser lidas por alguém em uma máquina, uma máquina em outro local poderá ser comprometida.

Assim, uma estratégia mais eficaz seria forçar os usuários a selecionar senhas difíceis de adivinhar.

Estratégias de seleção de senhas

A lição das duas experiências anteriormente descritas (Tabelas 18.3 e 18.4) é que, por decisão própria, muitos usuários escolhem uma senha muito curta ou muito fácil de descobrir. Em outro extremo, se os usuários receberem senhas consistindo de oito caracteres imprimíveis selecionados aleatoriamente, a quebra da senha será efetivamente impossível. Mas seria quase impossível que a maioria dos usuários se lembrasse de suas senhas. Felizmente, mesmo que limitemos o universo da senha a uma seqüência de caracteres que sejam razoavelmente fáceis de lembrar, o tamanho do universo ainda seria muito grande para permitir a quebra na prática. Nosso objetivo, então, é eliminar senhas de fácil descoberta e permitir que o usuário selecione uma senha que seja fácil de lembrar. Quatro técnicas básicas são utilizadas:

Tabela 18.4 Senhas quebradas a partir de uma amostra de 13.797 contas [KLEI90]

Tipo de senha	Tamanho da busca	Número de combinações	Porcentagem de senhas combinadas	Relação Custo/Benefício[a]
Nome/conta de usuário	130	368	2,7%	2,830
Seqüências de caracteres	866	22	0,2%	0,025
Números	427	9	0,1%	0,021
Chinês	392	56	0,4%	0,143
Nomes de lugares	628	82	0,6%	0,131
Nomes comuns	2239	548	4,0%	0,245
Nomes femininos	4280	161	1,2%	0,038
Nomes masculinos	2866	140	1,0%	0,049
Nomes incomuns	4955	130	0,9%	0,026
Mitos e lendas	1246	66	0,5%	0,053
Shakespeareanas	473	11	0,1%	0,023
Termos de esportes	238	32	0,2%	0,134
Ficção científica	691	59	0,4%	0,085
Filmes e atores	99	12	0,1%	0,121
Desenhos animados	92	9	0,1%	0,098
Pessoas famosas	290	55	0,4%	0,190
Frases e padrões	933	253	1,8%	0,271
Nomes de família	33	9	0,1%	0,273
Biologia	58	1	0,0%	0,017
Dicionário do sistema	19683	1027	7,4%	0,052
Nomes de máquina	9018	132	1,0%	0,015
Mnemônicos	14	2	0,0%	0,143
Bíblia na versão King James	7525	83	0,6%	0,011
Palavras variadas	3212	54	0,4%	0,017
Palavras iídiche	56	0	0,0%	0,000
Asteróides	2407	19	0,1%	0,007
TOTAL	62727	3340	24,2%	0,053

[a] Calculada como o número de combinações dividido pelo tamanho da busca. Quanto mais palavras precisaram ser testadas para uma combinação, menor a relação custo/benefício.

- Treinamento do usuário
- Senhas geradas pelo computador
- Verificação reativa da senha
- Verificação pró-ativa da senha

Os usuários podem ser informados da importância de usar senhas de difícil descoberta e podem receber orientações para selecionar senhas fortes. Essa estratégia de **treinamento do usuário** provavelmente não terá sucesso na maioria das instalações, particularmente onde houver um grande contingente ou muita rotatividade. Muitos usuários simplesmente virão ignorar as orientações. Outros podem não entender bem o que é uma senha forte. Por exemplo, muitos usuários (erroneamente) acreditam que reverter uma palavra ou colocar a última letra em maiúscula torna uma senha indecifrável.

Senhas geradas pelo computador também têm problemas. Se as senhas forem de natureza muito aleatória, os usuários não serão capazes de se lembrar delas. Mesmo que a senha seja pronunciável, o usuário poderá ter dificuldade para lembrá-la e, por isso será tentado a escrevê-la. Em geral, os esquemas de senha gerados pelo computador possuem um histórico de fraca aceitação pelos usuários. FIPS PUB 181 define um dos geradores de senha automatizados mais bem projetados. O padrão inclui não apenas uma descrição da técnica, mas também uma listagem completa do código-fonte do algoritmo em linguagem C. O algoritmo gera palavras formando sílabas pronunciáveis, concatenando-as para formar uma palavra. Um gerador de números aleatórios produz um fluxo aleatório de caracteres usados para criar sílabas e palavras.

Uma estratégia de **verificação reativa de senha** é aquela em que o sistema executa periodicamente seu próprio quebrador de senhas para encontrar senhas de fácil descoberta. O sistema cancela quaisquer senhas descobertas e noti-

fica o usuário. Essa tática tem diversas desvantagens. Primeiro, ela exige muitos recursos se a tarefa for feita corretamente. Como um determinado oponente que é capaz de roubar um arquivo de senhas pode dedicar tempo de CPU total para a tarefa por horas ou até mesmo dias, um verificador reativo de senhas eficaz está em clara desvantagem. Além disso, quaisquer senhas existentes permanecem vulneráveis até que o verificador de senhas reativo as encontre.

A técnica mais promissora para maior segurança das senhas é o **verificador proativo de senhas**. Nesse esquema, um usuário tem permissão para selecionar sua própria senha. Porém, no momento da seleção, o sistema verifica se a senha é permitida e, se não for, ela será rejeitada. Esses verificadores são baseados na filosofia de que, com orientação suficiente do sistema, os usuários podem selecionar senhas de fácil memorização a partir de um conjunto de senhas muito grande, as quais provavelmente não serão descobertas em um ataque de dicionário.

O truque com um verificador proativo de senhas é conseguir um equilíbrio entre aceitabilidade do usuário e força. Se o sistema rejeitar muitas senhas, os usuários reclamarão que é muito difícil selecionar uma senha. Se o sistema usar algum algoritmo simples para definir o que é aceitável, isso orientará os quebradores de senhas para que refinem sua técnica de adivinhação. No restante desta subseção, examinamos as técnicas possíveis para a verificação proativa de senhas.

A primeira técnica é um sistema simples para imposição de regras. Por exemplo, as seguintes regras poderiam ser impostas:

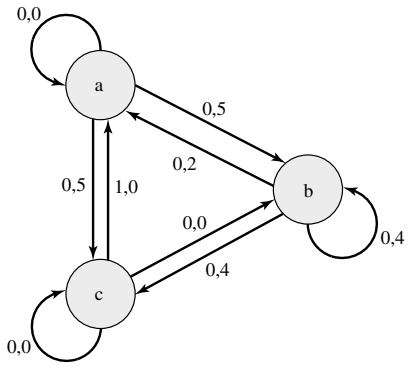
- Todas as senhas precisam ter pelo menos oito caracteres de extensão.
- Nos oito primeiros caracteres, as senhas precisam incluir pelo menos uma letra maiúscula, minúscula, dígito numérico e sinal de pontuação.

Essas regras poderiam ser fornecidas em conjunto com uma orientação aos usuários. Embora essa técnica seja superior a simplesmente treinar os usuários, pode não ser suficiente para deter os quebradores de senhas. Esse esquema alerta os quebradores sobre quais senhas não tentar, mas ainda pode tornar possível a quebra de senhas.

Outro procedimento possível é simplesmente compilar um grande dicionário de possíveis senhas 'ruins'. Quando um usuário seleciona uma senha, o sistema verifica se ela não está na lista de desaprovadas. Existem dois problemas com essa técnica:

- **Espaço:** O dicionário precisa ser muito grande para ser eficaz. Por exemplo, o dicionário usado no estudo da Purdue [SPAF92a] ocupa mais de 30 megabytes.
- **Tempo:** O tempo exigido para pesquisar um grande dicionário pode ser muito longo. Além disso, para verificar prováveis permutações de palavras do dicionário, essas palavras precisam ser incluídas no dicionário, tornando-o verdadeiramente imenso, ou cada busca também terá de envolver um processamento considerável.

Duas técnicas são promissoras para desenvolver um verificador de senhas proativo eficiente e eficaz baseado na rejeição de palavras em uma lista. Uma delas desenvolve um modelo de Markov para a geração de senhas previsíveis [DAVI93]. A Figura 18.5 mostra uma versão simplificada desse modelo. O modelo mostra uma linguagem consistindo em um alfabeto de três caracteres. O estado do sistema a qualquer momento é a identidade da letra mais recente. O valor na transição de um estado para outro representa a probabilidade de que uma letra venha após outra. Assim, a probabilidade de que a próxima letra seja b, dado que a letra atual é a, é de 0,5.



M = {3, {a, b, c} , T, 1 } onde

$$T = \begin{bmatrix} 0{,}0 & 0{,}5 & 0{,}5 \\ 0{,}2 & 0{,}4 & 0{,}4 \\ 1{,}0 & 0{,}0 & 0{,}0 \end{bmatrix}$$

por exemplo, a seqüência desta linguagem provavelmente é: abbcacaba

por exemplo, seqüência desta linguagem provavelmente não é: aacccbaaa

Figura 18.5 Um exemplo do modelo de Markov.

Em geral, um modelo de Markov é uma quádrupla $[m, A, \mathbf{T}, k]$, onde m é o número de estados no modelo, A é o espaço do estado, $\mathbf{T}$ é a matriz de probabilidades de transição e k é a ordem do modelo. Para um modelo de ordem k, a probabilidade de haver uma transição para uma letra em particular depende das k letras anteriores geradas. A Figura 18.5 mostra um modelo simples de primeira ordem.

Os autores relatam o desenvolvimento e uso de um modelo de segunda ordem. Para começar, um dicionário de senhas previsíveis é criado. Depois, a matriz de transição é calculada da seguinte maneira:

1. Determinar a matriz de freqüência $\mathbf{f}$, onde $\mathbf{f}(i, j, k)$ é o número de ocorrências do trigrama consistindo nos caracteres de ordem i, j e k. Por exemplo, a senha *parsnips* gera os trigramas par, ars, rsn, sni, nip e ips.
2. Para cada digrama ij, calcular $\mathbf{f}(i, j, \infty)$ como o número total de trigramas começando com ij. Por exemplo, $\mathbf{f}(\mathrm{a}, \mathrm{b}, \infty)$ seria o número total de trigramas da forma aba, abb, abc e assim por diante.
3. Calcular as entradas de $\mathbf{T}$ da seguinte maneira:

$$\mathbf{T}(i, j, k) = \frac{\mathbf{f}(i, j, k)}{\mathbf{f}(i, j, \infty)}$$

O resultado é um modelo que reflete a estrutura das palavras no dicionário. Com esse modelo, a pergunta "Esta é uma senha ruim?" é transformada na pergunta "Esta seqüência (senha) foi gerada por esse modelo de Markov?" Para determinada senha, as probabilidades de transição de todos os seus trigramas podem ser pesquisadas. Alguns testes estatísticos padrão podem então ser usados para determinar se a senha é provável ou improvável para esse modelo. As senhas prováveis de serem geradas pelo modelo são rejeitadas. Os autores relatam bons resultados para um modelo de segunda ordem. Seu sistema contém praticamente todas as senhas em seu dicionário e não exclui tantas senhas potencialmente boas, evitando ser não amigável ao usuário.

Uma técnica totalmente diferente foi relatada por Spafford [SPAF92a, SPAF92b]. Ela é baseada no uso de um filtro de Bloom [BLOO70]. Para começar, explicamos a operação do filtro de Bloom. Esse filtro de ordem k consiste em um conjunto de k funções de hash independentes $H_1(x), H_2(x),...,H_k(x)$, onde cada função relaciona uma senha a um valor de hash no intervalo de 0 a $N - 1$. Isto é,

$$H_i(\mathrm{X}_j) = y \qquad 1 \le i \le k; \qquad 1 \le j \le D; \qquad 0 \le y \le N - 1$$

onde

X_j = j-ésima palavra no dicionário de senhas
D = número de palavras no dicionário de senhas

O seguinte procedimento é então aplicado ao dicionário:

1. Uma tabela de hash de N bits é definida, com todos os bits inicialmente definidos como 0.
2. Para cada senha, seus k valores de hash são calculados, e os bits correspondentes na tabela de hash são definidos como 1. Assim, se $H_i(\mathrm{X}_j) = 67$ para algum (i, j), então o sexagésimo sétimo bit da tabela de hash é definido como 1; se o bit já tiver o valor 1, ele permanecerá em 1.

Quando uma nova senha é apresentada ao verificador, seus k valores de hash são calculados. Se todos os bits correspondentes da tabela de hash forem iguais a 1, então a senha será rejeitada. Todas as senhas no dicionário serão rejeitadas. Mas também haverá alguns 'falsos positivos' (ou seja, senhas que não estão no dicionário, mas que produzem uma combinação adequada na tabela de hash). Para verificar isso, considere um esquema com duas funções de hash. Suponha que as senhas *undertaker* e *hulkhogan* estejam no dicionário, mas *xG%#jj98* não esteja. Além disso, suponha que

H_1(undertaker) = 25	H_1(hulkhogan) = 83	H_1(xG%#jj98) = 665
H_2(undertaker) = 998	H_2(hulkhogan) = 665	H_2(xG%#jj98) = 998

Se a senha xG%#jj98 for apresentada ao sistema, ela será rejeitada, mesmo que não esteja no dicionário. Se houver muitos desses falsos positivos, será difícil que os usuários selecionem senhas. Portanto, gostaríamos de projetar um esquema de hash para minimizar falsos positivos. Pode-se mostrar que a probabilidade de um falso positivo pode ser aproximada por

$$P \approx (1 - \mathrm{e}^{kD/N})^k = (1 - \mathrm{e}^{k/R})^k$$

ou, de forma equivalente,

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

onde

k = número de funções de hash
N = número de bits na tabela de hash
D = número de palavras no dicionário
$R = N/D$, quociente do tamanho da tabela de hash (bits) pelo tamanho do dicionário (palavras)

A Figura 18.6 representa graficamente P como uma função de R para diversos valores de k. Suponha que tenhamos um dicionário de 1 milhão de palavras e desejamos ter uma probabilidade de 0,01 de rejeitar uma senha que não esteja no dicionário. Se escolhermos seis funções de hash, a razão exigida será $R = 9{,}6$. Portanto, precisamos de uma tabela de hash de $9{,}6 \times 10^6$ bits ou cerca de 1,2 Mbytes de armazenamento. Em contraste, o armazenamento do dicionário inteiro exigiria algo na ordem de 8 Mbytes. Assim, conseguimos uma compactação com um fator de quase 7. Além disso, a verificação de senhas envolve o cálculo direto de seis funções de hash e é independente do tamanho do dicionário, enquanto com o uso do dicionário completo existe uma busca substancial.[2]

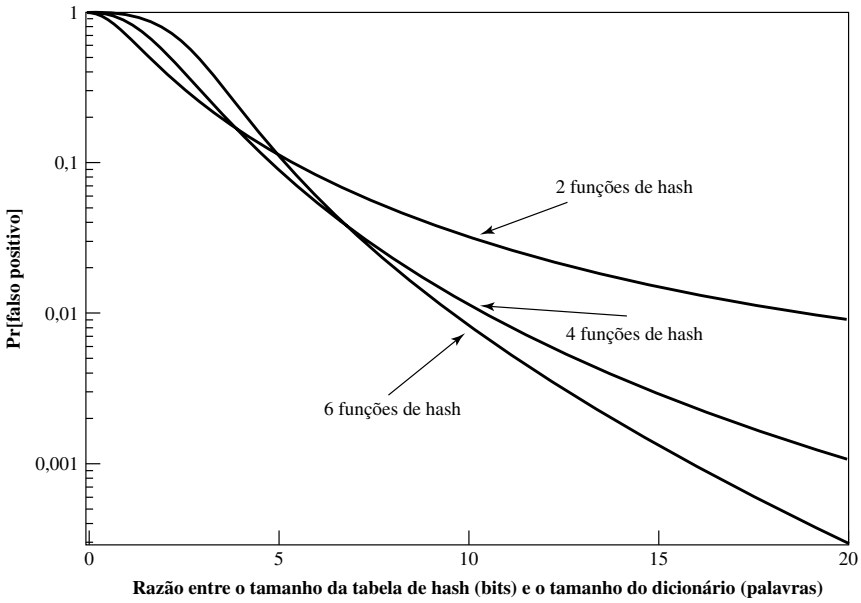


Figura 18.6 Desempenho do filtro de Bloom.

18.4 LEITURA E SITES WEB RECOMENDADOS

Duas abordagens completas da detecção de intrusão são [BACE00] e [PROC01]. Uma abordagem mais concisa, porém muito valiosa, é [BACE01]. Dois artigos de estudo curtos sobre o assunto, porém úteis, são [KENT00] e [MCHU00]. [NING04] analisa os avanços recentes em técnicas de detecção de intrusão. [HONE01] é um relato definitivo sobre honeypots, e oferece uma análise detalhada das ferramentas e métodos dos hackers.

BACE00 Bace, R. *Intrusion detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R. e Mell, P. *Intrusion detection systems*. NIST Special Publication SP 800-31, novembro de 2000.

HONE01 The Honeynet Project. *Know your enemy: revealing the security tools, tactics, and motives of the blackhat community*. Reading: Addison-Wesley, 2001.

KENT00 Kent, S. "On the trail of intrusions into information systems." *IEEE Spectrum*, dezembro de 2000.

MCHU00 McHugh, J.; Christie, A. e Allen, J. "The role of intrusion detection systems." *IEEE Software*, setembro/outubro de 2000.

NING04 Ning, P. e outros. "Techniques and tools for analyzing intrusion alerts." *ACM Transactions on Information and System Security*, maio de 2004.

PROC01 Proctor, P. *The practical intrusion detection handbook*. Upper Saddle River: Prentice Hall, 2001.

2. Tanto o modelo de Markov quanto o filtro de Bloom envolvem o uso de técnicas probabilísticas. No caso do modelo de Markov, existe uma pequena probabilidade de que algumas senhas no dicionário não sejam captadas e uma pequena probabilidade de que algumas senhas que não estejam no dicionário sejam rejeitadas. No caso do filtro de Bloom, existe uma pequena probabilidade de que algumas senhas que não estejam no dicionário sejam rejeitadas. Novamente, veremos que usar uma abordagem probabilística simplifica a solução (por exemplo, veja a nota de rodapé 1 no Capítulo 15).

Sites Web recomendados:

- **CERT Coordination Center:** A organização que cresceu a partir da equipe de resposta a emergências de computador, formada pela Defense Advanced Research Projects Agency. O site oferece boas informações sobre ameaças de segurança pela Internet, vulnerabilidades e estatísticas de ataque.
- **Honeynet Project:** Um projeto de pesquisa que estuda as técnicas de hackers predatórios e o desenvolvimento de produtos honeypot.
- **Honeypots:** Uma boa coleção de trabalhos de pesquisa e artigos técnicos.
- **Intrusion Detection Working Group:** Inclui todos os documentos gerados por este grupo.

18.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

detecção de intrusão
detecção de intrusão baseada em regras
detecção estatística de anomalia
formato de troca de detecção de intrusão
honeypot
intruso
mito da probabilidade de base
registro de auditoria
sal
senha
Teorema de Bayes

Perguntas de revisão

18.1 Liste e defina resumidamente três classes de intrusos.

18.2 Quais são as duas técnicas comuns usadas para proteger um arquivo de senhas?

18.3 Quais são os três benefícios que podem ser oferecidos por um sistema de detecção de intrusão?

18.4 Qual é a diferença entre detecção estatística de anomalia e detecção de intrusão baseada em regras?

18.5 Que medidas são úteis para a detecção de intrusão baseada em perfil?

18.6 Qual é a diferença entre detecção de anomalia baseada em regras e identificação de penetração baseada em regras?

18.7 O que é um honeypot?

18.8 O que é um 'sal' no contexto do gerenciamento de senhas do UNIX?

18.9 Liste e defina resumidamente quatro técnicas utilizadas para evitar senhas de fácil adivinhação.

Problemas

18.1 Um táxi esteve envolvido em um acidente fatal à noite e fugiu após a colisão. Duas empresas de táxi, a Verde e a Azul, operam na cidade. Você sabe que

- 85% dos táxis na cidade são da empresa Verde e 15% são da empresa Azul.
- Uma testemunha identificou o táxi como sendo da empresa Azul.
- O tribunal testou a confiabilidade da testemunha sob algumas circunstâncias que existiam na noite do acidente e concluiu que a testemunha estava correta na identificação da cor do táxi 80 por cento do tempo. Qual é a probabilidade de que o táxi envolvido no acidente seja da empresa Azul em vez de ser da empresa Verde?

18.2 Suponha que senhas sejam selecionadas a partir de combinações de quatro caracteres das 26 letras do alfabeto. Suponha que um adversário seja capaz de testar senhas a uma velocidade de uma por segundo.

a. Supondo que não haja qualquer retorno ao adversário até que cada tentativa tenha sido concluída, qual é o tempo esperado para descobrir a senha correta?

b. Supondo um retorno ao adversário, sinalizando um erro quando cada letra incorreta é inserida, qual é o tempo esperado para se descobrir a senha correta?

18.3 Suponha que elementos de origem de comprimento k sejam mapeados em algum padrão uniforme para elementos de destino de comprimento p. Se cada dígito puder assumir um dentre r valores, então o número de elementos de origem será r^k e o número de elementos de destino será o número menor r^p. Um elemento de origem em particular x_i é mapeado em um elemento de destino em particular y_j.

a. Qual é a probabilidade de que o elemento de origem correto possa ser selecionado por um adversário em apenas uma tentativa?

b. Qual é a probabilidade de que um diferente elemento de origem $x_k (x_i \neq x_k)$, que resulte no mesmo elemento de destino, y_j, possa ser produzido por um adversário?

c. Qual é a probabilidade de que o elemento de destino correto possa ser produzido por um adversário em apenas uma tentativa?

18.4 Um gerador de senhas fonéticas seleciona dois

segmentos aleatoriamente para cada senha de seis letras. A forma de cada segmento é CVC (consoante, vogal, consoante), onde V = <a,e,i,o,u> e C = $\overline{V}$.

a. Qual é o universo total de senhas?
b. Qual é a probabilidade de um adversário adivinhar uma senha corretamente?

18.5 Suponha que as senhas sejam limitadas ao uso dos 95 caracteres ASCII imprimíveis e que todas as senhas tenham 10 caracteres de comprimento. Considere um quebrador de senhas com uma taxa de criptografia de 6,4 milhões de criptografias por segundo. Quanto tempo levará para testar exaustivamente todas as senhas possíveis em um sistema UNIX?

18.6 Devido aos riscos conhecidos do sistema de senhas do UNIX, a documentação do SunOS-4.0 recomenda que o arquivo de senhas seja removido e substituído por um arquivo legível publicamente, chamado /etc/publickey. Uma entrada no arquivo para o usuário A consiste em um identificador de usuário ID_A, na chave pública do usuário, PU_a e na chave privada correspondente PR_a. Essa chave privada é criptografada usando DES com uma chave derivada da senha de login do usuário P_a. Quando A efetua o login, o sistema decriptografa $E[P_a,PR_a]$ para obter PR_a.

a. O sistema, então, verifica se P_a foi fornecida corretamente. Como?
b. Como um oponente pode atacar esse sistema?

18.7 O esquema de criptografia usado para senhas do UNIX é unidirecional; não é possível revertê-lo. Portanto, seria exato dizer que este é, na realidade, um código de hash em vez de uma criptografia da senha?

18.8 Foi dito que a inclusão do sal no esquema de senhas do UNIX aumenta a dificuldade de se descobrir por um fator de 4.096. Mas o sal é armazenado em texto claro na mesma entrada da senha de texto cifrado correspondente. Portanto, aqueles dois caracteres são conhecidos pelo atacante e não precisam ser descobertos. Por que se afirma que o sal aumenta a segurança?

18.9 Supondo que você tenha respondido corretamente o problema anterior e entendido o significado do sal, aqui está outra pergunta. Seria possível impedir completamente todos os quebradores de senhas aumentando drasticamente o tamanho do sal para, digamos, 24 ou 48 bits?

18.10 Considere o filtro de Bloom explicado na Seção 18.3. Defina k = número de funções de hash; N = número de bits na tabela de hash; e D = número de palavras no dicionário.

a. Mostre que o número esperado de bits na tabela de hash que são iguais a zero é expresso como:

$$\phi = \left(1 - \frac{k}{N}\right)^D$$

b. Mostre que a probabilidade de que uma palavra digitada, que não esteja no dicionário, seja falsamente aceita como estando no dicionário é:

$$P = (1 - \phi)^k$$

c. Mostre que a expressão anterior pode ser aproximada como

$$P \approx (1 - e^{-kD/N})^k$$

18.11 Projete um sistema de acesso a arquivos para permitir que certos usuários tenham acesso de leitura e de escrita a um arquivo, dependendo da autorização configurada no sistema. As instruções deverão estar no formato

READ (F, User A): tentativa pelo usuário A de ler o arquivo F
WRITE (F, User A): tentativa pelo usuário A de armazenar uma cópia possivelmente modificada de F

Cada arquivo tem um *registro de cabeçalho* que contém privilégios de autorização, ou seja, uma lista de usuários que podem ler e escrever. O arquivo deve ser criptografado por uma chave que não é compartilhada pelos usuários, mas conhecida apenas pelo sistema.

APÊNDICE 18A O MITO DA PROBABILIDADE DE BASE

Começamos com uma revisão dos resultados importantes da teoria da probabilidade e depois demonstramos o mito da probabilidade de base.

Probabilidade condicional e independência

Freqüentemente, queremos saber a probabilidade condicional de algum evento. O efeito da condição é remover alguns dos resultados do espaço amostral. Por exemplo, qual é a probabilidade de obter uma soma 8 ao jogar dois dados, se sabemos que a face de pelo menos um dado é um número par? Podemos raciocinar da seguinte maneira. Como um dado é par e a soma é par, o segundo dado precisa mostrar um número par. Assim, existem três resultados igualmente prováveis: (2, 6), (4, 4) e (6, 2), dentre um conjunto total de probabilidades de [36 − (número de eventos com os dois dados com faces com valor ímpar)] = 36 − 3 × 3 = 27. A probabilidade resultante é 3/27 = 1/9.

Formalmente, a **probabilidade condicional** de um evento A, supondo que o evento B tenha ocorrido, indicada por $\Pr[A|B]$, é definida como a razão

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]}$$

onde consideramos que Pr[B] não é zero.

Em nosso exemplo, A = {soma igual a 8} e B = {pelo menos um dado par}. A quantidade Pr[AB] abrange todos os resultados em que a soma é 8 e pelo menos um dado é par. Como vimos, existem três desses resultados. Assim, Pr[AB] = 3/36 = 1/12. Um momento de reflexão deverá convencê-lo de que Pr[B]= 3/4. Agora, podemos calcular

$$\Pr[A|B] = \frac{1/12}{3/4} = \frac{1}{9}$$

o que está de acordo com nosso raciocínio anterior.

Dois eventos A e B são chamados de **independentes** se Pr[AB] = Pr[A]Pr[B]. Pode ser facilmente visto que, se A e B são independentes, Pr[$A|B$] = Pr[A] e Pr[$B|A$] = Pr[B].

Teorema de Bayes

Um dos resultados mais importantes da teoria da probabilidade é conhecido como teorema de Bayes. Primeiro, precisamos indicar a fórmula de probabilidade total. Dado um conjunto de eventos mutuamente exclusivos E_1, E_1,...,E_n, tal que a união desses eventos abranja todos os resultados possíveis, e dado um evento qualquer A, então pode ser demonstrado que

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|E_i]\Pr[E_i] \tag{18.1}$$

O teorema de Bayes pode ser enunciado da seguinte forma:

$$\Pr[E_i|A] = \frac{\Pr[A|E_i]\mathrm{P}[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i]\mathrm{P}[E_i]}{\sum_{j=1}^{n} \Pr[A|E_j]\Pr[E_j]} \tag{18.2}$$

A Figura 18.7a ilustra os conceitos da probabilidade total e do teorema de Bayes.

O teorema de Bayes é usado para calcular 'chances posteriores', isto é, a probabilidade de que algo realmente seja o caso, dada a evidência em favor disso. Por exemplo, suponha que estejamos transmitindo uma seqüência de zeros e uns por uma linha de transmissão com ruído. Considere que S0 e S1 sejam os eventos em que 0 é enviado em determinado momento e 1 é enviado, respectivamente, e R0 e R1 sejam os eventos em que 0 é recebido e 1 é recebido. Suponha que saibamos as probabilidades na origem, a saber Pr[S1] = p e Pr[S0] = 1 − p. Agora a linha é observada para se determinar com que freqüência ocorre um erro quando 1 é enviado e quando 0 é enviado, e as probabilidades a seguir são calculadas: Pr[R0|S1] = p_a e Pr[R1|S0] = p_b. Se um 0 for recebido, podemos então calcular a probabilidade condicional de um erro, a saber, a probabilidade condicional de que 1 foi enviado dado que 0 foi recebido, usando o teorema de Bayes:

$$\Pr[\mathrm{S1}|\mathrm{R0}] = \frac{\Pr[\mathrm{R0}|\mathrm{S1}]\Pr[\mathrm{S1}]}{\Pr[\mathrm{R0}|\mathrm{S1}]\Pr[\mathrm{S1}] + \Pr[\mathrm{R0}|\mathrm{S0}]\Pr[\mathrm{S0}]} = \frac{p_a p}{p_a p + (1 - p_b)(1 - p)}$$

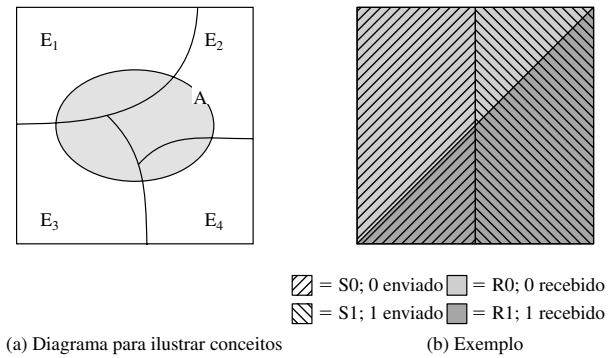


Figura 18.7 Ilustração da probabilidade total e do Teorema de Bayes.

A Figura 18.7b ilustra a equação anterior. Na figura, o espaço amostral é representado por um quadrado unitário. Metade do quadrado corresponde a S0 e metade a S1, de modo que Pr[S0] = Pr[S1] = 0,5. De modo semelhante, metade do quadrado corresponde a R0 e metade a R1, de modo que Pr[R0] = Pr[R1] = 0,5. Dentro da área representando S0, 1/4 dessa área corresponde a R1, de modo que Pr[R1/S0] = 0,25. Outras probabilidades condicionais são semelhantemente evidentes.

Demonstração do mito da probabilidade de base

Considere a situação a seguir. Um paciente faz um exame para alguma doença que retorna positivo (indicando que ele tem a doença). Você sabe que

- A precisão do exame é de 87 por cento (ou seja, se um paciente tem a doença, 87 por cento do tempo, o exame gera o resultado correto, e se o paciente não tem a doença, 87 por cento do tempo, o exame gera o resultado correto).
- A incidência da doença na população é de 1 por cento.

Dado que o teste é positivo, qual é a probabilidade de que o paciente não tenha a doença? Ou seja, qual é a probabilidade de que este seja um alarme falso? Precisamos do teorema de Bayes para obter a resposta correta:

$$\begin{aligned}\text{Pr[bem/positivo]} &= \frac{\text{Pr[positivo/bem]Pr[bem]}}{\text{Pr[positivo/doenca]Pr[doenca]} + \text{Pr[positivo/bem]Pr[bem]}} \\ &= \frac{(0{,}13)(0{,}99)}{(0{,}87)(0{,}01) + (0{,}13)(0{,}99)} = 0{,}937\end{aligned}$$

Assim, na grande maioria dos casos, quando uma condição de doença é detectada, ela é um alarme falso.

O problema, usado em um estudo [PIAT91], foi apresentado a uma série de pessoas. A maior parte delas forneceu a resposta de 13 por cento. A grande maioria, incluindo muitos médicos, forneceu um número abaixo de 50 por cento. Muitos médicos que adivinharam errado lamentaram: "Se você está certo, não há sentido em fazer exames clínicos!" O motivo pelo qual a maioria das pessoas está errada é que elas não levam em consideração a probabilidade de base de incidência (a taxa básica) quando solucionam o problema intuitivamente. Este erro é conhecido como 'mito da probabilidade de base'.

Como esse problema pode ser resolvido? Suponha que pudéssemos levar ambas as taxas de resultado correto para 99,9 por cento. Ou seja, suponha que tenhamos Pr[positivo/doença] = 0,999 e Pr[negativo/bem] = 0,999. Colocando esses números na Equação (18.2), obtemos Pr[bem/positivo] = 0,09. Assim, se pudermos detectar a doença com precisão e detectar a ausência de doença com precisão em um nível de 99,9 por cento, então a taxa de alarmes falsos será de 9 por cento. Isso é muito melhor, mas ainda não é ideal. Mais ainda, considere novamente 99,9 por cento de precisão, mas agora suponha que a incidência da doença na população seja de apenas 1/10000 = 0,0001. Então, acabamos com uma taxa de alarmes falsos de 91 por cento. Em situações reais, [AXEL00] descobriu que as probabilidades associadas aos sistemas de detecção de intrusão foram tais que a taxa de alarmes falsos era insatisfatória.

Software malicioso

19.1 **Vírus e ameaças relacionadas**

19.2 **Contramedidas para vírus**

19.3 **Ataques de negação de serviço distribuídos**

19.4 **Leitura e sites Web recomendados**

19.5 **Principais termos, perguntas de revisão e problemas**

O que é o conceito de defesa: o bloqueio de um golpe. Qual é seu aspecto característico: esperar pelo golpe.

— *Da guerra*, Carl Von Clausewitz

PONTOS PRINCIPAIS

- Software malicioso é o software intencionalmente incluído ou inserido em um sistema para um propósito prejudicial.
- Um vírus é um software que pode 'infectar' outros programas, modificando-os; a modificação inclui uma cópia do programa de vírus, que pode então prosseguir para infectar outros programas.
- Um verme (worm) é um programa que pode se replicar e enviar cópias de um computador para outro através de conexões de rede. Na chegada, o verme pode ser ativado para replicar-se e propagar-se novamente. Além da propagação, o verme normalmente realiza alguma função indesejada.
- Um ataque de negação de serviço (DoS — Denial of Service) é uma tentativa de impedir que usuários legítimos de um serviço usem esse serviço.
- Um ataque de negação de serviço distribuído é iniciado a partir de várias origens coordenadas.

Este capítulo examina o software malicioso (malware), especialmente vírus e vermes.

19.1 VÍRUS E AMEAÇAS RELACIONADAS

Talvez os tipos mais sofisticados de ameaças a sistemas de computador são apresentados por programas que exploram vulnerabilidades. Nesse contexto, estamos preocupados com os programas de aplicação, além de programas utilitários, como editores e compiladores.

Começamos esta seção com uma visão geral do espectro dessas ameaças de software. O restante da seção é dedicado a vírus e vermes.

Programas maliciosos

A terminologia nesta área apresenta problemas devido a uma falta de acordo universal sobre todos os termos e porque algumas das categorias se sobrepõem. A Tabela 19.1, baseada principalmente em [SZOR05], é um guia útil.

O software malicioso pode ser dividido em duas categorias: aqueles que precisam de um programa hospedeiro e aqueles que são independentes. Os primeiros são basicamente fragmentos de programas que não podem existir independentemente de algum programa de aplicação, utilitário ou programa do sistema. Vírus, bombas lógicas e backdoors são alguns exemplos. Os últimos são programas independentes que podem ser programados e executados pelo sistema operacional. Vermes e zumbis são alguns exemplos.

Também podemos fazer uma diferenciação entre aquelas ameaças de software que não se replicam e aquelas que se replicam. As primeiras são programas ou fragmentos de programas ativados por um disparador (trigger). Alguns exemplos são bombas lógicas, backdoors e zumbis. As últimas consistem em um fragmento de programa ou um programa independente que, quando executado, pode produzir uma ou mais cópias de si mesmo para serem ativadas mais tarde no mesmo sistema ou em algum outro sistema. Vírus e vermes são alguns exemplos.

No restante desta subseção, analisamos rapidamente algumas das principais categorias de software malicioso, com exceção de vírus e vermes, que são explicados com mais detalhes em outra parte desta seção.

Tabela 19.1 Terminologia dos programas maliciosos.

Nome	Descrição
Vírus	Anexa-se a um programa e propaga cópias de si mesmo a outros programas
Verme	Programa que propaga cópias de si mesmo a outros computadores
Bomba lógica	Dispara uma ação quando ocorre uma determinada condição
Cavalo-de-tróia	Programa que contém funcionalidade adicional inesperada
Backdoor (trapdoor)	Modificação de programa que permite o acesso não autorizado à funcionalidade
Exploit	Código específico para uma única vulnerabilidade ou conjunto de vulnerabilidades
Downloaders	Programa que instala outros itens em uma máquina sob ataque. Normalmente, um downloader é enviado em um e-mail
Auto-rooter	Ferramentas maliciosas de hacker usadas para invasão de novas máquinas remotamente
Kit (gerador de vírus)	Conjunto de ferramentas para gerar novos vírus automaticamente
Programas de spam	Usados para enviar grandes volumes de e-mail indesejado
Flooders	Usados para atacar sistemas de computador em rede com um grande volume de tráfego para executar um ataque de negação de serviço (DoS)
Keyloggers	Capta teclas digitadas em um sistema comprometido
Rootkit	Conjunto de ferramentas de hacker usadas após um atacante ter invadido um sistema de computador e obtido acesso em nível de root
Zumbi	Programa ativado em uma máquina infectada, que é preparado para desferir ataques a outras máquinas

Backdoor Uma backdoor (porta dos fundos), também conhecida como trapdoor, é um ponto de entrada secreto para um programa, que permite que alguém ciente da backdoor obtenha acesso sem passar pelos procedimentos normais de acesso de segurança. Os programadores têm usado backdoors legitimamente por muitos anos para depurar e testar programas. Isso normalmente é feito quando um programador está desenvolvendo uma aplicação que possui um procedimento de autenticação, ou uma configuração longa, exigindo que o usuário entre com muitos valores diferentes para executar a aplicação. Para depurar o programa, o desenvolvedor pode querer obter privilégios especiais ou evitar toda a preparação e autenticação necessária. O programador também pode querer garantir que haja um método de ativação do programa caso algo saia errado com o procedimento de autenticação que está sendo criado na aplicação. A backdoor é o código que reconhece alguma seqüência especial de entrada ou é disparado por ser executado a partir de uma certa ID de usuário ou por uma seqüência improvável de eventos.

As backdoors tornam-se ameaças quando programadores inescrupulosos as utilizam para obter acesso não autorizado. A backdoor foi a idéia básica para a vulnerabilidade retratada no filme *War games* (jogos de guerra). Outro exemplo é que, durante o desenvolvimento do Multics, testes de penetração foram realizados por uma 'equipe tigre' (tiger team) da força aérea norte-americana (simulando adversários). Uma tática empregada foi enviar uma atualização falsa de sistema operacional para um local executando o Multics. A atualização continha um cavalo-de-tróia (descrito mais adiante) que podia ser ativado por uma backdoor e que permitia que a equipe tigre obtivesse acesso. A ameaça foi tão bem implementada que os desenvolvedores do Multics não puderam descobri-la, mesmo depois de informados de sua presença [ENGE80].

É difícil implementar controles do sistema operacional para backdoors. As medidas de segurança precisam focalizar atividades de desenvolvimento de programa e atualização do software.

Bomba lógica Um dos tipos mais antigos de ameaça, criado antes mesmo de vírus e vermes, é a bomba lógica. A bomba lógica é o código embutido em algum programa legítimo configurado para 'explodir' quando certas condições forem satisfeitas. Alguns exemplos de condições que podem ser usadas como disparadores de uma bomba lógica são a presença ou ausência de certos arquivos, um dia da semana ou data em particular, ou determinado usuário executando a aplicação. Uma vez disparada, uma bomba pode alterar ou excluir dados ou arquivos inteiros, causar uma parada da máquina ou causar algum outro dano. Um exemplo marcante de como as bombas lógicas podem ser empregadas foi o caso de Tim Lloyd, sentenciado por preparar uma bomba lógica que custou ao seu empregador, a Omega Engineering, mais de US$ 10 milhões, desmontou sua estratégia de crescimento corporativo e por fim levou à demissão de 80 trabalhadores [GAUD00]. Lloyd foi condenado a 41 meses de prisão e a pagar US$ 2 milhões de restituição.

Cavalos-de-tróia Um cavalo-de-tróia é um programa ou procedimento de comando útil, ou aparentemente útil, contendo código oculto que, quando invocado, realiza alguma função indesejada ou prejudicial.

Os cavalos-de-tróia podem ser usados para executar funções indiretamente, que um usuário não autorizado não poderia executar diretamente. Por exemplo, para obter acesso aos arquivos de outro usuário em um sistema compartilhado, um usuário pode criar um cavalo-de-tróia que, quando executado, mude as permissões de arquivo do usuário, permitindo que os arquivos sejam lidos por qualquer usuário. O autor pode, então, induzir os usuários a executar o programa colocando-o em um diretório comum e nomeando-o de modo que pareça ser um utilitário. Um exemplo é um programa que produz aparentemente uma listagem dos arquivos do usuário em um formato desejável. Depois que outro usuário tiver executado o programa, o autor pode então acessar as informações nos arquivos do usuário. Um exemplo de cavalo-de-tróia que seria difícil de detectar é um compilador modificado para inserir código adicional em certos programas enquanto são compilados, como um programa de login do sistema [THOM84]. O código cria uma backdoor no programa de login, que permite ao autor efetuar o logon no sistema usando uma senha especial. Esse cavalo-de-tróia pode nunca ser descoberto lendo-se o código-fonte do programa de login.

Outra motivação comum para o cavalo-de-tróia é a destruição de dados. O programa parece estar realizando uma função útil (por exemplo, um programa de calculadora), mas também pode estar excluindo silenciosamente os arquivos do usuário. Por exemplo, um executivo da CBS foi vítima de um cavalo-de-tróia que destruiu todas as informações contidas na memória do seu computador [TIME90]. O cavalo-de-tróia foi implantado em uma rotina gráfica oferecida em uma BBS (Bulletin Board System).

Zumbi Um zumbi é um programa que controla secretamente outro computador conectado à Internet e depois usa esse computador para disparar ataques difíceis de rastrear até o criador do zumbi. Os zumbis são usados em ataques de negação de serviço, tipicamente contra sites Web . O zumbi é introduzido em centenas de computadores pertencentes a terceiros não suspeitos, e depois usados para inundar o site Web disparando um ataque esmagador de tráfego na Internet. A Seção 19.3 explica os zumbis no contexto dos ataques de negação de serviço.

A natureza dos vírus

Um vírus é um pedaço de software que pode 'infectar' outros programas modificando-os; a modificação inclui uma cópia do programa de vírus, a qual pode então prosseguir para infectar outros programas.

Os vírus biológicos são pequenos fragmentos de código genético – DNA ou RNA – que podem assumir o controle do mecanismo de uma célula viva e enganá-la para que faça milhares de réplicas sem falhas do vírus original. Assim como seu equivalente biológico, um vírus de computador carrega em seu código de instruções a receita para fazer cópias perfeitas de si mesmo. O vírus típico é embutido em um programa em um computador. Depois, sempre que o computador infectado entra em contato com um software não infectado, uma nova cópia do vírus passa para o novo programa. Assim, a infecção pode se espalhar de um computador para outro por usuários não suspeitos, que trocam discos ou enviam programas um para o outro por uma rede. Em um ambiente de rede, a capacidade de acessar aplicações e serviços do sistema em outros computadores oferece uma cultura perfeita para que um vírus se espalhe.

Um vírus pode fazer qualquer coisa que outros programas fazem. A única diferença é que ele se conecta a outro programa e é executado secretamente quando o programa hospedeiro é executado. Quando um vírus está sendo executado, ele pode realizar qualquer função, como apagar arquivos e programas.

Durante seu tempo de vida, um vírus típico passa pelas quatro fases a seguir:

- **Fase latente:** O vírus está inativo. O vírus será ativado por algum evento, como uma data, a presença de outro programa ou arquivo, ou a capacidade do disco de exceder algum limite. Nem todos os vírus têm esse estágio.
- **Fase de propagação:** O vírus coloca uma cópia idêntica de si mesmo em outros programas ou em certas áreas do sistema no disco. Cada programa infectado agora terá um clone do vírus que, por si só, entrará em uma fase de propagação.
- **Fase de disparo:** O vírus é ativado para realizar a função para a qual ele foi planejado. Assim como na fase latente, a fase de disparo pode ser causada por diversos eventos do sistema, incluindo a contagem do número de vezes que essa cópia do vírus fez cópias de si mesma.
- **Fase de execução:** A função é realizada. A função pode ser inofensiva, como uma mensagem na tela, ou danosa, como a destruição de programas e arquivos de dados.

A maioria dos vírus executa seu trabalho de maneira específica a determinado sistema operacional e, em alguns casos, específica a determinada plataforma de hardware. Assim, eles são criados para tirar proveito dos detalhes e pontos fracos dos sistemas em particular.

Estrutura do vírus Um vírus pode ser inserido antes ou após um programa executável, ou então pode ser embutido de alguma outra maneira. A chave para a sua operação é que o programa infectado, quando chamado, primeiro executará o código do vírus e depois executará o código original do programa.

Uma representação muito geral da estrutura do vírus aparece na Figura 19.1 baseada em [COHE94]. Nesse caso, o código do vírus, V, é anexado antes do código dos programas infectados, e presume-se que o ponto de entrada para o programa, quando chamado, é a primeira linha do programa.

Um programa infectado é iniciado com o código do vírus e trabalha da seguinte maneira. A primeira linha de código é um salto para o programa principal do vírus. A segunda linha é um marcador especial usado pelo vírus para determinar se um programa vítima em potencial já foi infectado ou não com esse vírus. Quando o programa

```
      program V :=

{goto main;
      1234567;

      subroutine infect-executable :=
            {loop:
            file := get-random-executable-file;
            if (first-line-of-file = 1234567)
               then goto loop
               else prepend V to file; }

      subroutine do-damage :=
            {whatever damage is to be done}

      subroutine trigger-pulled :=
            {return true if some condition holds}

main:     main-program :=
            {infect-executable;
            if trigger-pulled then do-damage;
            goto next;}
next:

}
```

Figura 19.1 Um vírus simples.

é chamado, o controle é transferido imediatamente para o programa principal do vírus. O programa do vírus primeiro procura arquivos executáveis não infectados e os infecta. Em seguida, o vírus pode realizar alguma ação, normalmente prejudicial ao sistema. Essa ação pode ser realizada toda vez que o programa for chamado, ou então pode ser uma bomba lógica que dispara apenas sob certas condições. Finalmente, o vírus transfere o controle ao programa original. Se a fase de infecção do programa for razoavelmente rápida, um usuário provavelmente não notará qualquer diferença entre a execução de um programa infectado e não infectado.

Um vírus como o que acabamos de descrever é facilmente detectado porque uma versão infectada de um programa é maior que a versão não infectada correspondente. Uma maneira de impedir esse modo simples de detectar um vírus é compactar o arquivo executável de modo que as versões infectadas e não infectadas tenham tamanhos idênticos. A Figura 19.2 [COHE94] mostra, em termos gerais, a lógica exigida. As linhas principais nesse vírus são numeradas, e a Figura 19.3 [COHE94] ilustra a operação. Consideramos que o programa P_1 está infectado com o vírus CV (Compression Virus). Quando esse programa é chamado, o controle passa para o vírus, que realiza as seguintes etapas:

1. Para cada arquivo não infectado P_2 encontrado, o vírus primeiro compacta esse arquivo para produzir P'_2, que é mais curto que o programa original pelo tamanho do vírus.
2. Uma cópia do vírus é inserida no início do programa compactado.
3. A versão compactada do programa infectado original, P'_1, é descompactada.
4. O programa original não compactado é executado.

Nesse exemplo, o vírus não faz nada além de se propagar. Como no exemplo anterior, o vírus pode incluir uma bomba lógica.

Infecção inicial Quando um vírus obtém acesso a um sistema, infectando um único programa, ele está em uma posição de infectar alguns ou todos os outros arquivos executáveis nesse sistema quando o programa infectado for executado. Assim, a infecção viral pode ser completamente impedida evitando-se que o vírus obtenha acesso ao sistema. Infelizmente, a prevenção é extraordinariamente difícil porque um vírus pode ser parte de qualquer programa fora de um sistema. Assim, a menos que uma pessoa esteja disposta a tomar uma atitude totalmente radical e escrever seus próprios programas de sistema e aplicação, essa pessoa está vulnerável.

```
     program CV :=

{goto main;
     01234567;

     subroutine infect-executable :=
              {loop:
                   file := get-random-executable-file;
              if (first-line-of-file = 01234567) then goto loop;
         (1)  compress file;
         (2)  prepend CV to file;
         }

main:    main-program :=
              {if ask-permission then infect-executable;
         (3)  uncompress rest-of-file;
         (4)  run uncompressed file;}
         }
```

Figura 19.2 Lógica para um vírus de compactação.

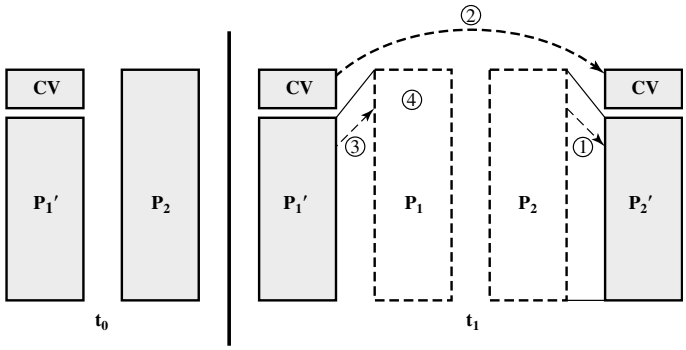


Figura 19.3 Um vírus de compactação.

Tipos de vírus

Há uma disputa contínua entre escritores de vírus e escritores de software antivírus desde que os vírus apareceram pela primeira vez. Conforme contramedidas eficazes são desenvolvidas para os tipos de vírus existentes, novos tipos são desenvolvidos. [STEP93] sugere as seguintes categorias como estando entre os tipos de vírus mais significativos:

- **Vírus parasitário:** A forma de vírus tradicional e ainda mais comum. Um vírus parasitário se conecta a arquivos executáveis e se replica quando o programa infectado é executado, localizando outros arquivos executáveis para infectar.
- **Vírus residente na memória:** Aloja-se na memória principal como parte de um programa residente no sistema. Desse ponto em diante, o vírus infecta todo programa executado.
- **Vírus do setor de inicialização (boot):** Infecta um registro-mestre de inicialização ou registro de inicialização e se espalha quando um sistema é inicializado a partir do disco contendo o vírus.
- **Vírus furtivo (stealth virus):** Uma forma de vírus projetado explicitamente para se esconder da detecção pelo software antivírus.
- **Vírus polimórfico:** Um vírus que se transforma a cada infecção, impossibilitando a detecção pela 'assinatura' do vírus.
- **Vírus metamórfico:** Assim como um vírus polimórfico, um vírus metamórfico muda a cada infecção. A diferença é que um vírus metamórfico se reescreve completamente a cada iteração, aumentando a dificuldade de detecção. Os vírus metamórficos podem mudar seu comportamento e também sua aparência.

Um exemplo de um **vírus furtivo** já foi explicado anteriormente: um vírus que usa compactação para que o programa infectado tenha exatamente o mesmo tamanho de uma versão não infectada. Técnicas muito mais sofisticadas são possíveis. Por exemplo, um vírus pode colocar uma lógica de interceptação em rotinas de E/S de disco, de modo que, quando houver uma tentativa de leitura de partes suspeitas do disco usando essas rotinas, o vírus apresente de volta o programa original, não infectado. Assim, *furtivo* não é um termo que se aplica a um vírus desse tipo, mas sim uma técnica usada por um vírus para evitar a detecção.

Um **vírus polimórfico** cria cópias durante a repetição funcionalmente equivalentes, mas que possuem padrões de bits distintamente diferentes. Assim como um vírus furtivo, a finalidade é enganar programas que procuram vírus. Nesse caso, a 'assinatura' do vírus variará a cada cópia. Para conseguir essa variação, o vírus pode inserir aleatoriamente instruções supérfluas ou trocar a ordem de instruções independentes. Uma técnica mais eficaz é usar a criptografia. Uma parte do vírus, geralmente chamada *mecanismo de mutação*, cria uma chave de criptografia aleatória para criptografar o restante do vírus. A chave é armazenada com o vírus, e o próprio mecanismo de mutação é alterado. Quando um programa infectado é chamado, o vírus usa a chave aleatória armazenada para decriptografar o vírus. Quando o vírus se replica, uma chave aleatória diferente é selecionada.

Outra arma no arsenal dos escritores de vírus é o kit de ferramentas de criação de vírus. Esse kit de ferramentas permite que alguém com relativamente pouca experiência crie rapidamente diversos vírus diferentes. Embora os vírus criados com os kits de ferramentas costumem ser menos sofisticados que os vírus projetados do zero, o grande número de novos vírus que podem ser gerados cria um problema para os esquemas antivírus.

Vírus de macro

Em meados da década de 1990, os vírus de macro se tornaram o tipo predominante de vírus. Os vírus de macro são particularmente ameaçadores por diversos motivos:

1. Um vírus de macro é independente de plataforma. Praticamente todos os vírus de macro infectam documentos do Microsoft Word. Qualquer plataforma de hardware e sistema operacional que aceite o Word pode ser infectados.
2. Os vírus de macro infectam documentos, e não partes executáveis do código. A maior parte das informações introduzidas em um sistema de computador está na forma de um documento, em vez de um programa.
3. Os vírus de macro são facilmente espalhados. Um método muito comum é por e-mail.

Os vírus de macro tiram proveito de um recurso encontrado no Word e em outras aplicações do Office, como o Microsoft Excel, a saber, a macro. Geralmente, uma macro é um programa executável embutido em um documento de processamento de textos ou outro tipo de arquivo. Normalmente, os usuários empregam macros para automatizar tarefas repetitivas e, desse modo, economizar digitação. A linguagem de macro normalmente é alguma forma de linguagem de programação Basic. Um usuário pode definir uma seqüência de digitação em uma macro e prepará-la para que esta seja chamada quando uma tecla de função ou combinação curta especial de teclas for utilizada.

Versões sucessivas do Word vêm oferencendo proteção contra vírus de macro. Por exemplo, a Microsoft oferece uma ferramenta de proteção de vírus de macro opcional que detecta arquivos do Word suspeitos e alerta o cliente para o risco em potencial de abrir um arquivo com macros. Vários fornecedores de produtos antivírus também desen-

volveram ferramentas para detectar e corrigir vírus de macro. Assim como em outros tipos de vírus, a corrida continua no campo de vírus de macro, mas eles não são mais as ameaças de vírus predominantes.

Vírus de e-mail

Um desenvolvimento mais recente em software malicioso é o vírus de e-mail. Os primeiros vírus de e-mail que se espalharam rapidamente, como o Melissa, utilizavam uma macro do Microsoft Word embutida em um anexo. Se o destinatário abrisse o anexo do e-mail, a macro do Word seria ativada. Então:

1. O próprio vírus do e-mail envia a si mesmo para todos na lista de correspondência do programa de e-mail do usuário.
2. O vírus gera danos locais.

No final de 1999, apareceu uma versão mais poderosa do vírus de e-mail. Essa versão mais nova pode ser ativada meramente abrindo-se um e-mail que contém o vírus, em vez de abrir um anexo. O vírus usa um script na linguagem Visual Basic aceita pelo programa de e-mail.

Assim, vemos uma nova geração de malware que chega por e-mail e utiliza recursos do software de e-mail para se replicar pela Internet. O vírus se propaga assim que ativado (seja pela abertura de um anexo de e-mail ou do e-mail) para todos os endereços de e-mail conhecidos do host infectado. Como resultado, enquanto os vírus costumavam levar meses ou anos para se propagar, estes agora fazem isso em horas. Isso torna muito difícil para o software antivírus responder antes que um grande dano seja causado. Finalmente, um maior grau de segurança precisa estar embutido no software utilitário da Internet e de aplicação em PCs para responder à crescente ameaça.

Vermes

Um verme (worm) é um programa que pode se replicar e enviar cópias de um computador para outro através de conexões de rede. Na chegada, o verme pode ser ativado para replicar-se e propagar-se novamente. Além da propagação, o verme normalmente realiza alguma função indesejada. Um vírus de e-mail tem algumas das características de um verme, pois se propaga de um sistema para outro. Porém, ainda podemos classificá-lo como um vírus, pois exige um humano para movê-lo adiante. Um verme busca ativamente mais máquinas para infectar e cada máquina infectada serve como uma plataforma de lançamento automatizada para ataques em outras máquinas.

Os vermes de rede utilizam conexões de rede para se espalharem de um sistema para outro. Uma vez ativo dentro de um sistema, um verme de rede pode se comportar como um vírus de computador ou bactéria, ou pode implantar cavalos-de-tróia ou realizar um número qualquer de ações destrutivas ou incômodas.

Para replicar-se, um verme de rede utiliza algum tipo de veículo de rede. Alguns exemplos são os seguintes:

- **Recursos de e-mail:** Um verme envia uma cópia de si mesmo para outros sistemas.
- **Capacidade de execução remota:** Um verme executa uma cópia de si mesmo em outro sistema.
- **Capacidade de login remoto:** Um verme se conecta a um sistema remoto como um usuário e depois utiliza comandos para copiar a si mesmo de um sistema para outro.

A nova cópia do verme é, então, executada no sistema remoto onde, além de quaisquer funções que ele realize nesse sistema, continua a se espalhar da mesma maneira.

Um verme de rede exibe as mesmas características de um vírus de computador: uma fase latente, uma fase de propagação, uma fase de disparo e uma fase de execução. A fase de propagação geralmente realiza as seguintes funções:

1. Procurar outros sistemas para infectar, examinando tabelas de host ou repositórios semelhantes de endereços de sistema remoto.
2. Estabelecer uma conexão com um sistema remoto.
3. Copiar-se para o sistema remoto e fazer com que a cópia seja executada.

O verme de rede também pode tentar determinar se um sistema foi previamente infectado antes de copiar a si mesmo para o sistema. Em um sistema com multiprocessamento, ele também pode disfarçar sua presença nomeando-se como um processo do sistema ou usando algum outro nome que pode não ser notado por um operador do sistema.

Assim como os vírus, os vermes de rede são difíceis de combater.

O verme Morris Até a geração atual dos vermes, o mais conhecido foi o verme lançado pela Internet por Robert Morris em 1988. O verme Morris foi projetado para se espalhar em sistemas UNIX e usava diversas técnicas diferentes para propagação. Quando uma cópia iniciava a execução, sua primeira tarefa era descobrir outros hosts conhecidos que permitissem a entrada a partir desse host. O verme realizava sua tarefa examinando diversas listas e tabelas, incluindo tabelas do sistema que declaravam quais outras máquinas eram confiáveis por esse host, arquivos de encaminhamento de correio dos usuários, tabelas pelas quais os usuários concediam a eles mesmos permissão para aces-

sar contas remotas, e a partir de um programa que informava o status das conexões da rede. Para cada host descoberto, o verme tentava diversos métodos para obter acesso.

1. Ele tentava efetuar o login em um host remoto como um usuário legítimo. Nesse método, o verme primeiro tentava quebrar o arquivo de senha local e depois usava as senhas descobertas e as IDs de usuário correspondentes. A suposição era de que muitos usuários usariam a mesma senha em diferentes sistemas. Para obter as senhas, o verme executava um programa para quebra de senhas que tentava
   (a) o nome da conta de cada usuário e permutações simples dele
   (b) uma lista de 432 senhas embutidas que o Morris achava que seriam candidatas prováveis
   (c) todas as palavras no dicionário do sistema local
2. Ele explorava um erro no protocolo finger, que informa o paradeiro de um usuário remoto.
3. Ele explorava uma trapdoor na opção de depuração do processo remoto que recebe e envia mensagens.

Se qualquer um desses ataques obtivesse sucesso, o verme conseguia a comunicação com o interpretador de comandos do sistema operacional. Depois, ele enviava a esse interpretador um pequeno programa autocarregador (bootstrap), emitia um comando para executar esse programa e depois fazia o logoff. O programa bootstrap, então, chamava de volta o programa pai (de origem) e baixava o restante do verme. O novo verme era então executado.

Ataques de vermes recentes A era contemporânea de ameaças de verme começaram com o lançamento do verme Code Red em julho de 2001. O Code Red tirou proveito de uma brecha de segurança no Microsoft Internet Information Server (IIS) para penetrar e se espalhar. Ele também desativou o verificador de arquivos do sistema no Windows. O verme sondou endereços de IP aleatórios para se espalhar para outros hosts. Durante um certo período de tempo, ele só se espalhou. Depois, ele iniciou um ataque de negação de serviço a um site Web do governo dos EUA, inundando o site com pacotes de diversos hosts. O verme, então, suspendeu as atividades e foi reativado periodicamente. Na segunda onda de ataque, o Code Red infectou quase 360 mil servidores em 14 horas. Além da devastação que ele causa no servidor, o Code Red pode consumir grandes quantidades de recursos da Internet, interrompendo o serviço.

O Code Red II é uma variante que tem como alvo os Microsoft IISs. Além disso, esse verme mais recente instala uma backdoor, permitindo que um hacker direcione suas atividades.

No final de 2001, apareceu um verme mais versátil, conhecido como Nimda. O Nimda se espalhava por vários mecanismos:

- de cliente para cliente via e-mail
- de cliente para cliente via compartilhamentos abertos de rede
- de servidor Web para cliente via navegação por sites Web comprometidos
- de cliente para servidor Web via varredura ativa e exploração de diversas vulnerabilidades de diretório transponíveis do Microsoft IIS 4.0/5.0
- de cliente para servidor Web via varredura das backdoors que deixadas pelos vermes Code Red II

O verme modifica documentos da Web (por exemplo, arquivos .htm, .html e .asp) e certos arquivos executáveis nos sistemas infectados e cria diversas cópias de si mesmo sob vários nomes de arquivo.

No início de 2003, apareceu o verme SQL Slammer. Esse verme explorava uma vulnerabilidade de estouro de buffer no Microsoft SQL Server. O Slammer era extremamente compacto e se espalhou rapidamente, infectando 90 por cento dos hosts vulneráveis dentro de 10 minutos. O final de 2003 viu a chegada do verme Sobig.f, que explorava servidores proxy abertos (open proxy) para transformar as máquinas infectadas em mecanismos de spam. Em seu pico, o Sobig.f foi responsável por uma em cada 17 mensagens e produziu mais de um milhão de cópias de si mesmo dentro das primeiras 24 horas.

Mydoom é um verme de remessa de e-mail em massa que apareceu em 2004. Ele seguia uma tendência crescente de instalar uma backdoor nos computadores infectados, permitindo assim que hackers obtivessem acesso remoto a dados como senhas e números de cartão de crédito. Mydoom se copiava até mil vezes por minuto e, conforme notícias, inundou a Internet com 100 milhões de mensagens infectadas em 36 horas.

Estado da tecnologia de verme

A última palavra na tecnologia de vermes inclui o seguinte:

- **Multiplataforma:** Os vermes mais recentes não estão limitados a máquinas com Windows, mas podem atacar diversas plataformas, especialmente as variedades populares do UNIX.
- **Multiexploração (Multiexploit):** Novos vermes penetram nos sistemas de diversas maneiras, usando exploits contra servidores Web, navegadores, e-mail, compartilhamento de arquivos e outras aplicações baseadas em rede.
- **Espalhamento ultra-rápido:** Uma técnica para acelerar o espalhamento de um verme é realizar uma varredura prévia na Internet para acumular endereços na Internet de máquinas vulneráveis.

- **Polimórfico:** Para evitar a detecção, escapar de filtros e driblar a análise em tempo real, os vermes adotam a técnica polimórfica dos vírus. Cada cópia do verme tem código novo, gerado instantaneamente, usando instruções funcionalmente equivalentes e técnicas de criptografia.
- **Metamórfico:** Além de mudar sua aparência, os vermes metamórficos possuem um repertório de padrões de comportamento desencadeados em diferentes estágios de propagação.
- **Veículos de transporte:** Como os vermes podem comprometer rapidamente uma grande quantidade de sistemas, eles são ideais para espalhar outras ferramentas de ataque distribuído, como zumbis de negação distribuída de serviço .
- **Exploração do dia zero (zero-day exploit):** Para conseguir o máximo de surpresa e distribuição, um verme deve explorar uma vulnerabilidade desconhecida que só é descoberta pela comunidade de rede quando o verme é desferido.

19.2 CONTRAMEDIDAS PARA VÍRUS

Técnicas antivírus

A solução ideal para a ameaça de vírus é a prevenção: Em primeiro lugar não permitir que um vírus entre no sistema. Esse objetivo, em geral, é impossível de se conseguir, embora a prevenção possa reduzir o número de ataques virais bem-sucedidos. A segunda melhor técnica é ser capaz de fazer o seguinte:

- Detecção: Uma vez que a infecção tenha ocorrido, localize o vírus.
- Identificação: Uma vez que a detecção tenha sido completada, identifique o vírus específico que infectou um programa.
- Remoção: Uma vez que o vírus específico tenha sido identificado, remova todos os traços do vírus do programa infectado e restaure-o ao seu estado original. Remova o vírus de todos os sistemas infectados, para que a infecção não se espalhe mais.

Se a detecção tiver sucesso, mas a identificação ou a remoção não for possível, então a alternativa será descartar o programa infectado e recarregar uma versão de backup limpa.

Os avanços na tecnologia de vírus e antivírus seguem lado a lado. Os primeiros vírus eram fragmentos de código relativamente simples, e podiam ser identificados e eliminados com pacotes de software antivírus relativamente simples. À medida que a corrida do vírus evoluiu, tanto os vírus quanto, necessariamente, o software antivírus, se tornaram mais complexos e sofisticados.

[STEP93] identifica quatro gerações de software antivírus:

- Primeira geração: scanner simples
- Segunda geração: scanner heurístico
- Terceira geração: interceptações de atividade
- Quarta geração: proteção completa

Um scanner de **primeira geração** exige que uma assinatura identifique um vírus. O vírus pode conter 'curingas', mas possui essencialmente a mesma estrutura e padrão de bits em todas as cópias. Esses scanners específicos de assinatura são limitados à detecção de vírus conhecidos. Outro tipo de scanner de primeira geração mantém um registro do tamanho dos programas e procura alterações no tamanho.

Um scanner de **segunda geração** não depende de uma assinatura específica. Em vez disso, o scanner usa regras heurísticas para procurar uma provável infecção de vírus. Uma classe desses scanners procura fragmentos de código que freqüentemente estão associados a vírus. Por exemplo, ele pode procurar o início de um loop de criptografia usado em um vírus polimórfico e descobrir a chave de criptografia. Quando a chave é descoberta, o scanner pode decriptografar o vírus para identificá-lo, depois remover a infecção e retornar o programa à operação normal.

Outra técnica de segunda geração é a verificação de integridade. Uma soma de verificação (checksum) pode ser anexada a cada programa. Se um vírus infectar o programa sem mudar a soma de verificação, então uma verificação de integridade detectará a mudança. Para enfrentar um vírus sofisticado o suficiente para alterar a soma de verificação ao infectar um programa, uma função de hash criptografada poderá ser usada. A chave de criptografia é armazenada separadamente do programa, para que o vírus não possa gerar um novo código de hash e criptografá-lo. Usando uma função de hash em vez de uma soma de verificação mais simples, o vírus é impedido de ajustar o programa para produzir o mesmo código de hash anterior.

Programas de **terceira geração** são programas residentes na memória, que identificam um vírus por suas ações, em vez de sua estrutura em um programa infectado. Esses programas têm a vantagem de que não é necessário desenvolver assinaturas e heurísticas para uma ampla variedade de vírus. Em vez disso, é necessário apenas identificar um pequeno conjunto de ações que indicam que uma infecção está sendo tentada e, então, intervir.

Produtos de **quarta geração** são pacotes compostos por uma série de técnicas antivírus usadas em conjunto. Estas incluem componentes de varredura e de interceptação de atividades. Além disso, esse tipo de pacote inclui recurso de

controle de acesso, que limita a capacidade dos vírus de penetrar em um sistema e, por conseqüência, limita a capacidade de um vírus de atualizar arquivos a fim de passar a infecção adiante.

A corrida continua. Com os pacotes de quarta geração, um estratégia de defesa mais abrangente é empregada, ampliando o escopo da defesa para medidas de segurança de computador de uso mais geral.

Técnicas antivírus avançadas

Técnicas e produtos antivírus mais sofisticados continuam a aparecer. Nesta subseção, destacamos duas das técnicas mais importantes.

Decriptografia genérica A tecnologia de decriptografia genérica (GD) permite que o programa antivírus detecte facilmente até mesmo os vírus polimórficos mais complexos, enquanto mantém velocidades rápidas de varredura [NACH97]. Lembre-se de que, quando um arquivo contendo um vírus polimórfico é executado, o vírus precisa se decriptografar para ser ativado. Para detectar essa estrutura, os arquivos executáveis são executados por meio de um scanner de GD, que contém os seguintes elementos:

- Emulador de CPU: Um computador virtual baseado em software. As instruções em um arquivo executável são interpretadas pelo emulador, em vez de executadas no processador principal. O emulador inclui versões de software de todos os registradores e outro hardware de processador, de modo que o processador principal não seja afetado por programas interpretados no emulador.
- Scanner de assinatura de vírus: Um módulo que varre o código de destino procurando assinaturas de vírus conhecidas.
- Módulo de controle de emulação: Controla a execução do código de destino.

No início de cada emulação, o emulador começa interpretando instruções no código de destino, uma de cada vez. Assim, se o código inclui uma rotina de decriptografia que decriptografa e, portanto, expõe o vírus, esse código é interpretado. Na verdade, o vírus realiza o trabalho para o programa antivírus, expondo o vírus. Periodicamente, o módulo de controle interrompe a interpretação para varrer o código de destino em busca de assinaturas de vírus.

Durante a interpretação, o código de destino pode não causar danos ao ambiente do computador pessoal real, pois está sendo interpretado em um ambiente completamente controlado.

O problema de projeto mais difícil em um scanner de GD é determinar por quanto tempo executar cada interpretação. Geralmente, os elementos de vírus são ativados logo depois que um programa começa a ser executado, mas isso não precisa acontecer necessariamente. Quanto mais o scanner emular um programa em particular, mais provável será que ele detecte quaisquer vírus ocultos. Porém, o programa antivírus só pode usar uma quantidade limitada de tempo e recursos antes que os usuários reclamem.

Sistema digital imune O sistema digital imune é uma abordagem abrangente para proteção contra vírus, desenvolvida pela IBM [KEPH97a, KEPH97b]. A motivação para esse desenvolvimento tem sido a ameaça cada vez maior da propagação de vírus baseada na Internet. Primeiramente, vamos falar um pouco sobre essa ameaça e depois resumiremos a técnica da IBM.

Tradicionalmente, a ameaça de vírus foi caracterizada pelo espalhamento relativamente lento de novos vírus e novas mutações. Geralmente, o software antivírus era atualizado mensalmente, e isso era suficiente para controlar o problema. Também tradicionalmente, a Internet desempenhou um papel comparativamente pequeno no espalhamento dos vírus. Mas, como [CHES97] indica, duas tendências importantes na tecnologia da Internet tiveram um impacto crescente na velocidade da propagação de vírus nos últimos anos:

- Sistemas de correio integrados: Sistemas como Lotus Notes e Microsoft Outlook tornam muito simples enviar qualquer coisa a qualquer pessoa e trabalhar com objetos recebidos.
- Sistemas de programa móvel: Capacidades como Java e ActiveX permitem que programas se movam por conta própria de um sistema para outro.

Em resposta à ameaça imposta por essas capacidades baseadas na Internet, a IBM desenvolveu um protótipo de sistema digital imune. Esse sistema expande o uso da emulação de programa explicada na subseção anterior e oferece um sistema de emulação e de detecção de vírus de uso geral. O objetivo desse sistema é oferecer um tempo de resposta rápido para que os vírus possam ser aniquilados quase na mesma hora em que são introduzidos. Quando um novo vírus entra em uma organização, o sistema imune automaticamente o capta, analisa e acrescenta detecção e proteção contra ele, remove-o e passa informações sobre esse vírus aos sistemas executando o IBM AntiVirus, para que possa ser detectado antes que tenha permissão para ser executado em outro lugar.

A Figura 19.4 ilustra as etapas típicas na operação do sistema digital imune:

1. Um programa de monitoração em cada PC usa uma variedade de heurísticas baseadas no comportamento do sistema, mudanças suspeitas nos programas, ou família de assinaturas para deduzir que um vírus pode estar presente. O programa de monitoração encaminha uma cópia de qualquer programa considerado infectado para uma máquina administrativa dentro da organização.

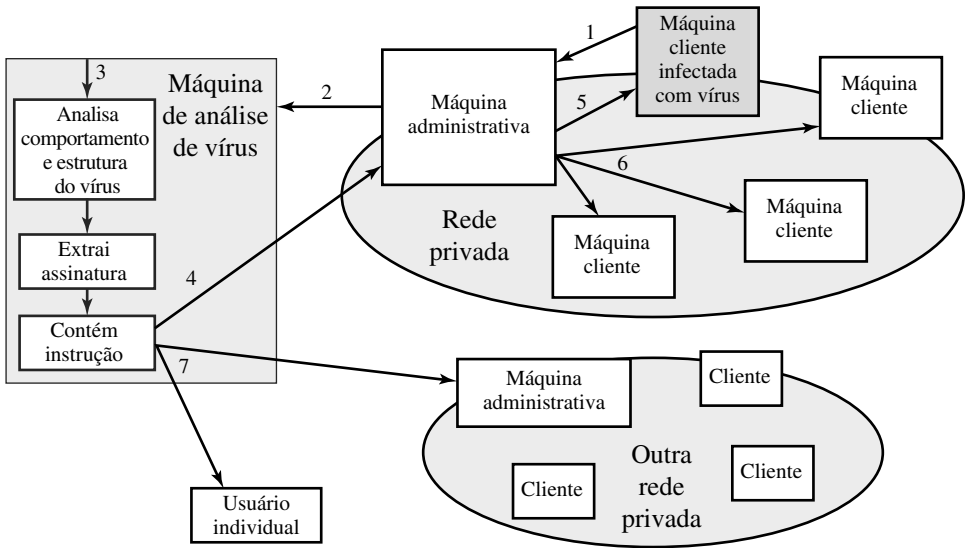


Figura 19.4 Sistema digital imune.

2. A máquina administrativa criptografa a amostra e a envia para uma máquina central de análise de vírus.
3. Essa máquina cria um ambiente em que o programa infectado pode ser seguramente executado para análise. As técnicas usadas para esse propósito incluem a emulação, ou a criação de um ambiente protegido dentro do qual um programa suspeito pode ser executado e monitorado. A máquina de análise de vírus produz então uma instrução para identificar e remover o vírus.
4. A instrução resultante é enviada de volta à máquina administrativa.
5. A máquina administrativa encaminha a instrução ao cliente infectado.
6. A instrução também é encaminhada a outros clientes na organização.
7. Os assinantes no mundo inteiro recebem atualizações regulares de antivírus que os protegem contra o novo vírus.

O sucesso do sistema digital imune depende da capacidade de a máquina de análise de vírus detectar composições novas e inovadoras de vírus. Analisando e monitorando constantemente os vírus encontrados em campo, deverá ser possível atualizar continuamente o software digital imune para que se mantenha parelho com a ameaça.

Software de bloqueio de comportamento

Ao contrário de scanners de heurística ou baseados em impressão digital, o software de bloqueio de comportamento se integra ao sistema operacional de um computador host e monitora o comportamento do programa em tempo real em busca de ações maliciosas. O software de bloqueio de comportamento bloqueia ações potencialmente maliciosas antes que elas tenham a chance de afetar o sistema. Os comportamentos monitorados podem incluir o seguinte:

- Tentativas de abrir, visualizar, excluir e/ou modificar arquivos;
- Tentativas de formatar unidades de disco e outras operações de disco que não podem ser desfeitas;
- Modificações na lógica de arquivos executáveis ou macros;
- Modificação de configurações críticas do sistema, como configurações de inicialização;
- Scripting de e-mail e clientes de mensagem instantânea para enviar conteúdo executável; e
- Inicialização de comunicações em rede.

Se o bloqueador de comportamento detectar que um programa está iniciando comportamentos provavelmente maliciosos enquanto é executado, ele pode bloquear esses comportamentos em tempo real e/ou encerrar o software ofensivo. Isso lhe dá uma vantagem fundamental em relação a técnicas de detecção antivírus estabelecidas, como impressão digital e heurísticas. Embora existam literalmente trilhões de maneiras diferentes de confundir e reorganizar as instruções de um vírus ou verme, muitas das quais evitam a detecção por um scanner de impressão digital ou de heurística, no fim o código malicioso precisa fazer uma solicitação bem definida ao sistema operacional. Visto que o bloqueador de comportamento pode interceptar todas essas solicitações, ele pode identificar e bloquear ações maliciosas independentemente de quão obscura a lógica do programa pareça ser.

A capacidade de observar o software enquanto ele é executado em tempo real claramente confere um grande benefício ao bloqueador de comportamento; porém, ele também possui desvantagens. Como o código malicioso precisa realmente ser executado na máquina de destino antes que todos os seus comportamentos possam ser identifica-

dos, ele pode causar muitos danos ao sistema antes que seja detectado e bloqueado pelo sistema de bloqueio de comportamento. Por exemplo, um novo vírus pode embaralhar diversos arquivos aparentemente não importantes pelo disco rígido antes de infectar um único arquivo e ser bloqueado. Embora a infecção real seja bloqueada, o usuário poderá ser incapaz de localizar seus arquivos, causando uma perda na produtividade ou, possivelmente, algo pior.

19.3 ATAQUES DE NEGAÇÃO DE SERVIÇO DISTRIBUÍDOS

Os ataques de negação de serviço distribuídos (DDoS — Distributed Denial of Service) apresentam uma ameaça de segurança significativa para as empresas, e a ameaça parece estar crescendo [VIJA02]. Em um estudo, abrangendo um período de três semanas em 2001, investigadores observaram mais de 12 mil ataques contra mais de 5 mil alvos distintos, variando desde empresas de e-commerce bem conhecidas, como Amazon e Hotmail, até pequenos provedores de serviços Internet fora dos EUA e conexões discadas [MOOR01]. Ataques de DDoS tornam os sistemas de computador inacessíveis inundando servidores, redes e até mesmo sistemas de usuário final com tráfego inútil, para que os usuários legítimos não possam mais ter acesso a esses recursos. Em um ataque de DDoS típico, um grande número de hosts comprometidos é reunido para enviar pacotes inúteis. Nos últimos anos, os métodos e as ferramentas de ataque se tornaram mais sofisticados, eficazes e mais difíceis de rastrear até os atacantes reais, enquanto as tecnologias de defesa se tornaram incapazes de suportar ataques em grande escala [CHAN02].

Um ataque de negação de serviço (DoS — Denial of Service) é uma tentativa de impedir que usuários legítimos de um serviço usem esse serviço. Quando esse ataque provém de um único host ou nó da rede, então ele é simplesmente chamado de ataque de DoS. Uma ameaça mais séria é imposta por um ataque de DDoS. Em um ataque de DDoS, um atacante é capaz de recrutar diversos hosts pela Internet para desferirem simultaneamente, ou de maneira coordenada, um ataque ao alvo. Esta seção trata dos ataques de DDoS. Primeiro, examinamos a natureza e os tipos dos ataques. Em seguida, examinamos meios pelos quais um atacante é capaz de recrutar uma rede de hosts para desferimento do ataque. Finalmente, esta seção examina as contramedidas.

Descrição do ataque de DDoS

Um ataque de DDoS tenta consumir os recursos do alvo de modo que ele não possa fornecer o serviço. Um modo de classificar os ataques de DDoS é em termos do tipo de recurso consumido. De modo geral, o recurso consumido é um recurso interno do host no sistema alvo ou a capacidade de transmissão de dados na rede local do alvo atacado.

Um exemplo simples de um **ataque de recurso interno** é o ataque de inundação de SYN. A Figura 19.5a mostra as etapas envolvidas:

1. O atacante toma o controle de vários hosts pela Internet, instruindo-os a entrar em contato com o servidor Web do alvo.
2. Os hosts escravos começam a enviar para o alvo pacotes SYN (sincronismo/inicialização) do TCP/IP, com informações de endereço de IP de retorno errado, para o alvo.
3. Cada pacote SYN é uma solicitação para abrir uma conexão TCP. Para cada pacote desse tipo, o servidor Web responde com um pacote SYN/ACK (sincronizar/confirmar), tentando estabelecer uma conexão TCP com uma entidade TCP em um endereço de IP falso. O servidor Web mantém uma estrutura de dados para cada solicitação SYN esperando por uma resposta de volta, e fica preso à medida que mais tráfego chega. O resultado é que as conexões legítimas são negadas enquanto a máquina vítima está esperando para completar as falsas conexões 'meio abertas'.

A estrutura de dados de estado do TCP é um recurso interno alvo bastante visado, mas de forma alguma é o único. [CERT01] fornece os seguintes exemplos:

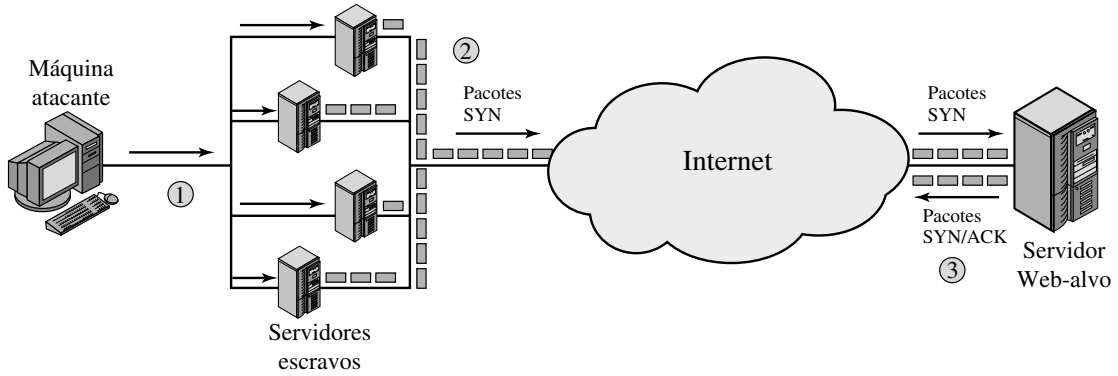
1. Em muitos sistemas, um número limitado de estruturas de dados está disponível para manter informações de processo (identificadores de processo, entradas de tabela de processo, slots de processamento etc.). Um intruso pode ser capaz de consumir essas estruturas de dados escrevendo um programa simples ou script que não faz nada além de criar repetidamente cópias de si mesmo.
2. Um intruso também pode tentar consumir espaço em disco de outras maneiras, incluindo
   - gerar quantidades excessivas de mensagens de e-mail
   - gerar intencionalmente erros que precisam ser registrados
   - colocar arquivos em áreas de ftp anônimas ou em áreas compartilhadas da rede

A Figura 19.5b ilustra o exemplo de um **ataque que consome recursos de transmissão de dados**. Nesse ataque, estão envolvidas as etapas relacionadas a seguir.

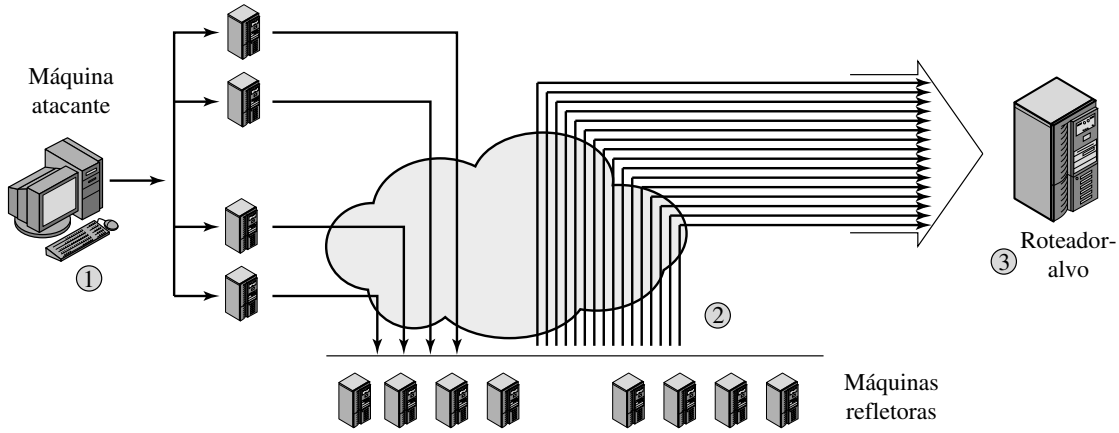
1. O atacante assume o controle de vários hosts pela Internet, instruindo-os a enviar pacotes ICMP ECHO[1] com o endereço de IP falsificado do alvo para um grupo de hosts que atuam como refletores, conforme descrevemos subseqüentemente.
2. Os nós no site de reflexão recebem várias solicitações falsificadas e respondem enviando pacotes de resposta de echo ao site-alvo.
3. O roteador-alvo é inundado com pacotes do site de reflexão, deixando-o sem capacidade de transmissão de dados para o tráfego legítimo.

Outra maneira de classificar ataques de DDoS é como ataques de DDoS diretos ou refletores. Em um ataque de **DDoS direto** (Figura 19.6a), o atacante é capaz de implantar software zumbi em diversos sites distribuídos pela Internet. Normalmente, o ataque de DDoS envolve dois níveis de máquinas zumbi: zumbis-mestres e zumbis escravos. Os hosts das duas máquinas foram infectados com código malicioso. O atacante coordena e dispara os zumbis-mestres, que por sua vez coordenam e disparam os zumbis escravos. O uso de dois níveis de zumbis torna mais difícil o trabalho de rastrear o ataque até a sua origem e fornece uma rede de atacantes mais flexível.

Um ataque de **DDoS refletor** acrescenta outra camada de máquinas (Figura 19.6b). Nesse tipo de ataque, os zumbis escravos criam pacotes solicitando uma resposta que contenha o endereço de IP do alvo como o endereço de IP de origem no cabeçalho do pacote de IP. Esses pacotes são enviados a máquinas não infectadas, conhecidas como refletoras. As máquinas não infectadas respondem com pacotes dirigidos para a máquina-alvo. Um ataque de DDoS refletor pode facilmente envolver mais máquinas e mais tráfego do que um ataque de DDoS direto e, portanto, ser mais prejudicial. Além disso, é mais difícil rastrear o ataque ou filtrar os pacotes de ataque, pois o ataque provém de máquinas não infectadas bastante dispersas.



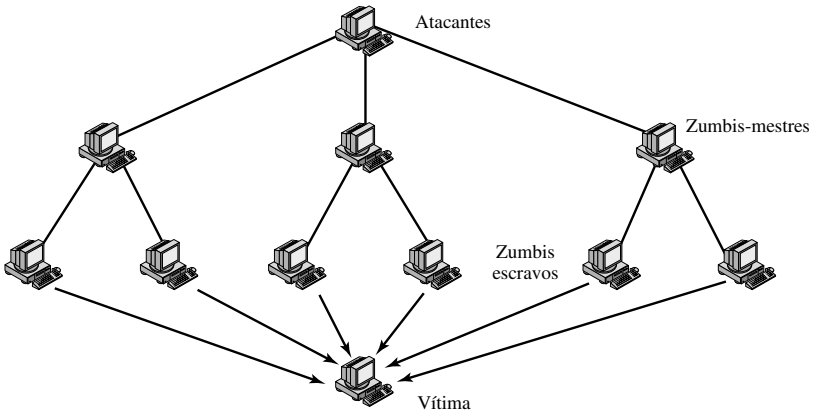
(a) Ataque distribuído de inundação de SYN



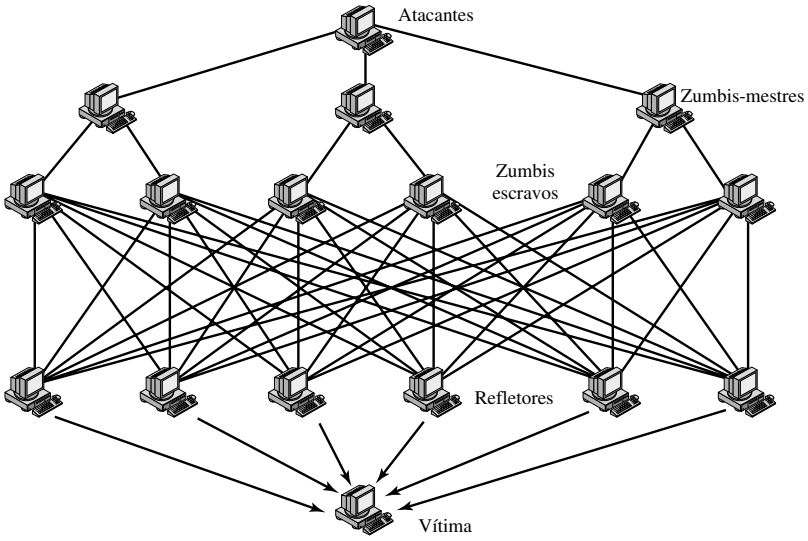
(a) Ataque distribuído de ICMP

Figura 19.5 Exemplos de ataques de DDoS simples.

[1] O Protocolo de Mensagem de Controle da Internet (ICMP — Internet Control Message Protocol) é um protocolo (de mesmo nível do IP) para o intercâmbio de pacotes de controle entre um roteador e um host ou entre hosts. O pacote ECHO solicita que o destinatário responda com uma resposta de echo (echo reply) para verificar se a comunicação é possível entre as entidades envolvidas.



(a) Tipos de ataque de DDoS direto



(b) Ataque de DDoS refletor

Figura 19.6 Tipos de ataques de DDoS baseados em inundação.

Criando a rede de ataque

A primeira etapa em um ataque de DDoS é o atacante infectar diversas máquinas com software zumbi que, por fim, serão usadas para executar o ataque. Os ingredientes essenciais nesta fase do ataque são os seguintes:

1. Software que possa efetuar o ataque de DDoS. O software precisa ser capaz de ser executado em um grande número de máquinas, de ocultar sua existência, de comunicar-se com o atacante ou ter algum tipo de mecanismo disparado por tempo e de iniciar o ataque pretendido contra o alvo.
2. Uma vulnerabilidade em uma grande quantidade de sistemas. O atacante precisa conhecer uma vulnerabilidade que muitos administradores de sistemas e usuários individuais não corrigiram e que permita ao atacante instalar o software zumbi.
3. Uma estratégia para localizar máquinas vulneráveis, um processo conhecido como varredura.

No processo de varredura, o atacante primeiro procura uma série de máquinas vulneráveis e as infecta. Depois, normalmente, o software zumbi instalado nas máquinas infectadas repete o mesmo processo de varredura, até que uma grande rede distribuída de máquinas infectadas seja criada. [MIRK04] lista os seguintes tipos de estratégias de varredura:

- Aleatória: Cada host comprometido sonda endereços aleatórios no espaço de endereços de IP usando uma semente diferente. Essa técnica produz um alto volume de tráfego na Internet, que pode causar interrupção generalizada até mesmo antes que o ataque real seja desferido.
- Lista de acerto: O atacante compila uma longa lista de máquinas vulneráveis em potencial. Esse pode ser um processo lento, feito por um longo período para evitar a detecção de que um ataque está em andamento. Quando a lista é compilada, o atacante começa a infectar máquinas na lista. Cada máquina infectada é abas-

tecida com uma parte da lista a ser varrida. Essa estratégia resulta em um período de varredura muito curto, que pode tornar difícil detectar que essa infecção está acontecendo.
- Topológica: Esse método usa informações contidas em uma máquina vítima infectada para encontrar mais hosts a serem varridos.
- Sub-rede local: Se um host puder ser infectado por trás de um firewall, esse host então procurará alvos em sua própria rede local. O host usa a estrutura de endereço de sub-rede para encontrar outros hosts que, de outra maneira, seriam protegidos pelo firewall.

Contramedidas para DDoS

Em geral, existem três linhas de defesa contra ataques de DDoS [CHAN02]:

- Prevenção de ataque e ação antecipada (antes do ataque): Estes mecanismos permitem à vítima resistir a tentativas de ataque sem negar o serviço a clientes legítimos. As técnicas incluem a imposição de políticas de consumo de recursos e o fornecimento de recursos reserva disponíveis de acordo com a necessidade. Além disso, os mecanismos de prevenção modificam sistemas e protocolos na Internet para reduzir a possibilidade de ataques de DDoS.
- Detecção e filtragem do ataque (durante o ataque): Estes mecanismos tentam detectar o ataque logo no início e responder imediatamente. Isso minimiza o impacto do ataque sobre o alvo. A detecção envolve procurar padrões de comportamento suspeitos. A resposta envolve a filtragem de pacotes que provavelmente fazem parte do ataque.
- Rastreamento e identificação da origem do ataque (durante e após o ataque): Esta é uma tentativa de identificar a origem do ataque como um primeiro passo na prevenção de ataques futuros. Porém, esse método normalmente não gera resultados com rapidez suficiente para mitigar um ataque em andamento.

O desafio ao lidar com ataques de DDoS é o grande número de maneiras como eles podem operar. Assim, as contramedidas para DDoS precisam evoluir com a ameaça.

19.4 LEITURA E SITES WEB RECOMENDADOS

Para ter um conhecimento completo dos vírus, o livro a ser lido é [SZOR05]. Outra abordagem excelente é [HARL01]. Bons artigos gerais sobre vírus e vermes são [CASS01], [FORR97], [KEPH97] e [NACH97]. [MEIN01] oferece uma boa explicação do verme Code Red.

[PATR04] é uma pesquisa que vale a pena sobre ataques de DDoS. [MIRK04] é uma descrição completa da variedade de ataques de DDoS e contramedidas. [CHAN02] é um bom exame das estratégias de defesa do DDoS.

CASS01 Cass, S. "Anatomy of malice." *IEEE Spectrum*, novembro 2001.

CHAN02 Chang, R. "Defending against flooding-based distributed denial-of-service attacks: a tutorial." *IEEE Communications Magazine*, outubro de 2002.

FORR97 Forrest, S.; Hofmeyr, S. e Somayaji, A. "Computer immunology". *Communications of the ACM*, outubro de 1997.

HARL01 Harley, D.; Slade, R. e Gattiker, U. *Viruses revealed*. Nova York: Osborne/McGraw-Hill, 2001.

KEPH97 Kephart, J.; Sorkin, G.; Chess, D. e White, S. "Fighting computer viruses". *Scientific American*, novembro de 1997.

MEIN01 Meinel, C. "Code Red for the Web". *Scientific American*, outubro de 2001.

MIRK04 Mirkovic, J. e Relher, P. "A taxonomy of DDoS attack and DDoS defense mechanisms". *ACM SIGCOMM Computer Communications Review*, abril de 2004.

NACH97 Nachenberg, C. "Computer virus-antivirus coevolution". *Communications of the ACM*, janeiro de 1997.

PATR04 Patrikakis, C.; Masikos, M. e Zouraraki, O. "Distributed denial of service attacks". *The Internet Protocol Journal*, dezembro de 2004.

SZOR05 Szor, P., *The art of computer virus research and defense*. Reading: Addison-Wesley, 2005.

Sites Web recomendados:

- **AntiVirus Online:** site da IBM com informações sobre vírus
- **Vmyths:** dedicado a revelar histórias falsas sobe vírus e dissipar falsos conceios sobre vírus reais
- **DDoS Attacks/Tools:** extensa lista de links e documentos

19.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

ataque de DDoS direto
ataque de DDoS refletor
auto-rooter
backdoor
bomba lógica
cavalo-de-tróia
downloaders
explorações
flooder
keylogger
kit
negação de serviço distribuída (DDoS)
programa de spam
rootkit
sistema digital imune
software malicioso (malware)
trapdoor
verme
vírus
vírus de e-mail
vírus de macro
vírus furtivo
vírus polimórfico
zumbi

Perguntas de revisão

19.1 Qual é o papel da compactação na operação de um vírus?

19.2 Qual é o papel da criptografia na operação de um vírus?

19.3 Quais são as fases típicas da operação de um vírus ou um verme?

19.4 Em termos gerais, como um verme se propaga?

19.5 O que é um sistema digital imune?

19.6 Como funciona o software de bloqueio de comportamento?

19.7 O que é um DDoS?

Problemas

19.1 Existe uma falha no programa de vírus da Figura 19.1. Qual é a falha?

19.2 Pode-se perguntar se é possível desenvolver um programa que possa analisar um fragmento de software para determinar se é um vírus. Considere que temos um programa D que deveria ser capaz de fazer isso. Ou seja, para qualquer programa P, se executarmos D(P), o resultado retornado será TRUE (P é um vírus) ou FALSE (P não é um vírus). Agora, considere o seguinte programa:

```
Program CV :=
  { . . .
  main-program :=
        {if D(CV) then goto next:
              else infect-executable;
        }
next:
  }
```

No programa anterior, infect-executable é um módulo que varre a memória em busca de programas executáveis e se copia nesses programas. Determine se D pode decidir corretamente se CV é um vírus.

CAPÍTULO 20

Firewalls

20.1 **Princípios de projeto de firewall**

20.2 **Sistemas confiáveis**

20.3 **Critérios comuns para avaliação de segurança da tecnologia da informação**

20.4 **Leitura e sites Web recomendados**

20.5 **Principais termos, perguntas de revisão e problemas**

A função de uma posição forte é tornar as forças que a mantêm praticamente inatacáveis.

— *Da guerra*, Carl Von Clausewitz

No dia em que você tomar seu comando, bloqueie as passagens da fronteira, destrua os registros oficiais e interrompa a passagem de todos os emissários.

— *A arte da guerra, Sun Tzu*

PONTOS PRINCIPAIS

- Um firewall forma uma barreira através da qual o tráfego indo em cada direção precisa passar. Uma política de segurança de firewall dita qual tráfego tem autorização para passar em cada direção.
- Um firewall pode ser projetado para operar como um filtro no nível de pacotes IP ou pode operar em uma camada de protocolo mais alta.
- Um sistema confiável é um computador e um sistema operacional que podem ser verificados para implementar determinada política de segurança. Normalmente, o foco de um sistema confiável é o controle de acesso. Uma política é implementada para ditar quais objetos podem ser acessados por quais sujeitos.
- O critério comum para a segurança da tecnologia de informação é uma iniciativa de padrões internacionais para definir um conjunto comum de requisitos de segurança e um meio sistemático de avaliar produtos em relação a esses requisitos.

Os firewalls podem ser um meio eficaz de proteger um sistema local ou um sistema de redes contra ameaças de segurança baseadas em rede, enquanto ao mesmo tempo garante acesso ao mundo exterior por meio de redes de longa distância e da Internet.

Começamos este capítulo com uma visão geral da funcionalidade e dos princípios de projeto de firewalls. Em seguida, tratamos da questão da segurança do próprio firewall e, em particular, o conceito de um sistema confiável, ou sistema operacional seguro.

20.1 PRINCÍPIOS DE PROJETO DE FIREWALL

Os sistemas de informação nas empresas, órgãos do governo e outras organizações têm passado por uma evolução contínua:

- O sistema de processamento de dados centralizado, com um mainframe central oferecendo suporte a diversos terminais diretamente conectados
- Redes locais (LANs) interconectando PCs e terminais uns aos outros e ao mainframe
- Rede local de instalações (premises network), composta por diversas LANs, interconectando PCs, servidores e talvez um ou dois mainframes
- Rede corporativa, consistindo de várias redes locais de instalações geograficamente distribuídas, interconectadas por uma rede geograficamente distribuída (WAN) privada
- Conectividade com a Internet, em que as diversas redes locais de instalações se conectam à Internet e podem ou não também estar conectadas por uma WAN privada

A conectividade com a Internet não é mais opcional para as organizações. A informação e os serviços disponíveis são essenciais à organização. Além disso, usuários individuais dentro da organização desejam e precisam de acesso à Internet, e se este não for fornecido através de sua LAN, eles usarão a capacidade de conexão discada de seus PCs com um provedor de serviços de Internet (ISP). Porém, embora o acesso à Internet ofereça benefícios à organização, ele permite que o mundo exterior alcance e interaja com os recursos da rede local. Isso cria uma ameaça à organização. Embora seja possível equipar cada estação de trabalho e servidor na rede local de um edifício ou conjunto de edifícios com fortes medidas de segurança, como proteção de intrusão, essa não é uma técnica prática. Considere uma rede com centenas ou até mesmo milhares de sistemas, executando uma mistura de diversas versões do UNIX mais Windows. Quando uma falha de segurança é descoberta, cada sistema potencialmente afetado precisa ser atualizado para corrigir essa falha. A alternativa, cada vez mais aceita, é o firewall. O firewall é inserido entre a rede local de instalações e a Internet, para estabelecer um enlace controlado e erguer uma parede ou um perímetro de segurança externo. O objetivo desse perímetro é proteger a rede local de instalações contra ataques baseados na Internet e oferecer um único ponto de controle onde a segurança e a auditoria podem ser impostas. O firewall pode ser um único sistema de computador ou um conjunto de dois ou mais sistemas que cooperam para realizar a função de firewall.

Nesta seção examinamos primeiro as características gerais dos firewalls. Depois, examinamos os tipos de firewalls atualmente em uso comum. Finalmente, examinamos algumas das configurações de firewall mais comuns.

Características do firewall

[BELL94b] lista os seguintes objetivos no projeto para um firewall:

1. Todo o tráfego de dentro para fora, e vice-versa, precisa passar pelo firewall. Isso é obtido bloqueando-se fisicamente todo o acesso à rede local, exceto por meio do firewall. Diversas configurações são possíveis, conforme explicamos mais adiante nesta seção.
2. Somente o tráfego autorizado, definido pela política de segurança local, terá permissão para passar. Diversos tipos de firewalls são usados, o que implementa diversos tipos de políticas de segurança, conforme explicado mais adiante nesta seção.
3. O próprio firewall é imune à penetração. Isso implica o uso de um sistema confiável com um sistema operacional seguro. Esse assunto é explicado na Seção 20.2.

[SMIT97] lista quatro técnicas gerais que os firewalls utilizam para controlar o acesso e impor a política de segurança do site. Originalmente, os firewalls focalizavam principalmente o controle de serviço, mas desde então eles evoluíram para fornecer todas as quatro:

- **Controle de serviço:** Determina os tipos de serviços de Internet que podem ser acessados, de entrada ou saída. O firewall pode filtrar o tráfego com base no endereço IP e número de porta TCP; pode oferecer software de proxy que recebe e interpreta cada solicitação de serviço antes de passá-la adiante, ou pode hospedar o próprio software do servidor, como um serviço Web ou de correio.
- **Controle de direção:** Determina a direção em que determinadas solicitações de serviço podem ser iniciadas e permitidas para fluir através do firewall.

- **Controle de usuário:** Controla o acesso a um serviço de acordo com qual usuário está tentando acessá-lo. Esse característica normalmente é aplicada aos usuários dentro do perímetro do firewall (usuários locais). Ela também pode ser aplicada ao tráfego que chega de usuários externos; esse último exige alguma forma de tecnologia de autenticação segura, como a que é fornecida no IPSec (Capítulo 16).
- **Controle de comportamento:** Controla como determinados serviços são usados. Por exemplo, o firewall pode filtrar e-mail para eliminar spam ou pode permitir o acesso externo a apenas uma parte da informação em um servidor Web local.

Antes de prosseguir para os detalhes dos tipos e configurações de firewall, é melhor resumir o que se pode esperar de um firewall. As capacidades a seguir estão dentro do escopo de um firewall:

1. Um firewall define um único ponto de estrangulamento que mantém usuários não autorizados fora da rede protegida, proíbe que serviços potencialmente vulneráveis entrem ou saiam da rede e oferece proteção contra diversos tipos de ataques de falsificação e roteamento do IP. O uso de um único ponto de estrangulamento simplifica o gerenciamento da segurança, pois as capacidades de segurança são consolidadas em um único sistema ou conjunto de sistemas.
2. Um firewall oferece um local para monitorar eventos relacionados à segurança. Auditorias e alarmes podem ser implementados no sistema de firewall.
3. Um firewall é uma plataforma conveniente para várias funções da Internet que não estão relacionadas à segurança. Estas incluem um tradutor de endereço de rede, que mapeia endereços locais para endereços da Internet, e uma função de gerenciamento de rede, que audita ou registra o uso da Internet.
4. Um firewall pode servir como a plataforma para o IPSec. Usando a capacidade do modo túnel descrita no Capítulo 16, o firewall pode ser usado para implementar redes privadas virtuais.

Os firewalls têm suas limitações, incluindo as seguintes:

1. O firewall não pode proteger contra ataques que contornam o firewall. Os sistemas internos podem ter capacidade de discagem para conectar-se com um provedor. Uma LAN interna pode dar suporte a um grupo de modems, que oferece capacidade de discagem para funcionários que estejam trabalhando fora das instalações da empresa.
2. O firewall não protege contra ameaças internas, como um funcionário insatisfeito ou um funcionário que coopera, sem querer, com um atacante externo.
3. O firewall não pode proteger contra a transferência de programas ou arquivos infectados com vírus. Devido à variedade de sistemas operacionais e aplicações aceitas dentro do perímetro, seria impraticável e talvez impossível que o firewall varresse todos os arquivos, e-mails e mensagens que chegam, em busca de vírus.

Tipos de firewalls

A Figura 20.1 ilustra os três tipos comuns de firewalls: filtros de pacotes, gateways em nível de aplicação e gateways em nível de circuito. Examinamos cada um destes por sua vez.

Roteador de filtragem de pacotes Um roteador de filtragem de pacotes aplica um conjunto de regras a cada pacote IP que entra e sai e depois encaminha ou descarta o pacote. O roteador normalmente é configurado para filtrar pacotes entrando em ambas direções (de e para a rede interna). As regras de filtragem são baseadas nas informações contidas em um pacote da rede:

- **Endereço IP de origem:** O endereço IP do sistema que originou o pacote IP (por exemplo, 192.178.1.1)
- **Endereço IP de destino:** O endereço IP do sistema que o pacote IP está tentando alcançar (por exemplo, 192.168.1.2)
- **Endereço de origem e destino em nível de transporte:** O número de porta em nível de transporte (por exemplo, TCP ou UDP), que define aplicações como SNMP ou TELNET
- **Campo de protocolo IP:** Define o protocolo de transporte
- **Interface:** Para um roteador com três ou mais portas, de qual interface do roteador o pacote veio ou para qual interface do roteador o pacote é destinado

O filtro de pacotes normalmente é configurado como uma lista de regras baseadas em combinações com campos no cabeçalho IP ou TCP. Se houver uma correspondência com uma das regras, essa regra é chamada para determinar se encaminhará ou descartará o pacote. Se não houver correspondência com qualquer regra, então a ação-padrão é tomada. Duas políticas-padrão são possíveis:

- **Padrão** = *descartar*: Aquilo que não é expressamente permitido é proibido.
- **Padrão** = *encaminhar*: Aquilo que não é expressamente proibido é permitido.

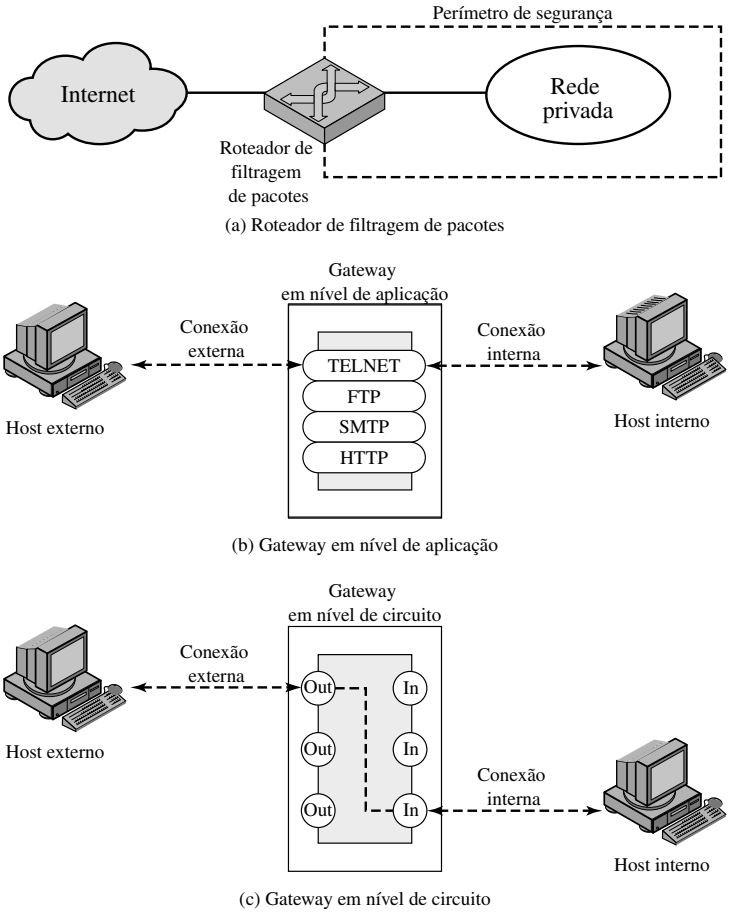


Figura 20.1 Tipos de firewalls.

A política de descarte padrão é mais conservadora. Inicialmente, tudo é bloqueado, e os serviços precisam ser acrescentados com base em cada caso. Essa política é mais visível aos usuários, que provavelmente verão o firewall como um obstáculo. A política de encaminhamento-padrão aumenta a facilidade de uso para os usuários finais, mas oferece menor segurança; o administrador de segurança precisa, basicamente, reagir a cada nova ameaça de segurança que se tornar conhecida.

A Tabela 20.1, de [BELL94b], contém alguns exemplos de conjuntos de regras de filtragem de pacotes. Em cada conjunto, as regras são aplicadas de cima para baixo. O "*" em um campo é um indicador de curinga, que corresponde a qualquer valor. Consideramos que a política-padrão = descartar está em vigor.

A. Correio recebido é permitido (porta 25 destina-se a SMTP entrante), mas somente para um host de gateway. Porém, os pacotes de determinado host externo, SPIGOT, são bloqueados porque esse host tem um histórico de enviar grandes arquivos em mensagens de e-mail.

B. Essa é uma declaração explícita da política-padrão. Todos os conjuntos de regras incluem essa regra implicitamente como a última regra.

C. Este conjunto de regras serve para especificar que qualquer host interno pode enviar correio para fora. Um pacote TCP com uma porta de destino 25 é roteado para o servidor SMTP na máquina de destino. O problema com essa regra é que o uso da porta 25 para recebimento de SMTP é apenas um padrão; uma máquina externa poderia ser configurada para ter alguma outra aplicação ligada à porta 25. Como essa regra é escrita, um atacante pode obter acesso às máquinas internas enviando pacotes com um número de porta de origem TCP 25.

D. Esse conjunto de regras alcança o resultado pretendido que não foi alcançado em C. As regras tiram proveito de um recurso das conexões TCP. Quando uma conexão é estabelecida, o flag ACK de um segmento TCP é configurado para confirmar segmentos enviados a partir do outro lado. Assim, esse conjunto de regras afirma que ele permite pacotes IP onde o endereço IP de origem consta em uma lista de hosts internos especificados e o número de porta TCP de destino é 25. Ele também permite pacotes que chegam com um número de porta de origem igual a 25, que incluem o flag ACK no segmento TCP. Observe que designamos explicitamente os sistemas de origem e destino para definir essas regras explicitamente.

Tabela 20.1 Exemplos de filtragem de pacotes

A

ação	nossohost	porta	hostdeles	porta	comentário
bloquear	*	*	SPIGOT	*	não confiamos nessas pessoas
permitir	OUR-GW	25	*	*	conexão com nossa porta SMTP

B

ação	nossohost	porta	hostdeles	porta	comentário
bloquear	*	*	*	*	padrão

C

ação	nossohost	porta	hostdeles	porta	comentário
permitir	*	*	*	25	conexão com a porta SMTP deles

D

ação	src	porta	dest	porta	flags	comentário
permitir	{our hosts}	*	*	25		nossos pacotes para a porta SMTP deles
permitir	*	25	*	*	ACK	respostas deles

E

ação	src	porta	dest	porta	flags	comentário
permitir	{our hosts}	*	*	*		nossas chamadas para fora
permitir	*	*	*	*	ACK	respostas às nossas chamadas
permitir	*	*	*	>1024		tráfego para não-servidores

E. Esse conjunto de regras é uma técnica para lidar com conexões FTP. Com FTP, duas conexões TCP são usadas: uma conexão de controle para configurar a transferência de arquivos e uma conexão de dados para a transferência de arquivos reais. A conexão de dados usa um número de porta distinto atribuído dinamicamente para a transferência. A maioria dos servidores e, portanto, a maior parte dos alvos de ataque, reside em portas com números baixos; a maioria das chamadas que saem costuma usar uma porta com número mais alto, normalmente acima de 1023. Assim, esse conjunto de regras permite

— Pacotes originados internamente
— Pacotes de resposta a uma conexão iniciada por uma máquina interna
— Pacotes destinados a uma porta alta em uma máquina interna
— Esse esquema exige que os sistemas sejam configurados de modo que somente os números de porta apropriados estejam em uso.

O conjunto de regras E aponta a dificuldade de lidar com aplicações no nível de filtragem de pacotes. Outra forma de lidar com FTP e aplicações semelhantes é um gateway em nível de aplicação, descrito mais adiante nesta seção.

Uma vantagem de um roteador de filtragem de pacotes é a sua simplicidade. Além disso, os filtros de pacotes normalmente são transparentes aos usuários e são muito rápidos. [WACK02] lista os seguintes pontos fracos dos firewalls de filtro de pacotes:

- Como os firewalls de filtro de pacotes não examinam dados da camada superior, eles não podem impedir ataques que empregam vulnerabilidades ou funções de aplicação específicas. Por exemplo, um firewall de filtro de pacotes não pode bloquear comandos de aplicação específicos; se um firewall de filtro de pacotes permitir determinada aplicação, todas as funções disponíveis dentro dessa aplicação serão permitidas.
- Devido às informações limitadas disponíveis ao firewall, a funcionalidade de registro (log) presente nos firewalls de filtro de pacotes é limitada. Os logs de filtro de pacotes normalmente contêm a mesma informação usada para tomar decisões de controle de acesso (endereço de origem, endereço de destino e tipo de tráfego).
- A maioria dos firewalls de filtro de pacotes não suportam esquemas avançados de autenticação de usuário. Mais uma vez, essa limitação ocorre principalmente devido à falta de funcionalidade da camada superior pelo firewall.
- Eles geralmente são vulneráveis a ataques e explorações que tiram proveito de problemas dentro da especificação e da pilha de protocolos TCP/IP, como a *falsificação do endereço de camada* da rede. Muitos firewalls de filtro de pacotes não podem detectar um pacote de rede em que a informação de endereçamento da Camada 3 do modelo OSI foi alterada. Os ataques de falsificação geralmente são empregados por intrusos para contornar os controles de segurança implementados em uma plataforma de firewall.
- Finalmente, devido ao pequeno número de variáveis usadas nas decisões de controle de acesso, os firewalls de filtro de pacotes são suscetíveis a brechas de segurança causadas por configurações impróprias. Em outras palavras, é fácil configurar acidentalmente um firewall de filtro de pacotes para permitir tipos de tráfego de origens e destinos que devem ser negados com base na política de segurança de informações de uma organização.

Alguns dos ataques que podem ser efetuados a roteadores de filtragem de pacotes e as contramedidas apropriadas são as seguintes:

- **Falsificação (spoofing) de endereço IP :** O intruso transmite pacotes vindos de fora com um campo de endereço IP de origem contendo o endereço de um host interno. O atacante espera que o uso de um endereço falsificado permita a penetração de sistemas que empregam segurança simples do endereço de origem, em que os pacotes de hosts internos confiáveis específicos são aceitos. A contramedida é descartar pacotes com um endereço de origem interno se o pacote chegar por uma interface externa.
- **Ataques de roteamento da origem:** A estação de origem especifica a rota que um pacote deverá seguir enquanto atravessa a Internet, na esperança de que isso evitará as medidas de segurança que não analisam as informações do roteamento de origem. A contramedida é descartar todos os pacotes que usam essa opção.
- **Ataques de fragmento pequeno:** O intruso usa a opção de fragmentação IP para criar fragmentos extremamente pequenos e forçar as informações de cabeçalho TCP em um fragmento de pacote separado. Esse ataque é preparado para contornar as regras de filtragem que dependem das informações do cabeçalho TCP. Normalmente, um filtro de pacotes tomará uma decisão de filtragem sobre o primeiro fragmento de um pacote. Todos os fragmentos subseqüentes desse pacote serão filtrados unicamente com base em que eles são parte do pacote cujo primeiro fragmento foi rejeitado. O atacante espera que o roteador de filtragem examine apenas o primeiro fragmento e que os fragmentos restantes tenham permissão para passar. Um ataque de fragmento pequeno pode ser neutralizado impondo-se uma regra em que o primeiro fragmento de um pacote deva conter uma quantidade mínima predefinida do cabeçalho de transporte. Se o primeiro fragmento for rejeitado, o filtro poderá se lembrar do pacote e descartar todos os fragmentos subseqüentes.

Firewalls de inspeção com estado (stateful inspection firewalls) Um filtro de pacotes tradicional toma decisões de filtragem com base em um pacote individual e não leva em consideração qualquer contexto de camada superior. Para entender o que significa contexto e por que o filtro de pacotes tradicional é limitado com relação ao contexto, é necessário um pouco de experiência. A maioria das aplicações padronizadas que são executadas em cima do TCP segue um modelo cliente/servidor. Por exemplo, para o Protocolo Simples de Transferência de Correio (SMTP – Simple Mail Transfer Protocl), o e-mail é transmitido de um sistema cliente para um sistema servidor. O sistema cliente gera novas mensagens de e-mail, normalmente a partir da entrada do usuário. O sistema servidor aceita as mensagens de e-mail que chegam e as coloca nas caixas de entrada dos usuários apropriados. O SMTP opera preparando uma conexão TCP entre cliente e servidor, em que o número de porta do servidor TCP, que identifica a aplicação de servidor SMTP, é 25. O número de porta TCP para o cliente SMTP é um número entre 1024 e 65535, que é gerado pelo cliente SMTP.

Em geral, quando uma aplicação que usa TCP cria uma sessão com um host remoto, ela cria uma conexão TCP em que o número de porta TCP para a aplicação remota (servidor) é um número menor que 1024 e o número de porta TCP para a aplicação local (cliente) é um número entre 1024 e 65535. Os números menores que 1024 são os números de porta "bem conhecidos" e são atribuídos permanentemente a determinadas aplicações (por exemplo, 25 para o servidor SMTP). Os números entre 1024 e 65535 são gerados dinamicamente e possuem significado temporário apenas pelo tempo de vida de uma conexão TCP.

Um firewall simples de filtragem de pacotes precisa permitir que ocorra tráfego de rede entrante em todas essas portas de número alto para o tráfego baseado em TCP. Isso cria uma vulnerabilidade que pode ser explorada por usuários não autorizados.

Um filtro de pacote com inspeção de estado estreita as regras para tráfego TCP, criando um catálogo de conexões TCP de saída, como mostra a Tabela 20.2. Existe uma entrada para cada conexão estabelecida atualmente. O filtro de pacotes agora só permitirá o tráfego que chega para portas de número alto para pacotes que se encaixam no perfil de uma das entradas nesse diretório.

Tabela 20.2 Exemplo de tabela de estados de conexão de firewall com estado [WACK02]

Endereço de origem	Porta de origem	Endereço de destino	Porta de destino	Estado da conexão
192.168.1.100	1030	210.9.88.29	80	Estabelecida
192.168.1.102	1031	216.32.42.123	80	Estabelecida
192.168.1.101	1033	173.66.32.122	25	Estabelecida
192.168.1.106	1035	177.231.32.12	79	Estabelecida
223.43.21.231	1990	192.168.1.6	80	Estabelecida
219.22.123.32	2112	192.168.1.6	80	Estabelecida
210.99.212.18	3321	192.168.1.6	80	Estabelecida
24.102.32.23	1025	192.168.1.6	80	Estabelecida
223.212.212	1046	192.168.1.6	80	Estabelecida

Um gateway em nível de aplicação, também chamado de servidor proxy, atua como um retransmissor de tráfego em nível de aplicação (Figura 20.1b). O usuário contata o gateway usando uma aplicação TCP/IP, como Telnet ou FTP, e o gateway solicita ao usuário o nome do host remoto a ser acessado. Quando o usuário responde e oferece uma ID de usuário e informações de autenticação válidas, o gateway contata a aplicação no host remoto e repassa os segmentos TCP contendo os dados da aplicação entre as duas extremidades. Se o gateway não implementar o código de proxy para a aplicação específica, o serviço não será aceito e não poderá ser encaminhado pelo firewall. Além disso, o gateway pode ser configurado para admitir apenas recursos específicos de uma aplicação, os quais o administrador da rede considera aceitáveis, enquanto nega todos os outros recursos.

Os gateways em nível de aplicação tendem a ser mais seguros do que os filtros de pacotes. Em vez de tentar lidar com as diversas combinações possíveis que devem ser permitidas e proibidas no nível de TCP e IP, o gateway em nível de aplicação só precisa examinar algumas aplicações permitidas. Além disso, é fácil registrar e auditar todo o tráfego que chega no nível da aplicação.

Uma desvantagem importante desse tipo de gateway é o overhead de processamento adicional em cada conexão. Na verdade, existem duas conexões divididas entre os usuários finais, com o gateway no ponto de divisão, e o gateway precisa examinar e encaminhar todo o tráfego nas duas direções.

Gateway em nível de circuito Um terceiro tipo de firewall é o gateway em nível de circuito (Figura 20.1c). Esse pode ser um sistema stand-alone ou pode ser uma função especializada, realizada por um gateway no nível de aplicação para certas aplicações. Um gateway em nível de circuito não permite uma conexão TCP de ponta a ponta; em vez disso, o gateway prepara duas conexões TCP, uma entre ele mesmo e um usuário TCP em um host interno e uma entre ele mesmo e um usuário TCP em um host externo. Quando as duas conexões são estabelecidas, o gateway normalmente repassa segmentos TCP de uma conexão para a outra sem examinar o conteúdo. A função de segurança consiste em determinar quais conexões serão permitidas.

Um uso típico dos gateways em nível de circuito é uma situação em que o administrador do sistema confia nos usuários internos. O gateway pode ser configurado para oferecer suporte a serviço em nível de aplicação ou proxy nas conexões entrantes e funções em nível de circuito para as conexões de saída. Nessa configuração, o gateway pode estar sujeito ao overhead de processamento de examinar os dados de aplicação que chegam para funções proibidas, mas não incorre nesse overhead nos dados de saída.

Um exemplo de uma implementação de gateway em nível de circuito é o pacote SOCKS [KOBL92]; a versão 5 do SOCKS é definida na RFC1928. A RFC define o SOCKS da seguinte maneira:

> O protocolo descrito aqui foi projetado para oferecer uma estrutura para aplicações cliente-servidor nos domínios TCP e UDP, para utilizar conveniente e seguramente os serviços de um firewall de rede. O protocolo é conceitualmente uma "camada de calço (shim-layer)" entre a camada de aplicação e a camada de transporte e, dessa maneira, não oferece serviços de gateway da camada de rede, como encaminhamento de mensagens ICMP.

O SOCKS é formado pelos seguintes componentes:

- O servidor SOCKS, que executa em um firewall baseado em UNIX.
- A biblioteca cliente SOCKS, executada nos hosts internos protegidos pelo firewall.
- Versões para SOCKS de diversos programas cliente padrão, como FTP e TELNET. A implementação do protocolo SOCKS normalmente envolve a recompilação ou religação de aplicações cliente baseadas em TCP para usar as rotinas de encapsulamento apropriadas na biblioteca SOCKS.

Quando um cliente baseado em TCP deseja estabelecer uma conexão com um objeto que só pode ser alcançado por um firewall (essa determinação é deixada para a implementação), ele precisa abrir uma conexão TCP com a porta SOCKS apropriada no sistema servidor SOCKS. O serviço SOCKS está localizado na porta TCP 1080. Se a solicitação de conexão obtiver sucesso, o cliente entrará em uma negociação para o método de autenticação a ser usado, autenticará com o método escolhido e depois enviará uma solicitação de retransmissão. O servidor SOCKS avaliará a solicitação e estabelecerá a conexão apropriada ou a negará. As trocas UDP são tratadas de uma maneira semelhante. Basicamente, uma conexão TCP é aberta para autenticar um usuário para enviar e receber segmentos UDP, e os segmentos UDP são encaminhados enquanto a conexão TCP estiver aberta.

Bastion host Um bastion host é um sistema identificado pelo administrador do firewall como um ponto forte crítico na segurança da rede. Normalmente, o bastion host serve como uma plataforma para um gateway em nível de aplicação ou em nível de circuito. Algumas características comuns de um bastion host são:

- A plataforma de hardware do bastion host executa uma versão segura de seu sistema operacional, tornando-o um sistema confiável.

- Somente os serviços que o administrador de rede considera essenciais são instalados no bastion host. Estes incluem aplicações proxy como Telnet, DNS, FTP, SMTP e autenticação do usuário.
- O bastion host pode exigir autenticação adicional antes que um usuário tenha permissão para acessar os serviços do proxy. Além disso, cada serviço do proxy pode exigir sua própria autenticação antes de conceder acesso ao usuário.
- Cada proxy é configurado para dar suporte a apenas um subconjunto do conjunto de comandos de aplicação padrão.
- Cada proxy é configurado para permitir o acesso apenas a sistemas host específicos. Isso significa que o conjunto de comandos/recursos limitado só pode ser aplicado a um subconjunto dos sistemas na rede protegida.
- Cada proxy mantém informações de auditoria detalhadas registrando todo o tráfego, cada conexão e a duração de cada conexão. O log de auditoria é uma ferramenta essencial para descobrir e finalizar ataques de intrusos.
- Cada módulo de proxy é um pacote de software muito pequeno, projetado especificamente para segurança da rede. Devido à sua relativa simplicidade, é mais fácil verificar esses módulos em busca de falhas de segurança. Por exemplo, uma aplicação de correio UNIX típica pode conter mais de 20.000 linhas de código, enquanto um proxy de correio pode conter menos de 1.000.
- Cada proxy é independente dos outros proxies no bastion host. Se houver um problema com a operação de qualquer proxy, ou se uma vulnerabilidade futura for descoberta, ele pode ser desinstalado sem afetar a operação das outras aplicações proxy. Além disso, se a população de usuários precisar de suporte para um novo serviço, o administrador da rede poderá facilmente instalar o proxy exigido no bastion host.
- Um proxy geralmente não realiza acesso ao disco além de ler seu arquivo de configuração inicial. Isso torna difícil para um intruso instalar sniffers (farejadores), cavalo-de-tróia ou outros arquivos perigosos no bastion host.
- Cada proxy é executado como um usuário não privilegiado em um diretório privado e protegido no bastion host.

Configurações de firewall

Além do uso de uma configuração simples, consistindo em um único sistema, como um único roteador de filtragem de pacote ou um único gateway (Figura 20.1), configurações mais complexas são possíveis e, na realidade, são mais comuns. A Figura 20.2 ilustra três configurações de firewall comuns. Vamos examinar cada uma delas.

Na configuração de **firewall com host filtrado, single-homed bastion (com uma única placa de rede)** (Figura 20.2a), o firewall é composto por dois sistemas: um roteador de filtragem de pacotes e um bastion host. Normalmente, o roteador é configurado de modo que

1. Para o tráfego que vem da Internet, somente pacotes IP destinados ao bastion host têm permissão para entrar.
2. Para o tráfego que vem da rede interna, somente pacotes IP do bastion host têm permissão para sair.

O bastion host realiza funções de autenticação e proxy. Essa configuração tem maior segurança do que simplesmente um roteador de filtragem de pacotes ou um gateway em nível de aplicação isolado, por dois motivos. Primeiro, essa configuração implementa a filtragem em nível de pacote e em nível de aplicação, permitindo considerável flexibilidade na definição da política de segurança. Segundo, um intruso geralmente precisa penetrar em dois sistemas separados antes que a segurança da rede interna seja comprometida.

Essa configuração também oferece flexibilidade fornecendo acesso direto à Internet. Por exemplo, a rede interna pode incluir um servidor de informações público, como um servidor Web, para o qual não é exigido um alto nível de segurança. Nesse caso, o roteador pode ser configurado para permitir o tráfego direto entre o servidor de informações e a Internet.

Na configuração single-homed que descrevemos, se o roteador de filtragem de pacotes for completamente comprometido, o tráfego poderá fluir diretamente pelo roteador entre a Internet e outros hosts na rede privada. A configuração **firewall com host filtrado, dual-homed bastion (conectado por uma placa à rede interna e por outra à rede externa)** impede fisicamente essa brecha de segurança (Figura 20.2b). As vantagens das duas camadas de segurança da configuração anterior também estão presentes aqui. Novamente, um servidor de informações ou outros hosts podem ter comunicação direta com o roteador se este estiver em conformidade com a política de segurança.

A configuração **firewall com sub-rede filtrada** da Figura 20.2c é a mais segura das que consideramos. Nessa configuração, são usados dois roteadores de filtragem de pacotes, um entre o bastion host e a Internet e um entre o bastion host e a rede interna. Essa configuração cria uma sub-rede isolada, que pode consistir simplesmente no bastion host, mas também pode incluir um ou mais servidores de informação e modems para recursos de discagem. Normalmente, tanto a Internet quanto a rede interna têm acesso aos hosts na sub-rede filtrada, mas o tráfego que atravessa a sub-rede filtrada é bloqueado. Essa configuração oferece diversas vantagens:

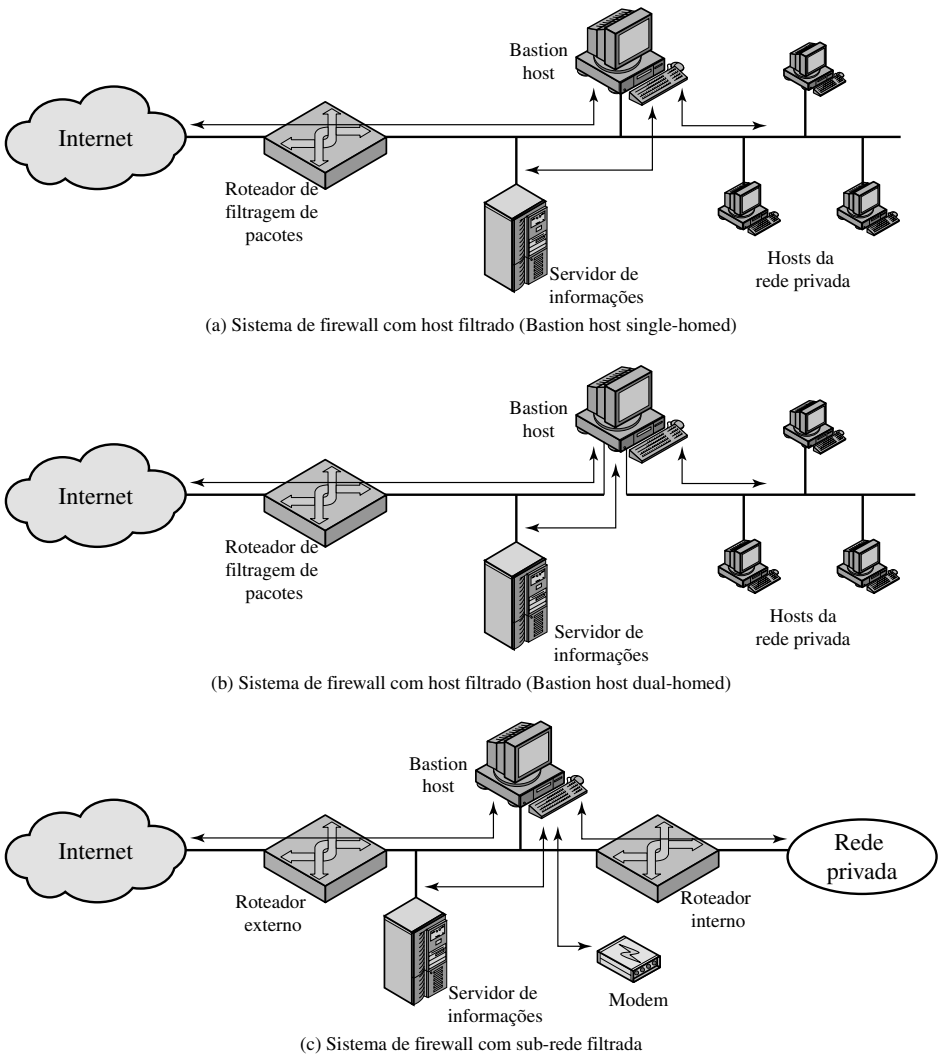


Figura 20.2 Configurações de firewall.

- Agora existem três níveis de defesa para impedir intrusos.
- O roteador externo anuncia apenas a existência da sub-rede filtrada à Internet; portanto, a rede interna fica invisível à Internet.
- De modo semelhante, o roteador interno anuncia apenas a existência da sub-rede filtrada à rede interna; portanto, os sistemas na rede interna não podem criar rotas diretas para a Internet.

20.2 SISTEMAS CONFIÁVEIS (TRUSTED SYSTEMS)

Uma maneira de aprimorar a capacidade de um sistema de defender-se contra intrusos e programas maliciosos é implementar a tecnologia de sistema confiável. Esta seção oferece uma breve idéia desse assunto. Começamos examinando alguns conceitos básicos do controle de acesso a dados.

Controle de acesso a dados

Após o logon bem-sucedido, o usuário recebe acesso a um host ou a um conjunto de hosts e aplicações. Isso geralmente não é suficiente para um sistema que inclui dados importantes em seu banco de dados. Por maio do procedimento de controle de acesso, um usuário pode ser identificado no sistema. Associado a cada usuário, pode existir um perfil que especifica operações e acessos de arquivos permissíveis. O sistema operacional pode, então, impor regras com base no perfil do usuário. Porém, o sistema de gerenciamento de banco de dados precisa controlar o acesso a registros específicos ou até mesmo a partes de registros. Por exemplo, pode ser permitido que qualquer um na adminis-

tração obtenha uma lista do pessoal da empresa, mas somente indivíduos selecionados podem ter acesso a informações de salário. A questão é mais do que apenas o nível de detalhes. Enquanto o sistema operacional pode conceder a um usuário permissão para acessar um arquivo ou usar uma aplicação, além do que não existem mais verificações de segurança, o sistema de gerenciamento de banco de dados precisa tomar uma decisão sobre cada tentativa de acesso individual. Essa decisão dependerá não apenas da identidade do usuário, mas também das partes específicas dos dados sendo acessados e até mesmo das informações já divulgadas ao usuário.

Um modelo geral de controle de acesso, como o exercido por um sistema de gerenciamento de arquivos ou banco de dados, é o de uma matriz de acesso (Figura 20.3a). Os elementos básicos do modelo são os seguintes:

- **Sujeito:** uma entidade capaz de acessar objetos. Geralmente, o conceito de sujeito é igual ao de processo. Qualquer usuário ou aplicação realmente obtém acesso a um objeto por meio de um processo que representa esse usuário ou aplicação.
- **Objeto:** qualquer coisa à qual o acesso seja controlado. Alguns exemplos incluem arquivos, partes de arquivos, programas e segmentos de memória.
- **Direito de acesso:** o modo com que um objeto é acessado por um sujeito. Alguns exemplos são ler, escrever e executar.

Um eixo da matriz consiste em sujeitos identificados que podem tentar o acesso aos dados. Normalmente, essa lista consistirá em usuários individuais ou grupos de usuários, embora o acesso possa ser controlado para terminais, hosts ou aplicações, em vez de ou além dos usuários. O outro eixo lista os objetos que podem ser acessados. Em um nível maior de detalhes, os objetos podem ser campos de dados individuais. Agrupamentos mais agregados, como registros, arquivos ou ainda o banco de dados inteiro, também podem ser objetos na matriz. Cada entrada na matriz indica os direitos de acesso desse sujeito a esse objeto.

Na prática, uma matriz de acesso normalmente é esparsa e é implementada pela decomposição em uma dentre duas maneiras. A matriz pode ser decomposta por colunas, gerando **listas de controle de acesso** (ACL— Access Control Lists)(Figura 20.3b). Assim, para cada objeto, uma lista de controle de acesso lista os usuários e seus direitos de acesso permitidos. A lista de controle de acesso pode conter uma entrada-padrão ou pública. Isso permite que os usuários que não estão listados explicitamente como tendo direitos especiais tenham um conjunto-padrão de direitos. Os elementos da lista podem incluir usuários individuais e também grupos de usuários.

	Programa1	**. . .**	**SegmentoA**	**SegmentoB**
Processo1	Ler Executar		Ler Escrever	
Processo2				Ler
. . .				

(a) Matriz de acesso

Lista de controle de acesso para Programa1: Processo1 (Ler, Executar)
Lista de controle de acesso para SegmentoA: Processo1 (Ler, Escrever)
Lista de controle de acesso para SegmentoB: Processo2 (Ler)

(b) Lista de controle de acesso

Lista de capacidades para Processo1: Programa1 (Ler, Executar) SegmentoA (Ler, Escrever)
Lista de capacidades para Processo2: SegmentoB (Ler)

(c) Lista de capacidades

Figura 20.3 Estrutura do controle de acesso.

A decomposição por linhas gera **tickets de capacidade** (Figura 20.3c). Um ticket de capacidade especifica objetos e operações autorizadas para um usuário. Cada usuário tem um número de tickets e pode ser autorizado a emprestá-los ou a oferecê-los a outros. Como os tickets podem estar dispersos pelo sistema, eles apresentam um problema maior de segurança do que as listas de controle de acesso. Particularmente, o ticket precisa ser impossível de ser falsificado. Um modo de conseguir isso é fazendo com que o sistema operacional mantenha todos os tickets em nome dos usuários. Esses tickets teriam de ser mantidos em uma região da memória inacessível aos usuários.

O conceito de sistemas confiáveis

Grande parte do que explicamos até aqui tem sido referente à proteção de determinada mensagem ou item contra ataques passivos ou ativos por determinado usuário. Um requisito um pouco diferente, mas bastante aplicável, é proteger dados ou recursos com base nos níveis de segurança. Isso normalmente é encontrado nos sistemas militares, onde as informações são categorizadas como não classificada (U – unclassified), confidencial (C), secreta (S), altamente secreta (TS – top secret) ou além disso. Esse conceito é igualmente aplicável em outras áreas, onde as informações podem ser organizadas em categorias gerais e os usuários podem ter permissões concedidas para acessar determinadas categorias de dados. Por exemplo, o nível mais alto de segurança pode ser referente a documentos e dados de planejamento estratégico corporativo, acessíveis apenas aos diretores corporativos e seu pessoal; em seguida, podem ser os dados financeiros e de pessoal, acessíveis apenas ao pessoal da administração, diretores etc.

Quando várias categorias ou níveis de dados são definidos, o requisito é chamado de **segurança multinível**. A declaração geral do requisito para a segurança multinível é que um sujeito em um nível alto não pode levar informações a um sujeito em um nível mais baixo ou não comparável, a menos que esse fluxo reflita com precisão a vontade de um usuário autorizado. Para propósitos de implementação, esse requisito é executado em duas partes e é simplesmente declarado. Um sistema de segurança multinível precisa impor o seguinte:

- **Nenhuma leitura acima (no read up):** Um sujeito só pode ler um objeto que tenha um nível de segurança inferior ou igual ao seu. Isso é conhecido na literatura como **Propriedade de Segurança Simples (Simple Security Property)**.
- **Nenhuma escrita abaixo (no write down):** Um sujeito só pode escrever em um objeto que tenha um nível de segurança maior ou igual ao seu. Isso é conhecido na literatura como **Propriedade-***[1] (pronuncia-se *propriedade estrela*).

Essas duas regras, se impostas corretamente, oferecem segurança multinível. Para um sistema de processamento de dados, a técnica usada, e que tem sido objeto de muita pesquisa e desenvolvimento, é baseada no conceito de *monitor de referência*. Essa técnica é representada na Figura 20.4. O monitor de referência é um elemento de controle no hardware e sistema operacional de um computador, que regula o acesso de sujeitos a objetos com base em parâmetros de segurança do sujeito e do objeto. O monitor de referência tem acesso a um arquivo, conhecido como *banco de dados do kernel de segurança*, que lista os privilégios de acesso (autorização de segurança) de cada objeto e atributos de proteção (nível de classificação) de cada objeto. O monitor de referência impõe as regras de segurança (nenhuma leitura acima, nenhuma escrita abaixo) e tem as seguintes propriedades:

- **Mediação completa:** As regras de segurança são impostas em cada acesso, não apenas, por exemplo, quando um arquivo é aberto.
- **Isolamento:** O monitor de referência e banco de dados são protegidos contra modificação não autorizada.
- **Verificabilidade:** A exatidão do monitor de referência precisa ser demonstrável. Ou seja, deverá ser possível demonstrar matematicamente que o monitor de referência impõe as regras de segurança e oferece mediação e isolamento completos.

Estes são requisitos rígidos. O requisito para mediação completa significa que cada acesso a dados dentro da memória principal, no disco e fita precisa ser mediado. Implementações de software puras impõem uma exigência de desempenho muito alta na prática; a solução precisa estar, pelo menos parcialmente, no hardware. O requisito para isolamento significa que não deverá ser possível que um atacante, não importa quão inteligente, altere a lógica do monitor de referência ou o conteúdo do banco de dados do kernel de segurança. Finalmente, o requisito para prova matemática é formidável para algo tão complexo quanto um computador de uso geral. Um sistema que pode provar tal verificação é conhecido como um **sistema confiável (trusted system)**.

Um elemento final ilustrado na Figura 20.4 é um arquivo de auditoria. Eventos de segurança importantes, como violações de segurança detectadas e alterações autorizadas no banco de dados do kernel de segurança, são armazenadas no arquivo de auditoria.

1. O "*" não tem significado algum. Ninguém conseguiu pensar em um nome apropriado para a propriedade durante a criação do primeiro relatório sobre o modelo. O asterisco foi um caractere fictício inserido no rascunho para que o revisor do texto pudesse rapidamente encontrar e substituir todas as ocorrências de seu uso quando a propriedade tivesse um nome. Nenhum nome foi criado, e por isso o relatório foi publicado com o "*" intacto.

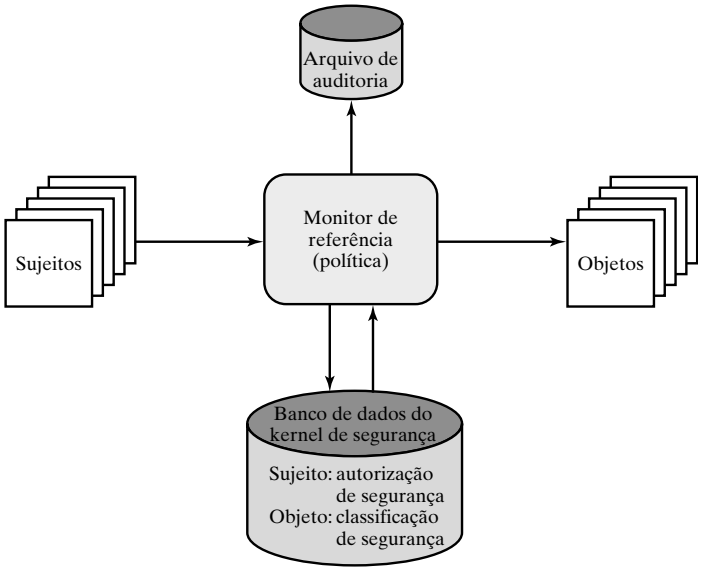


Figura 20.4 Conceito de monitor de referência.

Em um esforço para atender às suas próprias necessidades e como um serviço ao público, o Departamento de Defesa dos EUA, em 1981, estabeleceu o Computer Security Center dentro da National Security Agency (NSA) com o objetivo de encorajar a disponibilidade geral de sistemas de computador confiáveis. Esse objetivo é realizado por meio do centro do Programa de Avaliação de Produtos Comerciais (Commercial Product Evaluation Program). Basicamente, o centro tenta avaliar os produtos disponíveis comercialmente como obedecendo aos requisitos de segurança recém-esboçados. O centro classifica os produtos avaliados de acordo com a gama de recursos de segurança que eles oferecem. Essas avaliações são necessárias para as aquisições do Departamento de Defesa dos EUA, mas são publicadas e estão disponíveis gratuitamente. Logo, elas podem servir como orientação para os clientes comerciais na compra de equipamentos disponíveis comercialmente.

Defesa contra Cavalo-de-tróia

Uma forma de proteger-se contra ataques de Cavalo-de-tróia é o uso de um sistema operacional seguro, confiável. A Figura 20.5 ilustra um exemplo. Nesse caso, um Cavalo-de-tróia é usado para fugir do mecanismo de segurança padrão usado pela maioria dos sistemas de gerenciamento de arquivos e sistemas operacionais: a lista de controle de acesso. Neste exemplo, um usuário chamado Bob interage por meio de um programa com um arquivo de dados contendo a string de caracteres importante "CPE170KS". O usuário Bob criou o arquivo com permissão de leitura/escrita fornecida apenas a programas sendo executados em seu próprio nome: ou seja, somente processos que pertencem a Bob podem acessar o arquivo.

O ataque de cavalo-de-Tróia começa quando uma usuária hostil, chamada Alice, obtém acesso legítimo ao sistema e instala tanto um programa de cavalo-de-Tróia quanto um arquivo particular a ser usado no ataque como um "bolso-traseiro" (back-pocket). Alice concede permissão de leitura/escrita a si mesma para esse arquivo e concede permissão apenas de escrita a Bob (Figura 20.5a). Alice agora induz Bob a chamar o programa cavalo-de-Tróia, talvez anunciando-o como um utilitário. Quando o programa detecta que está sendo executado por Bob, ele lê a string de caracteres importante do arquivo de Bob e a copia no arquivo de bolso traseiro de Alice (Figura 20.5b). As operações de leitura e escrita satisfazem as restrições impostas pelas listas de controle de acesso. Alice, então, só precisa acessar o arquivo de Bob em outra oportunidade para descobrir o valor da string.

Agora, considere o uso de um sistema operacional seguro nesse cenário (Figura 20.5c). Os níveis de segurança são atribuídos aos sujeitos no logon com base em critérios como o terminal a partir do qual o computador está sendo acessado e o usuário envolvido, conforme identificado pela senha/ID. Nesse exemplo, existem dois níveis de segurança, restrito e público, ordenados de modo que o restrito tenha nível mais alto que o público. Os processos pertencentes a Bob e o arquivo de dados de Bob recebem o nível de segurança restrito; o arquivo e os processos de Alice, público. Se Bob chamar o programa Cavalo-de-tróia (Figura 20.5d), esse programa obterá o nível de segurança de Bob. Portanto, ele será capaz, sob a propriedade de segurança simples, de ler a string de caracteres sensível. Contudo, quando o programa tentar armazenar a string em um arquivo público (o arquivo de bolso traseiro), a Propriedade-* será violada e a tentativa será impedida pelo monitor de referência. Assim, a tentativa de escrever no arquivo de bolso traseiro é negada, embora a lista do controle de acesso o permita: a política de segurança tem precedência sobre o mecanismo da lista de controle de acesso.

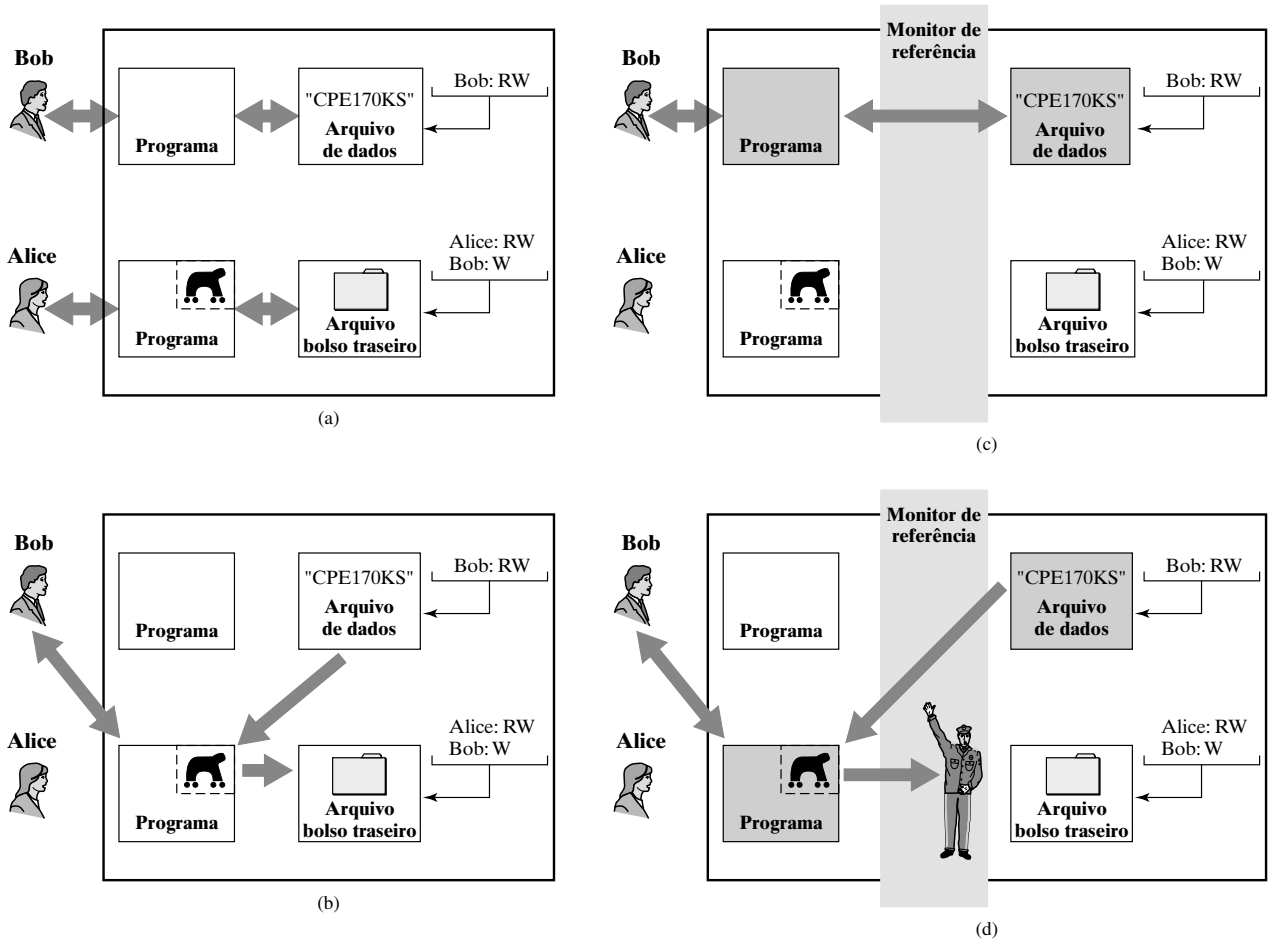


Figura 20.5 Cavalo-de-Tróia e sistema operacional seguro.

20.3 CRITÉRIOS COMUNS PARA AVALIAÇÃO DE SEGURANÇA DA TECNOLOGIA DA INFORMAÇÃO

O trabalho feito pela Agência de Segurança Nacional (NSA) e outras agências do governo dos EUA para desenvolver requisitos e critérios de avaliação para sistemas confiáveis tem paralelo em outros países. O Common Criteria (CC) for Information Technology and Security Evaluation é uma iniciativa internacional das agências de padrões em diversos países para desenvolver padrões internacionais para especificar requisitos de segurança e definir critérios de avaliação.

Requisitos

A CC define um conjunto comum de requisitos de segurança em potencial para uso na avaliação. O termo **alvo de avaliação (target of evaluation** – TOE) refere-se àquela parte do produto ou sistema sujeita à avaliação. Os requisitos podem estar em uma destas duas categorias:

- **Requisitos funcionais:** Definem o comportamento de segurança desejado. Documentos CC estabelecem um conjunto de componentes funcionais de segurança que oferece uma forma-padrão de expressar os requisitos funcionais de segurança para um TOE.
- **Requisitos de garantia:** A base para se assegurar de que as medidas de segurança afirmadas são eficazes e foram implementadas corretamente. Os documentos CC estabelecem um conjunto de componentes de segurança que oferecem um modo-padrão de expressar os requisitos de garantia para um TOE.

Tanto os requisitos funcionais quanto os requisitos de garantia são organizados em classes. Uma **classe** é um conjunto de requisitos que compartilham um foco ou intenção comum. As Tabelas 20.3 e 20.4 definem resumidamente as classes de requisitos funcionais e de garantia. Cada uma dessas classes contém diversas famílias. Os requisitos dentro de cada **família** compartilham objetivos de segurança, mas diferem em ênfase ou rigor. Por exemplo, a classe de audi-

toria contém seis famílias lidando com diversos aspectos da auditoria (por exemplo, geração de dados de auditoria, análise de auditoria e armazenamento de evento de auditoria). Cada família, por sua vez, contém um ou mais componentes. Um **componente** descreve um conjunto específico de requisitos de segurança e é o menor conjunto selecionável de requisitos de segurança para inclusão nas estruturas definidas na CC.

Por exemplo, a classe de suporte criptográfico dos requisitos funcionais inclui duas famílias: gerenciamento de chaves criptográficas e operação criptográfica. Existem quatro componentes sob a família de gerenciamento de chaves criptográficas usados para especificar: o algoritmo de geração de chaves e tamanho da chave; o método de distribuição de chaves; o método de acesso à chave e o método de destruição de chaves. Para cada componente, um padrão pode ser referenciado para definir o requisito. Sob a família de operação criptográfica, existe um único componente, que especifica um algoritmo e o tamanho da chave com base em um padrão atribuído.

Conjuntos de componentes funcionais e de garantia podem ser agrupados em pacotes reutilizáveis, reconhecidos como úteis para reunir objetivos identificados. Um exemplo desse pacote seriam os componentes funcionais exigidos para controles de acesso discricionário (sem restrições) (discretionary access controls).

Perfis e alvos

A iniciativa CC também define dois tipos de documentos que podem ser gerados usando os requisitos definidos pela CC.

- **Perfis de proteção (PPs)**: Definem um conjunto independente de implementação com requisitos e objetivos de segurança para uma categoria de produtos ou sistemas que atendem a necessidades semelhantes do consumidor para segurança de TI. Um PP deverá ser reutilizável e definir requisitos conhecidos como sendo úteis e eficazes para atender aos objetivos identificados. O conceito de PP tem sido desenvolvido para oferecer suporte à definição de padrões funcionais e como um auxílio para formular especificações de aquisição. O PP reflete os requisitos de segurança do usuário.

Tabela 20.3 Requisitos funcionais de segurança do CC

Classe	Descrição
Auditoria	Envolve reconhecimento, registro, armazenamento e análise de informações relacionadas a atividades de segurança. Registros de auditoria são produzidos por essas atividades e podem ser examinados para determinar sua relevância quanto à segurança.
Suporte criptográfico	Usado quando o TOE implementa funções criptográficas. Estas podem ser usadas, por exemplo, para oferecer suporte às comunicações, identificação e autenticação ou separação de dados.
Comunicações	Oferece duas famílias que tratam da irretratabilidade do originador e do destinatário dos dados.
Proteção de dados do usuário	Especificação de requisitos relacionados à proteção de dados do usuário dentro do TOE durante a importação, exportação e armazenamento, além dos atributos de segurança relacionados aos dados do usuário.
Identificação e autenticação	Garante a identificação não ambígua dos usuários autorizados e a correta associação dos atributos de segurança com usuários e sujeitos.
Gerenciamento de segurança	Especifica o gerenciamento de atributos, dados e funções de segurança.
Privacidade	Oferece a um usuário proteção contra descoberta e mau uso de sua identidade por outros usuários.
Proteção das funções TOE	Focaliza a proteção de dados do TSF (funções de segu-de segurança do rança do TOE), em vez dos dados do usuário. A classe se relaciona à integridade e ao gerenciamento dos mecanismos e dados do TSF.
Utilização de recursos	Admite a disponibilidade dos recursos exigidos, como capacidade de processamento e capacidade de armazenamento. Inclui requisitos para tolerância a falhas, prioridade de serviço e alocação de recursos.
Acesso ao TOE	Especifica requisitos funcionais, além daqueles especificados para identificação e autenticação, para controlar o estabelecimento da sessão de um usuário. Os requisitos para acesso ao TOE controlam aspectos como limitar o número e o escopo das sessões do usuário, exibir o histórico de acesso e a modificação dos parâmetros de acesso.
Caminho/canais confiáveis	Trata dos caminhos de comunicações confiáveis entre os usuários e o TSF, e entre TSFs.

Tabela 20.4 Requisitos de garantia de segurança do CC.

Classe	Descrição
Gerenciamento de configurações	Exige que a integridade do TOE seja adequadamente preservada. Especificamente, o gerenciamento de configurações oferece confiança de que o TOE e a documentação utilizada para avaliação são aqueles preparados para distribuição.
Entrega e operação	Trata das medidas, procedimentos e padrões para remessa segura, instalação e uso operacional do TOE, para garantir que a proteção de segurança oferecida pelo TOE não seja comprometida durante esses eventos.
Desenvolvimento	Trata do aprimoramento do TSF a partir da especificação definida no ST para a implementação, e um mapeamento dos requisitos de segurança para a representação de nível mais baixo.
Documentos de orientação	Trata do uso operacional seguro do TOE pelos usuários e administradores.
Suporte do ciclo de vida	Trata do ciclo de vida do TOE e inclui a definição do ciclo de vida, ferramentas e técnicas, segurança do ambiente de desenvolvimento e acerto das falhas encontradas pelos consumidores do TOE.
Testes	Trata da demonstração de que o TOE atende a seus requisitos funcionais. As famílias tratam da cobertura e da profundidade do teste do desenvolvedor, e requisitos para testes independentes.
Avaliação da vulnerabilidade	Define os requisitos voltados para vulnerabilidades exploráveis, que poderiam ser introduzidas pela construção, operação, mau uso ou configuração incorreta do TOE. As famílias identificadas aqui tratam da identificação de vulnerabilidades por meio de análise de canal oculto, análise da configuração do TOE, exame da força dos mecanismos das funções de segurança e identificação de falhas introduzidas durante o desenvolvimento do TOE. A segunda família abrange a categorização de segurança dos componentes do TOE. A terceira e quarta incluem a análise de mudanças para impacto na segurança e a provisão de evidência de que os procedimentos estão sendo seguidos. Esta classe oferece os elementos básicos para o estabelecimento de esquemas de manutenção da garantia.
Manutenção da garantia	Oferece requisitos cuja intenção é que sejam aplicados após um TOE ter sido certificado em relação à iniciativa CC. Esses requisitos visam garantir que o TOE continuará a atender a seu objetivo de segurança enquanto são feitas mudanças no TOE ou em seu ambiente.

- **Alvos de segurança (STs – Security targets):** Contêm os objetivos e requisitos de segurança da TI de um TOE específico identificado e define as medidas funcionais e de garantia oferecidas por esse TOE para atender aos requisitos declarados. O ST pode afirmar conformidade com um ou mais PPs, e isso forma a base para uma avaliação. O ST é fornecido por um fornecedor ou desenvolvedor.

A Figura 20.6 ilustra o relacionamento entre os requisitos de um lado e os perfis e alvos no outro. Para um PP, um usuário pode selecionar diversos componentes para definir os requisitos para o produto desejado. O usuário também pode se referir a pacotes predefinidos que montam uma série de requisitos normalmente agrupados juntos, dentro de um documento de requisitos de produto. De modo semelhante, um fornecedor ou projetista pode selecionar diversos componentes e pacotes para definir um ST.

A Figura 20.7 mostra o que é conhecido nos documentos CC como paradigma dos requisitos funcionais de segurança. Basicamente, essa ilustração é baseada no conceito de monitor de referência, mas utiliza a terminologia e filosofia de projeto do CC.

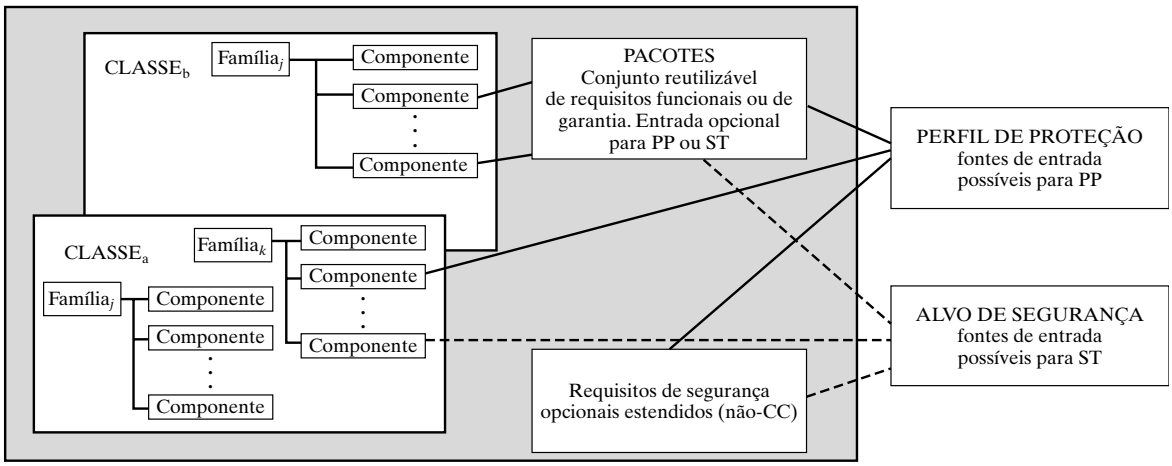


Figura 20.6 Organização e construção de requisitos dos critérios comuns.

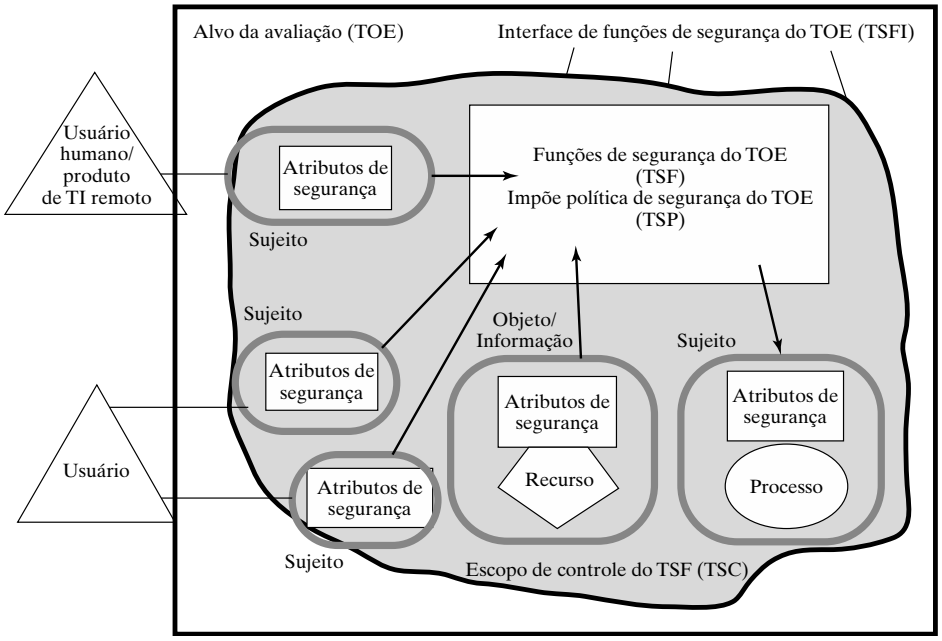


Figura 20.7 Paradigma dos requisitos funcionais de segurança.

20.4 LEITURA E SITES WEB RECOMENDADOS

Uma abordagem clássica dos firewalls é [CHAP00]. Outro clássico, recentemente atualizado, é [CHES03]. [LODI98], [OPPL97] e [BELL94b] são bons artigos gerais sobre o assunto. [WACK02] é uma excelente visão geral da tecnologia de firewall e políticas de firewall. [AUDI04] e [WILS05] oferecem explicações úteis sobre firewalls.

[GASS88] traz um estudo abrangente de sistemas de computador confiáveis. [PFLE03] e [GOLL99] também abordam o assunto. [FELT03] e [OPPL05] oferecem explicações úteis sobre computação confiável.

AUDI04 Audin, G. "Next-gen firewalls: what to expect". *Business Communications Review*, junho de 2004.

BELL94b Bellovin, S., and Cheswick, W. "Network firewalls". *IEEE Communications Magazine*, setembro de 1994.

CHAP00 Chapman, D. e Zwicky, E. *Building internet firewalls*. Sebastopol: O'Reilly, 2000.

CHES03 Cheswick, W. e Bellovin, S. *Firewalls and Internet security: repelling the wily hacker*. Reading: Addison-Wesley, 2003.

FELT03 Felten, E. "Understanding trusted computing: will its benefits outweigh its drawbacks?" *IEEE Security and Privacy*, maio/junho de 2003.

GASS88 Gasser, M. *Building a secure computer system*. Nova York: Van Nostrand Reinhold, 1988.

GOLL99 Gollmann, D. *Computer security*. Nova York: Wiley, 1999.

LODI98 Lodin, S. e Schuba, C. "Firewalls fend off invasions from the Net". *IEEE Spectrum*, fevereiro de 1998.

OPPL97 Oppliger, R. "Internet security: firewalls and beyond". *Communications of the ACM*, maio de 1997.

OPPL05 Oppliger, R. e Rytz, R. "Does trusted computing remedy computer security problems?" *IEEE Security and Privacy*, março/abril de 2005.

PFLE03 Pfleeger, C. *Security in Computing*. Upper Saddle River: Prentice Hall, 1997.

WACK02 Wack, J.; Cutler, K. e Pole, J. *Guidelines on firewalls and firewall policy*. NIST Special Publication SP 800-41, janeiro de 2002.

WILS05 Wilson, J. "The future of the firewall". *Business Communications Review*, maio de 2005.

Sites Web recomendados:

- **Firewall.com:** Diversos links para referências de firewall e recursos de software.
- **Trusted Computing Group:** Grupo de fornecedores envolvidos no desenvolvimento e promoção de padrões para computação confiável. O site inclui artigos, especificações e links para fornecedores.
- **Common Criteria Portal:** Site Web oficial do projeto Common Criteria.

20.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

Common Criteria (CC)
direito de acesso
firewall
firewall de inspeção com estado
gateway em nível de aplicação
gateway em nível de circuito
bastion host
lista de controle de acesso (ACL)
matriz de acesso
monitor de referência
objeto
roteador de filtragem de pacote
segurança multinível
sistema confiável
sujeito
ticket de capacidade

Perguntas de revisão

20.1 Liste três objetivos de projeto para um firewall.
20.2 Liste quatro técnicas usadas pelos firewalls para controlar o acesso e impor uma política de segurança.
20.3 Que informações são usadas por um roteador de filtragem de pacotes típico?
20.4 Quais são alguns pontos fracos de um roteador de filtragem de pacotes?
20.5 Qual é a diferença entre um roteador de filtragem de pacotes e um firewall de inspeção com estado?
20.6 O que é um gateway em nível de aplicação?
20.7 O que é um gateway em nível de circuito?
20.8 Quais são as diferenças entre as três configurações da Figura 20.2?
20.9 No contexto do controle de acesso, qual é a diferença entre um sujeito e um objeto?
20.10 Qual é a diferença entre uma lista de controle de acesso e um ticket de capacidade?
20.11 Quais são as duas regras que um monitor de referência impõe?
20.12 Que propriedades são exigidas de um monitor de referência?
20.13 Quais são os critérios comuns (common criteria)?

Problemas

20.1 Conforme mencionamos na Seção 20.1, uma técnica para impedir o ataque de fragmento pequeno é impor um tamanho mínimo do cabeçalho de transporte que precisa estar contido no primeiro fragmento de um pacote IP. Se o primeiro fragmento for rejeitado, todos os fragmentos subseqüentes

podem ser rejeitados. Porém, a natureza do IP é tal que os fragmentos podem chegar fora de ordem. Assim, um fragmento intermediário pode passar pelo filtro antes que o fragmento inicial seja rejeitado. Como essa situação pode ser resolvida?

20.2 Em um pacote IPv4, o tamanho do payload no primeiro fragmento, em octetos, é igual ao Tamanho Total – (4 X IHL). Se esse valor for menor que o mínimo exigido (8 octetos para TCP), então esse fragmento e o pacote inteiro podem ser rejeitados. Sugira um método alternativo para conseguir o mesmo resultado usando apenas o campo Deslocamento de Fragmento (Fragment Offset).

20.3 A RFC 791, a especificação do protocolo IPv4, descreve um algoritmo de remontagem que resulta em novos fragmentos sobrescrevendo quaisquer partes sobrepostas dos fragmentos recebidos anteriormente. Dada tal implementação de remontagem, um atacante poderia criar uma série de pacotes em que o fragmento mais baixo (deslocamento zero) tivesse dados inofensivos (e, portanto, passando por filtros administrativos de pacotes) e no qual algum pacote subseqüente, tendo um deslocamento diferente de zero, sobreporia as informações do cabeçalho TCP (porta de destino, por exemplo) e este seria modificado. O segundo pacote seria passado através da maioria das implementações de filtro, pois não tem um deslocamento de fragmento zero. Sugira um método que possa ser usado por um filtro de pacotes para impedir esse ataque.

20.4 A necessidade da regra "nenhuma leitura acima" para um sistema de segurança multinível é bastante óbvia. Qual é a importância da regra "nenhuma escrita abaixo"?

20.5 Na Figura 20.5, um elo da cadeia copiar-e-observar-depois do cavalo-de-Tróia está partido. Existem dois outros ângulos de ataque possíveis por Alice: Alice efetuando logon e tentando ler a string diretamente e Alice atribuindo um nível de segurança restrito ao arquivo de bolso traseiro. O monitor de referência impede esses ataques?

APÊNDICE A

PADRÕES E ORGANIZAÇÕES DE DEFINIÇÃO DE PADRÕES

A.1 **A importância dos padrões**

A.2 **Padrões da Internet e a Internet Society**

A.3 **National Institute of Standards and Technology**

Existem alguns cães que não depreciariam o que para eles são formas sagradas. Um pastor alemão muito bom, muito sério, com que trabalhei, por exemplo, rosnava alto para outros cães quando eles não obedeciam. Quando o treinava para pegar coisas, eu coloquei um brinquedo em formato de halteres de pé por brincadeira. Ele olhou fixamente para o brinquedo e para mim, desaprovando, e depois o empurrou cuidadosamente de volta à sua posição correta antes de apanhá-lo e retornar com ele, mal-humorado.

– Adam's task: calling animals by name, Vicki Hearne

Um conceito importante que se repete freqüentemente neste livro é o de padrões. Este apêndice oferece alguma base sobre a natureza e a relevância dos padrões e examina as principais organizações envolvidas no desenvolvimento de padrões para redes e comunicações.

A.1 A IMPORTÂNCIA DOS PADRÕES

Há muito tem sido aceito no setor de telecomunicações que os padrões são obrigatórios para controlar as características físicas, elétricas e de procedimento dos equipamentos de comunicação. No passado, essa visão não foi adotada pela indústria de computadores. Embora os fornecedores de equipamentos de comunicação reconheçam que seu equipamento geralmente irá se conectar e se comunicar com o equipamento de outros fornecedores, os fornecedores de computador tradicionalmente têm tentado monopolizar seus clientes. A proliferação de computadores e o processamento distribuído têm tornado essa posição insustentável. Computadores de diferentes fornecedores precisam se comunicar entre si e, com a evolução contínua dos padrões de protocolo, os clientes não aceitarão mais o desenvolvimento de software especial de conversão de protocolo. O resultado é que os padrões agora permeiam todas as áreas da tecnologia explicadas neste livro.

Existem diversas vantagens e desvantagens no processo de criação de padrões. As principais vantagens dos padrões são as seguintes:

- Um padrão garante que haverá um grande mercado para determinado equipamento ou software. Isso encoraja a produção em massa e, em alguns casos, o uso de técnicas de integração em larga escala (LSI) ou integração em escala muito grande (VLSI), resultando em menores custos.
- Um padrão permite que produtos de vários fornecedores se comuniquem, dando ao comprador mais flexibilidade na seleção e uso do equipamento.

As principais desvantagens dos padrões são as seguintes:

- Um padrão tende a congelar a tecnologia. Até que um padrão seja desenvolvido, sujeito à revisão e acordo, e promulgado, técnicas mais eficientes são possíveis.
- Existem muitos padrões para a mesma coisa. Essa não é uma desvantagem dos padrões por si só, mas do modo atual como as coisas são feitas. Felizmente, nos últimos anos, as diversas organizações de criação de padrões começaram a cooperar mais. Apesar disso, ainda existem áreas em que existem vários padrões em conflito.

A.2 PADRÕES DA INTERNET E A INTERNET SOCIETY

Muitos dos protocolos que compõem o conjunto de protocolos TCP/IP foram padronizados ou estão no processo de padronização. Por acordo universal, uma organização conhecida como Internet Society é responsável pelo desenvolvimento e publicação desses padrões. A Internet Society é uma organização de membros profissionais que supervisiona uma série de comitês e forças-tarefa envolvidos no desenvolvimento e na padronização da Internet.

Esta seção oferece uma rápida descrição do modo como os padrões para o conjunto de protocolos TCP/IP são desenvolvidos.

As organizações da Internet e publicação de RFCs

A Internet Society é o comitê coordenador para projeto, engenharia e gerenciamento da Internet. As áreas cobertas incluem a operação da própria Internet e a padronização de protocolos usados pelos sistemas finais na Internet para interoperabilidade. Três organizações sob a Internet Society são responsáveis pelo trabalho real de desenvolvimento e publicação de padrões:

- **Internet Architecture Board (IAB):** Responsável por definir a arquitetura geral da Internet, oferecendo orientação e direção ampla para o IETF
- **Internet Engineering Task Force (IETF):** O braço de engenharia e desenvolvimento de protocolos da Internet
- **Internet Engineering Steering Group (IESG):** Responsável pelo gerenciamento técnico das atividades do IETF e do processo de padrões da Internet

Os grupos de trabalho apoiados pelo IETF executam o desenvolvimento real de novos padrões e protocolos para a Internet. A entrada como membro em um grupo de trabalho é voluntária; qualquer interessado pode participar. Durante o desenvolvimento de uma especificação, um grupo de trabalho criará uma versão de rascunho do documento, disponível como um draft de Internet, que é colocado no diretório on-line "Internet Drafts" do IETF. O documento pode permanecer como um draft de Internet por até seis meses, e as partes interessadas podem rever e comentar o rascunho. Durante esse tempo, o IESG pode aprovar a publicação do rascunho como uma RFC (Request for Comment). Se o rascunho não tiver passado para o status de uma RFC durante o período de seis meses, ele é retirado do diretório. O grupo de trabalho pode posteriormente publicar uma versão revisada do rascunho.

O IETF é responsável por publicar as RFCs, com a aprovação do IESG. As RFCs são as notas de trabalho da comunidade de pesquisa e desenvolvimento da Internet. Um documento nessa série pode tratar de praticamente qualquer assunto relacionado a comunicações de computador, e pode ser qualquer coisa desde um relatório de reunião até a especificação de um padrão.

O trabalho do IETF é dividido em oito áreas, cada uma com um diretor de área e cada uma composta de diversos grupos de trabalho. A Tabela A.1 mostra as áreas do IETF e seu foco.

O processo de padronização

Área do IETF	Tema	Exemplos de grupos de trabalho
Geral	Processos e procedimentos do IETF	Estrutura política Processo para organização de padrões da Internet
Aplicações	Aplicações da Internet	Protocolos relacionados à Web (HTTP) Integração EDI-Internet LDAP
Internet	Infra-estrutura da Internet	IPv6 Extensões PPP
Operações e gerenciamento	Padrões e definições para operações de rede	SNMPv3 Monitoração de rede remota
Roteamento	Protocolos e gerenciamento para informações de roteamento	Roteamento multicast OSPF Roteamento QoS
Segurança	Protocolos e tecnologias de segurança	Kerberos IPSec X.509 S/MIME TLS
Transporte	Protocolos da camada de transporte	Serviços diferenciados Telefonia IP NFS RSVP
Serviços do usuário	Métodos para melhorar a qualidade as informações disponíveis aos usuários da Internet	Uso responsável da Internet Serviços do usuário Documentos FYI

A decisão de quais RFCs se tornam padrões da Internet é tomada pelo IESG, por recomendação do IETF. Para tornar-se um padrão, uma especificação precisa atender aos seguintes critérios:

Tabela A.1 Áreas do IETF

- Ser estável e bem entendida
-]Ser tecnicamente competente
- Ter várias implementações independentes e interoperáveis, com experiência operacional substancial
- Possuir significativo suporte público
- Ser reconhecidamente útil em alguma ou em todas as partes da Internet

A principal diferença entre esses critérios e aqueles usados para padrões internacionais do ITU é a ênfase na experiência operacional.

O lado esquerdo da Figura A.1 mostra a série de etapas, chamadas de *trilhas-padrão*, pelas quais uma especificação passa para se tornar um padrão; esse processo é definido na RFC 2026. As etapas envolvem o aumento das quantidades de exames detalhados e testes. Em cada etapa, o IETF precisa fazer uma recomendação para avanço do protocolo, e o IESG precisa ratificá-lo. O processo começa quando o IESG aprova a publicação de um documento de draft de Internet como uma RFC com o status de Proposed Standard (padrão proposto).

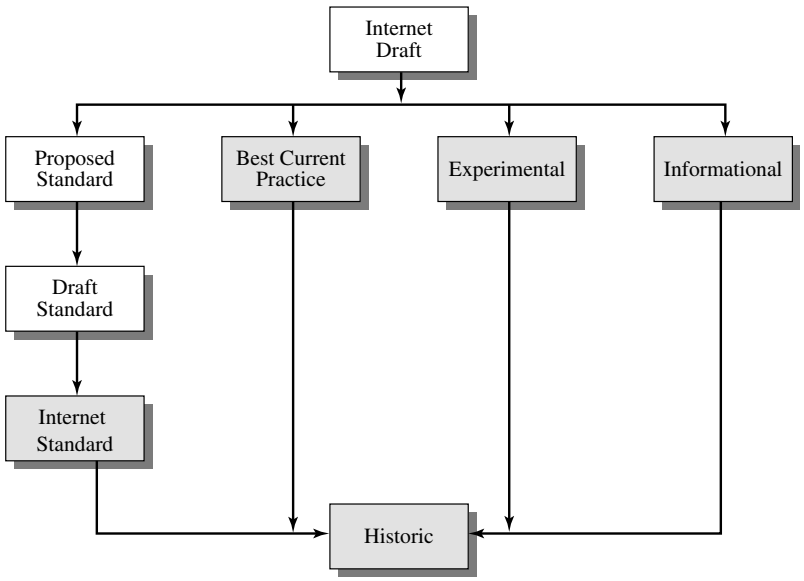


Figura A.1 Processo de publicação de RFC da Internet.

As caixas brancas no diagrama representam estados temporários, que devem ser ocupados pelo tempo prático mínimo. Porém, um documento precisa permanecer como um Proposed Standard pelo menos por seis meses e como Draft Standard por pelo menos quatro meses para permitir tempo para revisão e comentário. As caixas cinza representam estados de longo prazo que podem ser ocupados por anos.

Para uma especificação avançar para o status de Draft Standard, é preciso haver pelo menos duas implementações independentes e interoperáveis das quais foi obtida experiência operacional adequada.

Após implementação significativa e experiência operacional terem sido obtidas, uma especificação pode ser elevada a Internet Standard. Nesse ponto, a especificação recebe um número de STD além de um número de RFC. Finalmente, quando um protocolo se torna obsoleto, ele recebe o estado Historic.

Categorias de padrões da Internet

Todos os padrões da Internet seguem uma dentre duas categorias:

- **Especificação técnica (TS — Technical specification):** Uma TS define um protocolo, serviço, procedimento, convenção ou formato. A maior parte dos padrões da Internet são as TSs.
- **Declaração de aplicabilidade (AS — Applicability statement):** Uma AS especifica como, e sob quais circunstâncias, uma ou mais TSs podem ser aplicadas para dar suporte a determinado recurso da Internet. Uma AS identifica uma ou mais TSs relevantes ao recurso, e pode especificar valores ou intervalos para determinados parâmetros associados a uma TS ou subconjuntos funcionais de uma TS, que são relevantes para a capacidade.

Outros tipos de RFC

Existem várias RFCs que não são destinadas a se tornar padrões da Internet. Algumas RFCs padronizam os resultados de deliberações da comunidade sobre declarações de princípio ou conclusões sobre qual é a melhor maneira de realizar algumas operações ou função de processo do IETF. Essas RFCs são designadas como Best Current Practice (BCP). A aprovação das BCPs segue essencialmente o mesmo processo que a aprovação dos Proposed Standards. Diferentemente dos documentos da trilha-padrão, não existe um processo em três estágios para as BCPs; uma BCP vai do status de draft de Internet para BCP aprovada em uma etapa.

Um protocolo ou outra especificação que não é considerada pronta para padronização pode ser publicada como uma RFC Experimental. Depois de mais trabalho, a especificação pode ser enviada novamente. Se a especificação for estável, tiver resolvido opções de projeto conhecidas, for bem entendida, tiver recebido crítica significativa da comunidade e parecer gozar de interesse suficiente da comunidade para ser considerada valiosa, então a RFC será designada como um Proposed Standard.

Finalmente, uma Informational Specification é publicada para a informação geral da comunidade da Internet.

A.3 NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

O National Institute of Standards and Technology (NIST), parte do Departamento de Comércio dos EUA, emite padrões e orientações para uso pelos departamentos e órgãos do governo dos EUA. Esses padrões e orientações são emitidos na forma de Federal Information Processing Standards (FIPS). O NIST desenvolve FIPS quando existem requisitos exigidos pelo governo federal, como para segurança e interoperabilidade, e não existem padrões ou soluções aceitáveis no setor.

- O NIST anuncia os FIPS propostos no *Registro Federal* para revisão pública e comentários. Ao mesmo tempo em que um FIPS proposto é anunciado no *Registro Federal*, ele também é anunciado no site Web do NIST. O texto e as especificações associadas do FIPS proposto, se aplicáveis, são postadas no site Web do NIST.
- Um período de 90 dias é fornecido para revisão e para envio de comentários sobre o FIPS proposto ao NIST. A data na qual os comentários devem ser enviados ao NIST é especificada no *Registro Federal* e em outros anúncios.
- Comentários recebidos em resposta à nota do *Registro Federal* e a outras notas são revisados pelo NIST para determinar se as modificações ao FIPS proposto são necessárias.
- Um documento de justificativa detalhado é preparado, analisando os comentários recebidos e explicando se foram feitas modificações, ou explicando por que as mudanças recomendadas não foram feitas.
- O NIST envia o FIPS recomendado, o documento de justificativa detalhado, e recomendações quanto a se o padrão deve ser compulsório e vinculado para o uso do governo Federal, à Secretaria do Comércio para aprovação.
- Uma nota anunciando a aprovação do FIPS pela Secretaria do Comércio é publicada no *Registro Federal* e no site Web do NIST.

Embora os padrões do NIST sejam desenvolvidos para uso pelo governo dos EUA, muitos deles são bastante utilizados no setor. AES e DES são os principais exemplos.

APÊNDICE B

PROJETOS PARA ENSINO DE CRIPTOGRAFIA E SEGURANÇA DE REDE

B.1 **Projetos de pesquisa**
B.2 **Projetos de programação**
B.3 **Exercícios de laboratório**
B.4 **Tarefas escritas**
B.5 **Tarefas de leitura/relatório**

Análise e observação, teoria e experiência, nunca devem desprezar ou excluir uma à outra, mas sim se completarem.

— Da guerra, Carl Von Clausewitz

Muitos professores acreditam que os projetos de pesquisa e sua implementação são fundamentais para o claro entendimento da criptografia e da segurança de rede. Sem os projetos, pode ser difícil para os alunos entenderem alguns dos conceitos básicos e interações entre os componentes. Os projetos reforçam os conceitos introduzidos no livro, permitem que o aluno compreenda melhor como um algoritmo criptográfico ou protocolo funciona, e podem ajudar os alunos a terem confiança de que são capazes de não apenas entender, mas implementar os detalhes de um recurso de segurança.

Neste livro, tentei apresentar os conceitos da criptografia e segurança de rede da maneira mais didática possível, fornecendo diversos problemas como 'lição de casa' para reforçar esses conceitos. Porém, muitos professores desejarão complementar os conceitos aqui ensinados com projetos práticos. Este apêndice oferece uma breve orientação em relação a isso e explica mais detalhadamente o material de apoio disponível no site do livro (www.prenhal.com/stallings_br). Esse conteúdo abrange cinco tipos de projetos.

- Projetos de pesquisa
- Projetos de programação
- Exercícios de laboratório
- Tarefas escritas
- Tarefas de leitura/relatório

B.1 PROJETOS DE PESQUISA

Um modo eficaz de reforçar os conceitos básicos do curso e ensinar aos alunos habilidades de pesquisa é atribuir um projeto de pesquisa. Esse projeto pode envolver uma pesquisa literária e também uma busca na Internet pelos produtos de fornecedores, atividades de laboratório de pesquisa e esforços de padronização. Os projetos podem ser atribuídos a grupos de alunos ou propostos para serem desenvolvidos individualmente. De qualquer modo, é melhor exigir algum tipo de proposta de projeto desde cedo no semestre, de modo que o professor tenha tempo para avaliar o grau de dificuldade da proposta e o tempo e o esforço necessários para sua realização. O material a ser apresentado pelo aluno para os projetos de pesquisa deverá incluir:

- Um formato para a proposta
- Um formato para o relatório final
- Um cronograma com prazos intermediário e final
- Uma lista com propostas de temas para o projeto

Os alunos podem selecionar um desses temas ou criarem seu próprio projeto similar. No site do livro ele há um formato sugerido para a proposta e o relatório final, além de uma lista com várias opções de temas de pesquisa.

B.2 PROJETOS DE PROGRAMAÇÃO

O projeto de programação é uma ferramenta pedagógica útil. Existem várias características atraentes em projetos de programação independentes que não fazem parte de uma estrutura de segurança real:

1. O professor pode escolher dentre uma grande variedade de conceitos de criptografia e segurança de rede para propor projetos.
2. Os projetos podem ser programados pelos alunos em qualquer computador disponível e na linguagem que for mais apropriada; eles são independentes de plataforma e linguagem.
3. O professor não precisa baixar, instalar e configurar qualquer infra-estrutura em particular para os projetos independentes.

Há também flexibilidade no tamanho dos projetos. Projetos maiores dão aos alunos uma sensação de realização completa da tarefa, mas os alunos com menos capacidade ou menos habilidades organizacionais podem ficar para trás. Projetos maiores normalmente exigem mais esforço dos melhores alunos. Projetos menores podem ter uma relação conceito—código mais alta, e como pode haver maior quantidade deles, existe a oportunidade de abordar diversas áreas diferentes.

Novamente, assim como nos projetos de pesquisa, os alunos devem primeiro preparar uma proposta. O material do aluno deverá incluir os mesmos elementos listados na Seção A.1. O manual do professor inclui algumas sugestões de projetos de programação.

As seguintes pessoas forneceram os projetos de pesquisa e programação sugeridos no manual do professor: Henning Schulzrinne, da Columbia University; Cetin Kaya Koc, da Oregon State University; e David M. Balenson, da Trusted Information Systems e da George Washington University.

B.3 EXERCÍCIOS DE LABORATÓRIO

Os professores Sanjay Rao e Ruben Torres, da Purdue University, prepararam um conjunto de exercícios de laboratório que fazem parte do material de apoio disponível no site. Esses projetos são exercícios de implementação criados para serem programados no Linux, mas que podem ser adaptados para qualquer ambiente Unix, e oferecem uma experiência real da implementação de funções e aplicações de segurança.

B.4 TAREFAS ESCRITAS

Tarefas escritas podem ter um efeito multiplicador poderoso no processo de aprendizado em uma disciplina técnica, como criptografia e segurança de rede. Além disso, ajudam a superar a tendência dos alunos de buscar um assunto com o mínimo de envolvimento pessoal, motivados apenas pela busca de fatos e técnicas para a solução de problemas, sem adquirirem conhecimento profundo do assunto.

O site de apoio contém diversas sugestões de tarefas escritas, organizadas por capítulo. Os professores, talvez, poderão considerar essa a parte mais importante de sua abordagem de ensino. Gostaria muito de receber algum comentário dos professores em relação a essas tarefas escritas ou quaisquer outras sugestões.

B.5 TAREFAS DE LEITURA/RELATÓRIOS

Outra maneira excelente de reforçar os conceitos do curso e dar aos alunos experiência em pesquisa é sugerir artigos para serem lidos e analisados. No site do livro, também sugiro uma lista de artigos, um ou dois por capítulo, em inglês. Todos estão disponíveis na Internet ou em qualquer boa biblioteca técnica das universidades. Além disso, também há no site uma sugestão de estilo de texto a ser adotado pelo aluno ao redigir esse tipo de trabalho.

GLOSSÁRIO

Estudando o Império, Arrakis e toda a cultura que produziu o Maud'Dib, aparecem muitos termos incomuns. Aumentar o conhecimento é um objetivo louvável, daí as definições e explicações a seguir.
— *Dune*, Frank Herbert

Alguns termos deste glossário foram reproduzidos do *Internet Security Glossary* [RFC 2828]. Eles estão indicados por um asterisco.

algoritmo RSA Um algoritmo de criptografia por chave pública baseado na exponenciação em aritmética modular. Esse é o único algoritmo geralmente aceito como prático e seguro para a criptografia por chave pública.

ameaça à segurança* Um potencial para violação da segurança, que existe quando há uma circunstância, capacidade, ação ou evento que pode quebrar a segurança e causar danos. Ou seja, uma ameaça é um possível risco que pode explorar uma vulnerabilidade.

aritmética modular Um tipo de aritmética de inteiros que reduz todos os números a um número dentre um conjunto fixo $[0 \ldots n-1]$ para um dado número n. Qualquer inteiro fora desse intervalo é reduzido a um número nesse intervalo, tomando-se o resto após a divisão por n.

assinatura digital Um mecanismo de autenticação que permite que o criador de uma mensagem anexe um código que atua como uma assinatura. A assinatura é formada por meio do hash da mensagem e da codificação da mensagem com a chave privada do criador. A assinatura garante a origem e a integridade da mensagem.

ataque à segurança* Um assunto na segurança de sistemas que deriva de uma ameaça inteligente, ou seja, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) para escapar dos serviços de segurança e violar a política de segurança de um sistema.

ataque de aniversário Esse ataque de criptoanálise tenta encontrar dois valores no domínio de uma função que correspondem ao mesmo valor em seu intervalo.

ataque meet-in-the-middle (**encontro no meio**) Esse é um ataque criptoanalítico que tenta encontrar um valor em cada um dos intervalos e domínios da composição de duas funções, de modo que o mapeamento direto de um pela primeira função é igual à imagem inversa do outro pela segunda função - literalmente encontrando-se no meio da função composta.

ataque man-in-the-middle do (**homem no meio**) Uma forma de ataque de interceptação em que o atacante intercepta e modifica seletivamente os dados enviados, a fim de se disfarçar como uma ou mais das entidades envolvidas em uma comunicação.

ataques de replicação Um ataque em que um serviço já autorizado e completado é forjado por outra 'solicitação duplicada' em uma tentativa de repetir os comandos autorizados.

autenticação de mensagens Um processo usado para verificar a integridade de uma mensagem.

autenticação* O processo de verificar uma identidade afirmada por ou para uma entidade do sistema.

autenticador Informação adicional anexada a uma mensagem para permitir que o receptor verifique se a mensagem deve ser aceita como autêntica. O autenticador pode ser funcionalmente independente do conteúdo da própria mensagem (por exemplo, um *nonce* ou um identificador de origem), ou pode ser uma função do conteúdo da mensagem (por exemplo, um valor de hash ou uma soma de verificação criptográfica).

bactéria Programa que consome recursos do sistema pela própria replicação.

bomba lógica Lógica embutida em um programa de computador que verifica a ocorrência de um certo conjunto de condições no sistema. Quando essas condições são atendidas, ela executa alguma função que resulta em ações não autorizadas.

byte Uma seqüência de oito bits. Também conhecida como um octeto.

canal secreto Um canal de comunicação que permite a transferência de informações de uma maneira não pretendida pelos projetistas da facilidade de comunicação.

cavalo-de-tróia* Um programa de computador que parece ter uma função útil, mas também tem uma função escondida, e potencialmente maliciosa, que escapa dos mecanismos de segurança, às vezes explorando autorizações legítimas de uma entidade do sistema que invoca o programa.

centro de distribuição de chaves Um sistema autorizado a transmitir chaves de sessão temporárias para seus membros. Cada chave de sessão é transmitida em forma codificada, usando uma chave-mestra que o centro de distribuição de chaves compartilha com o membro de destino.

certificado de chave pública Consiste em uma chave pública mais uma identificação de usuário (User ID) do proprietário da chave, sendo o conjunto todo assinado por um terceiro elemento confiável. Normalmente, o terceiro é uma autoridade certificadora (CA) confiável pela comunidade de usuários, como um órgão do governo ou uma instituição financeira.

chave de sessão Uma chave de criptografia temporária, usada entre dois membros.

chave-mestra Uma chave de longa duração, usada entre um centro de distribuição de chaves e um membro com a finalidade de codificar a transmissão de chaves de sessão. Normalmente, as chaves-mestras são distribuídas por meios não criptográficos. Também chamada de chave de codificação de chave.

chave privada Uma das duas chaves usadas em um sistema de criptografia assimétrica. Para a comunicação segura, a chave privada só deverá ser conhecida por seu criador.

chave pública Uma das duas chaves usadas em um sistema de criptografia assimétrica. A chave pública se torna pública, para ser usada em conjunto com uma chave privada correspondente.

chave secreta A chave usada em um sistema de criptografia simétrica. Os dois participantes precisam compartilhar a mesma chave, e essa chave precisa permanecer secreta para proteger a comunicação.

cifra Um algoritmo para criptografia e decriptografia. Uma cifra substitui um trecho de informação (um elemento em texto claro) por outro objeto, com a intenção de ocultar seu significado. Normalmente, a regra de substituição é controlada por uma chave secreta.

cifra de fluxo Um algoritmo de codificação simétrico em que a saída do texto cifrado é produzida bit a bit ou byte a byte a partir de um fluxo de entrada de texto claro.

cifra em bloco Um algoritmo de criptografia simétrica em que um bloco de bits de texto claro (normalmente, 64 ou 128) é transformado como um todo em um bloco de texto cifrado de mesmo tamanho.

classe de resíduo Todos os inteiros que têm o mesmo resto quando divididos por n formam uma classe de resíduo (mod n). Assim, para determinado resto r, a classe de resíduo (mod n) à qual ele pertence, consiste nos inteiros r, $r \pm n$, $r \pm 2n$,...

código Uma regra invariável para substituir um trecho de informação (por exemplo, letra, palavra, frase) por outro objeto, não necessariamente do mesmo tipo. Em geral, não existe intenção de ocultar o significado. Alguns exemplos incluem o código de caracteres ASCII (cada caractere é representado por 7 bits) e a modulação por freqüência (cada valor binário é representado por uma determinada freqüência).

código de autenticação de mensagens (MAC — Message Authentication Code) Código de verificação criptográfica.

computacionalmente seguro Seguro porque o tempo e/ou custo de violar a segurança são muito altos para serem viáveis.

confusão Uma técnica criptográfica que busca tornar o relacionamento entre as estatísticas do texto cifrado e o valor da chave de codificação o mais complexo possível. Isso é obtido com o uso de um algoritmo complexo de embaralhamento, que depende da chave e da entrada.

controle de acesso discricionário* Um serviço de controle de acesso que impõe uma política de segurança baseada na identidade das entidades e suas autorizações para acessar recursos do sistema (*ver* lista de controle de acesso, política de segurança baseada em identidade, controle de acesso obrigatório). Esse serviço é chamado de 'discricionário' porque a entidade pode ter direitos de acesso que permitem que ela, a seu critério, possibilite que outra entidade acesse algum recurso.

controle de acesso obrigatório Um meio de restringir o acesso aos objetos com base nos atributos de segurança fixos atribuídos aos usuários e a arquivos e outros objetos. Os controles são obrigatórios no sentido de que não podem ser modificados por usuários ou seus programas.

criptoanálise O ramo da criptologia que trata da quebra de uma cifra para recuperar informações, ou falsificação de informações criptografadas para que sejam aceitas como autênticas.

criptoanálise diferencial Uma técnica que se baseia na criptografia de textos claros que apresentem determinados padrões de diferença XOR. Os padrões de diferença do texto cifrado resultantes oferecem informações que podem ser utilizadas para determinar a chave de criptografia.

criptografar Converter texto ou dados claros em uma forma não inteligível por meio de uma tradução reversível, com base em uma tabela ou algoritmo de tradução. Também chamado de cifrar.

criptografia O ramo da criptologia que trata do projeto dos algoritmos para criptografia e decriptografia, com o propósito de garantir o segredo e/ou a autenticidade das mensagens. Também designa o processo de criptografar textos claros.

criptografia assimétrica Uma forma de criptossistema em que a criptografia e a decriptografia são realizadas usando-se duas chaves diferentes, uma conhecida como chave pública e outra conhecida como chave privada. Também conhecida como codificação de chave pública.

criptografia convencional Criptografia simétrica.

criptografia de chave pública Criptografia assimétrica.

criptografia múltipla Uso repetido de uma função de criptografia, com diferentes chaves, para produzir um mapeamento mais complexo do texto claro para texto cifrado.

criptografia simétrica Uma forma de criptossistema em que a criptografia e a decriptografia são realizadas usando-se a mesma chave. Também conhecida como criptografia convencional.

criptologia O estudo das comunicações seguras, que abrange tanto a criptografia quanto a criptoanálise.

decriptografar Converter o texto ou dados codificados (crifrados) no texto ou dados originais (claros). Também chamado de decifrar.

decriptografia Conversão de texto ou dados codificados (chamados de texto cifrado) em texto ou dados originais (chamados de texto claro).

difusão Uma técnica criptográfica que busca obscurecer a estrutura estatística do texto claro, espalhando a influência de cada dígito de texto claro individual por muitos dígitos de texto cifrado.

digrama Uma seqüência de duas letras. Em inglês e em outros idiomas, a freqüência relativa dos diversos digramas em texto claro pode ser usada na criptoanálise de algumas cifras. Também chamado de dígrafo.

divisor Um inteiro é considerado um divisor de outro inteiro se não houver resto na divisão entre eles.

efeito avalanche Uma característica de um algoritmo de criptografia em que uma pequena mudança no texto claro ou na chave faz surgir uma grande mudança no texto cifrado. Para um código de hash, o efeito avalanche é uma característica em que uma pequena mudança na mensagem faz surgir uma grande mudança no resumo da mensagem.

encadeamento de bloco Um procedimento usado durante a criptografia simétrica de bloco, que torna um bloco de saída dependente não apenas da chave e do bloco de entrada em texto claro atual, mas também da entrada e/ou saída anterior. O efeito do encadeamento de bloco é que duas ocorrências do mesmo bloco de entrada em texto claro produzirão diferentes blocos de texto cifrado, tornando a criptoanálise mais difícil.

esteganografia Métodos de ocultar a existência de uma mensagem ou outros dados. Isso é diferente da criptografia, que esconde o significado de uma mensagem, mas não esconde a mensagem em si.

firewall Um computador dedicado que faz a interface com computadores fora de uma rede e possui precauções de segurança especiais embutidas, a fim de proteger arquivos confidenciais em computadores dentro da rede. É usado para atender a conexões de rede externas, especialmente a Internet, e linhas discadas.

função de hash Uma função que mapeia um bloco de dados ou mensagem de tamanho variável para um valor de tamanho fixo, chamado código de hash. A função é elaborada de tal forma que, quando protegida, oferece um autenticador para os dados ou a mensagem. Também conhecido como resumo da mensagem.

função unidirecional de trapdoor Uma função facilmente calculada, sendo o cálculo de seu inverso inviável, a menos que certa informação privilegiada seja conhecida.

função unidirecional Uma função facilmente calculada, mas o cálculo do seu inverso é inviável.

gerador de números pseudo-aleatórios Uma função que produz de forma determinística uma seqüência de números aparentemente aleatórios estatisticamente.

honeypot (pote de mel) Um sistema de chamariz projetado para atrair um atacante em potencial para fora dos sistemas críticos. Uma forma de detecção de intrusão.

incondicionalmente seguro Seguro até mesmo contra um oponente com tempo e recursos de computação ilimitados.

infra-estrutura de chave pública (PKI — Public-Key Infrastructure) O conjunto de equipamentos, programas, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais com base na criptografia assimétrica.

intruso Um indivíduo que obtém ou tenta obter acesso não autorizado a um sistema de computador ou obter privilégios não autorizados nesse sistema.

Kerberos O nome dado ao serviço de autenticação de código do Projeto Athena.

máximo divisor comum O maior divisor comum entre dois inteiros, a e b, é o maior positivo inteiro que divide a e b. Diz-se que um inteiro divide outro inteiro se não houver resto na divisão.

mecanismo de segurança Um processo (ou um dispositivo incorporando tal processo) elaborado para detectar, impedir ou recuperar-se de um ataque à segurança.

modo de operação Uma técnica para aprimorar o efeito de um algoritmo criptográfico ou adaptar o algoritmo a uma aplicação, como aplicar uma cifra de bloco a uma seqüência de blocos de dados ou cadeia de dados.

nibble Uma seqüência de quatro bits.

nonce Um identificador ou número usado apenas uma vez.

raiz primitiva Se r e n forem inteiros relativamente primos com $n > 0$, e se $\phi(n)$ for o menor expoente positivo m tal que $r^m \equiv 1 \bmod n$, então r é chamado de raiz primitiva módulo n.

rede privativa virtual Consiste em um conjunto de computadores interconectados por meio de uma rede relativamente insegura, e que utilizam a criptografia e protocolos especiais para obter segurança.

relativamente primos Dois números são relativamente primos se não tiverem fatores primos em comum, ou seja, seu único divisor comum for 1.

resíduo Quando o inteiro a é dividido pelo inteiro n, o resto r é conhecido como resíduo. De modo equivalente, $r = a \bmod n$.

resumo de mensagem Função de hash.

segurança multinível Uma capacidade que impõe controle de acesso ao longo de vários níveis de classificação de dados.

senha* Um valor secreto, normalmente uma cadeia de caracteres, usada como informação de autenticação. Uma senha normalmente está associada a um identificador de usuário apresentado explicitamente no processo de autenticação, mas, em alguns casos, a identidade pode estar implícita.

serviço de segurança Um serviço de processamento ou comunicação que melhora a segurança dos sistemas de processamento de dados e das transferências de informação de uma organização. Os serviços servem para se opor aos ataques à segurança, e utilizam um ou mais mecanismos de segurança para fornecer o serviço.

sistema confiável Um computador e sistema operacional que permitem a implementação de determinada política de segurança.

sistema de detecção de intrusão Um conjunto de ferramentas automatizadas projetadas para detectar acesso não autorizado a um sistema hospedeiro.

texto cifrado A saída de um algoritmo de criptografia; a forma criptografada de uma mensagem ou dados.

texto claro A entrada para uma função de criptografia ou a saída de uma função de decriptografia.

trapdoor Ponto de entrada secreto para um programa, não documentado, usado para conceder acesso sem os métodos normais de autenticação de acesso. Também conhecido como backdoor.

verificador criptográfico Um autenticador que é uma função criptográfica dos dados a serem autenticados e da chave secreta. Também considerado código de autenticação de mensagens.

verme Programa que pode se replicar e enviar cópias de um computador para outro pelas conexões da rede. Na chegada, o verme pode ser ativado para replicar-se e propagar-se novamente. Além da propagação, o verme normalmente realiza alguma função indesejada.

vetor de inicialização Um bloco aleatório de dados usado para iniciar a codificação de vários blocos de texto claro, quando é utilizada uma técnica de codificação por encadeamento de bloco. O vetor de inicialização serve para afastar os ataques de texto claro conhecido.

vírus Código embutido dentro de um programa, que faz com que uma cópia de si mesmo seja inserida em um ou mais outros programas. Além da propagação, o vírus normalmente realiza alguma função indesejada.

zumbi Um programa que secretamente assume o controle de outro computador conectado à Internet e depois utiliza esse computador como uma plataforma para desferir ataques difíceis de serem rastreados até o criador do zumbi.

REFERÊNCIAS

Em questões desse tipo, todos acham que é justificável escrever e publicar a primeira coisa que venha à cabeça ao apanhar uma caneta, e acham que sua própria idéia é axiomática, como o fato de que dois mais dois são quatro. Se os críticos tivessem o trabalho de pensar sobre o assunto por anos a fio e testar cada conclusão em relação à história real da guerra, como eu fiz, eles sem dúvida teriam mais cuidado com o que escrevem.

—On war, Carl von Clausewitz

ABREVIAÇÕES

ACM Association for Computing Machinery

IEEE Institute of Electrical and Electronics Engineers

NIST National Institute of Standards and Technology

ADAM90 Adams, C. e Tavares, S. "Generating and counting binary bent sequences". IEEE Transactions on Information Theory, 1990.

ADAM94 Adams, C. "Simple and effective key scheduling for symmetric ciphers". Proceedings, Workshop in Selected Areas of Cryptography, SAC' 94. 1994.

ADLE83 Adleman, L. "The function field sieve". Lecture Notes in Computer Science 877, Springer-Verlag, 1983.

AGRA02 Agrawal, M.; Keyal, N.; e Saxena, N. "PRIMES is in P". IIT Kanpur, Pré-impressão, agosto de 2002. **http://www.cse.iitk.ac.in/news/primality.pdf.**

AKL83 Akl, S. "Digital signatures: a tutorial survey". *Computer*, fevereiro de 1983.

ALVA90 Alvare, A. "How crackers crack passwords or what passwords to avoid". Proceedings, UNIX Security Workshop II, agosto de 1990.

ANDE80 Anderson, J. "Computer security threat monitoring and surveillance". Fort Washington, PA: James P. Anderson Co., abril de 1980.

AUDI04 Audin, G. "Next-gen firewalls: what to expect". *Business Communications Review*, junho de 2004.

AXEL00 Axelsson, S. "The base-rate fallacy and the difficulty of intrusion detection". *ACM Transactions and Information and System Security*, agosto de 2000.

BACE00 Bace, R. "Intrusion detection". Indianapolis: *Macmillan Technical Publishing*, 2000.

BACE01 Bace, R. e Mell, P. "Intrusion detection systems". *NIST Special Publication SP* 800-31, novembro de 2000.

BARK91 Barker, W. *Introduction to the analysis of the Data Encryption Standard (DES)*. Laguna Hills: Aegean Park Press, 1991.

BARR03 Barreto, P. e Rijmen, V. "The whirlpool hashing function". Submission to NESSIE, setembro de 2000, revisado em maio de 2003.

BAUE88 Bauer, D. e Koblentz, M. "NIDX—An expert system for real-time network intrusion detection". Proceedings, Computer Networking Symposium, abril de 1988.

BELL90 Bellovin, S. e Merritt, M. "Limitations of the Kerberos authentication system". *Computer Communications Review*, outubro de 1990.

BELL92 Bellovin, S. "There be dragons". Proceedings, UNIX Security Symposium III, setembro de 1992.

BELL93 Bellovin, S. "Packets found on an internet". *Computer Communications Review*, julho de 1993.

BELL94a Bellare, M e Rogaway, P. "Optimal asymmetric encryption—how to encrypt with RSA". Proceedings, Eurocrypt '94, 1994.

BELL94b Bellovin, S. e Cheswick, W. "Network firewalls". *IEEE Communications Magazine*, setembro de 1994.

BELL96a Bellare, M.; Canetti, R.; e Krawczyk, H. "Keying hash functions for message authentication". Proceedings, CRYPTO '96, agosto de 1996; publicado por Springer-Verlag. Uma versão detalhada está disponível em **http://www-cse.ucsd.edu/users/mihir.**

BELL96b Bellare, M.; Canetti, R.; e Krawczyk, H. "The HMAC construction". *CryptoBytes*, Spring 1996.

BELL97 Bellare, M. e Rogaway, P. "Collision-resistant hashing: towards making UOWHF's practical". Proceedings, CRYPTO '97, 1997; publicado por Springer-Verlag.

BELL00 Bellare, M.; Kilian, J.; e Rogaway, P. "The security of the cipher block chaining message authentication code". *Journal of Computer and System Sciences*, dezembro de 2000.

BERL84 Berlekamp, E. *Algebraic coding theory*. Laguna Hills: Aegean Park Press, 1984.

BETH91 Beth, T.; Frisch, M.; e Simmons, G. eds. *Public-Key cryptography: state of the art and future directions*. Nova York: Springer-Verlag, 1991.

BIHA93 Biham, E. e Shamir, A. *Differential cryptanalysis of the data encryption standard*. Nova York: Springer-Verlag, 1993.

BIHA00 Biham, E. e Shamir, A. "Power analysis of the key scheduling of the AES candidates" Proceedings, Second AES Candidate Conference, 24 de outubro de 2000. **http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm.**

BISH03 Bishop, M. *Computer security: art and science*. Boston: Addison-Wesley, 2003.

BISH05 Bishop, M. *Introduction to computer security*. Boston: Addison-Wesley, 2005.

BLAC00 Black, J. e Rogaway, P.; e Shrimpton, T. "CBC MACs for arbitrary-length messages: the three-key constructions". Advances in Cryptology — CRYPTO '00, 2000.

BLAC02 Black, J. e Rogaway, P. "Black-box analysis of the block-cipher-based hash function constructions from PGV". Advances in Cryptology — CRYPTO '02, 2002.

BLAK99 Blake, I.; Seroussi, G.; e Smart, N. *Elliptic curves in cryptography*. Cambridge: Cambridge University Press, 1999.

BLOO70 Bloom, B. "Space/time trade-offs in hash coding with allowable errors". *Communications of the ACM*, julho de 1970.

BLUM86 Blum, L.; Blum, M.; e Shub, M. "A simple unpredictable pseudo-random number generator". *SIAM Journal on Computing*, Nº 2, 1986.

BONE99 Boneh, D. "Twenty years of attacks on the rsa cryptosystem". *Notices of the American Mathematical Society*, fevereiro de 1999.

BONE02 Boneh, D. e Shacham, H. "Fast variants of RSA". *CryptoBytes*, inverno/primavera de 2002. **http://www.rsasecurity.com/rsalabs.**

BORN03 Bornemann, F. "PRIMES is in P: a breakthrough for everyman". *Notices of the American Mathematical Society*, maio de 2003.

BRIG79 Bright, H. e Enison, R. "Quasi-random number sequences from long-period tlp generator with remarks on application to cryptography". *Computing Surveys*, dezembro de 1979.

BRYA88 Bryant, W. "Designing an authentication system: a dialogue in four scenes". Project Athena document, fevereiro de 1988. Disponível em **http://web.mit.edu/kerberos/www/dialogue.html.**

BURN97 Burn, R. *A pathway to number theory*. Cambridge, England: Cambridge University Press, 1997.

CAMP92 Campbell, K. e Wiener, M. "Proof that DES is not a group". Proceedings, Crypto '92, 1992; publicado por Springer-Verlag.

CASS01 Cass, S. "Anatomy of malice". *IEEE Spectrum*, novembro de 2001.

CERT01 CERT Coordination Center. "Denial of service attacks". Junho de 2001. **http://www. cert.org/tech_tips /denial_of_service.html.**

CHAN02 Chang, R. "Defending against flooding-based distributed denial-of-service attacks: a tutorial". *IEEE Communications Magazine*, outubro de 2002.

CHAP00 Chapman, D. e Zwicky, E. Building Internet Firewalls. Sebastopol, CA: O'Reilly, 2000.

CHEN98 Cheng, P. e outros. "A security architecture for the Internet protocol". *IBM Systems Journal*, número 1, 1998.

CHES97 Chess, D. "The future of viruses on the Internet". Proceedings, Virus Bulletin International Conference, outubro de 1997.

CHES03 Cheswick, W. e Bellovin, S. *Firewalls and Internet security: repelling the wily hacker*. Reading: Addison-Wesley, 2003.

COCK73 Cocks, C. "A note on non-secret encryption". CESG Report, novembro de 1973.

COHE94 Cohen, F. *A short course on computer viruses*. Nova York: Wiley, 1994.

COPP94 Coppersmith, D. "The Data Encryption Standard (DES) and its strength against attacks". *IBM Journal of Research and Development*, maio de 1994.

CORM01 Cormen, T.; Leiserson, C.; Rivest, R.; e Stein, C. *Introduction to algorithms*. Cambridge: MIT Press, 2001.

CRAN01 Crandall, R. e Pomerance, C. Prime Numbers: *A computational perspective*. Nova York: Springer-Verlag, 2001.

DAEM99 Daemen, J. e Rijmen, V. "AES proposal: Rijndael, version 2". Submission to NIST, março de 1999. **http://csrc.nist.gov/encryption/aes.**

DAEM01 Daemen, J. e Rijmen, V. "Rijndael: the Advanced Encryption Standard". *Dr. Dobb's Journal*, março de 2001.

DAEM02 Daemen, J. e Rijmen, V. *The design of Rijndael: the wide trail strategy explained*. Nova York, Springer-Verlag, 2002.

DAMG89 Damgard, I. "A design principle for hash functions". Proceedings, CRYPTO '89, 1989; publicado por Springer-Verlag.

DAVI89 Davies, D. e Price, W. *Security for computer networks*. Nova York: Wiley, 1989.

DAVI93 Davies, C. e Ganesan, R. "BApasswd: a new proactive password checker". Proceedings, 16th National Computer Security Conference, setembro de 1993.

DAWS96 Dawson, E. e Nielsen, L. "Automated cryptoanalysis of xor plaintext strings". *Cryptologia*, abril de 1996.

DENN81 Denning, D. "Timestamps in key distribution protocols". *Communications of the ACM*, agosto de 1981.

DENN82 Denning, D. *Cryptography and data security*. Reading: Addison-Wesley, 1982.

DENN83 Denning, D. "Protecting public keys and signature keys". *Computer*, fevereiro de 1983.

DENN87 Denning, D. "An intrusion-detection model". *IEEE Transactions on Software Engineering*, fevereiro de 1987.

DESK92 Deskins, W. *Abstract algebra*. Nova York: Dover, 1992.

DIFF76a Diffie, W. e Hellman, M. "New directions in cryptography". Proceedings of the AFIPS National Computer Conference, junho de 1976.

DIFF76b Diffie, W. e Hellman, M. "Multiuser cryptographic techniques". *IEEE Transactions on Information Theory*, novembro de 1976.

DIFF77 Diffie, W. e Hellman, M. "Exhaustive cryptanalysis of the NBS Data Encryption Standard". *Computer*, junho de 1977.

DIFF79 Diffie, W. e Hellman, M. "Privacy and authentication: an introduction to cryptography". *Proceedings of the IEEE*, março de 1979.

DIFF88 Diffie, W. "The first ten years of public-key cryptography". *Proceedings of the IEEE*, maio de 1988. Reimpresso em [SIMM92].

DOBB96 Dobbertin, H. "The status of MD5 after a recent attack". *CryptoBytes*, Summer 1996.

DORA03 Doraswamy, N. e Harkins, D. *IPSec*. Upper Saddle River: Prentice Hall, 2003.

EFF98 Electronic Frontier Foundation. Cracking DES: *secrets of encryption research, wiretap politics, and chip design*. Sebastopol, CA: O'Reilly, 1998

ELGA85 ElGamal, T. "A public-key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transactions on Information Theory*, julho de 1985.

ELLI70 Ellis, J. "The possibility of secure non-secret digital encryption". *CESG Report*, janeiro de 1970.

ELLI99 Ellis, J. "The history of non-secret encryption". *Cryptologia*, julho de 1999.

ENGE80 Enger, N. e Howerton, P. *Computer security*. Nova York: Amacom, 1980.

ENGE99 Enge, A. *Elliptic curves and their applications to cryptography*. Norwell: Kluwer Academic Publishers, 1999.

FEIS73 Feistel, H. "Cryptography and computer privacy". *Scientific American*, maio de 1973.

FEIS75 Feistel, H.; Notz, W.; e Smith, J. "Some cryptographic techniques for machine-to-machine data communications". *Proceedings of the IEEE*, novembro de 1975.

FELT03 Felten, E. "Understanding trusted computing: will its benefits outweigh its drawbacks?" *IEEE Security and Privacy*, maio/junho de 2003.

FERN99 Fernandes, A. "Elliptic curve cryptography". *Dr. Dobb's Journal*, dezembro de 1999.

FLUH00 Fluhrer, S. e McGrew, D. "Statistical analysis of the alleged RC4 key stream generator". Proceedings, Fast Software Encryption 2000, 2000.

FLUH01 Fluhrer, S.; Mantin, I.; e Shamir, A. "Weakness in the key scheduling algorithm of RC4". Proceedings, Workshop in Selected Areas of Cryptography, 2001.

FORD95 Ford, W. "Advances in public-key certificate standards". *ACM SIGSAC Review*, julho de 1995.

FORR97 Forrest, S.; Hofmeyr, S.; e Somayaji, A. "Computer immunology". *Communications of the ACM*, outubro de 1997.

FRAN01 Frankel, S. *Demystifying the IPSec puzzle*. Boston: Artech House, 2001.

FREE93 Freedman, D. "The goods on hacker hoods". *Forbes ASAP*, 13 de setembro de 1993.

FUMY93 Fumy, S. e Landrock, P. "Principles of key management". *IEEE Journal on Selected Areas in Communications*, junho de 1993.

GARD72 Gardner, M. *Codes, ciphers, and secret writing*. Nova York: Dover, 1972.

GARD77 Gardner, M. "A new kind of cipher that would take millions of years to break". *Scientific American*, agosto de 1977.

GARR01 Garrett, P. Making, *Breaking codes: an introduction to cryptology*. Upper Saddle River: Prentice Hall, 2001.

GASS88 Gasser, M. *Building a secure computer system*. Nova York: Van Nostrand Reinhold, 1988.

GAUD00 Gaudin, S. "The omega files". *Network World*, 26 de junho de 2000.

GILB03 Gilbert, H. and Handschuh, H. "Security analysis Of SHA-256 and sisters". Proceedings, CRYPTO '03, 2003; publicado por Springer-Verlag.

GOLL99 Gollmann, D. *Computer security*. Nova York: Wiley, 1999.

GONG92 Gong, L. "A security risk of depending on synchronized clocks". *Operating Systems Review*, janeiro de 1992.

GONG93 Gong, L. "Variations on the themes of message freshness and replay". Proceedings, IEEE Computer Security Foundations Workshop, junho de 1993.

GRAH94 Graham, R.; Knuth, D.; e Patashnik, O. *Concrete mathematics: a foundation for computer science*. Reading: Addison-Wesley, 1994.

GUTM02 Gutmann, P. "PKI: it's not dead, just resting". *Computer*, agosto de 2002.

HAMM91 Hamming, R. *The art of probability for scientists and engineers*. Reading: Addison-Wesley, 1991.

HANK04 Hankerson, D.; Menezes, A.; e Vanstone, S. *Guide to elliptic curve cryptography*. Nova York: Springer, 2004.

HARL01 Harley, D.; Slade, R.; e Gattiker, U. *Viruses revealed*. Nova York: Osborne/McGraw-Hill, 2001.

HEBE92 Heberlein, L.; Mukherjee, B.; e Levitt, K. "Internetwork security monitor: an intrusion-detection system for large-scale networks". Proceedings, 15th National Computer Security Conference, outubro de 1992.

HELD96 Held, G. *Data and image compression: tools and techniques*. Nova York: Wiley, 1996.

HELL78 Hellman, M. "An overview of public key cryptography". *IEEE Communications* Magazine, novembro de 1978.

HERS75 Herstein, I. *Topics in algebra*. Nova York: Wiley, 1975.

HEVI99 Hevia, A. e Kiwi, M. "Strength of two data encryption standard implementations under timing attacks". *ACM transactions on information and system security*, novembro de 1999.

HEYS95 Heys, H. e Tavares, S. "Avalanche characteristics of substitution-permutation encryption networks". *IEEE Transactions on Computers*, setembro de 1995.

HONE01 The Honeynet Project. *Know your enemy: revealing the security tools, tactics, and motives of the blackhat community*. Reading, MA: Addison-Wesley, 2001.

HORO71 Horowitz, E. "Modular arithmetic and finite field theory: a tutorial". Proceedings of the Second ACM Symposium and Symbolic and Algebraic Manipulation, março de 1971.

HUIT98 Huitema, C. IPv6: *The new Internet protocol*. Upper Saddle River: Prentice Hall, 1998.

IANS90 I'Anson, C. e Mitchell, C. "Security Defects in CCITT recommendation x.509 — the directory authentication framework". *Computer Communications Review*, abril de 1990.

ILGU93 Ilgun, K. "USTAT: a real-time intrusion detection system for UNIX". Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy, maio de 1993.

IWAT03 Iwata, T. e Kurosawa, K. "OMAC: one-key CBC MAC". Proceedings, Fast Software Encryption, FSE '03, 2003.

JAIN91 Jain, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Nova York: Wiley, 1991.

JAKO98 Jakobsson, M.; Shriver, E.; Hillyer, B.; e Juels, A. "A practical secure physical random bit generator". Proceedings of The Fifth ACM Conference on Computer and Communications Security, novembro de 1998.

JAVI91 Javitz, H. e Valdes, A. "The SRI IDES statistical anomaly detector". Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy, maio de 1991.

JONE82 Jones, R. "Some techniques for handling encipherment keys". *ICL Technical Journal*, novembro de 1982.

JUEN85 Jueneman, R.; Matyas, S.; e Meyer, C. "Message authentication". *IEEE Communications Magazine*, setembro de 1988.

JUEN87 Jueneman, R. "Electronic document authentication". *IEEE Network Magazine*, abril de 1987.

JUN99 Jun, B. e Kocher, P. "The Intel random number generator". *Intel White Paper*, 22 de abril de 1999.

JUNO04 Junod, P. e Vaudenay, S. "Perfect diffusion primitives for block ciphers: building efficient mds matrices". Selected Areas in Cryptography 2004, Waterloo, Canada, 9-10 de agosto de 2004.

JURI97 Jurisic, A. e Menezes, A. "Elliptic curves and cryptography". *Dr. Dobb's Journal*, abril de 1997.

KAHN96 Kahn, D. *The codebreakers: the story of secret writing*. Nova York: Scribner, 1996.

KALI95 Kaliski, B. e Robshaw, M. "The secure use of RSA". *CryptoBytes*, Autumn 1995.

KALI96a Kaliski, B. e Robshaw, M. "Multiple encryption: weighing security and performance". *Dr. Dobb's Journal*, janeiro de 1996.

KALI96b Kaliski, B. "Timing attacks on cryptosystems". *RSA Laboratories Bulletin*, janeiro de 1996. **http://www.rsasecurity.com/rsalabs.**

KATZ00 Katzenbeisser, S., ed. *Information hiding techniques for steganography and digital watermarking*. Boston: Artech House, 2000.

KEHN92 Kehne, A.; Schonwalder, J.; e Langendorfer, H. "A nonce-based protocol for multiple authentications" *Operating Systems Review*, outubro de 1992.

KELS98 Kelsey, J.; Schneier, B.; e Hall, C. "Cryptanalytic attacks on pseudorandom number generators". Proceedings, Fast Software Encryption, 1998. **http://www.schneier.com/paper-prngs.html.**

KENT00 Kent, S. "On the trail of intrusions into information systems". *IEEE Spectrum*, dezembro de 2000.

KEPH97a Kephart, J.; Sorkin, G.; Chess, D.; e White, S. "Fighting computer viruses". *Scientific American*, novembro de 1997.

KEPH97b Kephart, J.; Sorkin, G.; Swimmer, B.; e White, S. "Blueprint for a computer immune system". Proceedings, Virus Bulletin International Conference, outubro de 1997.

KITS04 Kitsos, P. e Koufopaviou, O. "Architecture and hardware implementation of the whirlpool hash function". *IEEE Transactions on Consumer Electronics*, fevereiro de 2004.

KLEI90 Klein, D. "Foiling the cracker: a survey of, and improvements to, password security". Proceedings, UNIX Security Workshop II, agosto de 1990.

KNUD98 Knudsen, L. e outros. "Analysis method for alleged RC4". Proceedings, ASIACRYPT '98, 1998.

KNUT97 Knuth, D. *The art of computer programming, volume 1: fundamental algorithms*. Reading: Addison-Wesley, 1997.

KNUT98 Knuth, D. *The art of computer programming, volume 2: seminumerical algorithms*. Reading: Addison-Wesley, 1998.

KOBL92 Koblas, D. e Koblas, M. "SOCKS". Proceedings, UNIX Security Symposium III, setembro de 1992.

KOBL94 Koblitz, N. *A course in number theory and cryptography*. Nova York: Springer-Verlag, 1994.

KOCH96 Kocher, P. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". Proceedings, Crypto '96, agosto de 1996.

KOCH98 Kocher, P.; Jaffe, J.; e Jun, B. "Introduction to Differential power analysis and related attacks". **http://www.cryptography.com/dpa/technical/index.html.**

KOHL89 Kohl, J. "The use of encryption in Kerberos for network authentication". Proceedings, Crypto '89, 1989; publicado por Springer-Verlag.

KOHL94 Kohl, J.; Neuman, B.; e Ts'o, T. "The evolution of the Kerberos authentication service". in Brazier, F. e Johansen, D. Distributed Open Systems. Los Alamitos: IEEE Computer Society Press, 1994. Disponível em **http://web.mit.edu/kerberos/ www/papers.html.**

KOHN78 Kohnfelder, L. Towards a practical public-key cryptosystem. Tese de bacharelado, M.I.T., maio de 1978.

KONH81 Konheim, A. *Cryptography: a primer*. Nova York: Wiley, 1981.

KORN96 Korner, T. *The pleasures of counting*. Cambridge, England: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills: Aegean Park Press, 1997.

KUMA98 Kumanduri, R. e Romero, C. *Number theory with computer applications*. Upper Saddle River: Prentice Hall, 1998.

LAM92a Lam, K. e Gollmann, D. "Freshness assurance of authentication protocols" Proceedings, ESORICS 92, 1992; publicado por Springer-Verlag.

LAM92b Lam, K. e Beth, T. "Timely authentication in distributed systems". Proceedings, ESORICS 92, 1992; publicado por Springer-Verlag.

LAND04 Landau, S. "Polynomials in the nation's service: using algebra to design the advanced encryption standard". *American Mathematical Monthly*, fevereiro de 2004.

LE93 Le, A.; Matyas. S.; Johnson, D.; e Wilkins, J. "A public key extension to the common cryptographic architecture". *IBM Systems Journal*, Nº 3, 1993.

LEHM51 Lehmer, D. "Mathematical methods in large-scale computing". Proceedings, 2nd Symposium on Large-Scale Digital Calculating Machinery, Cambridge: Harvard University Press, 1951.

LEUT94 Leutwyler, K. "Superhack". *Scientific American*, julho de 1994.

LEVE90 Leveque, W. *Elementary theory of numbers*. Nova York: Dover, 1990.

LEWA00 Lewand, R. *Cryptological mathematics*. Washington: Mathematical Association of America, 2000.

LEWI69 Lewis, P.; Goodman, A.; e Miller, J. "A pseudo-random number generator for the system/360". *IBM Systems Journal*, Nº 2, 1969.

LIDL94 Lidl, R. e Niederreiter, H. *Introduction to finite fields and their applications*. Cambridge: Cambridge University Press, 1994.

LIPM00 Lipmaa, H.; Rogaway, P.; e Wagner, D. "CTR mode encryption". NIST First Modes of Operation Workshop, outubro de 2000. **http://csrc.nist.gov/encryption/modes.**

LODI98 Lodin, S. e Schuba, C. "Firewalls fend off invasions from the net". *IEEE Spectrum*, fevereiro de 1998.

LUNT88 Lunt, T. e Jagannathan, R. "A prototype real-time intrusion-detection expert system". Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy, abril de 1988.

MADS93 Madsen, J. "World record in password checking". Usenet, comp.security.misc newsgroup, 18 de agosto de 1993.

MANT01 Mantin, I., Shamir, A. "A practical attack on broadcast RC4". Proceedings, Fast Software Encryption, 2001.

MARK97 Markham, T. "Internet security protocol". *Dr. Dobb's Journal*, junho de 1997.

MATS93 Matsui, M. "Linear cryptanalysis method for DES cipher". Proceedings, EUROCRYPT '93, 1993; publicado por Springer-Verlag.

MATY91a Matyas, S. "Key handling with control vectors". *IBM Systems Journal*, Nº 2, 1991.

MATY91b Matyas, S.; Le, A.; e Abraham, D. "A Key-management scheme based on control vectors". *IBM Systems Journal*, Nº 2, 1991.

MCHU00 McHugh, J.; Christie, A.; e Allen, J. "The role of intrusion detection systems". *IEEE Software*, setembro/outubro de 2000.

MEIN01 Meinel, C. "Code red for the Web". *Scientific American*, outubro de 2001.

MENE97 Menezes, A.; van Oorschot, P.; e Vanstone, S. *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997.

MERK78 Merkle, R. e Hellman, M. "Hiding information and signatures in trap door knapsacks". *IEEE Transactions on Information Theory*, setembro de 1978.

MERK79 Merkle, R. "Secrecy, authentication, and public key systems". Tese de Ph.D., Stanford University, junho de 1979.

MERK81 Merkle, R. e Hellman, M. "On the security of multiple encryption". *Communications of the ACM*, julho de 1981.

MERK89 Merkle, R. "One way hash functions and DES". Proceedings, CRYPTO '89, 1989; publicado por Springer-Verlag.

MEYE82 Meyer, C. e Matyas, S. *Cryptography: a new dimension in computer data security*. Nova York: Wiley, 1982.

MEYE88 Meyer, C. e Schilling, M. "Secure program load with modification detection code". Proceedings, SECURICOM 88, 1988.

MILL75 Miller, G. "Riemann's Hypothesis and Tests for Primality". Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing, maio de 1975.

MILL88 Miller, S.; Neuman, B.; Schiller, J.; e Saltzer, J. "Kerberos authentication and authorization system". Section E.2.1, Project Athena Technical Plan, M.I.T. Project Athena, Cambridge. 27 de outubro de 1988.

MIRK04 Mirkovic, J. e Relher, P. "A taxonomy of DDoS attack and DDoS defense mechanisms". *ACM SIGCOMM Computer Communications Review*, abril de 2004.

MIST96 Mister, S. e Adams, C. "Practical s-box design". Proceedings, Workshop in Selected Areas of Cryptography, SAC' 96. 1996.

MIST98 Mister, S. e Tavares, S. "Cryptanalysis of RC4-like ciphers". Proceedings, Workshop in Selected Areas of Cryptography, SAC' 98. 1998.

MITC90 Mitchell, C.; Walker, M.; e Rush, D. "CCITT/ISO standards for secure message handling". *IEEE Journal on Selected Areas in Communications*, maio de 1989.

MITC92 Mitchell, C.; Piper, F.; e Wild, P. "Digital signatures". Em [SIMM92].

MIYA90 Miyaguchi, S.; Ohta, K.; e Iwata, M. "Confirmation that Some hash functions are not collision free". Proceedings, EUROCRYPT '90, 1990; publicado por Springer-Verlag.

MOOR01 Moore, M. "Inferring Internet denial-of-service activity". Proceedings of the 10th USENIX Security Symposium, 2001.

MUFT89 Muftic, S. *Security mechanisms for computer networks*. Nova York: Ellis Horwood, 1989.

MURP90 Murphy, S. "The cryptanalysis of FEAL-4 with 20 chosen plaintexts". *Journal of Cryptology*, Nº 3, 1990.

MUSA03 Musa, M.; Schaefer, E.; e Wedig, S. "A simplified AES algorithm and its linear and differential cryptanalyses". *Cryptologia*, abril de 2003.

MYER91 Myers, L. *Spycomm: covert communication techniques of the underground*. Boulder: Paladin Press, 1991.

NACH97 Nachenberg, C. "Computer virus-antivirus coevolution". *Communications of the ACM*, janeiro de 1997.

NECH92 Nechvatal, J. "Public key cryptography". Em [SIMM92].

NECH00 Nechvatal, J. e outros. "Report on the development of the advanced encryption standard. national institute of standards and technology". 2 de outubro de 2000.

NEED78 Needham, R. e Schroeder, M. "Using encryption for authentication in large networks of computers". *Communications of the ACM*, dezembro de 1978.

NEUM90 Neumann, P. "Flawed computer chip sold for years". *RISKS-FORUM Digest*, Vol.10, Nº 54, 18 de outubro de 1990.

NEUM93a Neuman, B. e Stubblebine, S. "A note on the use of timestamps as nonces". *Operating Systems Review*, abril de 1993.

NEUM93b Neuman, B. "Proxy-based authorization and accounting for distributed systems". Proceedings of the 13th International Conference on Distributed Computing Systems, maio de 1993.

NICH96 Nichols, R. *Classical cryptography course*. Laguna Hills: Aegean Park Press, 1996.

NING04 Ning, P. e outros. "Techniques and tools for analyzing intrusion alerts". *ACM Transactions on Information and System Security*, maio de 2004.

NIST97 National Institute of Standards and Technology. "Request for candidate algorithm nominations for the Advanced Encryption Standard". Federal Register, 12 de setembro de 1997.

ODLY95 Odlyzko, A. "The future of integer factorization". *CryptoBytes*, Summer 1995.

OPPL97 Oppliger, R. "Internet security: firewalls and beyond". *Communications of the ACM*, maio de 1997.

OPPL05 Oppliger, R. e Rytz, R. "Does trusted computing remedy computer security problems?" *IEEE Security and Privacy*, março/abril de 2005.

ORE67 Ore, O. "Invitation to number theory". Washington: The Mathematical Association of America, 1967,

ORE76 Ore, O. *Number theory and its history*. Nova York: Dover, 1976.

PARK88 Park, S. e Miller, K. "Random number generators: good ones are hard to find". *Communications of the ACM*, outubro de 1988.

PATR04 Patrikakis, C.; Masikos, M.; e Zouraraki, O. "Distributed denial of service attacks". *The Internet Protocol Journal*, dezembro de 2004.

PERL99 Perlman, R. "An overview of PKI trust models". *IEEE Network*, novembro/dezembro de 1999.

PFLE02 Pfleeger, C. *Security in computing*. Upper Saddle River: Prentice Hall, 2002.

PFLE03 Pfleeger, C. *Security in computing*. Upper Saddle River: Prentice Hall, 1997.

PIAT91 Piattelli-Palmarini, M. "Probability: neither rational nor capricious". Bostonia, março de 1991.

PIEP03 Pieprzyk, J.; Hardjono, T.; e Seberry, J. *Fundamentals of computer security*. Nova York: Springer-Verlag, 2003.

POHL81 Pohl, I. e Shaw, A. *The nature of computation: an introduction to computer science*. Rockville: Computer Science Press, 1981.

POIN02 Pointcheval, D. "How to encrypt properly with RSA". *CryptoBytes*, Winter/Spring 2002. **http://www.rsasecurity.com/rsalabs.**

POPE79 Popek, G. e Kline, C. "Encryption and secure computer networks". *ACM Computing Surveys*, dezembro de 1979.

PORR92 Porras, P. STAT: "A state transition analysis tool for intrusion detection". Tese de mestrado, University of California at Santa Barbara, julho de 1992.

PREN93a Preneel, B.; Govaerta, R.; e Vandewalle, J. "Hash functions based on block ciphers: a synthetic approach". Proceedings, Advances in Cryptology — CRYPTO '93, 1993.

PREN93b Preneel, B. "Cryptographic hash functions". Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, 1993.

PREN96 Preneel, B. e Oorschot, P. "On the security of two MAC algorithms". Lecture Notes in Computer Science 1561; Lectures on Data Security, 1999; publicado por Springer-Verlag.

PREN99 Preneel, B. "The State of cryptographic hash functions". Proceedings, EUROCRYPT '96, 1996; publicado por Springer-Verlag.

PROC01 Proctor, P. *The practical intrusion detection handbook*. Upper Saddle River: Prentice Hall, 2001.

RABI78 Rabin, M. "Digitalized signatures". Em Foundations of Secure Computation, DeMillo, R.; Dobkin, D.; Jones, A.; e Lipton, R., eds. Nova York: Academic Press, 1978.

RABI80 Rabin, M. "Probabilistic algorithms for primality testing". *Journal of Number Theory*, dezembro de 1980.

RAND55 Rand Corporation. *A million random digits*. Nova York: The Free Press, 1955. **http://www.rand.org/publications/classics/randomdigits**.

RESC01 Rescorla, E. *SSL and TLS: designing and building secure systems*. Reading: Addison-Wesley, 2001.

RIBE96 Ribenboim, P. *The new book of prime number records*. Nova York: Springer-Verlag, 1996.

RIJM96 Rijmen, V.; Daemen, J.; Preneel, B.; Bosselares, A.; e Win, E. "The cipher SHARK". *Fast Software Encryption*, FSE '96, 1996.

RITT91 Ritter, T. "The efficient generation of cryptographic confusion sequences". *Cryptologia*, vol. 15 nº 2, 1991. **http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM.**

RIVE78 Rivest, R.; Shamir, A.; e Adleman, L. "A method for obtaining digital signatures and public key cryptosystems". *Communications of the ACM*, fevereiro de 1978.

RIVE84 Rivest, R. e Shamir, A. "How to expose an eavesdropper". *Communications of the ACM*, abril de 1984.

ROBS95a Robshaw, M. "Stream ciphers". RSA Laboratories Technical Report TR-701, julho de 1995. **http://www.rsasecurity.com/rsalabs.**

ROBS95b Robshaw, M. "Block ciphers". RSA Laboratories Technical Report TR-601, agosto de 1995. **http://www.rsasecurity.com/rsalabs.**

ROBS95c Robshaw, M. "MD2, MD4, MD5, SHA and other hash functions". RSA Laboratories Technical Report TR-101, julho de 1995. **http://www.rsasecurity.com/rsalabs.**

RODR02 Rodriguez, A. e outros. *TCP/IP tutorial and technical overview*. Upper Saddle River: Prentice Hall, 2002.

ROSE05 Rosen, K. *Elementary number theory and its applications*. Reading: Addison-Wesley, 2000.

ROSI99 Rosing, M. *Implementing elliptic curve cryptography*. Greenwich: Manning Publications, 1999.

RUBI97 Rubin, A. "An experience teaching a graduate course in cryptography". *Cryptologia*, abril de 1997.

RUEP92 Rueppel, T. "Stream ciphers". Em [SIMM92].

SAFF93 Safford, D.; Schales, D.; e Hess, D. "The TAMU security package: an ongoing response to internet intruders in an academic environment". Proceedings, UNIX Security Symposium IV, outubro de 1993.

SAUE81 Sauer, C. e Chandy, K. *Computer systems performance modeling*. Englewood Cliffs: Prentice Hall, 1981.

SCHA96 Schaefer, E. "A simplified data encryption standard algorithm". *Cryptologia*, janeiro de 1996.

SCHN91 Schnorr, C. "Efficient signatures for smart card". *Journal of Cryptology*, Nº 3, 1991.

SCHN96 Schneier, B. *Applied cryptography*. Nova York: Wiley, 1996.

SCHN00 Schneier, B. *Secrets and lies: digital security in a networked world*. Nova York: Wiley, 2000.

SHAM82 Shamir, A. "A Polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem". Proceedings, 23rd IEEE Symposium on the Foundations of Computer Science, 1982.

SHAM03 Shamir, A. e Tromer, E. "On the cost of factoring RSA-1024". *CryptoBytes*, Summer 2003. **http://www.rsasecurity.com/rsalabs.**

SHAN49 Shannon, C. "Communication theory of secrecy systems". *Bell Systems Technical Journal*, Nº 4, 1949.

SIMM92 Simmons, G., ed. *Contemporary cryptology: the science of information integrity*. Piscataway: IEEE Press, 1992.

SIMM93 Simmons, G. "Cryptology". Encyclopaedia Britannica, Fifteenth Edition, 1993.

SIMO95 Simovits, M. *The DES: An extensive documentation and evaluation*. Laguna Hills: Aegean Park Press, 1995.

SING99 Singh, S. *The code book: the science of secrecy from ancient egypt to quantum cryptography*. Nova York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary cryptanalysis: a mathematical approach*. Washington: The Mathematical Association of America, 1966.

SMIT97 Smith, R. *Internet cryptography*. Reading: Addison-Wesley, 1997.

SNAP91 Snapp, S. e outros. "A system for distributed intrusion detection". Proceedings, COMPCON Spring '91, 1991.

SPAF92a Spafford, E. "Observing reusable password choices". Proceedings, UNIX Security Symposium III, setembro de 1992.

SPAF92b Spafford, E. "OPUS: Preventing weak password choices". *Computers and Security*, Nº 3, 1992.

STAL02 Stallings, W. "The Advanced Encryption Standard". *Cryptologia*, julho de 2002.

STAL04 Stallings, W. *Computer networking with Internet protocols and technology*. Upper Saddle River: Prentice Hall, 2004.

STAL06 Stallings, W. "The Whirlpool secure hash function". *Cryptologia*, janeiro de 2006.

STEI88 Steiner, J.; Neuman, C.; e Schiller, J. "Kerberos: an authentication service for open networked systems". Proceedings of the Winter 1988 USENIX Conference, fevereiro de 1988.

STEP93 Stephenson, P. "Preventive medicine". *LAN Magazine*, novembro de 1993.

STER92 Sterling, B. *The hacker crackdown: law and disorder on the electronic frontier*. Nova York: Bantam, 1992.

STIN02 Stinson, D. *Cryptography: theory and practice*. Boca Raton: CRC Press, 2002.

STOL88 Stoll, C. "Stalking the wily hacker". *Communications of the ACM*, maio de 1988.

STOL89 Stoll, C. *The cuckoo's egg*. Nova York: Doubleday, 1989.

SZOR05 Szor, P. *The art of computer virus research and defense*. Reading: Addison-Wesley, 2005.

THOM84 Thompson, K. "Reflections on trusting trust (deliberate software bugs)". *Communications of the ACM*, agosto de 1984.

TIME90 Time, Inc. *Computer security, understanding computers series*. Alexandria: Time-Life Books, 1990.

TIPP27 Tippett, L. *Random sampling numbers*. Cambridge, England: Cambridge University Press, 1927.

TSUD92 Tsudik, G. "Message authentication with one-way hash functions". Proceedings, INFOCOM '92, maio de 1992.

TUCH79 Tuchman, W. "Hellman presents no shortcut solutions to DES". *IEEE Spectrum*, julho de 1979.

TUNG99 Tung, B. Kerberos: *A network authentication system*. Reading: Addison-Wesley, 1999.

VACC89 Vaccaro, H. e Liepins, G. "Detection of anomalous computer session activity". Proceedings of the IEEE Symposium on Research in Security and Privacy, maio de 1989.

VANO90 van Oorschot, P. e Wiener, M. "A known-plaintext attack on two-key triple encryption". Proceedings, EUROCRYPT '90, 1990; publicado por Springer-Verlag.

VANO94 van Oorschot, P. e Wiener, M. "Parallel collision search with application to hash functions and discrete logarithms". Proceedings, Second ACM Conference on Computer and Communications Security, 1994.

VIJA02 Vijayan, J. "Denial-of-service attacks still a threat". *ComputerWorld*, 8 de abril de 2002.

VOYD83 Voydock, V. e Kent., S. "Security mechanisms in high-level network protocols". *Computing Surveys*, junho de 1983.

WACK02 Wack, J.; Cutler, K.; e Pole, J. *Guidelines on firewalls and firewall policy*. NIST Special Publication SP 800-41, janeiro de 2002.

WAGN00 Wagner, D. e Goldberg, I. "Proofs of security for the unix password hashing algorithm". Proceedings, ASIACRYPT '00, 2000.

WANG05 Wang, X.; Yin, Y.; e Yu, H. "Finding collisions in the full SHA-1. Proceedings, Crypto '05, 2005; publicado por Springer-Verlag.

WAYN96 Wayner, P. *Disappearing cryptography*. Boston: AP Professional Books, 1996.

WEBS86 Webster, A. e Tavares, S. "On the design of s-boxes". Proceedings, Crypto '85, 1985; publicado por Springer-Verlag.

WIEN90 Wiener, M. "Cryptanalysis of short RSA secret exponents". *IEEE Transactions on Information Theory*, vol. IT-36, 1990.

WILS05 Wilson, J. "The future of the firewall". *Business Communications Review*, maio de 2005.

WOO92a Woo, T. e Lam, S. "Authentication for distributed systems". *Computer*, janeiro de 1992.

WOO92b Woo, T. e Lam, S. "Authentication revisited". *Computer*, abril de 1992.

YUVA79 Yuval, G. "How to swindle rabin". *Cryptologia*, julho de 1979.

ZENG91 Zeng. K.; Yang, C.; Wei, D.; e Rao, T. "Pseudorandom bit generators in stream-cipher cryptography". *Computer*, fevereiro de 1991.

ZIV77 Ziv, J. e Lempel, A. "A universal algorithm for sequential data compression". *IEEE Transactions on Information Theory*, maio de 1977.

ÍNDICE REMISSIVO

A

abeliano, grupo, 65, 214-215
acesso aleatório, modo CTR, 131
ações do modelo USTAT, detecção de intrusão, 413
AddRoundKey, transformação, 105, 107
Adicionar chave, camada AK, 264
Adicionar chave, função, S-AES, 116
adjacência de transporte, 367
Advanced Encryption Standard (AES), 91-120
  AddRoundKey, transformação, 105
  aspectos de implementação, 108-110
  caixa-S, 98
  cifra, 95-110
  critérios de avaliação, 92-95
  estrutura, 97
  expansão de chave, 105-107
  MixColumns, transformações, 103-105, 109, 113
  origens, 92
  polinômios com coeficientes em GF(28), 112-114
  processador de 32 bits, 109-110
  processador de 8 bits, 108-109
  Rijndael, proposta, 95, 108
  ShiftRows, transformação, 102, 107
  simplificado, 114-120
  Substituir bytes (SubBytes), transformação, 98-102, 107
  transformações, 98-105, 107, 113
Agrawal-Kayal-Saxena (AKS), algoritmo, 173
algoritmo de descompressão, 344
algoritmo de escalonamento de chave, 60
algoritmo de geração de subchave, 45
algoritmo SHA-512, 253-257
  etapas de processamento, 253-255
  função round, 255-257
  lógica, 253-255
algoritmos, 18, 45, 45-46, 55-56, 60, 72-73, 105-107, 133-135, 158, 171-173, 183, 238. 281-283, 334-337, 343-344
  AES, expansão de chave, 105-107
  Agrawal-Kayal-Saxena (AKS), 173
  Assinatura digital (DSA), 281-283
  CMAC, 268-269
  compressão, 343-344
  criptografia, 18, 183
  criptográfico, S/MIME, 334-337
  Data Authentication, 238
  Data Encryption (DEA), 47
  decriptografia, 18, 183
  descompressão, 344
  deskewing, 158
  escalonamento de chave, 60
  Euclideanos, 72-73
  Feistel, 45-46
  geração de sub-chave, 45
  Hash seguro (SHA), 253-257
  HMAC, 265-268
  Miller-Rabin, 171-173
  RC4, 133-135
  Rivest-Shamir-Adleman (RSA), 268–280, 189, 202
  Whirlpool, 257-264
alvo de avaliação (TOE), 456-458
alvos de segurança (STs), 456-458
ameaças de acesso à informação, 13
ameaças de serviço, 13
ameaças, 6, 13
análise de tráfego, 6, 227
análise, facilidade de, 45
Anel (R), 66-67
anel comutativo, 67
ANSI X.9.17 PRNG, 156-157
Anti-Repetição, IPSec, 353, 355-356
aritmética de curva elíptica, 214-220
  curva binária, 217
  curva de primos, 217
  curvas sobre GF(2^m), 219-220
  curvas sobre Z_p, 217-219
  descrição algébrica da adição, 216
  descrição geométrica da adição, 215-216
  grupos abelianos, 214-215
  números reais, 215
aritmética modular, 67-72, 176-178
  divisores, 68
  logaritmos, 176-178
  operações, 69-70
  propriedades das congruências, 68
  propriedades, 70-72
aritmética polinomial, 76-80, 81-83, 112-114
  anel polinomial, 77
  coeficientes em GF(28), 112-114
  coeficientes em Z*p*, 77-79
  comum, 76-77
  máximo divisor comum, encontrando, 79-80
  modular, 81-83
  multiplicação por *x*, 114
  transformação MixColumns, 113
aritmética, 67-72, 76-80, 176-178, 214-220. *Ver também* teoria dos números
  curva elíptica, 214-220
  modular, 67-72, 176-178
  polinomial, 76-80
assinatura (SIG), payload, 368, 369
assinatura digital arbitrada, 274-275
assinatura digital direta, 273-274
assinatura dual, SET, 395-396
assinatura, 303, 304, 323, 327
  campo de confiança, 327
  componente, 323-324
  identificador de algoritmo, 303
assinaturas digitais, 272-285, 365
  algoritmo (DSA), 281-283
  arbitradas, 274-275
  chave Oakley, protocolo de troca, 366
  diretas, 273-374

padrão (DSS), 280-283
protocolos de autenticação, 275-279
requisitos, 273
associação de segurança (SA), 352-353, 361-363, 368
adjacência de transporte, 362
autenticação mais confidencialidade, 362
bundle de túnel de transporte, 362
combinações básicas, exemplos de, 362-363
combinando protocolos, 361-363
Internet Protocol Security (IPSec), 351-353
parâmetros, 352-353
payload, 367, 368
seletores, 353
ataque criptoanalítico, 20
Ataque de texto cifrado escolhido (CCA), 196-198
ataque de texto claro conhecido, 124
ataque man-in-the-middle, 210, 214
ataque meet-in-the-middle, 123, 243
ataque por força bruta, 20, 21, 243
Código de autenticação de mensagens (MAC), 244
funções de hash, 243
ataques ativos, 7-8
ataques de dia do aniversário, 241-242, 247-251
base matemática, 247-251
desigualdade, 249-250
duplicações, 250
paradox, 248, 249
problema relacionado, 248
sobreposição entre dois conjuntos, 250
ataques de temporização, 55, 195-196
ataques passivos, 6-7
ataques por repetição, 275
ataques suppress-replay, 277
ataques, 2, 6, 7-8, 20, 56-57, 123, 125, 140-141, 195-198, 210, 214, 228, 241-242, 243, 244, 245-246, 247-251, 276, 277, 438-441. *Ver também* software malicioso; vírus; vermes
ataque de texto cifrado escolhido (CCA), 196-198
ataques de dia do aniversário, 241-242, 247-251
ataques de negação de serviços distribuídos (DDoS), 455-458, 438-411
ativos, 6-7
confidencialidade, locais em, 140-141
criptoanálise diferencial, 56-57
criptoanálise, 20-22, 245-246
força bruta, 20, 21, 2433
identificação, 228
man-in-the-middle, 243
meet-in-the-middle, 123, 124
passivos, 6-7
repetição, 275
segurança, 3, 5, 6-8
suppress-replay, 277
temporização, 195-196
texto claro conhecido, 124
atraso aleatório, 196
atributos do emissor, 308
atributos de diretório, 308
identificador de chave, 308
identificador exclusivo, 303
informação de chave pública, 303
nome alternativo, 308
nome, 303
atributos do titular, 308
identificador exclusivo, 303
nome alternativo, 308
nome, 303
atualização do par de chaves, 310
autenticação de conta do proprietário do cartão, SET, 394
autenticação de entidade par, 9
Autenticação de transações de intercâmbio eletrônico de dados (EDI), 308
autenticação do comerciante, SET, 394
autenticação mútua, 275-279
técnicas de criptografia de chave pública, 278-279
técnicas de criptografia simétrica, 276-278
autenticação unidirecional, 279, 306-307
serviço X.509, 307
técnicas de criptografia de chave pública, 280
técnicas de criptografia simétrica, 279
autenticação, 9, 210-211, 226-251, 275-279, 289-314, 317, 351, 352, 355-358, 365
algoritmo, IPSec, 351
aplicações, 289-314
ataques de dia do aniversário, 241-242, 247-251
cabeçalho (AH), 351, 353, 355-358, 375
chave secreta, 210-211
código de autenticação de mensagens (MAC), 228, 232-233, 236-238, 244
códigos, 236-238
criptografia de mensagem, 228-230, 231
Data Authentication Algorithm, 238
duas vias, 307
funções de hash, 228, 233-236, 239-243, 243-246, 247-251
funções, 228-236
infra-estrutura de chave pública (PKI), 309-310
Kerberos, 290-302, 312-314
mútua, 275-276
Oakley, protocolo de determinação de chave, 366
origem de dados, 9
Pretty Good Privacy (PGP), 316, 319
protocolos, 275-280
requisitos, 227
segurança das funções de hash e MACs, 243-246
serviços de segurança de rede, 289-314
técnicas de criptografia por chave pública, 278-279, 280
técnicas de criptografia simétrica, 279
três vias, 307
uma via, 279, 306-307
X.509, 302-309
autoridade de certificação (CA), 309
autoridade de certificação (RA), 309
avalanche garantida (GA), 60

B

backdoor, 429
Banco de dados de políticas de segurança (SPD), 353
Base 64, codificação de transferência, 334
bastion host, 449-450
bijeção, 173
blinding, 196
Bloom, filtro, 421-422
Blum Blum Shub (BBS), gerador, 157
bomba lógica, 429
buffer de histórico deslizante, 343
buffer look-ahead, 343
bundle do túnel de transporte, 362

C

cabeçalho de autenticação (AH), 351, 353, 355-358, 375
  campos, 355
  definição, 351
  informação, 353
  IPv6, 375
  modo de transporte, 356-358
  modo túnel, 356-358
  serviço anti-repetição, 355-356
  valor de verificação de integridade (ICV), 356
cabeçalho de Fragmento, 375, 377
cabeçalho de opções de destino, 375, 377
cabeçalho de opções salto a salto, 375, 376
cabeçalho de roteamento, 375, 377
cabeçalho IPv6, 375
cabeçalhos de extensão, IPv6, 375-377
caixa-S, 52, 53, 59-60, 98-102, 119
  AES, 98-102
  papel, 52, 53
  projeto, 59-60
  S-AES, 119
cálculos criptográficos, 389, 392-393
  criação de segredo mestre, 389
  geração de parâmetros, 389
  SSL, 390
  TLS, 392
camada não linear SB, 261-262
campo de legitimidade de chave, 327
canal secreto, 145
carimbos de tempo, 276, 323, 324
cavalos de tróia, 429, 454-455
certificação cruzada, 310
certificação, PKIX, 310
certificados, 208, 303-306, 308, 339-340, 367, 369, 387-389, 392
  elementos, 303
  mensagens, 387-389, 392
  obtendo, 304-305
  payload, 367, 369
  políticas, 308
  processamento, 339-340
  reversos, 305
  revogação, 305-306
  tipos de cliente TLS, 392
  usuário, 304-305
Certificate Management Messages (CMC), 310
Certificate Management Protocols (CMP), 310
Certificate_verify, mensagem, 388, 392
César, cifra, 22-23
chave de sessão, 148, 150
chave mestra, definição, 147
chave pública, 308, 323-326
  criptossistemas, 182-183
  lista, 323-326
  período de uso, 308
  RSA, operação eficiente usando, 192
chave secreta, 18, 209-211
  ataque man-in-the-middle, 210
  autenticação, 210-211
  confidencialidade, 210-211
  distribuição, 209-210
  técnica híbrida, mainframes IBM, 211
chaves de 56 bits, uso, 54
chaves públicas, 230-231, 272-285, 326-330, 366. *Ver também* criptografia de chave pública
  autoridade, 207-208
  certificados, 208-209
  criptografia, 163-225, 230-231, 278-279, 280, 366
  criptografia, 272-285
  diretório, 207
  gerenciamento, 206-211
  operação eficiente de RSA usando, 192
  Pretty Good Privacy (PGP), 326-330
  revogando, 329
cifra em bloco W, 259-264
  adicionar camada AK, 264
  camada de difusão MR, 263-264
  CState, 260
  expansão de chave para, 264
  KState, 261
  camada de permutação SC, 263
  camada não linear SB, 261-262
  estrutura, 260-261
cifra inversa equivalente, 107-108
cifra polialfabética, 28-31
cifra, 18, 22-32, 40-63. *Ver também* cifras em bloco; cifras simétricas
  bloco, 40-63
  César, 22-32
  Feistel, 43-48
  fluxo, 41
  Hill, 27-28
  monoalfabéticas, 23-25
  one-time pad, 31-32
  Playfair, 25-26
  polialfabética, 28-31
  transposição, 32-33
  Vigenère, 29
cifras de bloco, 40-46, 58-63, 126-131
  Cipher Block Chaining (CBC), modo, 127-128
  Cipher Feedback (CFB), modo, 128-129
  counter (CTR), modo, 130-131
  Electronic Codebook (ECB), modo, 126-127
  ideal, 42
  modos de operação, 126-131
  motivação para estrutura Feistel, 41-43
  Output Feedback (OFB), modo, 129, 130
  princípios de projeto, 58-60
  princípios, 41-46
cifras de fluxo, 41, 132-135

algoritmo RC4, 133-135
considerações de projeto, 132
estrutura, 132-133
fluxo de chaves, 132
cifras monoalfabéticas, 23-25
cifras simétricas, 15-181
Advanced Encryption Standard (AES), 91-121
cifras de fluxo, 132-135
cifras em bloco, 62–94, 40-63, 126-131
confidencialidade, 139-161
corpos finitos, 64-90
criptografia múltipla e triple DES, 121-125
Data Encryption Standard (DES), 40-41, 46-63
modelo, 18-22
RC4, 132-135
técnicas de criptografia, 17-39
Cipher Block Chaining (CBC), modo, 238
Cipher feedback (CFB), modo, 128-129
classe, 472
codificações de transferência, MIME, 333
Código de autenticação de mensagens (MAC), 228, 232-233, 236-239, 383, 390
Algoritmo de autenticação de dados, 238
ataques de força bruta, 243
baseado em DES, 238
Cipher-based Message Authentication Code (CMAC), 268-269
criptoanálise, 245, 246
HMAC, 265-268
padrão da Internet (HMAC), 265-268
requisitos, 236-238
segurança, 243-246
SSL, 383
TLS, 396
uso, 232-235
códigos de alerta, TLS, 391-392
complementação de tráfego, 146
complementação, 146, 196-198, 359, 374, 392-393
cálculos criptográficos, 392
ESP, 359
Optimal Assymetric Encryption Padding (OAEP), 197-198
tráfego, 145-146
complexidade de tempo, 203
componente de chave de sessão, 323
componente, 473
compressão, 245, 319-320, 343-344
algoritmo, 343-344
função, 245
PGP, 319-320
Comprimento do cabeçalho de internet (IHL), 373
Computer Emergency Response Team (CERT), 4
comunicações store-and-forward, 144
conexão SSL, 381-382
confiabilidade da rede, 372
confiança no proprietário, campo, 327
confiança, 326-329
campo de assinatura, 327
campo de legitimidade de chave, 327
campo no proprietário, 327
exemplo de, 329
flags, 327, 328
uso de PGP, 326-327
confidencialidade de tráfego, 145-146
confidencialidade, 139-161, 317-319, 382, 394
ataques, locais em, 140-141
distribuição de chaves, 146-152
função de criptografia, posicionamento, 140-145
geração de número aleatório, 152, 152-157
Pretty Good Privacy (PGP), 317-319
redes locais (LANs), 141
SET, 394
SSL, 382
tráfego, 145-146
confusão, 43-44
congruência, propriedades, 68
conjunto de coeficientes, 76
conjuntos de cifras, TLS, 392
contador de números de seqüência, 352
controle de acesso, 9, 407,4183
controle de chave descentralizado, 151
controle de chave transparente, 150
controle de comportamento, 445
controle de direção, 444
controle de serviço, 444
controle de usuário, 445
corpos finitos (F), 64-90
algoritmo Euclidiano, 72-73
anéis, 65, 66-67
aritmética modular, 67-72
aritmética polinomial, 76-80
campo de Galois (GF), 73
corpos, 65, 67
da forma GF(2^n), 80-88
da forma GF(p), 73-76
grupos, 65-66
correio eletrônico (e-mail), segurança, 315-347
compatibilidade de e-mail, 320-321
compressão de dados, 342-344
conversão radix-64, 344-345
geração de número aleatório usando PGP, 345-347
Pretty Good Privacy (PGP), 287, 316-330, 345-347
Secure/Multipurpose Internet Mail Extension (S/MIME), 330-341
ZIP, compressão de dados usando, 342-344
counter (CTR), modo, 130-131
criação de segredo mestre, 389
criptoanálise diferencial, 56-58
criptoanálise linear, 57-58
criptoanálise, 18, 20-22, 56-58, 57-58, 188, 245
chave pública, 188
Códigos de autenticação de mensagens (MAC), 246
diferencial, 56-58
funções de hash, 245-246
linear, 57-58
tipos de ataques, 20
criptografar-decriptografar-criptografar (EDE), seqüência, 124
criptografia cíclica, 155-156
criptografia de chave pública, 163-225, 230-231, 278-279, 280, 366

algoritmo RSA, 188-198, 202
autenticação de mensagem, 230-231
criptografia de curva elíptica (ECC), 214-223
criptografia, 181-204
criptosistemas, 182-188, 205-225
gerenciamento de chave, 205-2117
protocolo de determinação de chave Oakley, 366
técnicas de algoritmo, 278-279, 280
teoria de números, 165-180
troca de chave Diffie-Hellman, 211-214
criptografia de chave pública, 272-285
assinaturas digitais, 273-275
Padrão de assinatura digital (DSS), 280-283
protocolos de autenticação, 275-280
Criptografia de curva elíptica (ECC), 221-223
criptografia/decriptografia, 222
Diffie-Hellman, acordo de chave, 221-222
segurança, 222-223
criptografia de enlace, 141-142
criptografia de mensagem, 228-230, 231
chave pública, 230-231
simétrica, 228-230
criptografia de ponta a ponta, 141-145
posicionamento da função, 142-145
técnicas básicas, 141-142
criptografia simétrica, 18-22, 228-230, 276-278, 279-280, 366
função de autenticação, 228-230
modelo de cifra, 18-22
protocolo de determinação de chave Oakley, 366
técnicas de autenticação, 276-278, 279-280
criptografia, 17-39, 46-54, 115, 116-118, 122-125, 140-145, 155, 228-231, 276-280, 312-314, 351, 383. *Ver também* cifra em bloco; Data Encryption Standard (DES); criptografia de chave pública
algoritmo, 18, 183, 351
camada de aplicação, 144
camada de rede, 142
chave pública, 163-225, 230-232
cíclica, 155
cifras de bloco, 40-46, 58-63, 389
cifras de fluxo, 383
computacionalmente seguro, 21
criptoanálise, 18, 20-22
criptografia, 18, 19
curvas elípticas, 222
Data Encryption Standard (DES), 40-41, 46-60, 122-123, 124-126
esteganografia, 34-35
função, posicionamento da, 140-145
gerenciamento de chave, 205-211
incondicionalmente seguro, 21
IPSec, 351
link, 141-142
máquinas de rotor, 33-35
mensagem, 228-231
múltipla, 122-125
Optimal Assymetric Encryption Padding (OAEP), 197
ponta a ponta, 141-145
protocolos de autenticação, 276-280
S-AES, 114-115, 116-118
simétrica, 18-22, 228-230, 276-278, 279-280, 366
SSL, 384
técnicas clássicas, 17-39
técnicas de autenticação de chave pública, 278-279, 280
técnicas de substituição, 22-32
técnicas de transposição, 32-33
técnicas Kerberos, 312-314
criptografia, 18, 19, 181-204
chave pública, 181-204
dimensões do sistema, 19
criptografia/decriptografia rápida de software, 45
criptologia, 18
criptosistemas, 182-188
aplicações, 187
chave pública, 182-188
criptoanálise, 188
princípios, 182-188
requisitos, 187-188
critério de avalanche estrito (SAC), 59
Critério de independência de Bit (BIC), 59
Critérios comuns (CC), 455-458
alvo de avaliação (TOE), 456-458
alvos de segurança (STs), 456-458
perfis de proteção (PPs), 456-458
requisitos, 455-456
CRL, emissor, 309
CRT, *ver* Teorema chinês do resto (CRT)
CState, 260
curva binária, 217
curva de primos, 217
curvas, *ver* aritmética de curva elíptica; criptografia de curva elíptica

D

dados, 10, 342-344, 353, 394, 451-453
autenticação da origem, 9
compactação usando ZIP, 342-344
confidencialidade, 10
controle de acesso, 451-453
integridade, 10, 394
nível de importância, 353
Data Encryption Algorithm (DEA), 72, 47
Data Encryption Standard (DES), 40-41, 46-63, 122-123, 124-125, 238
ataque de texto claro conhecido, 125
ataque man-in-the-middle, 123
ataques de temporização, 55
chaves de 56 bits, uso de, 54
criptoanálise diferencial, 56-57
criptoanálise linear, 57-58
criptografia múltipla, 122-123, 124-125
criptografia, 48-52
critérios de projeto, 58-59
Data Authentication Algorithm, 238
decriptografia, 52
duplo, 122-123
efeito avalanche, 52-54
força, 54-56
geração de chave, 51
Message Authentication Code (MAC), 238
natureza do algoritmo, 55
permutação inicial, 48

redução a único estágio, 123
rodada individual, detalhes, 50-51
triplo, 124-125
DDoS direto, 439
DDoS refletor, 439
DDoS, *ver* negação de serviços distribuídos (DDoS), ataques
decriptografia, 17, 18, 45-46, 52, 115, 116-118, 183, 222
algoritmo Feistel, 45-46
algoritmo, 18, 183
curvas elípticas, 222
DES, 52
S-AES, 165–166, 115, 116-118
desafio/resposta, 276
descriptografia genérica (GD), 436
deskewing, algoritmos, 157
deslocamento de fragmento, 373
detecção de intrusão baseada em regra, 409, 411-413
detecção de intrusão distribuída, 414-415
detecção de intrusão, 408-416
baseada em regra, 409, 411-413
distribuída, 414-415
estatística de anomalia, 409, 410-411
formato de troca, 415
mito da probabilidade de base, 413
registros de auditoria, 409-410
detecção estatística de anomalia, 409, 410-411
diálogo de autenticação, 291-292, 292-293, 293-296, 300-302
Kerberos versão 4, 291-292, 292-293, 293-296
Kerberos versão 5, 300-302
simples, 291-292
troca de serviço de autenticação, 300
Diffie-Hellman, acordo de chave, 211-214, 221-222, 387
algoritmo, 212-213
análogo, 221-222
anônimo, 387
ataque man-in-the-middle, 214
efêmero, 387
fixo, 387
protocolos, 213
RSA, 393-394
difusão, 43-44, 263-264
camada MR, 263-264
método, 43-44
Distância máxima separável (MDS), 264
distribuição de chave, 146-152, 278-279
autenticação, 276-278, 278-279
cenário, 148-149
centro (CDC), 148-149, 276
controlando o uso, 151-152
controle de chave descentralizado, 150, 151
controle de chave transparente, 150
técnica, 146-147
tempo de vida da chave da sessão, 149
divisores, 68
divulgação de conteúdo da mensagem, 6
divulgação, 227
domínio de integridade, 67
Domínio de interpretação (DOI), 351
DS/ECN, 373, 375

E

efeito avalanche, 52-54
eficiência do hardware, modo CTR, 131
eficiência do software, modo CTR, 131
Electronic Codebook (ECB), modo, 126-127
Encapsulamento de segurança do payload (ESP), 351, 353, 358-361, 362
algoritmos, criptografia e autenticação, 359
formato, 358
informação, 353
modo transporte, 359-361, 362
modo túnel, 361, 362
opção de autenticação, 362
endereço IP de destino, 353, 374, 375
endereço IP de origem, 353, 374, 375
entidade final, 309
EnvelopedData, 337-338
esquemas de endereçamento, 372
esteganografia, 34-35
estouro do contador de seqüência, 352
etiqueta de fluxo, 375
Euclidiano, algoritmo, 72-73
Euler, função tociente, 169, 170
Euler, teorema, 168, 170-171
Excusão (D), payload, 368, 369
expansão de chave, 105-107, 118, 264
algoritmo AES, 105-107
algoritmo S-AES, 118
cifra de bloco W, 264
extensões, 303

F

família, definição, 455
Feistel, cifra, 41-66
algoritmo de decriptografia, 45-46
confusão, 43-44
difusão, 43-44
estrutura, 44-45
motivação, 41-43
Fermat, teorema, 168-169
firewalls de inspeção com estado, 448
firewalls, 403, 443-460
características, 444-445
configurações, 450-451
Critérios comuns (CC), 455-458
princípios de projeto, 444-451
sistemas confiáveis, 451-455
tipos, 445-449
flags de ticket, 301-302
flags, 373
fluxo de chaves, 132
forma canônica, MIME, 334
Fortezza, esquema, 383, 387-388
função F, projeto, 59
função pseudo-aleatória (PRF), TLS, 390-391
função unidirecional, 407
funcionalidade da camada de transporte (TCP), 144
funções de hash, 228, 233-236, 239, 243-246, 247-251. *Ver também* HMAC; Algoritmo de Hash (SHA)
algoritmos, 252-257, 265-268
ataques de dia do aniversário, 241-242, 247-251

ataques por força bruta, 243
criptoanálise, 245-246
requisitos, 239
segurança, 243-246
SHA, 253-257
simples, 240-241
técnicas de encadeamento em bloco, 243
uso, 235
Whirlpool, 257-264

G

Galois, corpo (GF), 73
gateway em nível de aplicação, 449
gateway em nível de circuito, 449
geração de chave, 51, 193, 340
algoritmo DES, 51
algoritmo RSA, 188-198, 193, 194
S/MIME, 340
geração de fluxo, 134
geração de números aleatórios, 152, 345-347
aleatoriedade, 153
ANSI X.9.17 PGP, 345-347
ANSI X.9.17 PRNG, 156-157
criptografia cíclica, 155
criptograficamente, 155-156
gerador Blum Blum Shub (BBS), 156-157
gerador de número aleatório verdadeiro (TRNG), 157, 345
geradores congruentes lineares, 154-155
geradores de números pseudo-aleatórios (PRNGs), 153, 155-156, 156-157, 157, 345-347
imprevisibilidade, 153
Output Feedback (OFB), modo, DES, 155
Pretty Good Privacy (PGP), 345-347
uso, 152-153
gerador de bit pseudo-aleatório criptograficamente seguros (CSPRBG), 157
geradores congruenciais lineares, 154-155
geradores de números aleatórios verdadeiros (TRNG), 157, 345
geradores, 66, 86-87, 153-157
bit pseudo-aleatório criptograficamente seguros (CSPRBG), 157
Blum Blum Shub (BBS), 156-157
congruentes lineares, 154-155
número pseudo-aleatório (PRNGs), 153, 155-156, 156-157, 157
usando, 86-87
gerenciamento de chave pública, 326-330
confiança, uso de, 326-329
criptografia para distribuição de chave secreta, 209-211
distribuição, 206-210
Pretty Good Privacy (PGP), 326-330
técnicas de PGP, 326
gerenciamento de chave, 205-211, 351, 363-370
algoritmo RSA, 188-193
automatizado, 364
chaves públicas, 206-209
chaves secretas, 209-211
distribuição, 206-211
IPSec, 351, 363-370
manual, 364
Oakley, protocolo de determinação de chave, 364-366
Protocolo de gerenciamento de chaves e de associações de segurança na Internet (ISAKMP), 364, 366-370
gerenciamento de senha, 416-422
controle de acesso, 418
esquema do UNIX, 416-418
estratégias de seleção, 418-422
filtro Bloom, 427
gerado por computador, 419
modelo de Markov, 420-421
proteção, 416-418
treinamento do usuário, 419
verificação reativa, 419
verificador proaativo, 420
vulnerabilidade das senhas, 416-418
GF(2^m), curvas elípticas sobre, 219-220
GF(2^n), 80-88
adição, 85
aritmética polinomial modular, 81-83, 84
considerações computacionais, 83
corpos finitos na forma, 80-88
gerador usando, 86-87
inverso multiplicativo, encontrando, 83
motivação, 80-81
multiplicação, 85-86
GF(p), 73-76
corpos finitos da forma, 73-76
corpos finitos de ordem p, 73-74
inverso multiplicativo, 74-76
grupo cíclico, 66
grupo finito, 65
grupo infinito, 65
grupos (G), 65-66

H

hash (HASH), payload, 368, 369
Hill, cifra, 27-28
HMAC, 265-268
estrutura do algoritmo, 265-267
objetivos de projeto, 265
segurança, 267-268

I

ID da chave, 323, 324
ID de usuário, 324, 354
identificação (ID), payload, 367, 368
identificação, 373
identificador de chave de autoridade, 308
identificador de protocolo de segurança, 352
identificadores de chave, 321-323
Índice de parâmetros de segurança (SPI), 352
infra-estrutura de chave pública (PKI), 309-310
funções de gerenciamento, 310
protocolos de gerenciamento, 310
X.509 (PKIX), 309-310
inicialização, PKIX, 310
integridade da mensagem, SSL, 382
interfaces de rede, 372
interligação de redes, 372

International Telecommunication Union (ITU), 5
Internet Architecture Board (IAB), 4
Internet Engineering Task Force (IETF), 309
Internet protocol (IP), 348-377
IPv4, 373-374
IPv6, 374-377
papel, 372-373
segurança (IPSec), 348-377
unidade de protocolo de dados (PDU), 372
Internet Protocol Security (IPSec), 348-377
AAutenticação de cabeçalho (AH), 351, 353, 355-358
aplicações de roteamento, 350
aplicações, 349
arquitetura, 350-354
associação de segurança (SA), 352-353, 361-363
benefícios, 349
documentos, 350-351
Encapsulamento de segurança do payload (ESP), 351, 353, 358-361, 362
gerenciamento de chave, 351, 363-370
modo de protocolo, 353
modo transporte, 354, 356-358, 359-361, 362
modo túnel, 354, 356-358, 359, 361, 362
serviços, 351
visão geral, 349-350
intrusos, 403. *Ver também* detecção de intruso
detecção, 408-416
gerenciamento de senha, 416-422
infrator, 406
mascarado, 406
mito da probabilidade de base, 413, 424-426
técnicas, 407-408
usuário clandestino, 406
inverso multiplicativo, 74-75, 83
IP, endereço de destino, 352
IPv4, 373-374
IPv6, 374-377

K

Kerberos, 290-302, 312-314
diálogo de autenticação, 291-292, 292-293, 293-296, 300-301
diferenças entre versões, 298-300
domínio, 296-298
flags de ticket, versão 5, 301-302
justificativa de elementos, versão 4, 295-296
motivação, 290-291
principal, 297
técnicas de criptografia, 312-314
trocas de mensagem, versão 4, 294
versão 4, 291-298, 298-300
versão 5, 298-302
KState, 264

L

limite de altos, 375
lista de chave pública, 324
lista de chaves, 323-326
listas de correspondência seguras, 341
logaritmos discretos, 175-178
aritmética modular, 176-178
cálculo, 178
módulo n, 175-176
potências de um inteiro, 175-176

M

MAC, *ver* Código de autenticação de mensagens (MAC)
Mailing List Agent (MLA), 341
mapeamentos de política, 308
máquinas de rotor, 33-34
marcação de caractere, 35
Markov, modelo do processo, 420-421
mascarado, 406
mascaramento, 227
máximo divisor comum, 72-73, 79-80
algoritmo Euclidiano, 72-73
aritmética polinomial, 79-80
mecanismos de segurança, 5, 10-11
serviços de segurança, relacionamento entre, 11
X.800, 10-11
mensagem apenas com certificados, 339
mensagem, componente, 322
mensagens de conclusão, 389-392
mensagens PGP, transmissão e recepção, 324-326
Miller-Rabin, algoritmo, 171-173
MIME, *ver* Multipurpose Internet Mail Extensions (MIME)
misfeasor, 406
mito da probabilidade de base, 413, 424-426
demonstração, 426
independência, 424-425
probabilidade condicional, 424-425
problema, 413
teorema de Bayes, 425-426
MixColumns, transformações, 103-105, 107, 113
modelo de segurança de sessão (SSM), 150
modelo de série temporal, 411
modelo multivariado, 411
modelo operacional, 411
modificação de conteúdo, 227
modificação de mensagens, 7
modificação de seqüência, 227
modificação de tempo, 227
modo de transporte, 354, 356-358, 359-361, 362
AH, 355-358
ESP, 359, 361, 362
visão geral do IPSec, 354
modo hash, definição, 233
modo túnel, 354, 356-358, 359, 361, 362
AH, 355-358
ESP, 359, 361, 362
visão geral do IPSec, 354
módulo congruente (n), 68
módulo, definição, 67
Morris, verme, 450
MTU do caminho, 353
Multipurpose Internet Mail Extensions (MIME), 330-334
codificações de transferência, 333
forma canônica, 334
multipart, exemplo, 334

tipos de conteúdo, 331-333
visão geral, 331

N

não-repudiação, 10
National Institute of Standards and Technology (NIST), 46, 92-94, 253, 280
Negação de serviço distribuído (DDoS), ataques, 438-441
consome recursos de transmissão de dados, 455
contramedidas, 441
criando, 440-441
diretos, 439
processo de varredura, 440
recurso interno, 438
refletores, 439
negação de serviço, 7
New European Schemes for Signatures, Integrity, and Encryption (NESSIE), 257
nonce (NONCE), payload, 368, 369
nonce, 148, 371
notificação (N), payload, 368, 369
número de série, 303
número de versão do TLS, 390
números aleatórios gerados criptograficamente, 155-157
números primos, 166-168, 171-173
algoritmo de Miller-Rabin, 172-173
determinação, 166-168
distribuição, 173
propriedades, 171-172
testando, 171-173
números pseudo-aleatórios, 153, 155-156, 156-157, 346-347
ANSI X9.17, 155-156
gerador Blum Blum Shub (BBS), 156-157
geradores (PRNGs), 153, 155-156, 156-157, 346-347
PGP, geração usando, 346-347

O

Oakley, protocolo de determinação de chave, 364
características, 365
exemplo de troca, 366
métodos de autenticação, 365
troca de cookies, 365
one-time pad, 31-32
Open Systems Interconnection (OSI), 5, 142-143
arquitetura de segurança, 5
modelo, 142-143
Optimal Assymetric Encryption Padding (OAEP), 197
ordem, 65
Output Feedback (OFB), modo, 129, 130, 156

P

pagamento, SET, 396-400
autorização, 399-400
captação, 400
processamento, 396
solicitação de compra, 396-399
Papel do agente do usuário, 340
payload de encapsulamento de segurança, cabeçalho, 375
payload, 351, 353, 358-361, 361-362, 367-369, 375
Encapsulamento de segurança (ESP), 351, 353, 358-361, 362
tamanho, 375
tipos ISAKMP, 367-369
perfis de proteção (PPs), 456-458
perfurações, 35
período de validade, 303
permutação, 45, 48, 50, 263
inicial, 48, 49
SC da camada, 263
tabelas DES, 49
PGP, *ver* Pretty Good Privacy (PGP)
Playfair, cifra, 25-26
polinômio constante, 76
polinômio irredutível, 79
polinômio mônico, 76
polinômio primo, 79
portas, origem e destino, 359
pré-processamento, modo CTR, 131
pré-saída, definição, 48
Pretty Good Privacy (PGP), 287, 316-320, 342-344, 345-347
autenticação, 317, 319
compatibilidade de e-mail, 320-321
compressão de dados, 342-344
compressão, 319-320
confiança, uso de, 326-329
confidencialidade, 317-319
descrição operacional, 317-321
geração da chave de sessão, 321
geração de número aleatório, 345-347
gerenciamento de chave pública, 326-330
identificadores de chave, 321-323
lista de chave criptográficos, 321, 323-326
lista de chaves, 323-326
notação, 316
remontagem, 321
revogando chaves públicas, 329
segmentação, 321
ZIP, compressão de dados usando, 324-344
probabilidade condicional, 424-425
processador de 32 bits, implementação do AES, 109-110
processador de 8 bits, implementação do AES, 108-109
processador front-end (FEP), 143
projeto, 58-60, 265, 461-468, 443-451
algoritmo de escalonamento de chave, 60
caixa-S, 59-60
critérios DES, 58-59
firewalls, 443-451
função F, 59
HMAC, 265
princípios de cifra em bloco, 58-60
rodadas, número de, 59
proposta (P), payload, 368
Protocolo da camada de transporte, 353
Protocolo de alerta, 384
protocolo de estabelecimento de sessão, 385-389
Protocolo de gerenciamento de chaves e de associações de segurança na Internet (ISAKMP), 364, 366-370
formatos de cabeçalho, 366-367
tipos de payload, 367-369

trocas, 369-370
protocolo de mudança de especificação de cifra, 384
protocolo de registro, 382-384
protocolos, autenticação, 275-280
próximo cabeçalho, 375, 376

R

radix-64, conversão, 344-345
raiz primitiva, 176
raiz, polinômio, 86
raízes, 45, 50-51, 59
função, 45
isoladas, detalhes, 50-51
número de, 59
RC4, algoritmo, 133
força, 133
geração de fluxo, 134
inicialização de S, 133-134
lógica, 134
recibos assinados, 341
recuperação do par de chaves, 310
Rede, segurança, 3
redes locais (LANs), 140
registros de auditoria, 409-410
relativamente primo, 71
repetição, 7
repositório, 309
resíduo, 67, 70
resistência forte à colisões, 239, 244
resistência fraca à colisões, 239, 244
resistência q computação, 244
restrições básicas, 309
restrições de nome, 309
restrições de política, 309
restrições do caminho de certificação, 309
restrições, certificação X.509, 309
resumo de mensagem, 233
retratação da origem, 227
retratação do destino, 227
Revogação de certificação (CRL), 306
RFC 822, 330
Rijndael, proposta, 95, 108
Rivest-Shamir-Adleman (RSA), algoritmo, 188-279, 202-204
aspectos computacionais, 191-193
ataque de texto cifrado escolhido (CCA), 196-198
ataques de temporização, 195-196
complexidade, 203-204
descrição, 189-191
exponenciação na aritmética modular, 191-192
geração de chave, 193
operação eficiente, 192
Optimal Assymetric Encryption Padding (OAEP), 197
problema de fatoração, 194-195
prova, 202
segurança, 193-194
rodada individual, detalhes, 50-51
roteador de filtragem de pacotes, 445-448
rótulos de segurança, 341
RSA, algoritmo, *ver* Rivest-Shamir-Adleman (RSA), algoritmo

S

S/MIME, *ver* Secure/Multipurpose Internet Mail Extension (S/MIME)
S-AES, *ver* Simplified Advanced Encryption Standard (S-AES)
Secure Electronic Transaction (SET), 393-400
assinatura dual, 395-396
pagamento, 396-400
participantes, 394-395
recursos, 393-394
requisitos, 393
solicitação de compra, 396-399
tipos de transação, 397
visão geral, 393
Secure Hash Algorithms (SHA), 253-257
parâmetros, 253
SHA-512, 253-255
Secure Socket Layer (SSL), 381-390
arquitetura, 381-382
conexão, 381
criação de segredo mestre, 389
parâmetros criptográficos, geração, 389
Protocolo de alerta, 384
Protocolo de estabelecimento de sessão, 385-389
Protocolo de mudança de especificação de cifra, 384
Protocolo de registro, 382-384
sessão, 381
Secure/Multipurpose Internet Mail Extension (S/MIME), 330-341
algoritmos criptográficos, 336
assinatura às claras, 339
certificados VeriSign, 340, 341
entidade MIME, protegendo, 337-338
envelopedData, 338
funcionalidade, 334-337
funções, 334
listas de correspondência seguras, 341
Mailing List Agent (MLA), 341
mensagens apenas com certificados, 339
mensagens, 337-339
Multipurpose Internet Mail Extensions (MIME), 330-334
papel do agente do usuário, 340
processamento de certificado, 339-340
recibos assinados, 341
RFC 822, 330
rótulos de segurança, 341
serviços de segurança aperfeiçoados, 340
signedData, 338-339
solicitação de registro, 339
segurança da informação, 3
segurança de computador, 3
segurança de rede, 3, 12-13, 287-288, 289-314, 315-347, 348-377, 378-402
aplicações de autenticação, 289-314
correio eletrônico, 315-347
infra-estrutura de chave pública (PKI), 309-310
Internet Protocol (IP), 348-377
Kerberos, 290-302, 312-314
modelo, 12-13
tipos de aplicações, 287-288
Web, 378-402
X.509, 302-309

segurança demonstrável, modo de CTR, 131
segurança do sistema, 405-460
  firewalls, 443-460
  intrusos, 405-426
  software malicioso, 427-442
segurança Web, 378-402
  ameaças, 379, 380
  cálculos criptográficos, 389-390
  protocolo de alerta, 384
  protocolo de especificação de cifra de mudança, considerações, 379-380
  protocolo de estabelecimento de sessão, 385-389
  Secure Electronic Transaction (SET), 393-400
  Secure Socket Layer (SSL), 381-389
  técnicas de tráfego, 379, 380
  Transport Layer Security (TLS), 390-393
segurança, 2-14, 188-198, 222-223, 243-246, 405-460*Ver também* autenticação; segurança de rede; segurança de sistema
  algoritmo RSA, 193-198
  arquitetura OSI, 5
  ataques por força bruta, 243
  ataques, 2, 5, 6-8
  autenticação, 25-320
  código de autenticação de mensagem (MAC), 243-244
  computador, 3
  criptoanálise, 245-246
  criptografia de curva elíptica (ECC), 222-223
  funções de hash, 243-246
  informação, 3
  inter-rede, 3
  introdução, 2-16
  mecanismo, 5, 10-11
  segurança de rede, 3, 5, 12-13, 287-400
  segurança do sistema, 405-460
  serviços, 2, 5, 8-10
  tendências, 4-5
serviço de disponibilidade, 10
serviços de segurança aperfeiçoados, 340
serviços de segurança, 2, 5, 8-10
  autenticação, 11
  confidencialidade de dados, 10
  controle de acesso, 11
  definição, 2, 5
  disponibilidade, 11
  integridade de dados, 10
  mecanismos de segurança, relacionamento entre, 11
  não repudiação, 10
  X.800, 10-11
Servidor de concessão de ticket (TGS), 292-293
sessão SSL, 381-382
ShiftRows, transformação, 102, 107
SignedData, 338-339
simplicidade do modo CTR, 131
Simplified Advanced Encryption Standard (S-AES), 114-120
  caixa-S, 119
  decriptografia, 116-118
    criptografia, 116-118
  estrutura, 119-120
  expansão de chave, 118
  função de adicionar chave, 116
  função deslocamento de linhas, 117
  função embaralhamento de colunas, 117-118
  substituição de nibble, 117
  visão geral, 114-116
sistema digital imune, 436-437
sistemas confiáveis, 451-455
  conceito, 453-454
  controle de acesso a dados, 451-454
  defesa contra cavalo de tróia, 454-455
  definição, 453
software antivírus, 435-437
software de bloqueio de comportamento, 437-438
software malicioso, 403, 427-442
  backdoor, 429
  bomba lógica, 429
  cavalos de tróia, 429
  contramedidas, 435-438, 441
  introdução, 427-430
  Negação de serviço distribuído (DDoS), ataques, 438-441
  vermes, 433-434
  vírus, 429-433, 435-438
  zumbi, 429
Solicitação de certificado (CR), payload, 367, 369
solicitação de compra, SET, 396-399
solicitação de registro, 339
solicitação de revogação, 310
soma de verificação do cabeçalho, 374
substituição de nibble, 115-116
substituir bytes (SubBytes), transformação, 98-102, 107
subtipos MIME, 332-334
SunOS, eventos do sistema, detecção de intrusão, 413

T

tamanho de chave, 45
tamanho do bloco, 45
técnicas de encadeamento em bloco, 243
técnicas de substituição, 22-32
  cifra de César, 22-23
  cifra de Hill, 27-28
  cifra monoalfabética, 23-26
  cifra Playfair, 25-26
  cifra polialfabética, 28-31
  one-time pad, 31-32
técnicas de transporte, 32-33
tempo de exponenciação constante, 196
Tempo de vida (TTL), 373
tempo de vida da associação de segurança, 353
Teorema chinês do resto (CRT), 173-175, 192
teorema de Bayes, 425-426
teoria dos números, 165-180
  algoritmo de Miller-Rabin, 171-173
  distribuição de primos, 173
  função tociente de Euler, 169-170
  logaritmos discretos, 175-178
  números primos, 166-168
  teorema chinês do resto (CRT), 173-175

teorema de Euler, 168, 170-171
teorema de Fermat, 168-169
testando se o número é primo, 171-173
texto cifrado, 17, 18, 183
texto claro, 18, 183
tipos de certificado do cliente, TLS, 392
transformação (T), payload, 367, 368
transformações, 98-107, 107-108, 113, 116-120
AddRoundKey, 105
AES, 98-107, 107-108
caixa-S, 98-102, 199
cifra inversa equivalente, 107
encaminhamento, 98-101, 102, 103, 105
intercâmbio de AddRoundKey e InvMixColumns, 107
inversas, 98-102, 102, 103-105, 105
S-AES, 116-118
substituição de nibble, 117
substituir bytes (SubBytes), 145-149, 107
transformação ShiftRows, 102, 107
transformações MixColumns, 103-105, 113, 117
Transport Layer Security (TLS), 390-393
certificate_verify, mensagem, 392
código de autenticação de mensagem, 390
códigos de alerta, 391-392
conjuntos de cifra, 392
função pseudo-aleatória (PRF), 390-391
mensagens de conclusão, 392
número de versão, 390
tipos de certificado do cliente, 392
triplo EDS, 124-126
troca básica, 369
troca de chave (KE) 367, 368, 387-389
Diffie-Hellman, 211-214, 221-222, 387-389
Fortezza, 387-388
métodos, SSL, 390
payload, 367, 368
RSA, 387-389
troca de cookies, 365
troca de proteção de identidade, 369, 370
troca informativa, 369, 370
trocas ISAKMP, 369-370

U

uso de chave, 308
usuário clandestino, 406

V

Valor de verificação de integridade (ICV), 356-358
variável indeterminada (x), 76
VeriSign, certificados, 340, 341
vermes, 433-435
ataques recentes, 434
Morris, 433
tecnologia, estado da, 434
versão, 303, 373, 375
Vigenère, cifra, 29
vírus de e-mail, 433
vírus de macro, 432
vírus do setor de inicialização, 432
vírus furtivo, 432
vírus metamórfico, 432
vírus parasitário, 449
vírus polimórfico, 432
vírus residente na memória, 432
vírus, 429-433, 435-438
contramedidas, 435-438
decriptografia genérica (GD), 436
estrutura, 430-431
fases, 430
infecção inicial, 431
natureza, 429-431
sistema imune digital, 436-437
software de bloqueio de comportamento, 437-438
técnicas antivírus, 435-437
tipos, 432
vírus de e-mail, 433
vírus de macro, 432

W

Whirlpool, 257-264
características, 257
cifra em bloco W, 259-364
desvantagens, 257
estrutura de hash, 258-259
etapas de processamento, 259
fundamentos, 258

X

X.509, infra-estrutura de chave pública (PKIX), 309-310
X.509, serviço de autenticação, 302-309
certificados, 303-306
duas vias, 307
informação de chave, 308
informação de política, 308
procedimentos, 306-308
restrições do caminho de certificação, 309
três vias, 307
uma via, 306-307
versão 3, 307-308
X.800, recomendação da ITU-T, 5

Z

ZIP, 342-344
algoritmo de compressão, 343-344
algoritmo de descompressão, 344
compressão de dados usando, 342-344
Z_p, 114–117, 217-219, 77-79
coeficientes, 77-79
curvas elípticas sobre, 217-219
zumbi, 429

William Stallings

Criptografia e segurança de redes

4ª Edição

Princípios e práticas

Ciência da computação/Segurança de redes

Guia definitivo sobre o assunto, *Criptografia e segurança de redes*, 4ª edição, apresenta a teoria, a prática e os princípios para o desenvolvimento de sistemas que visam a segurança de computadores e redes, bem como das informações armazenadas e que trafegam nesses meios.

Seguindo o estilo didático característico de William Stallings, o livro inicialmente traz, em detalhes, conceitos e padrões criados e utilizados para a criptografia, tais como cifras, assinaturas digitais e criptografia de chaves pública e simétrica. Em seguida, aborda a parte prática da segurança de redes, apresentando os recursos de última geração para a segurança de autenticação, IP e Web. Por fim, discute os desafios da segurança de sistemas, explicando como se dão e como evitar os mais comuns e danosos ataques da atualidade.

Uma ampla gama de recursos e informações disponíveis on-line, no site exclusivo do livro, complementa a obra, indispensável para professores e estudantes de segurança de redes e criptografia, nos cursos de ciência da computação, e para qualquer profissional que queira se destacar no mercado de trabalho.

William Stallings é PhD em ciência da computação pelo MIT e fez uma contribuição única aos diferentes aspectos do desenvolvimento nas áreas de arquitetura e redes de computadores. É membro do *board* editorial da *Cryptologia*, conceituada revista acadêmica dedicada a todos os aspectos da criptologia, e venceu seis vezes o prêmio anual norte-americano Best Computer Science and Engineering Textbook, concedido pela Academic Authors Association. Como consultor independente, atende a diversas empresas privadas de desenvolvimento de hardware e software, bem como instituições de pesquisa governamentais. É autor do livro *Arquitetura e organização de computadores*, também publicado pela Pearson Education.

www.pearson.com.br/stallings
Site de apoio com apresentações em PowerPoint, respostas dos exercícios e recursos adicionais para professores e alunos.

ISBN 978-85-7605-119-0

www.pearson.com.br