

Segurança em redes de computadores

Na Seção 1.6, descrevemos algumas das categorias mais predominantes e prejudiciais dos ataques na Internet, incluindo ataques *malware*, recusa de serviço, análise de pacotes, disfarce da origem e modificação e exclusão de mensagem. Embora tenhamos aprendido muito sobre redes de computadores, ainda não analisamos como protegê-las desses ataques. Com nosso conhecimento recém-adquirido em rede de computadores e protocolos da Internet, estudaremos minuciosamente a comunicação segura e, em particular, como as redes podem ser protegidas desses vilões.

Queremos lhe apresentar Alice e Bob, duas pessoas que desejam se comunicar, porém “com segurança”. Como esse texto refere-se a redes, gostaríamos de observar que Alice e Bob podem ser dois roteadores que querem trocar tabelas de roteamento com segurança, um cliente e um servidor que querem estabelecer uma conexão de transporte segura, ou duas aplicações de e-mail que querem trocar e-mails com segurança – todos esses tópicos serão examinados mais adiante neste capítulo. Alice e Bob são componentes conhecidos da comunidade de segurança, talvez porque o nome deles seja mais interessante do que uma entidade genérica denominada “A” que quer se comunicar com segurança com uma entidade genérica denominada “B”. Amores proibidos, comunicações em tempo de guerra e transações financeiras são as necessidades dos seres humanos mais citadas quando o assunto é segurança nas comunicações; preferimos a primeira necessidade às duas últimas, e vamos usar, com muito prazer, Alice e Bob como nosso remetente e nosso destinatário e imaginá-los nesse primeiro cenário.

Dissemos que Alice e Bob querem se comunicar, porém “com segurança”, mas o que isso significa exatamente? Como veremos, a segurança (assim como o amor) é repleta de maravilhas; isto é, ela tem muitas facetas. É certo que Alice e Bob gostariam que o conteúdo de sua comunicação permanecesse secreto, a salvo de um bisbilhoteiro. Provavelmente, eles também gostariam de ter certeza de que estão se comunicando mesmo um com o outro e de que, caso algum bisbilhoteiro interfira na comunicação, essa interferência seja detectada. Na primeira parte deste capítulo, estudaremos as técnicas que permitem criptografar/decriptar comunicações, autenticar a parte com quem estamos nos comunicando e assegurar a integridade da mensagem.

Na segunda parte deste capítulo, examinaremos como os princípios da criptografia podem ser usados para criar protocolos de rede seguros. Utilizando mais uma abordagem top-down, examinaremos os protocolos seguros em cada uma das (quatro principais) camadas, iniciando pela camada de aplicação. Verificaremos como proteger o e-mail e uma conexão TCP (do inglês *Transmission Control Protocol* – Protocolo de Controle de Transmissão), como prover segurança total na camada de rede, e como proteger uma rede local (LAN, do inglês *local area network*) sem fio. Na terceira parte deste capítulo, avaliaremos a segurança operacional, que tem o objetivo de proteger redes organizacionais de ataques. Em particular, verificaremos, de forma minuciosa, como os firewalls e os sistemas de detecção de invasão podem aprimorar a segurança de uma rede organizacional.

8.1 O QUE É SEGURANÇA DE REDE?

Vamos iniciar nosso estudo de segurança de redes voltando aos namorados citados, Alice e Bob, que querem se comunicar “com segurança”. O que isso significa ao certo? Com certeza, Alice quer que somente Bob entenda a mensagem que ela enviou, mesmo que eles *estojam* se comunicando por um meio inseguro, em que um intruso (Trudy, a intrusa) pode interceptar qualquer dado que seja transmitido. Bob também quer ter certeza de que a mensagem que recebe de Alice foi de fato enviada por ela, e Alice quer ter certeza de que a pessoa com quem está se comunicando é de fato Bob. Alice e Bob também querem ter certeza de que o conteúdo de suas mensagens não foi alterado em trânsito. Também querem, antes de tudo, ter certeza de que podem se comunicar (i.e., de que ninguém lhes negue acesso aos recursos necessários à comunicação). Dadas essas considerações, podemos identificar as seguintes propriedades desejáveis da **comunicação segura**.

- *Confidencialidade*. Apenas o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida. O fato de intrusos conseguirem interceptar a mensagem exige, necessariamente, que esta seja **cifrada** de alguma maneira para impedir que seja entendida por um interceptador. Esse aspecto de confidencialidade é, provavelmente, o significado mais comumente percebido na expressão *comunicação segura*. Estudaremos técnicas de criptografia para cifrar e decifrar dados na Seção 8.2.
- *Integridade de mensagem*. Alice e Bob querem assegurar que o conteúdo de sua comunicação não seja alterado, por acidente ou por má intenção, durante a transmissão. Extensões das técnicas de soma de verificação que encontramos em protocolos de transporte e de enlace confiáveis podem ser utilizadas para proporcionar integridade à mensagem. Estudaremos autenticação do ponto de chegada e integridade da mensagem na Seção 8.3.
- *Autenticação do ponto final*. O remetente e o destinatário precisam confirmar a identidade da outra parte envolvida na comunicação – confirmar que a outra parte é de verdade quem alega ser. A comunicação pessoal entre seres humanos resolve facilmente esse problema por reconhecimento visual. Quando entidades comunicantes trocam mensagens por um meio pelo qual não podem ver a outra parte, a autenticação não é assim tão simples. Por que, por exemplo, você deveria acreditar que o e-mail que recebeu e que contém uma sentença afirmando que aquele e-mail veio de um amigo seu vem mesmo dele? Estudamos a autenticação do ponto final na Seção 8.4.
- *Segurança operacional*. Hoje quase todas as organizações (empresas, universidades etc.) possuem redes conectadas à Internet pública. Essas redes podem ser comprometidas por atacantes que ganham acesso a elas por meio da Internet pública. Os atacantes podem tentar colocar *worms* nos hospedeiros na rede, adquirir segredos corporativos, mapear as configurações internas da rede e lançar ataques de recusa de serviços (DoS, do inglês *Denial-of-Service*). Veremos na Seção 8.9 que os mecanismos operacionais, como firewalls e sistemas de detecção de invasão, são usados para deter ataques contra a rede de uma organização. Um firewall localiza-se entre a rede da organização e a rede

pública, controlando os acessos de pacote de e para a rede. Um sistema de detecção de invasão realiza uma “inspeção profunda de pacote”, alertando os administradores da rede sobre alguma atividade suspeita.

Agora que já determinamos o que significa segurança de rede, vamos considerar em seguida quais são, exatamente, as informações às quais um intruso pode ter acesso e quais ações podem ser executadas por ele. A Figura 8.1 ilustra o cenário. Alice, a remetente, quer enviar dados a Bob, o destinatário. Para trocar dados com segurança, além de atender aos requisitos de confidencialidade, autenticação e integridade de mensagens, Alice e Bob trocarão mensagens de controle e de dados (algo muito semelhante ao modo como remetentes e destinatários TCP trocam segmentos de controle e segmentos de dados). Todas ou algumas dessas mensagens costumam ser criptografadas. Conforme discutimos na Seção 1.6, um intruso passivo consegue, potencialmente, fazer o seguinte:

- *monitorar* – identificar e gravar as mensagens de controle e de dados no canal;
- *modificar, inserir ou eliminar* mensagens ou conteúdo de mensagens.

Como veremos, a menos que sejam tomadas contramedidas adequadas, essas capacidades permitem que um intruso monte uma grande variedade de ataques à segurança: monitorar comunicações (talvez roubando senhas e dados), fazer-se passar por outra entidade, sequestrar uma sessão em curso, recusar serviço a usuários legítimos da rede sobrecarregando os recursos do sistema e assim por diante. O CERT Coordination Center (CERT, 2020) mantém um resumo de ataques comunicados.

Agora que já temos certeza de que há ameaças reais à solta na Internet, quais são os equivalentes de Alice e Bob na Internet, esses nossos amigos que precisam se comunicar com segurança? Decerto, Bob e Alice podem ser dois usuários humanos em dois sistemas finais, por exemplo, uma Alice real e um Bob real que de fato querem trocar e-mails seguros. Eles podem também ser os participantes de uma transação de comércio eletrônico. Por exemplo, um Bob real pode querer transmitir com segurança o número de seu cartão de crédito a um servidor Web para comprar um produto pela rede. As partes que necessitam de uma comunicação segura podem fazer parte de uma infraestrutura da rede. Lembre-se de que o sistema de nomes de domínio (DNS, do inglês *domain name system*, veja a Seção 2.4) ou programas de roteadores que trocam informações de roteamento (veja o Capítulo 5) requerem comunicação segura entre dois participantes. O mesmo é válido para aplicações de gerenciamento de rede, um tópico que examinamos no Capítulo 5. Um intruso que conseguisse interferir em consultas do DNS (como discutido na Seção 2.4), processamentos de roteamento (Seções 5.3 e 5.4) ou funções de gerenciamento de rede (Seções 5.5 e 5.7) poderia causar uma devastação na Internet.

Estabelecida a estrutura, apresentadas algumas das definições mais importantes e justificada a necessidade de segurança na rede, vamos examinar a criptografia mais a fundo. Embora sua utilização para prover confidencialidade seja evidente por si só, veremos em breve que a criptografia também é essencial para prover autenticação do ponto final e integridade de mensagem – o que faz dela uma pedra fundamental da segurança na rede.

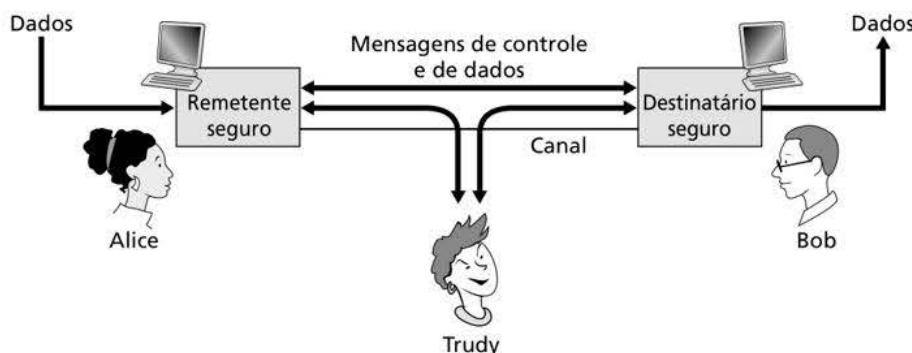


Figura 8.1 Remetente, destinatário e intruso (Alice, Bob e Trudy).

8.2 PRINCÍPIOS DE CRIPTOGRAFIA

Embora a criptografia tenha uma longa história que remonta, no mínimo, a Júlio César, técnicas modernas, incluindo muitas das usadas na Internet, são baseadas em progressos feitos nos últimos 30 anos. O livro de Kahn, *The Codebreakers* (Kahn, 1967), e o livro de Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* (Singh, 1999), nos oferecem um panorama fascinante dessa longa história. Uma discussão completa sobre a criptografia exige um livro inteiro (Bishop, 2003; Kaufman, 2002; Schneier, 2015); portanto, trataremos apenas de seus aspectos essenciais, em particular do modo como as técnicas criptográficas são postas em prática na Internet. Observamos também que, quanto nesta seção focalizemos a utilização da criptografia aplicada à confidencialidade, em breve veremos que as técnicas criptográficas estão inextricavelmente entrelaçadas com a autenticação, a integridade de mensagens, o não repúdio, etc.

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário, é claro, deve estar habilitado a recuperar os dados originais a partir dos dados disfarçados. A Figura 8.2 apresenta alguns dos componentes mais importantes da terminologia usada em criptografia.

Suponha agora que Alice queira enviar uma mensagem a Bob. A mensagem de Alice em sua forma original (p. ex., "Bob, I love you. Alice") é conhecida como **texto aberto** ou **texto claro**. Alice criptografa sua mensagem em texto aberto usando um **algoritmo de criptografia**, de modo que a mensagem criptografada, conhecida como **texto cifrado**, pareça ininteligível para qualquer intruso. O interessante é que em muitos sistemas criptográficos modernos, incluindo os usados na Internet, a técnica de codificação é *conhecida* – publicada, padronizada e disponível para qualquer um (p. ex., [RFC 1321; RFC 3447; RFC 2420; NIST, 2001]), mesmo para um potencial intruso! Evidentemente, se todos conhecem o método para codificar dados, então deve haver alguma informação secreta que impede que um intruso decifre os dados transmitidos. É aqui que entra a chave.

Na Figura 8.2, Alice fornece uma **chave**, K_A , uma cadeia de números ou de caracteres, como entrada para o algoritmo de criptografia. O algoritmo pega essa chave e o texto aberto da mensagem, m , como entrada e produz texto cifrado como saída. A notação $K_A(m)$ refere-se à forma do texto cifrado (criptografado usando a chave K_A) da mensagem em texto aberto, m . O algoritmo criptográfico propriamente dito, que usa a chave K_A , ficará evidente do próprio contexto. De maneira semelhante, Bob fornecerá uma chave, K_B , ao **algoritmo de decriptação**, que pega o texto cifrado e a chave de Bob como entrada e produz o texto aberto original como saída. Isto é, se Bob receber uma mensagem criptografada $K_A(m)$, ele a decriptará calculando $K_B(K_A(m)) = m$. Em **sistemas de chaves simétricas**, as chaves de

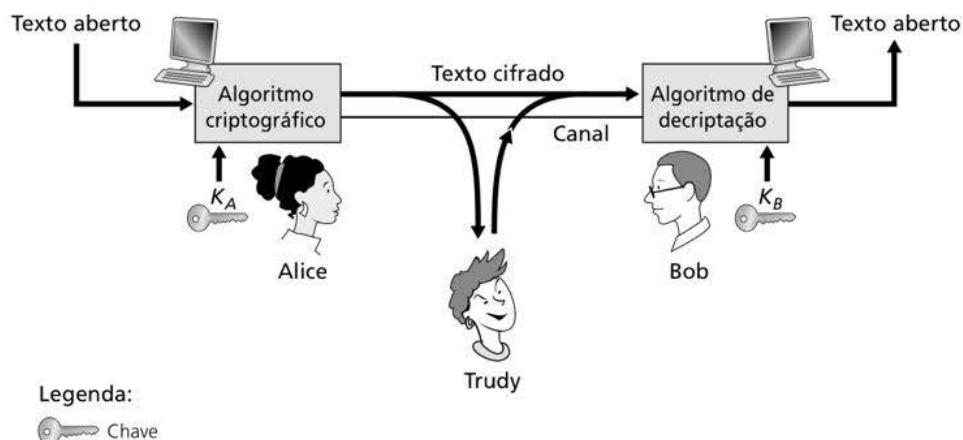


Figura 8.2 Componentes criptográficos.

Bob e de Alice são idênticas e secretas. Em **sistemas de chaves públicas**, é usado um par de chaves. Uma delas é conhecida por Bob e por Alice (na verdade, é conhecida pelo mundo inteiro). A outra chave é conhecida apenas por Bob ou por Alice (mas não por ambos). Nas duas subseções seguintes, examinaremos com mais detalhes sistemas de chaves simétricas e de chaves públicas.

8.2.1 Criptografia de chaves simétricas

Todos os algoritmos criptográficos envolvem a substituição de um dado por outro, como tomar um trecho de um texto aberto e então, calculando e substituindo esse texto por outro cifrado apropriado, criar uma mensagem cifrada. Antes de estudar um sistema criptográfico moderno baseado em chaves, vamos abordar um algoritmo de chaves simétricas muito antigo, muito simples, atribuído a Júlio César e conhecido como **cifra de César** (uma cifra é um método para codificar dados).

A cifra de César funciona tomando cada letra da mensagem do texto aberto e substituindo-a pela k -ésima letra sucessiva do alfabeto (permitindo a rotatividade do alfabeto, isto é, a letra “z” seria seguida novamente da letra “a”). Por exemplo, se $k = 3$, então a letra “a” do texto aberto fica sendo “d” no texto cifrado; “b” no texto aberto se transforma em “e” no texto cifrado, e assim por diante. Nesse caso, o valor de k serve de chave. Por exemplo, a mensagem “Bob, i love you. Alice” se torna “ere, l oryh brx. dolfh” em texto cifrado. Embora o texto cifrado na verdade pareça não ter nexo, você não levaria muito tempo para quebrar o código se soubesse que foi usada a cifra de César, pois há somente 25 valores possíveis para as chaves.

Um aprimoramento da cifra de César é a **cifra monoalfabética**, que também substitui uma letra do alfabeto por outra. Contudo, em vez de fazer isso seguindo um padrão regular (p. ex., substituição por um deslocamento de k para todas as letras), qualquer letra pode ser substituída por qualquer outra, contanto que cada letra tenha uma única substituta e vice-versa. A regra de substituição apresentada na Figura 8.3 mostra uma regra possível para codificar textos abertos.

A mensagem do texto aberto “Bob, I love you. Alice” se torna “nkn, s gktc wky. Mgsbc.” Assim, como aconteceu no caso da cifra de César, o texto parece sem nexo. A cifra monoalfabética também parece ser melhor que a cifra de César, pois há $26!$ (da ordem de 10^{26}) possíveis pares de letras, em vez de 25 pares possíveis. Uma técnica de força bruta que experimentasse todos os 10^{26} pares possíveis demandaria um esforço grande demais e impediria que esse fosse um método viável para quebrar o algoritmo criptográfico e decodificar a mensagem. Contudo, pela análise estatística da linguagem do texto aberto, por exemplo, sabendo que as letras “e” e “t” são as mais frequentes em textos em inglês (13% e 9% das ocorrências de letras, respectivamente) e sabendo que determinados grupos de duas e de três letras aparecem com bastante frequência em inglês (p. ex., “in”, “it”, “the”, “ion”, “ing”, e assim por diante), torna-se relativamente fácil quebrar esse código. Se o intruso tiver algum conhecimento sobre o possível texto da mensagem, então ficará mais fácil ainda quebrar o código. Por exemplo, se a intrusa Trudy for a esposa de Bob e suspeitar que ele está tendo um caso com Alice, ela poderá facilmente imaginar que os nomes “Bob” e “Alice” apareçam no texto. Se Trudy tivesse certeza de que esses dois nomes aparecem no texto cifrado e tivesse uma cópia do texto cifrado da mensagem do exemplo, então ela poderia determinar imediatamente sete dos 26 pares de letras, o que resultaria em 10^9 possibilidades a menos para verificar pelo método da força bruta. Na verdade, se Trudy

Letra do texto aberto: 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
Letra no texto cifrado: m n b v c x z a s d f g h j k l p o i u y t r e w q

Figura 8.3 Uma cifra monoalfabética.

suspeitasse que Bob estava tendo um caso, ela poderia muito bem esperar encontrar algumas outras palavrinhas especiais no texto também.

Considerando como seria fácil para Trudy quebrar o código criptográfico de Bob e Alice, podemos distinguir três cenários diferentes, dependendo do tipo de informação que o intruso tem.

- *Ataque exclusivo a texto cifrado.* Em alguns casos, o intruso pode ter acesso somente ao texto cifrado interceptado, sem ter nenhuma informação exata sobre o conteúdo do texto aberto. Já vimos como a análise estatística pode ajudar o **ataque exclusivo ao texto cifrado** em um esquema criptográfico.
- *Ataque com texto aberto conhecido.* Vimos anteriormente que, se Trudy, por alguma razão, tivesse certeza de que “bob” e “alice” apareciam no texto cifrado, ela poderia determinar os pares (texto cifrado, texto aberto) para as letras *a, l, i, c, e, b* e *o*. Trudy poderia também ser muito sortuda e ter gravado todas as transmissões de texto cifrado e descoberto uma versão decifrada pelo próprio Bob escrita em um pedaço de papel. Quando um intruso conhece alguns dos pares (texto aberto, texto cifrado), referimo-nos a isso como ataque ao esquema criptográfico **a partir de texto aberto conhecido**.
- *Ataque com texto aberto escolhido.* Nesse tipo de ataque, o intruso pode escolher a mensagem em texto aberto e obter seu texto cifrado correspondente. Para os algoritmos criptográficos simples que vimos até aqui, se Trudy conseguisse que Alice enviasse a mensagem “The quick brown fox jumps over the lazy dog”, ela poderia decifrar completamente o esquema criptográfico. Veremos em breve que, para técnicas de criptografia mais sofisticadas, um ataque com um texto aberto escolhido não significa necessariamente que a técnica criptográfica possa ser decifrada.

Quinhentos anos atrás, foram inventadas técnicas que aprimoravam a cifra monoalfabética, conhecidas como **cifras polialfabéticas**. A ideia subjacente à criptografia polialfabética é usar várias cifras monoalfabéticas com uma cifra monoalfabética específica para codificar uma letra em uma posição específica no texto aberto da mensagem. Assim, a mesma letra, quando aparece em posições diferentes no texto aberto da mensagem, pode ser codificada de maneira diferente. Um exemplo de esquema criptográfico polialfabético é mostrado na Figura 8.4, na qual há duas cifras de César (com $k = 5$ e $k = 19$), que aparecem nas linhas da figura. Podemos optar pelo uso dessas duas cifras de César, C_1 e C_2 , seguindo o modelo de repetição C_1, C_2, C_2, C_1, C_2 . Isto é, a primeira letra do texto deve ser cifrada usando-se C_1 , a segunda e a terceira, C_2 , a quarta, C_1 , e a quinta, C_2 . O modelo, então, se repete, com a sexta letra sendo cifrada usando-se C_1 , a sétima, com C_2 , e assim por diante. Dessa maneira, a mensagem em texto aberto “Bob, I love you.” é cifrada como “ghu, n etox dhz.”. Note que o primeiro “*b*” da mensagem em texto aberto é cifrado usando-se C_1 , ao passo que o segundo “*b*” é cifrado usando-se C_2 . Nesse exemplo, a “chave” da codificação e da decodificação é o conhecimento das duas cifras de César ($k = 5, k = 19$) e do modelo C_1, C_2, C_2, C_1, C_2 .

Cifras de bloco

Vamos agora nos direcionar a tempos modernos e analisar como a criptografia de chaves simétricas é feita atualmente. Focaremos em cifras de bloco, que são utilizadas em muitos protocolos seguros da Internet, incluindo PGP (do inglês *Pretty Good Privacy* – privacidade razoável) (para e-mail seguro), Segurança na Camada de Transporte (TLS, do inglês *Transport Layer Security*) (para conexões TCP seguras) e IPsec (para proteger o transporte da camada de rede).

Letra do texto aberto:	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
$C_1(k = 5)$:	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
$C_2(k = 19)$:	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

Figura 8.4 Uma cifra polialfabética que utiliza duas cifras de César.

Na cifra de bloco, a mensagem a ser criptografada é processada em blocos de k bits. Por exemplo, se $k = 64$, então a mensagem é dividida em blocos de 64 bits, e cada bloco é criptografado de maneira independente. Para criptografar um bloco, a cifra utiliza um mapeamento um para um para mapear o bloco de k bits de texto aberto para um bloco de k bits de texto cifrado. Vamos examinar um exemplo. Suponha que $k = 3$, de modo que a cifra de bloco mapeie entradas de 3 bits (texto aberto) para saídas de 3 bits (texto cifrado). Um possível mapeamento é determinado na Tabela 8.1. Observe que esse é um mapeamento um para um; ou seja, há uma saída diferente para cada entrada. Essa cifra de bloco divide a mensagem em blocos de até 3 bits e criptografa cada bloco de acordo com o mapeamento acima. Você deve verificar que a mensagem 010110001111 é criptografada para 101000111001.

Ainda no exemplo do bloco de 3 bits, observe que o mapeamento na Tabela 8.1 é um de muitos possíveis mapeamentos. Quantos deles existem? Para responder a essa questão, observe que um mapeamento nada mais é do que uma permutação de todas as possíveis entradas. Há $2^3 (= 8)$ possíveis entradas (relacionadas nas colunas de entrada). Elas podem ser permutadas em $8! = 40.320$ formas diferentes. Uma vez que cada permutação especifique um mapeamento, há 40.320 mapeamentos possíveis. Podemos ver cada mapeamento como uma chave – se Alice e Bob sabem o mapeamento (a chave), conseguem criptografar e decodificar as mensagens enviadas entre eles.

O ataque de força bruta para essa cifra é tentar decodificar o texto cifrado usando todos os mapeamentos. Com apenas 40.320 mapeamentos (quando $k = 3$), isso pode ser rapidamente realizado em um computador de mesa. Para impedir os ataques de força bruta, as cifras de bloco normalmente usam blocos muito maiores, consistindo em $k = 64$ bits ou ainda maior. Observe que o número de mapeamentos possíveis para uma cifra de bloco k geral é $2^k!$, o qual é extraordinário até mesmo para valores moderados de k (como $k = 64$).

Embora as cifras de bloco da tabela completa, como descritas, com valores moderados de k possam produzir esquemas robustos de criptografia de chaves simétricas, eles infelizmente são difíceis de implementar. Para $k = 64$ e para um determinado mapeamento, Alice e Bob precisariam manter uma tabela com 2^{64} valores de entrada, uma tarefa impraticável. Além disso, se Alice e Bob quiserem trocar chaves, cada um teria de renovar a tabela. Assim, a cifra de bloco da tabela completa, que fornece mapeamentos predeterminados entre todas as entradas e saídas (como no exemplo anterior), está simplesmente fora de cogitação.

Em vez disso, as cifras de bloco costumam utilizar funções que simulam, de maneira aleatória, tabelas permutadas. Um exemplo (adaptado de Kaufman [2002]) de tal função para $k = 64$ bits é mostrado na Figura 8.5. A função primeiro divide um bloco de 64 bits em 8 blocos, cada qual consistindo em 8 bits. Cada bloco de 8 bits é processado por uma tabela de 8 bits para 8 bits, a qual possui um tamanho controlável. Por exemplo, o primeiro bloco é processado pela tabela denotada por T_1 . Em seguida, os oito blocos de saída são reunidos em um bloco de 64 bits. As posições dos 64 bits no bloco são, então, permutadas para produzir uma saída de 64 bits. Essa saída é devolvida à entrada de 64 bits, onde se inicia outro ciclo. Após n ciclos, a função apresenta um bloco de 64 bits de texto cifrado. O objetivo de cada ciclo é fazer cada bit de entrada afetar a maioria (se não todos) dos bits finais de saída. (Se somente um ciclo fosse usado, um determinado bit de entrada afetaria somente 8 dos 64 bits de saída.) A chave para esse algoritmo das cifras de bloco seria as oito tabelas de permutação (admitindo que a função de permutação seja publicamente conhecida).

TABELA 8.1 Uma cifra de bloco de 3 bits específica

Entrada	Saída	Entrada	Saída
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

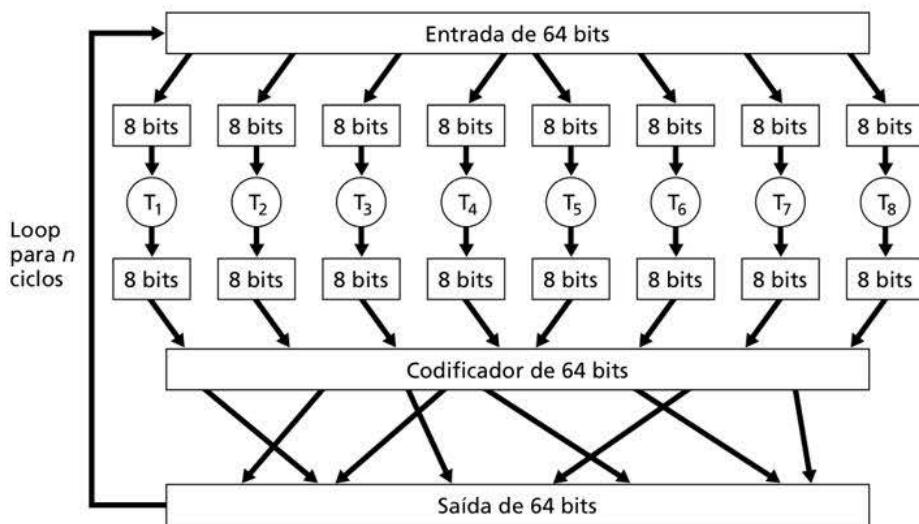


Figura 8.5 Exemplo de uma cifra de bloco.

Hoje, existem diversas cifras de bloco conhecidas, incluindo DES (do inglês *Data Encryption Standard* – Padrão de Criptografia de Dados), 3DES e AES (do inglês *Advanced Encryption Standard* – Padrão de Criptografia Avançada). Cada padrão utiliza funções em vez de tabelas predeterminadas, segundo a Figura 8.5 (embora mais complexa e específica para cada cifra). Cada um desses algoritmos também utiliza uma cadeia de bits para chave. Por exemplo, o DES usa blocos de 64 bits com uma chave de 56 bits. O AES usa blocos de 128 bits e pode operar com chaves de 128, 192 e 256 bits de comprimento. A chave de um algoritmo determina os mapeamentos da “minitabela” e permutações dentro do algoritmo. O ataque de força bruta para cada uma dessas cifras é percorrer todas as chaves, aplicando o algoritmo de decriptografia com cada chave. Observe que com o comprimento de chave n , há 2^n chaves possíveis. NIST (2001) estima que uma máquina que pudesse decifrar um DES de 56 bits em um segundo (i.e., testar 2^{56} chaves em um segundo) levaria, mais ou menos, 149 trilhões de anos para decifrar uma chave AES de 128 bits.

Encadeamento de blocos de cifras

Em aplicações de redes de computadores, em geral precisamos criptografar mensagens longas (ou fluxos de dados longos). Se aplicarmos uma cifra de bloco, como descrita, apenas partindo a mensagem em blocos de k bits e criptografando de modo independente cada bloco, um problema sutil mas importante ocorrerá. Para entendê-lo, note que dois ou mais blocos de texto aberto podem ser idênticos. Por exemplo, o texto aberto em dois ou mais blocos poderia ser “HTTP/1.1”. Em relação a esses blocos idênticos, uma cifra de bloco produziria, é claro, o mesmo texto cifrado. Um atacante poderia talvez adivinhar o texto aberto ao ver blocos de texto cifrado idênticos e ser capaz de decodificar a mensagem inteira identificando os blocos de texto cifrado idênticos e usando o que sabe sobre a estrutura do protocolo subjacente (Kaufman, 2002).

Para abordar esse problema, podemos associar um pouco de aleatoriedade ao texto cifrado para que blocos de texto aberto idênticos produzam blocos de texto cifrado diferentes. Para explicar essa ideia, $m(i)$ representará o i -ésimo bloco de texto aberto, $c(i)$ representará o i -ésimo bloco de texto cifrado, e $a \oplus b$ representará o ou exclusivo (XOR, do inglês *exclusive-or*) das duas cadeias de bits, a e b . (Lembre-se de que $0 \oplus 0 = 1 \oplus 1 = 0$ e $0 \oplus 1 = 1 \oplus 0 = 1$, e o XOR das duas cadeias de bits é feito em uma base de bit por bit. Então, p. ex., $10101010 \oplus 11110000 = 01011010$.) Ademais, denote o algoritmo de criptografia da cifra de bloco com a chave S como K_S . Eis a ideia básica. O emissor cria um número aleatório $r(i)$ de k bits para o i -ésimo bloco e calcula $c(i) = K_S(m(i) \oplus r(i))$. Observe que um novo

número aleatório de k bits é escolhido para cada bloco. O emissor envia, então, $c(1), r(1), c(2), r(2), c(3), r(3)$ etc. Visto que o receptor recebe $c(i)$ e $r(i)$, ele pode recuperar cada bloco de texto aberto computando $m(i) = K_s(c(i)) \oplus r(i)$. É importante observar que, embora $r(i)$ seja enviado de modo inocente e, portanto, possa ser analisado por Trudy, ela não consegue obter blocos de texto aberto $m(i)$, pois não conhece a chave K_s . Observe também que se dois blocos de texto aberto $m(i)$ e $m(j)$ são iguais, os blocos de texto cifrado correspondentes $c(i)$ e $c(j)$ serão diferentes (contanto que os números $r(i)$ e $r(j)$ aleatórios sejam diferentes, o que acontece com alta probabilidade).

Como um exemplo, considere a cifra de bloco de 3 bits na Tabela 8.1. Suponha que o texto aberto seja 010010010. Se Alice criptografá-lo diretamente, sem incluir a aleatoriedade, o texto cifrado resultante se torna 101101101. Se Trudy analisar esse texto cifrado, como cada uma das três cifras de blocos é igual, ela pode pensar que cada um dos blocos de texto aberto são iguais. Agora suponha que em vez disso Alice cria blocos aleatórios $r(1) = 001$, $r(2) = 111$ e $r(3) = 100$ e use a técnica anterior para criar o texto cifrado $c(1) = 100$, $c(2) = 010$ e $c(3) = 000$. Observe que os três blocos de texto cifrado são diferentes mesmo se os blocos de texto aberto forem iguais. Alice então envia $c(1), r(1), c(2)$ e $r(2)$. Você deve verificar que Bob pode obter o texto aberto original usando a chave K_s compartilhada.

O leitor atento observará que introduzir a aleatoriedade resolve o problema, mas cria outro: ou seja, Alice deve transmitir duas vezes mais bits que antes. De fato, para cada bit de cifra, ela deve agora enviar um bit aleatório, dobrando a largura de banda exigida. Para obtermos o melhor dos dois mundos, as cifras de bloco em geral usam uma técnica chamada **Encadeamento do Bloco de Cifra (CBC)**, do inglês *Cipher Block Chaining*). A ideia básica é enviar somente *um valor aleatório junto com a primeira mensagem e, então, fazer o emissor e o receptor usarem blocos codificados em vez do número aleatório subsequente*. O CBC opera da seguinte forma:

1. Antes de criptografar a mensagem (ou o fluxo de dados), o emissor cria uma cadeia de k bits chamada **Vetor de Inicialização (IV, do inglês Initialization Vector)**. Indique esse vetor de inicialização por $c(0)$. O emissor envia o IV ao receptor *em texto aberto*.
2. Em relação ao primeiro bloco, o emissor calcula $m(1) \oplus c(0)$, ou seja, calcula o XOR do primeiro bloco de texto aberto com o IV. Ele então codifica o resultado através do algoritmo de cifra de bloco para obter o bloco de texto cifrado correspondente; isto é, $c(1) = K_s(m(1) \oplus c(0))$. O emissor envia o bloco criptografado $c(1)$ ao receptor.
3. Para o i -ésimo bloco, o emissor cria o i -ésimo bloco de texto cifrado de $c(i) = K_s(m(i) \oplus c(i - 1))$.

Vamos agora examinar algumas das consequências dessa abordagem. Primeiro, o receptor ainda será capaz de recuperar a mensagem original. De fato, quando o receptor recebe $c(i)$, ele o decodifica com K_s para obter $s(i) = m(i) \oplus c(i - 1)$; uma vez que o receptor também conhece $c(i - 1)$, ele então obtém o bloco de texto aberto de $m(i) = s(i) \oplus c(i - 1)$. Segundo, mesmo se dois blocos de texto aberto forem idênticos, os texto cifrados correspondentes (quase sempre) serão diferentes. Terceiro, embora o emissor envie o IV aberto, um invasor ainda não será capaz de decodificar os blocos de texto cifrado, visto que o invasor não conhece a chave secreta, S . Por fim, o emissor somente envia um bloco auxiliar (o IV), fazendo aumentar, de forma insignificante, o uso da largura de banda para longas mensagens (consistindo em centenas de blocos).

Como um exemplo, vamos determinar o texto cifrado para uma cifra de bloco de 3 bits na Tabela 8.1 com texto aberto 010010010 e $IV = c(0) = 001$. O emissor primeiro usa o IV para calcular $c(1) = K_s(m(1) \oplus c(0)) = 100$. O emissor calcula, então, $c(2) = K_s(m(2) \oplus c(1)) = K_s(010 \oplus 100) = 000$ e $c(3) = K_s(m(3) \oplus c(2)) = K_s(010 \oplus 000) = 101$. O leitor deve verificar que o receptor, conhecendo o IV e K_s , pode recuperar o texto aberto original.

O CBC possui uma consequência importante ao projetar protocolos de rede seguros: precisaremos fornecer um mecanismo dentro do protocolo para distribuir o IV do emissor ao receptor. Veremos como isso é feito para vários protocolos mais adiante, neste capítulo.

8.2.2 Criptografia de chave pública

Por mais de 2 mil anos (desde a época da cifra de César até a década de 1970), a comunicação cifrada exigia que as duas partes comunicantes compartilhassem um segredo – a chave simétrica usada para cifrar e decifrar. Uma dificuldade dessa abordagem é que as duas partes têm de concordar, de alguma maneira, com a chave compartilhada, mas, para fazê-lo, é preciso comunicação segura. Talvez as partes pudessem se encontrar antes, escolher a chave (p. ex., dois dos centuriões de César poderiam se encontrar nos banhos romanos) e, mais tarde, se comunicar de modo cifrado. Em um mundo em rede, contudo, o mais provável é que as partes comunicantes nunca possam se encontrar e jamais consigam conversar a não ser pela rede.

É possível que elas se comuniquem por criptografia sem compartilhar uma chave comum secreta conhecida com antecedência? Em 1976, Diffie e Hellman (Diffie, 1976) apresentaram um algoritmo (conhecido como Troca de Chaves Diffie-Hellman – Diffie-Hellman Key Exchange) que faz exatamente isso – uma abordagem da comunicação segura radicalmente diferente e de uma elegância maravilhosa que levou ao desenvolvimento dos atuais sistemas de criptografia de chaves públicas. Veremos em breve que os sistemas de criptografia de chaves públicas também têm diversas propriedades maravilhosas que os tornam úteis não só para criptografia, mas também para autenticação e assinaturas digitais. É interessante que veio à luz que ideias semelhantes às de (Diffie, 1976) e às da (RSA, 1978) foram desenvolvidas independentemente no início da década de 1970 em uma série de relatórios secretos escritos por pesquisadores do Grupo de Segurança para Comunicação e Eletrônica (Communications-Electronics Security Group) do Reino Unido (Ellis, 1987). Como acontece com frequência, grandes ideias podem surgir de modo independente em diversos lugares; felizmente, os progressos da criptografia de chaves públicas ocorreram não apenas no âmbito privado, mas também no público.

A utilização de criptografia de chaves públicas, como conceito, é bastante simples. Suponha que Alice queira se comunicar com Bob. Como ilustra a Figura 8.6, em vez de Bob e Alice compartilharem uma única chave secreta (como no caso dos sistemas de chaves simétricas), Bob (o destinatário das mensagens de Alice) tem duas chaves – uma **chave pública**, que está à disposição do mundo *todo* (inclusive de Trudy, a intrusa), e uma **chave privada**, que apenas ele (Bob) conhece. Usaremos a notação K_B^+ e K_B^- para nos referirmos às chaves pública e privada de Bob, respectivamente. Para se comunicar com Bob, Alice busca primeiro a chave pública de Bob. Em seguida, ela criptografa sua mensagem, m , usando a chave pública de Bob e um algoritmo criptográfico conhecido (p. ex., padronizado), isto é, Alice calcula $K_B^+(m)$. Bob recebe a mensagem criptografada de Alice e usa sua chave privada e um algoritmo de decriptação conhecido (p. ex., padronizado) para decifrar a mensagem de Alice, isto é, Bob calcula $K_B^-(K_B^+(m))$. Veremos, mais adiante, que há algoritmos e

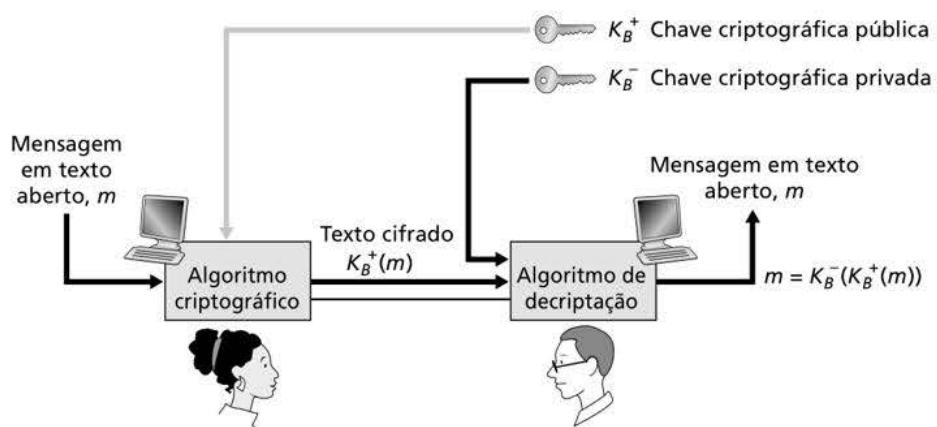


Figura 8.6 Criptografia de chaves públicas.

técnicas de criptografia/decriptação para escolher chaves públicas e privadas de modo que $K_B^-(K_B^+(m)) = m$; isto é, aplicando a chave pública de Bob, K_B^+ , à mensagem m , para obter $K_B^+(m)$, e então aplicando a chave privada de Bob, K_B^- , à versão criptografada de m , isto é, calculando $K_B^-(K_B^+(m))$, obtemos m novamente. Esse resultado é notável! Dessa maneira, Alice pode utilizar a chave de Bob disponível publicamente para enviar uma mensagem secreta a Bob sem que nenhum deles tenha de permutar nenhuma chave secreta! Veremos em breve que nós podemos permutar a chave pública e a chave privada de criptografia e obter o mesmo resultado notável, isto é, $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$.

Embora a criptografia de chave pública seja atraente, uma preocupação vem imediatamente à mente. Como a chave criptográfica de Bob é pública, qualquer um pode enviar uma mensagem cifrada a Bob, incluindo Alice ou alguém *afirmando* ser Alice. No caso de uma única chave secreta compartilhada, o fato de o remetente conhecer a chave secreta identifica implicitamente o remetente para o destinatário. No caso da criptografia de chave pública, contudo, isso não acontece, já que qualquer um pode enviar uma mensagem cifrada a Bob usando a chave dele, que está publicamente disponível a todos. É preciso uma assinatura digital, um tópico que estudaremos na Seção 8.3, para vincular um remetente a uma mensagem.

RSA

Embora existam muitos algoritmos e chaves que tratam dessas preocupações, o **algoritmo RSA** (cujo nome se deve a seus inventores, Ron Rivest, Adi Shamir e Leonard Adleman) tornou-se quase um sinônimo de criptografia de chave pública. Vamos, primeiro, ver como o RSA funciona e, depois, examinar por que ele funciona.

O RSA faz uso extensivo das operações aritméticas usando a aritmética de módulo- n . Vamos revisar de maneira breve a aritmética modular. Lembre-se de que $x \bmod n$ simplesmente significa o resto de x quando dividido por n , de modo que, por exemplo, $19 \bmod 5 = 4$. Na aritmética modular, uma pessoa executa as operações comuns de adição, multiplicação e exponenciação. Entretanto, o resultado de cada operação é substituído pelo resto inteiro que sobra quando o resultado é dividido por n . A adição e a multiplicação com a aritmética modular são facilitadas com as seguintes propriedades úteis:

$$\begin{aligned} [(a \bmod n) + (b \bmod n)] \bmod n &= (a + b) \bmod n \\ [(a \bmod n) - (b \bmod n)] \bmod n &= (a - b) \bmod n \\ [(a \bmod n) \cdot (b \bmod n)] \bmod n &= (a \cdot b) \bmod n \end{aligned}$$

Segue da terceira propriedade que $(a \bmod n)^d \bmod n = a^d \bmod n$, uma identidade que em breve acharemos muito útil.

Agora suponha que Alice queira enviar a Bob uma mensagem criptografada por meio do RSA, conforme ilustrado na Figura 8.6. Em nossa discussão sobre o RSA, vamos sempre manter em mente que uma mensagem não é nada mais do que um padrão de bits, e cada padrão de bits pode ser representado unicamente por um número inteiro (junto com o comprimento do padrão de bits). Por exemplo, suponha que uma mensagem tenha o padrão de bits 1001; essa mensagem pode ser representada pelo número inteiro decimal 9. Assim, criptografar uma mensagem com RSA é equivalente a criptografar um número inteiro que representa a mensagem.

Existem dois componentes inter-relacionados do RSA:

- A escolha da chave pública e da chave privada.
- O algoritmo de criptografia/decriptação.

Para escolher as chaves pública e privada no RSA, Bob deve executar as seguintes etapas:

1. Escolher dois números primos grandes, p e q . Que ordem de grandeza devem ter p e q ? Quanto maiores os valores, mais difícil será quebrar o RSA, mas mais tempo se levará

para realizar a codificação e a decodificação. O RSA Laboratories recomenda que o produto de p e q seja da ordem de 1.024 bits. Para uma discussão sobre como achar números primos grandes, consulte Caldwell (2012).

2. Calcular $n = pq$ e $z = (p - 1)(q - 1)$.
3. Escolher um número e menor do que n que não tenha fatores comuns (exceto o 1) com z . (Nesse caso, dizemos que e e z são números primos entre si.) A letra “ e ” é usada já que esse valor será utilizado na criptografia (“*encryption*”, em inglês).
4. Achar um número d , tal que $ed - 1$ seja divisível exatamente (i.e., não haja resto na divisão) por z . A letra “ d ” é usada porque seu valor será utilizado na decriptação. Em outras palavras, dado e , escolhemos d tal que

$$ed \bmod z = 1$$

5. A chave pública que Bob põe à disposição de todos, K_B^+ , é o par de números (n, e) ; sua chave privada, K_B^- , é o par de números (n, d) .

A criptografia feita por Alice e a decriptação feita por Bob acontecem da seguinte forma:

- Suponha que Alice queira enviar a Bob um padrão de bits, ou número m , tal que $m < n$. Para codificar, Alice calcula a potência m^e e, então, determina o resto inteiro da divisão de m^e por n . Assim, o valor cifrado, c , da mensagem em texto aberto de Alice, m , é:

$$c = m^e \bmod n$$

O padrão de bits correspondente a esse texto cifrado c é enviado a Bob.

- Para decifrar a mensagem em texto cifrado recebida, c , Bob calcula

$$m = c^d \bmod n$$

que exige o uso de sua chave secreta (n, d) .

Como exemplo simples de RSA, suponha que Bob escolha $p = 5$ e $q = 7$. (Admitimos que esses valores são muito pequenos para ser seguros.) Então, $n = 35$ e $z = 24$. Bob escolhe $e = 5$, já que 5 e 24 não têm fatores comuns. Por fim, ele escolhe $d = 29$, já que $5 \cdot 29 - 1$ (i.e., $ed - 1$) é divisível exatamente por 24. Ele divulga os dois valores, $n = 35$ e $e = 5$, e mantém em segredo o valor $d = 29$. Observando esses dois valores públicos, suponha que Alice queira agora enviar as letras l, o, v e e a Bob. Interpretando cada letra como um número entre 1 e 26 (com a sendo 1 e z sendo 26), Alice e Bob realizam a criptografia e a decriptação mostradas nas Tabelas 8.2 e 8.3, respectivamente. Observe que, neste exemplo, consideramos cada uma das quatro letras como uma mensagem distinta. Um exemplo mais realista seria converter as quatro letras em suas representações ASCII de 8 bits e então codificar o número inteiro correspondente ao padrão de bits de 32 bits resultante. (Esse exemplo realista cria números muito longos para publicar neste livro!)

Dado que o exemplo fictício das Tabelas 8.2 e 8.3 já produziu alguns números extremamente grandes, e visto que sabemos, porque vimos antes, que p e q devem ter, cada um, algumas centenas de bits de comprimento, várias questões práticas nos vêm à mente no caso do RSA. Como escolher números primos grandes? Como escolher e e d ? Como calcular exponenciais de números grandes? A discussão desses assuntos está além do escopo deste livro; consulte Kaufman (2002) e as referências ali citadas para obter mais detalhes.

Chaves de sessão

Observamos aqui que a exponenciação exigida pelo RSA é um processo que consome tempo considerável. Como resultado, o RSA é frequentemente usado na prática em combinação com a criptografia de chaves simétricas. Por exemplo, se Alice quer enviar a Bob uma

TABELA 8.2 Criptografia RSA para Alice, $e = 5, n = 35$

Letra do texto aberto	m : representação numérica	m^e	Texto cifrado $c = m^e \text{ mod } n$
I	12	248832	17
O	15	759375	15
V	22	5153632	22
E	5	3125	10

TABELA 8.3 Decriptação RSA para Bob, $d = 29, n = 35$

Texto cifrado c	c^d	$m = c^d \text{ mod } n$	Letra do texto aberto
17	4819685721067509150915091411825223071697	12	I
15	127834039403948858939111232757568359375	15	O
22	851643319086537701956194499721106030592	22	V
10	100	5	E

grande quantidade de dados cifrados a alta velocidade, ela pode fazer o seguinte. Primeiro, ela escolhe uma chave que será utilizada para codificar os dados em si; essa chave às vezes é denominada **chave de sessão**, representada por K_S . Alice deve informar a Bob essa chave de sessão, já que essa é a chave simétrica compartilhada que eles usarão com uma cifra de chave simétrica (p. ex., DES ou AES). Alice criptografa o valor da chave de sessão usando a chave pública RSA de Bob, isto é, ela processa $c = (K_S)^e \text{ mod } n$. Bob recebe a chave de sessão codificada RSA, c , e a decifra para obter a chave de sessão K_S . Ele agora conhece a chave que Alice usará para transferir dados cifrados.

Por que o RSA funciona?

A criptografia/decriptação do RSA parece mágica. Por que será que, aplicando o algoritmo de criptografia e , em seguida, o de decriptação, podemos recuperar a mensagem original? Para entender por que o RSA funciona, tome de novo $n = pq$, onde p e q são os números primos grandes usados no algoritmo RSA.

Lembre-se de que, na criptografia RSA, uma mensagem (representada exclusivamente por um número inteiro) m é elevada primeiro à potência e usando-se aritmética de módulo n , ou seja,

$$c = m^e \text{ mod } n$$

A decriptação é feita elevando-se esse valor à potência d , novamente usando a aritmética de módulo n . O resultado de uma etapa de criptografia, seguida de uma etapa de decriptação, é então $(m^e \text{ mod } n)^d \text{ mod } n$. Vamos ver agora o que podemos dizer sobre essa quantidade. Como mencionado antes, uma propriedade importante da aritmética modular é $(a \text{ mod } n)^d \text{ mod } n = a^d \text{ mod } n$ para quaisquer valores a , n e d . Então, usando $a = m^e$ nesta propriedade, temos:

$$(m^e \text{ mod } n)^d \text{ mod } n = m^{ed} \text{ mod } n$$

Portanto, falta mostrar que $m^{ed} \text{ mod } n = m$. Embora estejamos tentando eliminar um pouco da mágica do modo de funcionamento do RSA, para explicá-lo, precisaremos usar, aqui, outro resultado bastante mágico da teoria dos números. Especificamente, precisamos

do resultado que diga que, se p e q forem primos, $n = pq$ e $z = (p - 1)(q - 1)$, então $x^y \bmod n$ será o mesmo que $x^{(y \bmod z)} \bmod n$ (Kaufman, 2002). Aplicando esse resultado com $x = m$ e $y = ed$, temos

$$m^{ed} \bmod n = m^{(ed \bmod z)} \bmod n$$

Mas lembre-se de que escolhemos e e d tais que $e^d \bmod z = 1$. Isso nos dá

$$m^{ed} \bmod n = m^1 \bmod n = m$$

que é exatamente o resultado que esperávamos! Efetuando primeiro a exponenciação da potência e (i.e., criptografando) e depois a exponenciação da potência d (i.e., decriptando), obtemos o valor original m . E mais notável ainda é o fato de que, se elevarmos primeiro à potência d e, em seguida, à potência e , isto é, se invertermos a ordem da criptografia e da decriptação, realizando primeiro a operação de decriptação e, em seguida, aplicando a de criptografia – também obteremos o valor original m . Esse resultado extraordinário resulta imediatamente da aritmética modular:

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

A segurança do RSA reside no fato de que não se conhecem algoritmos para fatorar rapidamente um número, nesse caso, o valor público n , em números primos p e q . Se alguém conhecesse os números p e q , então, dado o valor público e , poderia com facilidade processar a chave secreta d . Por outro lado, não se sabe se existem ou não algoritmos rápidos para fatorar um número e, nesse sentido, a segurança do RSA não é garantida. Dados os avanços recentes da computação quântica e os algoritmos publicados de fatoração rápida para computadores quânticos, surge a preocupação de que o RSA possa não ser seguro para sempre (MIT TR, 2019). Mas a realização prática desses algoritmos ainda parece estar no futuro distante.

Outro conhecido algoritmo de criptografia de chave pública é o Diffie-Hellman, que será explorado resumidamente nos Exercícios de Fixação. O Diffie-Hellman não é tão versátil quanto o RSA, pois não pode ser usado para cifrar mensagens de comprimento arbitrário; ele pode ser usado, entretanto, para determinar uma chave de sessão simétrica, que, por sua vez, é utilizada para codificar mensagens.

8.3 INTEGRIDADE DE MENSAGEM E ASSINATURAS DIGITAIS

Na seção anterior, vimos como a criptografia pode ser usada para oferecer sigilo a duas entidades em comunicação. Nesta seção, nos voltamos ao assunto igualmente importante da criptografia que é prover **integridade da mensagem** (também conhecida como autenticação da mensagem). Além da integridade da mensagem, discutiremos dois assuntos parecidos nesta seção: assinaturas digitais e autenticação do ponto final.

Definimos o problema de integridade da mensagem usando, mais uma vez, Alice e Bob. Suponha que Bob receba uma mensagem (que pode ser cifrada ou estar em texto aberto) acreditando que tenha sido enviada por Alice. Para autenticar a mensagem, Bob precisa verificar se:

1. A mensagem foi, realmente, enviada por Alice.
2. A mensagem não foi alterada em seu caminho para Bob.

Veremos, nas Seções 8.4 a 8.7, que esse problema de integridade da mensagem é uma preocupação importante em todos os protocolos de rede seguros.

Como um exemplo específico, considere uma rede de computadores que está utilizando um algoritmo de roteamento de estado de enlace (como *Open Shortest Path First* [OSPF]) para determinar rotas entre cada dupla de roteadores na rede (veja Capítulo 5). Em um algoritmo de estado de enlace, cada roteador precisa transmitir uma mensagem de estado de enlace a todos os outros roteadores na rede. Uma mensagem de estado de enlace do roteador inclui uma relação de seus vizinhos diretamente conectados e os custos diretos a eles. Uma vez que o roteador recebe mensagens de estado de enlace de todos os outros roteadores, ele pode criar um mapa completo da rede, executar seu algoritmo de roteamento de menor custo e configurar sua tabela de repasse. Um ataque relativamente fácil no algoritmo de roteamento é Trudy distribuir mensagens de estado de enlace falsas com informações incorretas sobre o estado de enlace. Assim, a necessidade de integridade da mensagem – quando o roteador B recebe uma mensagem de estado de enlace do roteador A, deve verificar que o roteador A na verdade criou a mensagem e que ninguém a alterou em trânsito.

Nesta seção, descrevemos uma técnica conhecida sobre integridade da mensagem usada por muitos protocolos de rede seguros. Mas, antes disso, precisamos abordar outro importante tópico na criptografia – as funções de hash criptográficas.

8.3.1 Funções de hash criptográficas

Como mostrado na Figura 8.7, a função de hash recebe uma entrada, m , e calcula uma cadeia de tamanho fixo $H(m)$ conhecida como hash. A soma de verificação da Internet (Capítulo 3) e as verificações de redundância cílica (CRCs, do inglês *cyclic redundancy checks*) (Capítulo 6) satisfazem essa definição. Uma **função hash criptográfica** deve apresentar a seguinte propriedade adicional:

- Em termos de processamento, é impraticável encontrar duas mensagens diferentes x e y tais que $H(x) = H(y)$.

Informalmente, essa propriedade significa que, em termos de processamento, é impraticável que um invasor substitua uma mensagem por outra que está protegida pela função hash. Ou seja, se $(m, H(m))$ é a mensagem e o hash da mensagem criada pelo emissor, um invasor não pode forjar os conteúdos de outra mensagem, y , que possui o mesmo valor de hash da mensagem original.

É bom nos convencermos de que uma soma de verificação simples, como a soma de verificação da Internet, criaria uma função de hash criptográfica fraca. Em vez de processar a aritmética de complemento de 1 (como é feito para a soma de verificação da Internet), vamos efetuar uma soma de verificação tratando cada caractere como um byte e somando os bytes usando porções de 4 bytes por vez. Suponha que Bob deva a Alice US\$ 100,99 e lhe envie um vale contendo a cadeia de texto “IOU100.99BOB”. (IOU, do inglês *I Owe You* – Eu lhe devo.)

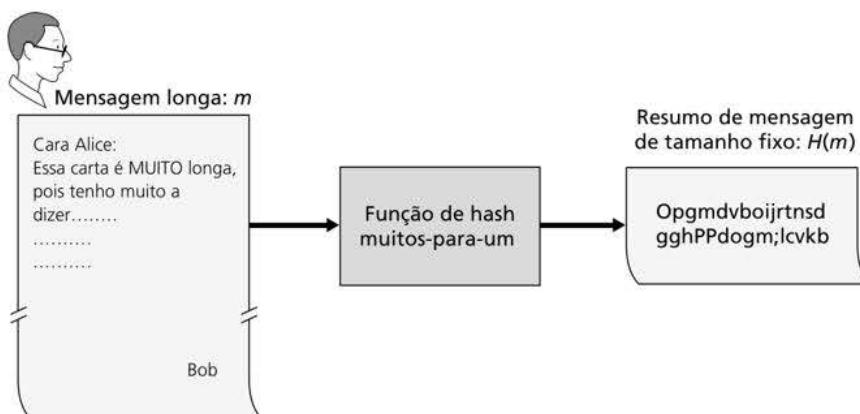


Figura 8.7 Funções de hash.

A representação ASCII (em notação hexadecimal) para essas letras é 49, 4F, 55, 31, 30, 30, 2E, 39, 39, 42, 4F, 42.

A Figura 8.8 (parte de cima) mostra que a soma de verificação de 4 bytes para essa mensagem é B2 C1 D2 AC. Uma mensagem ligeiramente diferente (e que sairia muito mais cara para Bob) é mostrada na parte de baixo da Figura 8.8. As mensagens “IOU100.99BOB” e “IOU900.19BOB” têm a *mesma* soma de verificação. Assim, esse algoritmo de soma de verificação simples viola a exigência anterior. Fornecidos os dados originais, é simples descobrir outro conjunto de dados com a mesma soma de verificação. É claro que, para efeito de segurança, precisaremos de uma função de hash muito mais poderosa do que uma soma de verificação.

O algoritmo de hash MD5 de Ron Rivest (RFC 1321) é amplamente usado hoje. Ele processa um resumo de mensagem de 128 bits por meio de um processo de quatro etapas, constituído de uma etapa de enchimento (adição de um “um” seguido de “zeros” suficientes para que o comprimento da mensagem satisfaça determinadas condições), uma etapa de anexação (anexação de uma representação de 64 bits do comprimento da mensagem antes do enchimento), uma etapa de inicialização de um acumulador e uma etapa final iterativa, na qual os blocos de 16 palavras da mensagem são processados (misturados) em quatro rodadas de processamento. Para ver uma descrição do MD5 (incluindo uma implementação em código fonte C), consulte (RFC 1321).

O segundo principal algoritmo de hash em uso atualmente é o SHA-1 (do inglês *Secure Hash Algorithm* – algoritmo de hash seguro) (FIPS, 1995). Esse algoritmo se baseia em princípios similares aos usados no projeto do MD4 (RFC 1320), o predecessor do MD5. O uso do SHA-1, um padrão federal norte-americano, é exigido sempre que um algoritmo de hash criptográfico for necessário para uso federal nos EUA. Ele produz uma mensagem de resumo de 160 bits. O resultado mais longo torna o SHA-1 mais seguro.

8.3.2 Código de autenticação da mensagem

Retornemos ao problema da integridade da mensagem. Agora que compreendemos as funções de hash, vamos tentar entender como podemos garantir a integridade da mensagem:

1. Alice cria a mensagem m e calcula o hash $H(m)$ (p. ex., com SHA-1).
2. Alice então anexa $H(m)$ à mensagem m , criando uma mensagem estendida $(m, H(m))$, e a envia para Bob.
3. Bob recebe uma mensagem estendida (m, h) e calcula $H(m)$. Se $H(m) = h$, Bob conclui que está tudo certo.

Mensagem	Representação				Soma de verificação
	ASCII				
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	

Mensagem	Representação				Soma de verificação
	ASCII				
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	

Figura 8.8 Mensagem inicial e mensagem fraudulenta têm a mesma soma de verificação!

Essa abordagem é, obviamente, errônea. Trudy pode criar uma mensagem m' falsa, passar-se por Alice, calcular $H(m')$ e enviar $(m', H(m'))$ a Bob. Quando Bob receber a mensagem, tudo se encaixa na etapa 3, então ele não suspeita de nada.

Para realizar a integridade da mensagem, além de usar as funções de hash criptográficas, Alice e Bob precisarão de um segredo compartilhado s , que não é nada mais do que uma cadeia de bits denominada **chave de autenticação**. Utilizando esse segredo compartilhado, a integridade da mensagem pode ser realizada da seguinte maneira:

1. Alice cria a mensagem m , concatena s com m para criar $m + s$, e calcula o hash $H(m + s)$ (p. ex., com SHA-1). $H(m+s)$ é denominado o **código de autenticação da mensagem (MAC, do inglês message authentication code)**.
2. Alice então anexa o MAC à mensagem m , criando uma mensagem estendida $(m, H(m + s))$, e a envia para Bob.
3. Bob recebe uma mensagem estendida (m, h) e, conhecendo s , calcula o MAC $H(m + s)$. Se $H(m + s) = h$, Bob conclui que está tudo certo.

Um resumo desse processo é ilustrado na Figura 8.9. É importante observar que o MAC, neste caso, não é o mesmo MAC utilizado nos protocolos da camada de enlace (abreviação de *medium access control* [controle de acesso ao meio]!).

Um bom recurso do MAC é o fato de ele não exigir um algoritmo de criptografia. De fato, em muitas aplicações, incluindo o algoritmo de roteamento de estado do enlace, descrito antes, as entidades de comunicação somente estão preocupadas com a integridade da mensagem, e não com o seu sigilo. Utilizando um MAC, as entidades podem autenticar as mensagens que enviam uma à outra sem ter de integrar algoritmos de criptografia complexos ao processo de integridade.

Como você pode esperar, muitos padrões diferentes para os MACs foram propostos ao longo dos anos. Hoje, o mais popular é o **HMAC**, que pode ser usado com o MD5 ou SHA-1. O HMAC, na verdade, passa os dados e a chave de autenticação duas vezes pela função de hash (Kaufman, 2002; RFC 2104).

Ainda resta um assunto importante. Como distribuímos a chave de autenticação compartilhada às entidades de comunicação? No algoritmo de roteamento de estado do enlace, por exemplo, precisaríamos, de alguma forma, distribuir a chave de autenticação secreta a cada um dos roteadores no sistema independente. (Observe que os roteadores podem usar a mesma chave de autenticação.) Um administrador de rede conseguiria, de fato, esse feito visitando fisicamente cada um dos roteadores. Ou, se o administrador de rede for preguiçoso, e se cada roteador possuir sua própria chave pública, ele poderia distribuir a chave de autenticação a qualquer um dos roteadores criptografando-a com a chave pública e, então, enviar a chave cifrada por meio da rede ao roteador.

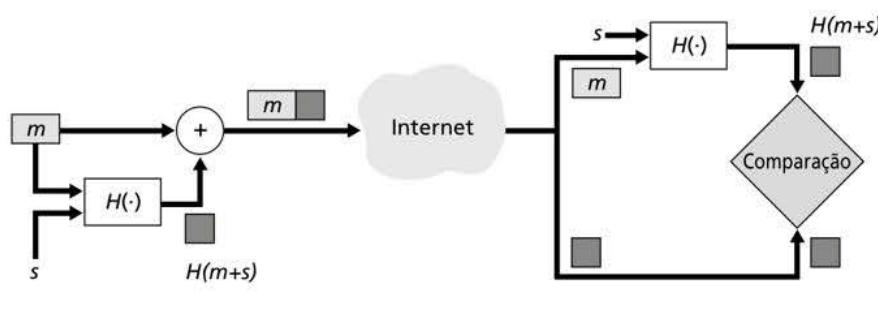


Figura 8.9 Código de autenticação de mensagem (MAC).

8.3.3 Assinaturas digitais

Pense no número de vezes em que você assinou seu nome em um pedaço de papel durante a última semana. Você assina cheques, comprovantes de operação de cartões de crédito, documentos legais e cartas. Sua assinatura atesta o fato de que você (e não outra pessoa) conhece o conteúdo do documento e/ou concorda com ele. No mundo digital, com frequência deseja-se indicar o dono ou o criador de um documento ou deixar claro que alguém concorda com o conteúdo de um documento. A **assinatura digital** é uma técnica criptográfica que cumpre essas finalidades no mundo digital.

Exatamente como acontece com as assinaturas por escrito, a assinatura digital deve ser verificável e não falsificável. Isto é, deve ser possível provar que um documento assinado por um indivíduo foi na verdade assinado por ele (a assinatura tem de ser verificável) e que *somente* aquele indivíduo poderia ter assinado o documento (a assinatura não pode ser falsificável).

Vamos considerar agora como podemos criar um método de assinatura digital. Observe que, quando Bob assina uma mensagem, deve colocar algo nela que seja único para ele. Bob poderia adicionar um MAC à assinatura, sendo o MAC criado ao adicionar sua chave (única para ele) à mensagem e, depois, formar o hash. Mas para Alice verificar a assinatura, ela deve também ter uma cópia da chave, que não seria única para Bob. Portanto, os MACs não se incluem nesse processo.

Lembre-se de que, com a criptografia de chave pública, Bob possui tanto uma chave pública como uma privada, as quais são únicas para ele. Dessa maneira, a criptografia de chave pública é uma excelente candidata para prover assinaturas digitais. Vamos verificar como isso é feito.

Suponha que Bob queira assinar digitalmente um documento, m . Imagine que o documento seja um arquivo ou uma mensagem que ele vai assinar e enviar. Como mostra a Figura 8.10, para assinar esse documento, Bob apenas usa sua chave criptográfica privada K_B^- para processar $K_B^-(m)$. A princípio, pode parecer estranho que Bob esteja usando sua chave privada (que, como vimos na Seção 8.2, foi usada para decriptar uma mensagem que tinha sido criptografada com sua chave pública) para assinar um documento. Mas lembre-se de que criptografia e decriptação nada mais são do que uma operação matemática (exponenciação à potência e ou d no RSA; veja a Seção 8.2), e que a intenção de Bob não é embaralhar ou disfarçar o conteúdo do documento, mas assiná-lo de maneira que este seja verificável, não falsificável e incontestável. Bob tem o documento, m , e sua assinatura digital do documento é $K_B^-(m)$.

A assinatura digital $K_B^-(m)$ atende às exigências de ser verificável e não falsificável? Suponha que Alice tenha m e $K_B^-(m)$. Ela quer provar na Justiça (em ação litigiosa) que Bob de-

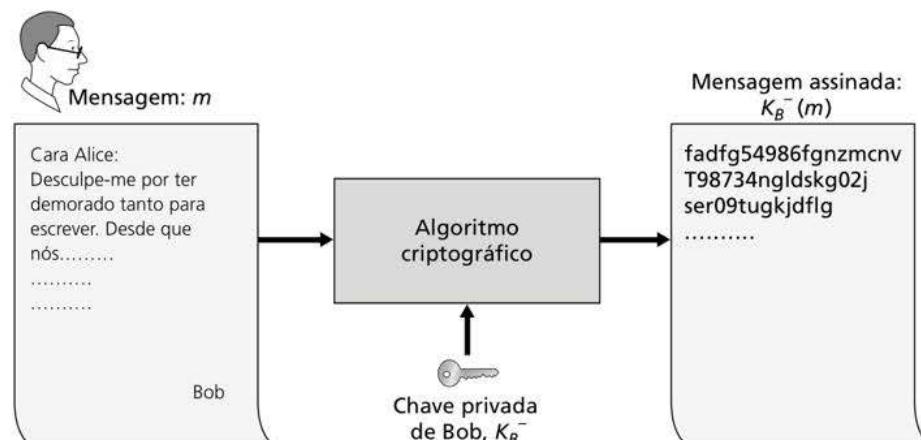


Figura 8.10 Criando uma assinatura digital para um documento.

fato assinou o documento e que ele era a única pessoa que poderia tê-lo assinado. Alice pega a chave pública de Bob, K_B^+ , e lhe aplica a assinatura digital $K_B^-(m)$ associada ao documento m . Isto é, ela processa $K_B^+(K_B^-(m))$, e, *voilà*, com dramática encenação, produz m , que é uma reprodução exata do documento original! Ela então argumenta que somente Bob poderia ter assinado o documento pelas seguintes razões:

- Quem quer que tenha assinado o documento deve ter usado a chave criptográfica privada, K_B^- , para processar a assinatura $K_B^-(m)$, de modo que $K_B^+(K_B^-(m)) = m$.
- A única pessoa que poderia conhecer a chave privada, K_B^- , é Bob. Lembre-se de que dissemos em nossa discussão do RSA, na Seção 8.2, que conhecer a chave pública, K_B^+ , não serve para descobrir a chave privada, K_B^- . Portanto, a única pessoa que poderia conhecer K_B^- é aquela que gerou o par de chaves (K_B^+, K_B^-) em primeiro lugar, Bob. (Note que, para isso, admitimos que Bob não passou K_B^- a ninguém e que ninguém roubou K_B^- de Bob.)

Também é importante notar que, se o documento original, m , for modificado para alguma forma alternativa, m' , a assinatura que Bob criou para m não será válida para m' , já que $K_B^+(K_B^-(m))$ não é igual a m' . Assim, observamos que as assinaturas digitais também oferecem integridade da mensagem, permitindo que o receptor verifique se ela foi alterada, bem como sua origem.

Uma preocupação com os dados de assinatura é que a criptografia e a decriptação prejudicam o desempenho do computador. Sabendo do trabalho extra de criptografia e decriptação, os dados de assinatura por meio de criptografia/decriptação completa podem ser excessivos. Uma técnica mais eficiente é introduzir as funções de hash na assinatura digital. Na Seção 8.3.2, um algoritmo de hash pega uma mensagem, m , de comprimento qualquer e calcula uma “impressão digital” de comprimento fixo da mensagem, representada por $H(m)$. Usando uma função de hash, Bob assina o hash da mensagem em vez de assinar a própria mensagem, ou seja, Bob calcula $K_B^-(H(m))$. Como $H(m)$ é em geral muito menor do que a mensagem original m , o esforço computacional necessário para criar a assinatura digital é reduzido consideravelmente.

Em relação a Bob enviar uma mensagem para Alice, a Figura 8.11 apresenta um resumo do procedimento operacional de criar uma assinatura digital. Bob coloca sua longa mensagem original em uma função de hash. Ele, então, assina digitalmente o hash resultante com sua chave privada. A mensagem original (em texto aberto) junto com o resumo de mensagem assinada digitalmente (de agora em diante denominada assinatura digital) é então enviada para Alice. A Figura 8.12 apresenta um resumo do processo operacional da assinatura. Alice aplica a função de hash à mensagem para obter um resultado de hash. Ela também aplica a função de hash à mensagem em texto aberto para obter um segundo resultado de hash. Se os dois combinarem, então Alice pode ter certeza sobre a integridade e o autor da mensagem.

Antes de seguirmos em frente, vamos fazer uma breve comparação entre as assinaturas digitais e os MACs, visto que eles são paralelos, mas também têm diferenças sutis e importantes. As assinaturas digitais e os MACs iniciam com uma mensagem (ou um documento). Para criar um MAC por meio de mensagem, inserimos uma chave de autenticação à mensagem e, depois, pegamos o hash do resultado. Observe que nem a chave pública nem a criptografia de chaves simétricas estão envolvidas na criação do MAC. Para criar uma assinatura digital, primeiro pegamos o hash da mensagem e, depois, ciframos a mensagem com nossa chave privada (usando a criptografia de chave pública). Assim, uma assinatura digital é uma técnica “mais pesada”, pois exige uma Infraestrutura de Chave Pública (PKI, do inglês *Public Key Infrastructure*) subjacente com autoridades de certificação, conforme descrito abaixo. Veremos na Seção 8.4 que o PGP – um sistema de e-mail seguro e popular – utiliza assinaturas digitais para a integridade da mensagem. Já vimos que o OSPF usa MACs para a integridade da mensagem. Veremos nas Seções 8.5 e 8.6 que os MACs são também usados por conhecidos protocolos de segurança da camada de transporte e da camada de rede.

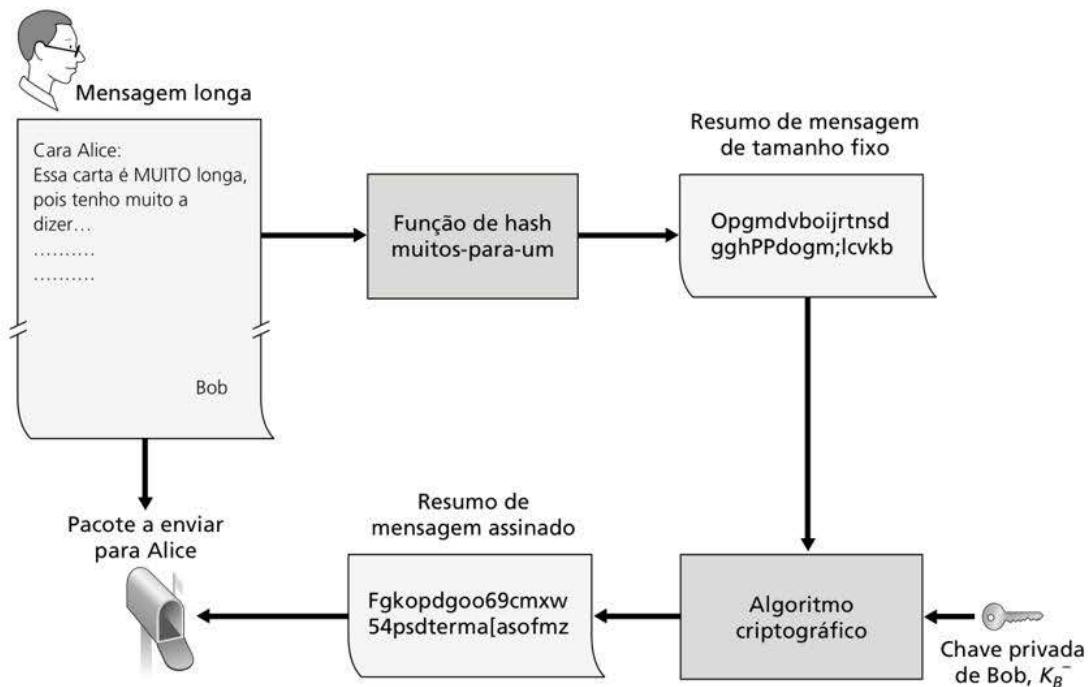


Figura 8.11 Enviando uma mensagem assinada digitalmente.

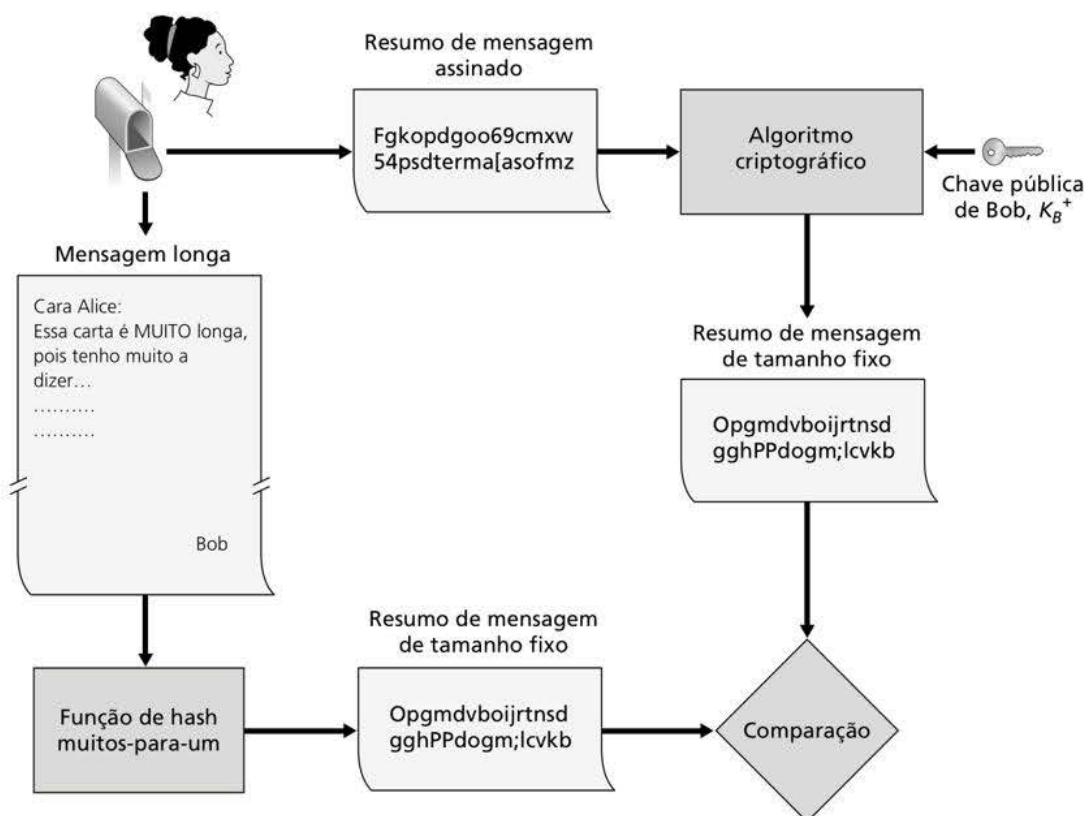


Figura 8.12 Verificando uma mensagem assinada.

Certificação de chaves públicas

Uma aplicação importante de assinaturas digitais é a **certificação de chaves públicas**, ou seja, certificar que uma chave pública pertence a uma entidade específica. A certificação de chaves públicas é usada em muitos protocolos de rede seguros, incluindo o IPsec e o TLS.

Para compreender mais sobre o problema, consideremos uma versão de comércio eletrônico para o clássico “trote da pizza”. Suponha que Alice trabalhe no ramo de pizzas para viagem e que aceite pedidos pela Internet. Bob, que adora pizza, envia a Alice uma mensagem em texto aberto que contém o endereço de sua casa e o tipo de pizza que quer. Nessa mensagem, ele inclui também uma assinatura digital (i.e., um hash assinado extraído da mensagem original em texto aberto) para provar a Alice que ele é a origem verdadeira da mensagem. Para verificar a assinatura, Alice obtém a chave pública de Bob (talvez de um servidor de chaves públicas ou de uma mensagem de e-mail) e verifica a assinatura digital. Dessa maneira, ela se certifica de que foi Bob, e não algum adolescente brincalhão, quem fez o pedido.

Tudo parece caminhar bem até que entra em cena a esperta Trudy. Como mostrado na Figura 8.13, Trudy decide fazer uma travessura. Ela envia uma mensagem a Alice na qual diz que é Bob, fornece o endereço de Bob e pede uma pizza. Nessa mensagem, ela também inclui sua (de Trudy) chave pública, embora Alice suponha, claro, que seja a de Bob. Trudy também anexa uma assinatura digital, a qual foi criada com sua própria (de Trudy) chave privada. Após receber a mensagem, Alice aplica a chave pública de Trudy (pensando que é a de Bob) à assinatura digital e concluirá que a mensagem em texto aberto foi, na verdade, criada por Bob. Este ficará muito surpreso quando o entregador aparecer em sua casa com uma pizza de calabresa e anchovas!

A partir desse exemplo, vemos que, para que a criptografia de chaves públicas seja útil, você precisa ser capaz de verificar se tem a chave pública verdadeira da entidade (pessoa,

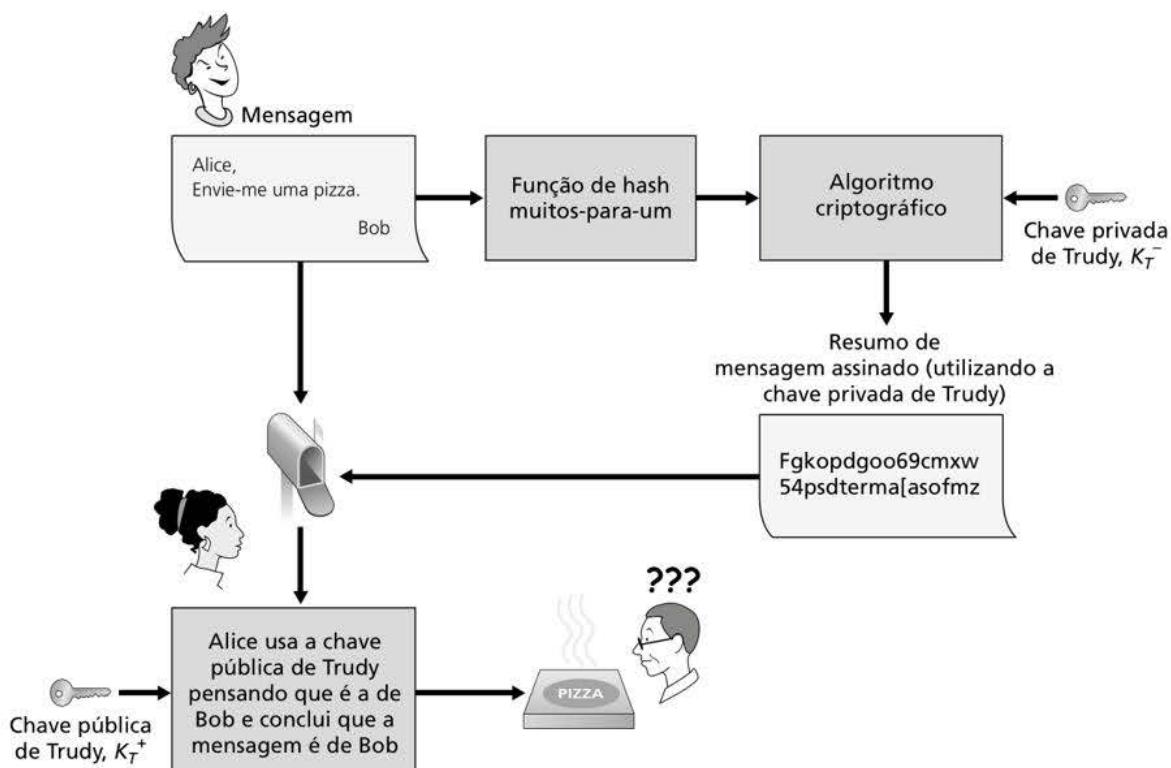


Figura 8.13 Trudy se passa por Bob usando criptografia de chaves públicas.

roteador, navegador e outras) com a qual quer se comunicar. Por exemplo, quando Alice estiver se comunicando com Bob usando criptografia de chaves públicas, ela precisará saber, com certeza, que a chave pública que supostamente é de Bob é, de fato, dele.

A vinculação de uma chave pública a uma entidade particular é feita, em geral, por uma **Autoridade Certificadora (CA)**, do inglês *Certification Authority*), cuja tarefa é validar identidades e emitir certificados. Uma CA tem as seguintes funções:

1. Uma CA verifica se uma entidade (pessoa, roteador e assim por diante) é quem diz ser. Não há procedimentos obrigatórios para o modo como deve ser feita a certificação. Quando tratamos com uma CA, devemos confiar que ela tenha realizado uma verificação de identidade adequadamente rigorosa. Por exemplo, se Trudy conseguisse entrar na autoridade certificadora Fly-by-Night, e apenas declarasse “Eu sou Alice” e recebesse certificados associados à identidade de Alice, então não se deveria dar muita credibilidade a chaves públicas certificadas pela autoridade certificadora Fly-by-Night. Por outro lado, seria mais sensato (ou não!) estar mais inclinado a confiar em uma CA que faz parte de um programa federal ou estadual. O grau de confiança que se tem na identidade associada a uma chave pública equivale apenas ao grau de confiança depositada na CA e em suas técnicas de verificação de identidades. Que teia emaranhada de confiança estamos tecendo!
2. Tão logo verifique a identidade da entidade, a CA cria um **certificado** que vincula a chave pública da entidade à identidade verificada. O certificado contém a chave pública e a informação exclusiva que identifica mundialmente o proprietário da chave pública (p. ex., o nome de alguém ou um endereço IP). O certificado é assinado digitalmente pela CA. Essas etapas são mostradas na Figura 8.14.

Vejamos, agora, como certificados podem ser usados para combater os espertinhos das pizzas, como Trudy e outros indesejáveis. Quando Bob faz seu pedido, ele também envia seu certificado assinado por uma CA. Alice usa a chave pública da CA para verificar a validade do certificado de Bob e extrair a chave pública dele.

Tanto a International Telecommunication Union (ITU) como a IETF desenvolveram padrões para autoridades certificadoras. Na recomendação ITU X.509 (ITU, 2005a), encontramos a especificação de um serviço de autenticação, bem como uma sintaxe específica para certificados. O RFC 1422 descreve um gerenciamento de chaves baseado em CA para utilização com o e-mail seguro pela Internet. Essa recomendação é compatível com X.509,

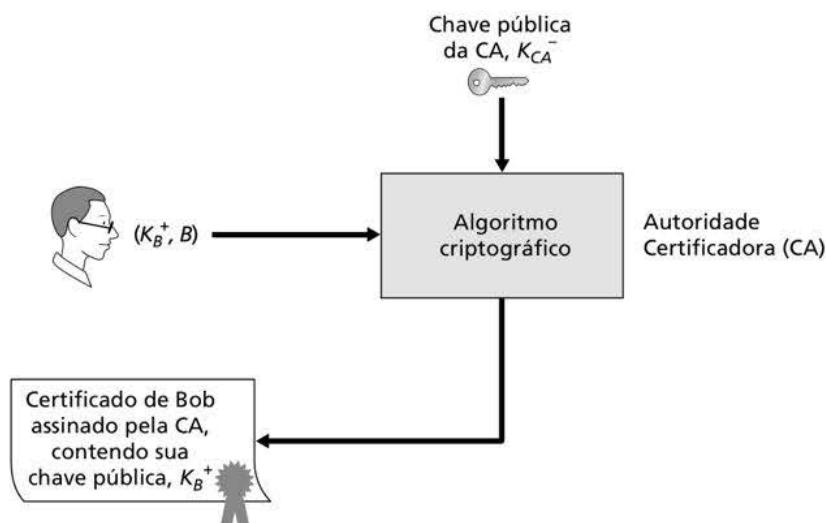


Figura 8.14 Bob obtém sua chave pública certificada pela CA.

TABELA 8.4 Campos selecionados de uma chave pública X.509 e RFC 1422

Nome do campo	Descrição
Versão	Número da versão da especificação X.509
Número de série	Identificador exclusivo emitido pela CA para um certificado
Assinatura	Especifica o algoritmo usado pela CA para assinar esse certificado
Nome do emissor	Identidade da CA que emitiu o certificado, em formato de nome distinto (DN) (RFC 4514)
Período de validade	Início e fim do período de validade do certificado
Nome do sujeito	Identidade da entidade cuja chave pública está associada a esse certificado, em formato DN
Chave pública do sujeito	A chave pública do sujeito, bem como uma indicação do algoritmo de chave pública (e parâmetros do algoritmo) a ser usado com essa chave

mas vai além, pois estabelece procedimentos e convenções para uma arquitetura de gerenciamento de chaves. A Tabela 8.4 apresenta alguns campos importantes de um certificado.

8.4 AUTENTICAÇÃO DO PONTO FINAL

A **autenticação do ponto final** é o processo de provar a identidade de uma entidade a outra entidade por uma rede de computadores, por exemplo, um usuário provando sua identidade a um servidor de correio eletrônico. Como seres humanos, autenticamos uns aos outros de diversas maneiras: reconhecemos o rosto do outro ao nos encontrarmos, reconhecemos a voz no telefone, somos autenticados por um oficial da Alfândega que compara a nossa foto com a do passaporte.

Nesta seção, vamos considerar como uma entidade pode autenticar outra quando as duas estão se comunicando por uma rede. Vamos nos atentar aqui para a autenticação de uma entidade “ao vivo”, no momento em que a comunicação está de fato ocorrendo. Um exemplo concreto é um usuário autenticando-se em um servidor de correio eletrônico. Este é um problema sutilmente diferente de provar que uma mensagem recebida em algum ponto no passado veio mesmo do remetente afirmado, conforme estudamos na Seção 8.3.

Ao realizar autenticação pela rede, as entidades comunicantes não podem contar com informações biométricas, como aparência visual ou timbre de voz. Na verdade, veremos em nossos estudos de caso que, geralmente, são os elementos da rede, como roteadores e processos cliente/servidor, que precisam se autenticar. Aqui, a autenticação precisa ser feita unicamente com base nas mensagens e nos dados trocados como parte de um **protocolo de autenticação**. Em geral, um protocolo de autenticação seria executado *antes* que as duas entidades comunicantes executassem algum outro protocolo (p. ex., um protocolo de transferência confiável de dados, um de troca de informações de roteamento ou um de correio eletrônico). O protocolo de autenticação primeiro estabelece as identidades das partes para a satisfação mútua; somente depois da autenticação é que as entidades “põem as mãos” no trabalho real.

Assim como no caso do nosso desenvolvimento de um protocolo de transferência confiável de dados (rdt, do inglês *reliable data transfer*) no Capítulo 3, será esclarecedor desenvolvermos aqui diversas versões de um protocolo de autenticação, que chamaremos de **ap** (do inglês *authentication protocol* – protocolo de autenticação), explicando cada versão enquanto prosseguimos. (Se você gostar da evolução passo a passo de um projeto, também poderá gostar de Bryant [1988], que relata uma narrativa fictícia entre projetistas de um sistema aberto de autenticação de rede e sua descoberta das muitas questões sutis que estão envolvidas.)

Vamos imaginar que Alice precise se autenticar com Bob.

Talvez o protocolo de autenticação mais simples que possamos imaginar seja aquele onde Alice apenas envia uma mensagem a Bob dizendo ser Alice. Esse protocolo aparece na Figura 8.15. A falha aqui é óbvia – não há como Bob realmente saber que a pessoa enviando a mensagem “Eu sou Alice” é mesmo Alice. Por exemplo, Trudy (a intrusa) poderia enviar tal mensagem.

Protocolo de autenticação ap2.0

Se Alice possui um endereço de rede conhecido (p. ex., um endereço IP) do qual ela sempre se comunica, Bob poderia tentar autenticar Alice verificando se o endereço de origem no datagrama IP que transporta a mensagem de autenticação combina com o endereço conhecido de Alice. Nesse caso, Alice seria autenticada. Isso poderia impedir que um intruso muito ingênuo em redes se passe por Alice, mas não o aluno determinado, que está estudando este livro, ou muitos outros!

Pelo estudo das camadas de rede e enlace de dados, sabemos que não é difícil (p. ex., se alguém tivesse acesso ao código do sistema operacional e pudesse criar seu próprio núcleo do sistema operacional, como acontece com o Linux e vários outros sistemas operacionais disponíveis livremente) criar um datagrama IP, colocar qualquer endereço de origem IP que se desejar (p. ex., o endereço IP conhecido de Alice) no datagrama IP e enviar o datagrama pelo protocolo da camada de enlace para o roteador do primeiro salto. Daí em diante, o datagrama com endereço de origem incorreto seria zelosamente repassado para Bob. Essa técnica, mostrada na Figura 8.16, é uma forma de falsificação de IP. Tal falsificação

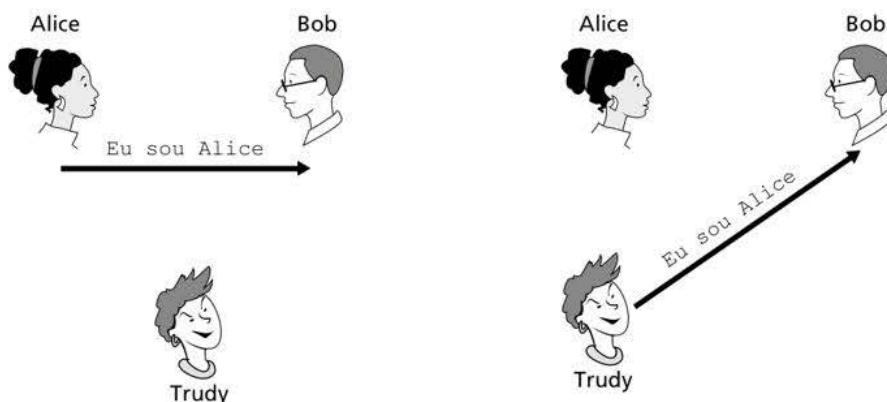


Figura 8.15 Protocolo ap1.0 e um cenário de falha.

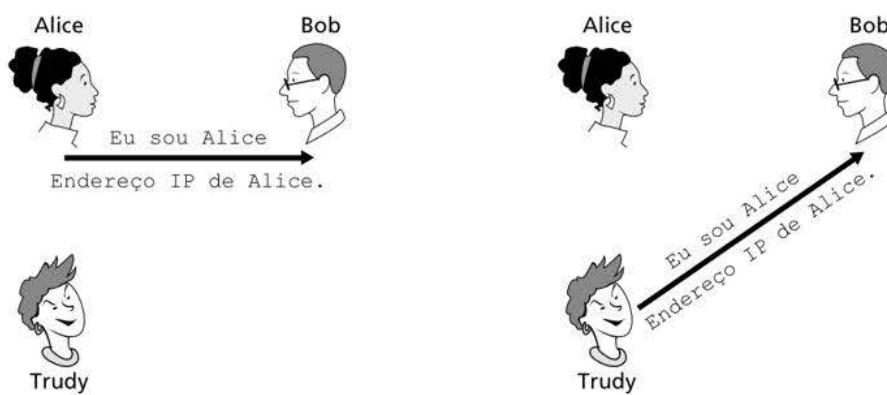


Figura 8.16 Protocolo ap2.0 e um cenário de falha.

pode ser evitada se o roteador do primeiro salto de Trudy for configurado para encaminhar apenas datagramas contendo o endereço IP de origem de Trudy (RFC 2827). Porém, essa capacidade não é implementada ou imposta de modo universal. Assim, Bob poderia ser enganado a achar que o gerente de rede de Trudy (que poderia ser a própria Trudy) configurou o roteador do primeiro salto de Trudy para repassar somente datagramas corretamente endereçados.

Protocolo de autenticação ap3.0

Uma técnica clássica de autenticação é usar uma senha secreta. A senha é um segredo compartilhado entre o autenticador e a pessoa sendo autenticada. Gmail, Facebook, Telnet, FTP e muitos outros serviços usam a autenticação por senha. No protocolo ap3.0, Alice envia sua senha secreta a Bob, como mostra a Figura 8.17.

Como as senhas são muito utilizadas, poderíamos supor que o protocolo *ap3.0* é bastante seguro. Nesse caso, estariamos errados! A falha de segurança aqui é clara. Se Trudy interceptar a comunicação de Alice, então ela poderá descobrir a senha de Alice. Se você acha que isso é improvável, considere o fato de que, quando você usa Telnet com outra máquina e se autentica, a senha de autenticação é enviada abertamente para o servidor Telnet. Alguém conectado ao cliente Telnet ou à LAN do servidor possivelmente poderia investigar (ler e armazenar) todos os pacotes transmitidos na LAN e, assim, roubar a senha de autenticação. De fato, essa é uma técnica bem conhecida para roubar senhas (p. ex., ver Jimenez [1997]). Essa ameaça, obviamente, é bastante real, de modo que *ap3.0* certamente não funcionará.

Protocolo de autenticação ap3.1

Nossa próxima ideia para consertar o ap3.0, naturalmente, é criptografar a senha. Criptografando a senha, podemos impedir que Trudy descubra a senha de Alice. Se considerarmos que Alice e Bob compartilham uma chave secreta simétrica, K_{A-B} , então Alice pode criptografar a senha e enviar sua mensagem de identificação, “Eu sou Alice”, e sua senha criptografada para Bob. Então, Bob decodifica a senha e, supondo que a senha esteja correta, autentica Alice. Bob está confiante na autenticação de Alice, pois Alice não apenas

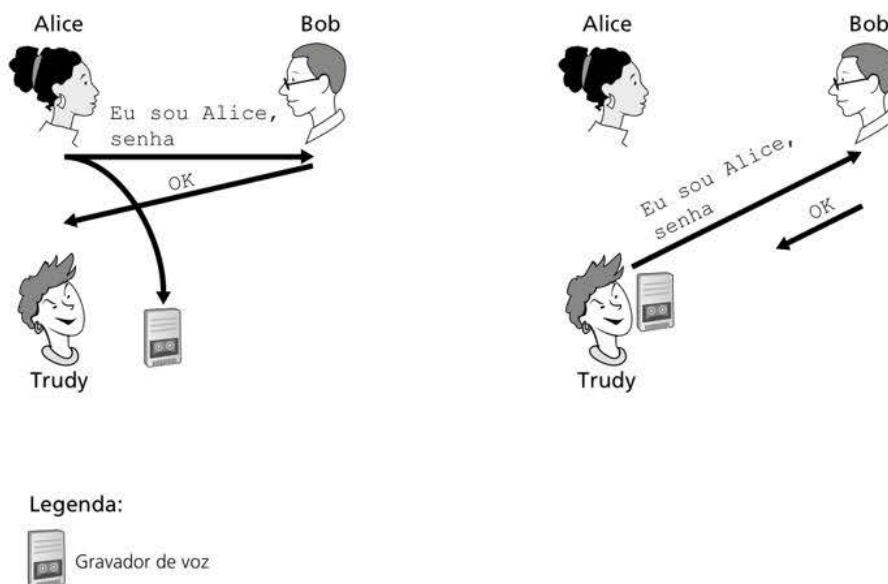


Figura 8.17 Protocolo ap3.0 e um cenário de falha.

conhece a senha, mas também sabe o valor da chave secreta compartilhada necessária para criptografar a senha. Vamos chamar esse protocolo de *ap3.1*.

Embora seja verdade que *ap3.1* impede que Trudy descubra a senha de Alice, o uso da criptografia aqui não resolve o problema da autenticação. Bob está sujeito a um **ataque de reprodução**: Trudy só precisa investigar a comunicação de Alice, gravar a versão criptografada da senha e reproduzir a versão criptografada da senha para Bob, fingindo ser Alice. O uso de uma senha criptografada em *ap3.1* não torna a situação muito diferente daquela do protocolo *ap3.0*, na Figura 8.17.

Protocolo de autenticação *ap4.0*

O cenário de falha na Figura 8.17 resultou do fato de Bob não conseguir distinguir entre a mensagem original enviada por Alice e a reprodução dessa mensagem. Ou seja, Bob não conseguiu saber se Alice estava ao vivo (i.e., realmente estava no outro ponto da conexão) ou se a mensagem que ele recebeu foi uma reprodução de uma autenticação anterior. O leitor muito (*muito*) atento se lembrará de que o protocolo de apresentação de três vias precisou resolver o mesmo problema – o lado do servidor de uma conexão TCP não queria aceitar uma conexão se o segmento SYN recebido fosse uma cópia antiga (retransmissão) de um segmento SYN de uma conexão anterior. Como o lado do servidor TCP resolveu o problema de determinar se o cliente estava mesmo ao vivo? Ele escolheu um número de sequência inicial que não havia sido usado por um bom tempo, enviou esse número ao cliente, e então aguardou que o cliente respondesse com um segmento ACK contendo esse número. Podemos adotar a mesma ideia para fins de autenticação.

Um **nonce** é um número que o protocolo usará somente uma vez por toda a vida. Ou seja, uma vez que o protocolo utilizar um nonce, esse número nunca mais será utilizado. Nossa protocolo *ap4.0* usa um nonce da seguinte forma:

1. Alice envia a mensagem “Eu sou Alice” para Bob.
2. Bob escolhe um nonce, R , e o envia a Alice.
3. Alice criptografa o nonce usando a chave secreta simétrica de Alice e Bob, K_{A-B} , e envia o nonce criptografado, $K_{A-B}(R)$, de volta a Bob. Assim como no protocolo *ap3.1*, o fato de Alice conhecer K_{A-B} e o utilizar para criptografar o valor permite a Bob saber que a mensagem recebida foi gerada por Alice. O nonce é usado para garantir que Alice está ao vivo.
4. Bob decodifica a mensagem recebida. Se o nonce decodificado for igual ao que ele enviou a Alice, então Alice está autenticada.

O protocolo *ap4.0* é ilustrado na Figura 8.18. Usando o valor exclusivo de R e depois verificando o valor retornado, $K_{A-B}(R)$, Bob pode estar certo de que Alice tanto é quem ela diz ser (pois conhece o valor da chave secreta necessária para criptografar R) quanto está ao vivo (pois criptografou o nonce, R , que Bob acabou de criar).

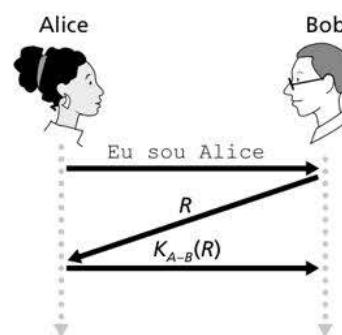


Figura 8.18 Protocolo *ap4.0* e um cenário de falha.

O uso de um nonce e das formas de criptografia por chave simétrica forma a base do *ap4.0*. Uma pergunta natural é se podemos usar um nonce e a criptografia de chave pública (em vez da criptografia de chave simétrica) para resolver o problema de autenticação. Essa questão é explorada nos problemas ao final deste capítulo.

8.5 PROTEGENDO O E-MAIL

Nas seções anteriores, analisamos as questões fundamentais em segurança de redes, incluindo a criptografia de chaves simétricas e de chaves públicas, autenticação do ponto final, distribuição de chave, integridade da mensagem e assinaturas digitais. Agora vamos analisar como essas ferramentas estão sendo usadas para oferecer segurança na Internet.

Curiosamente, é possível oferecer serviços de segurança em cada uma das quatro principais camadas da pilha de protocolos da Internet. Quando a segurança é fornecida para um protocolo específico da camada de aplicação, a aplicação que o emprega utilizará um ou mais serviços de segurança, como sigilo, autenticação ou integridade. Quando a segurança é oferecida para um protocolo da camada de transporte, todas as aplicações que usam esse protocolo aproveitam os seus serviços de segurança. Quando a segurança é fornecida na camada de rede em base hospedeiro para hospedeiro, todos os segmentos da camada de transporte (e, portanto, todos os dados da camada de aplicação) aproveitam os serviços de segurança dessa camada. Quando a segurança é oferecida em um enlace, os dados em todos os quadros que percorrem o enlace recebem os serviços de segurança do enlace.

Nas Seções 8.5 a 8.8, verificamos como as ferramentas de segurança estão sendo usadas nas camadas de enlace, rede, transporte e aplicação. Obedecendo à estrutura geral deste livro, começamos no topo da pilha de protocolo e discutimos segurança na camada de aplicação. Nossa técnica é usar uma aplicação específica, e-mail, como um estudo de caso para a segurança da camada de aplicação. Então, descemos a pilha de protocolo. Examinaremos o protocolo TLS (que provê segurança na camada de transporte), o IPsec (que provê segurança na camada de rede) e a segurança do protocolo LAN IEEE 802.11 sem fio.

Você deve estar se perguntando por que a funcionalidade da segurança está sendo fornecida em mais de uma camada na Internet. Já não bastaria prover essa funcionalidade na camada de rede? Há duas respostas para a pergunta. Primeiro, embora a segurança na camada de rede possa oferecer “cobertura total” cifrando todos os dados nos datagramas (i.e., todos os segmentos da camada de transporte) e autenticando todos os endereços IP destinatários, ela não pode prover segurança no nível do usuário. Por exemplo, um site de comércio não pode confiar na segurança da camada IP para autenticar um cliente que vai comprar mercadorias. Assim, existe a necessidade de uma funcionalidade da segurança em camadas superiores bem como cobertura total em canais inferiores. Segundo, em geral, é mais fácil implementar serviços da Internet, incluindo os de segurança nas camadas superiores da pilha de protocolo. Enquanto aguardamos a ampla implementação da segurança na camada de rede, o que ainda levará muitos anos, muitos criadores de aplicação “já fazem isso” e introduzem a funcionalidade da segurança em suas aplicações favoritas. Um exemplo clássico é o PGP, que oferece e-mail seguro (discutido mais adiante nesta seção). Necessitando de apenas um código de aplicação do cliente e do servidor, o PGP foi uma das primeiras tecnologias de segurança a ser amplamente utilizada na Internet.

8.5.1 E-mail seguro

Agora usamos os princípios de criptografia das Seções 8.2 a 8.3 para criar um sistema de e-mail seguro. Criamos esse projeto de alto nível de maneira incremental, introduzindo, a cada etapa, novos serviços de segurança. Em nosso projeto de um sistema de e-mail seguro, vamos manter em mente o exemplo atrevido apresentado na Seção 8.1 – o caso de amor entre

Alice e Bob. Imagine que Alice quer enviar uma mensagem de e-mail para Bob e Trudy quer bisbilhotar.

Antes de avançar e projetar um sistema de e-mail seguro para Alice e Bob, devemos considerar quais características de segurança seriam as mais desejáveis para eles. A primeira, e mais importante, é a *confidencialidade*. Como foi discutido na Seção 8.1, nem Alice nem Bob querem que Trudy leia a mensagem de e-mail de Alice. A segunda característica que Alice e Bob provavelmente gostariam de ver no sistema de e-mail seguro é a *autenticação do remetente*. Em particular, quando Bob receber a seguinte mensagem: “Eu não o amo mais. Nunca mais quero vê-lo. Da anteriormente sua, Alice”, ele naturalmente gostaria de ter certeza de que a mensagem veio de Alice, e não de Trudy. Outra característica de segurança de que os dois amantes gostariam de dispor é a *integridade de mensagem*, isto é, a certeza de que a mensagem que Alice enviar não será modificada no trajeto até Bob. Por fim, o sistema de e-mail deve fornecer *autenticação do receptor*, isto é, Alice quer ter certeza de que de fato está enviando a mensagem para Bob, e não para outra pessoa (p. ex., Trudy) que esteja se passando por ele.

Portanto, vamos começar abordando a preocupação mais premente, a confidencialidade. A maneira mais direta de conseguir confidencialidade é Alice criptografar a mensagem com tecnologia de chaves simétricas (como DES ou AES) e Bob decriptar a mensagem ao recebê-la. Como discutido na Seção 8.2, se a chave simétrica for longa o suficiente e se apenas Alice e Bob possuírem a chave, então será dificílimo que alguém (incluindo Trudy) leia a mensagem. Embora essa seja uma abordagem direta, ela apresenta a dificuldade fundamental que discutimos na Seção 8.2 – é difícil distribuir uma chave simétrica de modo que apenas Bob e Alice tenham cópias dela. Portanto, é natural que consideremos uma abordagem alternativa – a criptografia de chaves públicas (p. ex., usando RSA). Nessa abordagem, Bob disponibiliza publicamente sua chave pública (p. ex., em um servidor de chaves públicas ou em sua página pessoal) e Alice criptografa sua mensagem com a chave pública de Bob, e envia a mensagem criptografada para o endereço de e-mail de Bob. Quando recebe a mensagem, ele simplesmente a decodifica com sua chave privada. Supondo que Alice tenha certeza de que aquela chave pública é a de Bob, essa técnica é um meio excelente de fornecer a confidencialidade desejada. Um problema, contudo, é que a criptografia de chaves públicas é relativamente ineficiente, sobretudo para mensagens longas.

Para superar o problema da eficiência, vamos fazer uso de uma chave de sessão (discutida na Seção 8.2.2). Em particular, Alice (1) escolhe uma chave simétrica, K_S , aleatoriamente, (2) criptografa sua mensagem m com a chave simétrica, (3) criptografa a chave simétrica com a chave pública de Bob, K_B^+ , (4) concatena a mensagem criptografada e a chave simétrica criptografada de modo que formem um “pacote”, e (5) envia o pacote ao endereço de e-mail de Bob. Os passos estão ilustrados na Figura 8.19. (Nessa figura e nas subsequentes, o sinal “+” dentro de um círculo representa formar a concatenação, e o sinal “-” dentro de um círculo, desfazer a concatenação.) Quando Bob receber o pacote, ele (1) usará sua chave

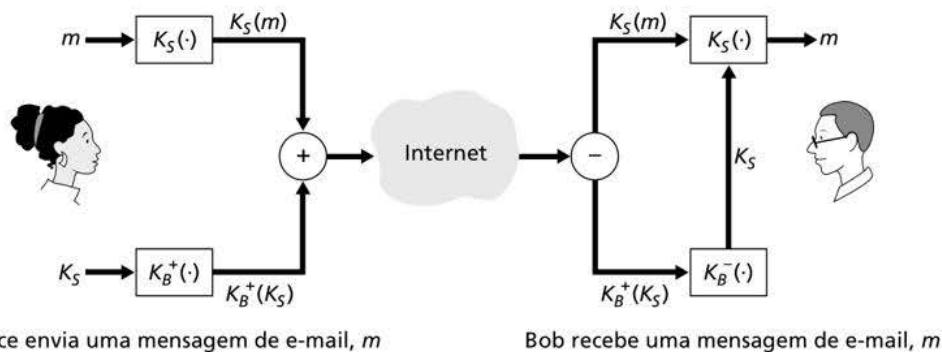


Figura 8.19 Alice usou uma chave de sessão simétrica, K_S , para enviar um e-mail secreto para Bob.

privada, K_B^- , para obter a chave simétrica, K_S , e (2) utilizará a chave simétrica K_S para decodificar a mensagem m .

Agora que projetamos um sistema de e-mail seguro que fornece confidencialidade, vamos desenvolver um outro sistema que forneça autenticação do remetente e integridade de mensagem. Vamos supor, por enquanto, que Alice e Bob não estejam mais preocupados com confidencialidade (querem compartilhar seus sentimentos com todos!) e que só estejam preocupados com a autenticação do remetente e com a integridade da mensagem. Para realizar essa tarefa, usaremos assinaturas digitais e resumos de mensagem, como descrito na Seção 8.3. Especificamente, Alice (1) aplica uma função de hash, H (p. ex., MD5), à sua mensagem, m , para obter um resumo, (2) assina o resultado da função de hash com sua chave privada K_A^- para criar uma assinatura digital, (3) concatena a mensagem original (não criptografada) com a assinatura para criar um pacote, e (4) envia o pacote ao endereço de e-mail de Bob. Quando Bob recebe o pacote, ele (1) aplica a chave pública de Alice, K_A^+ , ao resumo de mensagem assinado e (2) compara o resultado dessa operação com o próprio hash, H , da mensagem. As etapas são ilustradas na Figura 8.20. Como discutimos na Seção 8.3, se os dois resultados forem iguais, Bob poderá ter razoável certeza de que a mensagem veio de Alice e não foi alterada.

Vamos considerar agora o projeto de um sistema de e-mail que forneça confidencialidade, autenticação de remetente e integridade de mensagem. Isso pode ser feito pela combinação dos procedimentos das Figuras 8.19 e 8.20. Primeiro, Alice cria um pacote preliminar, exatamente como ilustra a Figura 8.20, constituído de sua mensagem original junto com um hash da mensagem assinado digitalmente. Em seguida, ela trata esse pacote preliminar como uma mensagem em si e a envia seguindo as etapas do remetente mostradas na Figura 8.19, criando um novo pacote, que é enviado a Bob. As etapas seguidas são mostradas na Figura 8.21. Quando Bob recebe o pacote, ele aplica primeiro seu lado da Figura 8.19 e depois seu lado da Figura 8.20. É preciso ficar claro que esse projeto atinge o objetivo de fornecer confidencialidade, autenticação de remetente e integridade de mensagem. Note que, nesse esquema, Alice utiliza criptografia de chaves públicas duas vezes: uma vez com sua chave privada e uma vez com a chave pública de Bob. De maneira semelhante, Bob também aplica a criptografia de chaves públicas duas vezes – uma vez com sua chave privada e uma vez com a chave pública de Alice.

O projeto de e-mail seguro ilustrado na Figura 8.21 provavelmente fornece segurança satisfatória para os usuários de e-mail na maioria das ocasiões. Mas ainda resta uma questão importante a ser abordada. O projeto da Figura 8.21 requer que Alice obtenha a chave pública de Bob e que este obtenha a chave pública de Alice. A distribuição dessas chaves públicas é um problema nada trivial. Por exemplo, Trudy poderia se disfarçar de Bob e dar a Alice sua própria chave pública, dizendo que é a de Bob. Como aprendemos na Seção 8.3, uma tática popular para distribuir chaves públicas com segurança é *certificar* as chaves públicas usando uma autoridade certificadora (CA).

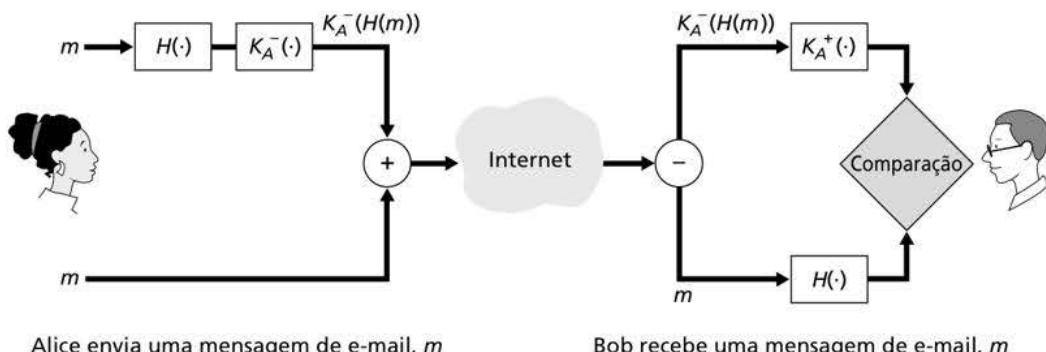


Figura 8.20 Usando funções de hash e assinaturas digitais para fornecer autenticação de remetente e integridade de mensagem.

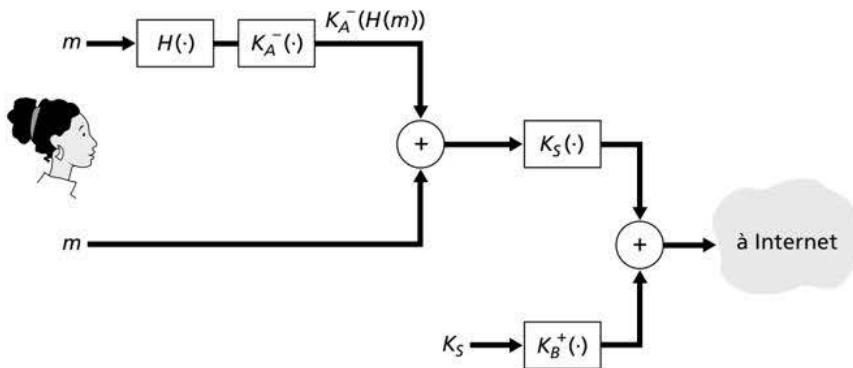


Figura 8.21 Alice usa criptografia de chaves simétricas, criptografia de chaves públicas, uma função de hash e uma assinatura digital para fornecer sigilo, autenticação de remetente e integridade de mensagem.

8.5.2 PGP

Projetado originalmente por Phil Zimmermann em 1991, o PGP é um belo exemplo de esquema de criptografia para e-mail (PGP, 2020). O projeto do PGP é, basicamente, idêntico ao projeto apresentado na Figura 8.21. Dependendo da versão, o software do PGP usa MD5 ou SHA para processar o resumo de mensagem; CAST, DES triplo ou IDEA para criptografar chaves simétricas, e RSA para criptografar chaves públicas.

Quando o PGP é instalado, o software cria um par de chaves públicas para o usuário. A chave pública pode então ser colocada no site do usuário ou em um servidor de chaves públicas. A chave privada é protegida pelo uso de uma senha. A senha tem de ser informada todas as vezes que o usuário acessar a chave privada. O PGP oferece ao usuário a opção de assinar digitalmente a mensagem, criptografar a mensagem ou, ainda, ambas as opções: assinar digitalmente e criptografar a mensagem. A Figura 8.22 mostra uma mensagem PGP assinada. Essa mensagem aparece após o cabeçalho MIME (do inglês *Multipurpose Internet Mail Extensions* – Extensões Multifunção para Mensagens da Internet). Os dados codificados da mensagem correspondem a $K_A^-(H(m))$, isto é, ao resumo de mensagem assinado digitalmente. Como discutimos antes, para que Bob verifique a integridade da mensagem, ele precisa ter acesso à chave pública de Alice.

A Figura 8.23 mostra uma mensagem PGP secreta. Essa mensagem também aparece após o cabeçalho MIME. É claro que a mensagem em texto aberto não está incluída na de e-mail secreta. Quando um remetente (como Alice) quer não apenas a confidencialidade, mas também a integridade, o PGP contém uma mensagem como a da Figura 8.23 dentro daquela da Figura 8.22.

```

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Bob:
Can I see you tonight?
Passionately yours, Alice
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset: noconv
yHJRHHGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----

```

Figura 8.22 Uma mensagem PGP assinada.

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=1KhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE
```

Figura 8.23 Uma mensagem PGP secreta.

O PGP também fornece um mecanismo para certificação de chaves públicas, mas esse mecanismo é bem diferente daquele da CA mais convencional. As chaves públicas do PGP são certificadas por uma *rede de confiança*. A própria Alice pode certificar qualquer par chave/usuário quando ela achar que esse par realmente está correto. Além disso, o PGP permite que Alice declare que ela confia em outro usuário para atestar a autenticidade de mais chaves. Alguns usuários do PGP assinam reciprocamente suas chaves montando grupos de assinatura de chaves. Os usuários se reúnem fisicamente, trocam chaves públicas e certificam suas chaves reciprocamente, assinando-as com suas chaves privadas.

8.6 PROTEGENDO CONEXÕES TCP: TLS

Na seção anterior, vimos como as técnicas criptográficas podem prover sigilo, integridade dos dados e autenticação do ponto final a aplicações específicas, ou seja, e-mail. Nesta seção, desceremos uma camada na pilha de protocolo e examinaremos como a criptografia pode aprimorar o TCP com os serviços de segurança, incluindo sigilo, integridade dos dados e autenticação do ponto final. Essa versão aprimorada do TCP é denominada **protocolo TLS**, padronizado pelo IETF (RFC 4346). Uma versão anterior e semelhante desse protocolo é o SSL versão 3.

O SSL foi na origem projetado pela Netscape, mas as ideias básicas por trás da proteção do TCP antecedem o trabalho da Netscape (p. ex., consulte Woo [1994]). Desde sua concepção, o SSL e seu sucessor, o TLS, obtiveram uma ampla implementação. O TLS é suportado por todos os navegadores Web e servidores Web populares, e é usado pelo Gmail e por basicamente todos os sites de comércio eletrônico na Internet (incluindo Amazon, eBay e TaoBao). Centenas de bilhões de dólares são gastos com o TLS a cada ano. Na verdade, se você já comprou qualquer coisa pela Internet com seu cartão de crédito, a comunicação entre seu navegador e servidor para essa compra foi quase certamente por meio do TLS. (Você pode identificar que o TLS está sendo usado por seu navegador quando o URL se iniciar com https: em vez de http:.)

Para entender a necessidade do TLS, vamos examinar um cenário de comércio pela Internet típico. Bob está navegando na Web e acessa o site Alice Incorporated, que está vendendo perfume. O site Alice Incorporated exibe um formulário no qual Bob deve inserir o tipo de perfume e a quantidade desejados, seu endereço e o número de seu cartão de crédito. Bob insere essas informações, clica em Enviar, e espera pelo recebimento (via correio postal comum) de seus perfumes; ele também espera pelo recebimento de uma cobrança pelo seu pedido na próxima fatura do cartão de crédito. Até o momento, tudo bem, mas se nenhuma medida de segurança for tomada, Bob poderia esperar por surpresas.

- Se nenhum sigilo (criptografia) for utilizado, um invasor poderia interceptar o pedido de Bob e receber suas informações sobre o cartão. O invasor poderia, então, fazer compras à custa de Bob.
- Se nenhuma integridade de dados for utilizada, um invasor poderia modificar o pedido de Bob, fazendo-o comprar dez vezes mais frascos de perfumes que o desejado.

- Finalmente, se nenhuma autenticação do servidor for utilizada, um servidor poderia exibir o famoso logotipo da Alice Incorporated, quando na verdade o site é mantido por Trudy, que está se passando por Alice Incorporated. Após receber o pedido de Bob, Trudy poderia ficar com o dinheiro dele e sumir. Ou Trudy poderia realizar um roubo de identidade obtendo o nome, endereço e número do cartão de crédito de Bob.

O TLS resolve essas questões aprimorando o TCP com sigilo, integridade dos dados, autenticação do servidor e autenticação do cliente.

Muitas vezes, o TLS é usado para oferecer segurança em transações que ocorrem pelo HTTP. Entretanto, como o TLS protege o TCP, ele pode ser empregado por qualquer aplicação que execute o TCP. O TLS provê uma interface de programação de aplicações (API, do inglês *application programming interface*) com *sockets*, semelhante à API do TCP. Quando uma aplicação quer empregar o TLS, ela inclui classes/bibliotecas SSL. Como mostrado na Figura 8.24, embora o TLS resida tecnicamente na camada de aplicação, do ponto de vista do desenvolvedor, ele é um protocolo de transporte que provê serviços do TCP aprimorados com serviços de segurança.

8.6.1 Uma visão abrangente

Começamos descrevendo uma versão simplificada do TLS, que nos permitirá obter uma visão abrangente de *por que* e *como* funciona o TLS. Vamos nos referir a essa versão simplificada do TLS como “quase-TLS”. Após descrever o quase-TLS, na próxima subseção, descreveremos o TLS de verdade, preenchendo as lacunas. O quase-TLS (e o TLS) possui três fases: *apresentação (handshake)*, *derivação de chave* e *transferência de dados*. Agora descreveremos essas três fases para uma sessão de comunicação entre um cliente (Bob) e um servidor (Alice), o qual possui um par de chaves pública/privada e um certificado que associa sua identidade à chave pública.

Apresentação (*handshake*)

Durante a fase de apresentação, Bob precisa (a) estabelecer uma conexão TCP com Alice, (b) verificar se ela é *realmente* Alice, e (c) enviar-lhe uma chave secreta mestre, que será utilizada por Alice e Bob para criar todas as chaves simétricas de que eles precisam para a sessão TLS. Essas três etapas estão ilustradas na Figura 8.25. Observe que, uma vez estabelecida a conexão TCP, Bob envia a Alice uma mensagem “hello”. Alice, então, responde com seu certificado, que contém sua chave pública. Conforme discutido na Seção 8.3, como o certificado foi assinado por uma CA, Bob sabe, com certeza, que a chave pública no certificado pertence a Alice. Ele então cria um Segredo Mestre (MS, do inglês *Master Secret*) (que será usado somente para esta sessão TLS), codifica o MS com a chave pública de Alice para criar o Segredo Mestre Cifrado (EMS, do inglês *Encrypted Master Secret*), enviando-o para Alice, que o decodifica com sua chave privada para obter o MS. Após essa fase, Bob e Alice (e mais ninguém) sabem o segredo mestre para esta sessão TLS.

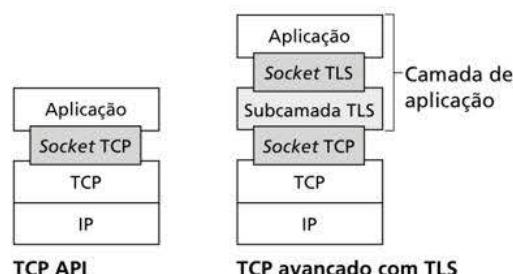


Figura 8.24 Embora o TLS resida tecnicamente na camada de aplicação, do ponto de vista do desenvolvedor, ele é um protocolo da camada transporte.

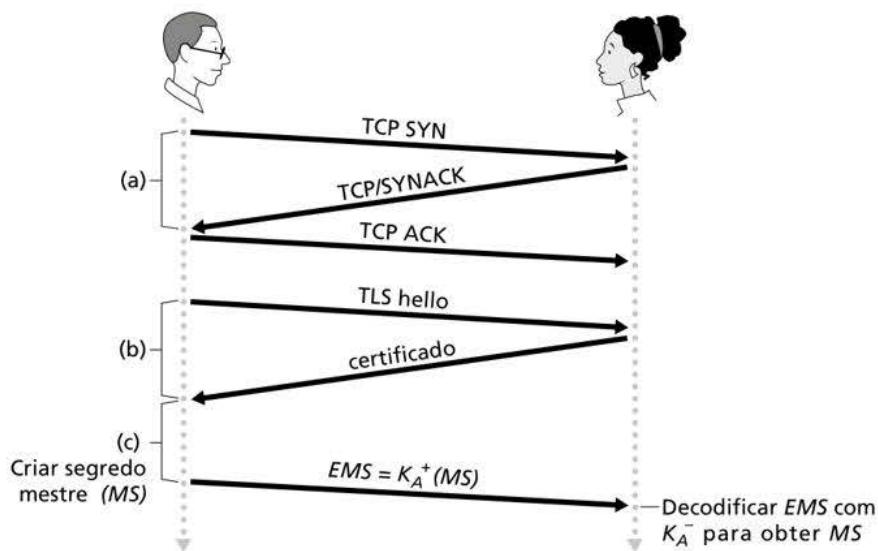


Figura 8.25 A apresentação quase-TLS, iniciando com uma conexão TCP.

Derivação de chave

A princípio, o MS, agora compartilhado por Bob e Alice, poderia ser usado como a chave de sessão simétrica para toda a verificação subsequente de criptografia e integridade dos dados. Entretanto, em geral, é considerado mais seguro para Alice e Bob usarem, individualmente, chaves criptográficas diferentes, bem como chaves diferentes para criptografia e verificação da integridade. Assim, Alice e Bob usam o MS para criar quatro chaves:

- E_B = chave de criptografia de sessão para dados enviados de Bob para Alice.
- M_B = chave HMAC de sessão para dados enviados de Bob para Alice, onde HMAC (RFC 2104) é um MAC resumido padronizado que encontramos na Seção 8.3.2.
- E_A = chave de criptografia de sessão para dados enviados de Alice para Bob.
- M_A = chave HMAC de sessão para dados enviados de Alice para Bob.

Alice e Bob podem criar quatro chaves a partir do MS. Isso poderia ser feito apenas dividindo o MS em quatro chaves. (Mas, no TLS *real*, é um pouco mais complicado, como veremos.) Ao final da fase de derivação de chave, Alice e Bob têm todas as quatro chaves. As duas chaves de criptografia serão usadas para cifrar dados; as duas chaves MAC serão usadas para verificar a integridade dos dados.

Transferência de dados

Agora que Alice e Bob compartilham as mesmas quatro chaves de sessão (E_B , M_B , E_A e M_A), podem começar a enviar dados protegidos um ao outro por meio da conexão TCP. Como o TCP é um protocolo de fluxo de bytes, uma abordagem natural seria o TLS cifrar dados da aplicação enquanto são enviados e, então, transmitir os dados cifrados para o TCP. Mas, se fizéssemos isso, onde colocaríamos o HMAC para a verificação da integridade? Decerto não queremos esperar até o fim da sessão TCP para verificar a integridade de todos os dados de Bob que foram enviados por toda a sessão! Para abordar essa questão, o TLS divide o fluxo de dados em registros, anexa um HMAC a cada registro para a verificação da integridade, e cifra o registro+HMAC. Para criar o HMAC, Bob insere os dados de registro junto com a chave MB em uma função de hash, conforme discutido na Seção 8.3. Para cifrar o pacote registro+HMAC, Bob usa sua chave de criptografia de sessão E_B . Esse pacote cifrado é então transmitido ao TCP para o transporte pela Internet.

Embora essa abordagem seja robusta, o fornecimento de integridade dos dados para um fluxo inteiro de mensagem ainda não é à prova de falhas. Em particular, suponha que Trudy seja uma mulher no meio e possua a habilidade de inserir, excluir e substituir segmentos no fluxo dos segmentos TCP enviados entre Alice e Bob. Trudy, por exemplo, poderia capturar dois segmentos enviados por Bob, inverter sua ordem, ajustar os números de sequência TCP (que não estão cifrados) e enviar dois segmentos na ordem inversa para Alice. Supondo que cada segmento TCP encapsula exatamente um registro, vamos verificar como Alice os processaria.

1. O TCP que está sendo executado em Alice pensaria que está tudo bem e passaria os dois registros para a subcamada TLS.
2. O TLS em Alice decodificaria os dois registros.
3. O TLS em Alice usaria o HMAC em cada registro para verificar a integridade dos dados dos dois registros.
4. O TLS passaria, então, os fluxos de bytes decifrados dos dois registros à camada de aplicação; mas o fluxo de bytes completo recebido por Alice não estaria na ordem correta em razão da inversão dos registros!

Recomendamos que você analise cenários diferentes para quando Trudy remove segmentos ou para quando Trudy repete segmentos.

A solução para esse problema, como você deve ter imaginado, é usar números de sequência. O TLS os utiliza da seguinte maneira. Bob mantém um contador de números de sequência, que se inicia no zero e vai aumentando para cada registro TLS que ele envia. Bob, na verdade, não inclui um número de sequência no próprio registro, mas no cálculo do HMAC. Desse modo, o HMAC é agora um hash dos dados mais uma chave HMAC M_B *mais o número de sequência atual*. Alice rastreia os números de sequência de Bob, permitindo que ela verifique a integridade dos dados de um registro incluindo o número de sequência apropriado no cálculo do HMAC. O uso de números de sequência TLS impede que Trudy realize um ataque *woman-in-the-middle*, como reordenar ou repetir os segmentos. (Por quê?)

Registro TLS

O registro TLS (assim como o registro quase-TLS) é ilustrado na Figura 8.26. Ele consiste em um campo de tipo, um campo de versão, um campo de comprimento, um campo de dados e um campo HMAC. Observe que os primeiros três campos não estão cifrados. O campo de tipo indica se o registro é uma mensagem de apresentação ou uma mensagem que contém dados da aplicação. É também usado para encerrar a conexão TLS, como discutido a seguir. Na extremidade receptora, o TLS usa o campo de comprimento para extrair os registros TLS do fluxo de bytes TCP da entrada. O campo de versão não precisa de explicações.

8.6.2 Uma visão mais completa

A subseção anterior abordou o protocolo quase-TLS; serviu para nos dar uma compreensão básica de por que e como funciona o TLS. Agora que temos essa compreensão básica sobre o TLS, podemos nos aprofundar mais e examinar os princípios básicos do verdadeiro

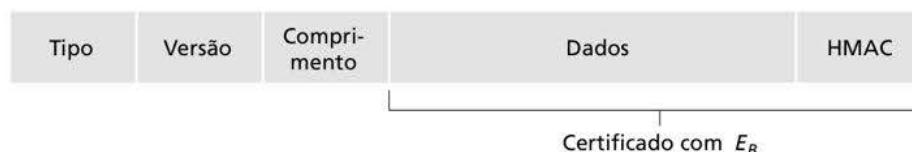


Figura 8.26 Formato de registro para o TLS.

protocolo TLS. Além da leitura desta descrição, recomendamos que você complete o laboratório Wireshark TLS, disponível no site deste livro.

Apresentação TLS

O TLS não exige que Alice e Bob usem um algoritmo específico de chave simétrica ou um algoritmo de chave pública. Em vez disso, permite que Alice e Bob combinem os algoritmos criptográficos no início da sessão TLS, durante a fase de apresentação. Ademais, durante essa fase, Alice e Bob enviam nonces um ao outro, que são usados na criação de chaves de sessão (E_B , M_B , E_A e M_A). As etapas da apresentação do TLS real são:

1. O cliente envia uma lista de algoritmos criptográficos que ele suporta, junto com um nonce do cliente.
2. A partir da lista, o servidor escolhe um algoritmo simétrico (p. ex., AES), um algoritmo de chave pública (p. ex., RSA com um comprimento de chave específico) e um algoritmo HMAC (MD5 ou SHA-1) junto com as chaves HMAC. Ele devolve ao cliente suas escolhas, bem como um certificado e um nonce do servidor.
3. O cliente verifica o certificado, extrai a chave pública do servidor, gera um Segredo Pré-mestre (PMS, do inglês *Pre-Master Secret*), cifra o PMS com a chave pública do servidor, e envia o PMS cifrado ao servidor.
4. Utilizando a mesma função de derivação de chave (conforme especificado pelo padrão TLS), o cliente e o servidor calculam independentemente o MS do PMS e dos nonces. O MS é então dividido para gerar as duas chaves de criptografia e duas chaves HMAC. Além disso, quando a cifra simétrica selecionada emprega o CBC (como 3DES ou AES), então dois IVs – um para cada lado da conexão – são também obtidos do MS. De agora em diante, todas as mensagens enviadas entre o cliente e o servidor são cifradas e autenticadas (com o HMAC).
5. O cliente envia um HMAC de todas as mensagens de apresentação.
6. O servidor envia um HMAC de todas as mensagens de apresentação.

As duas últimas etapas protegem a apresentação da adulteração. Para entender, observe que, no passo 1, o cliente normalmente oferece uma lista de algoritmos – alguns fortes e outros fracos. Essa lista é enviada em texto aberto, visto que os algoritmos de criptografia e chaves ainda não foram consentidos. Trudy, como uma *woman-in-the-middle*, poderia excluir os algoritmos mais fortes da lista, obrigando o cliente a selecionar um algoritmo fraco. Para evitar o ataque de adulteração, na etapa 5, o cliente envia um HMAC da concatenação de todas as mensagens de apresentação que ele enviou e recebeu. O servidor pode comparar esse HMAC com o das mensagens de apresentação que ele enviou e recebeu. Se houver uma inconsistência, o servidor pode finalizar a conexão. De maneira semelhante, o servidor envia um HMAC das mensagens de apresentação que encontrou, permitindo que o cliente verifique a presença de inconsistências.

Você talvez esteja questionando por que existem nonces nas etapas 1 e 2. Os números de sequência não são suficientes para prevenir o ataque de repetição de segmento? A resposta é sim, mas eles não previnem sozinhos “o ataque de repetição de conexão”. Considere o seguinte ataque de repetição de conexão. Suponha que Trudy analise todas as mensagens entre Alice e Bob. No dia seguinte, ela se passa por Bob e envia a Alice exatamente a mesma sequência de mensagens que Bob enviou a Alice no dia anterior. Se Alice não usar os nonces, ela responderá exatamente com a mesma sequência de mensagens que enviou no dia anterior. Alice não suspeitará de nada estranho, pois cada mensagem que ela recebe passará pela verificação de integridade. Se Alice for um servidor de comércio eletrônico, pensará que Bob está efetuando um segundo pedido (para a mesma coisa). Por outro lado, ao incluir um nonce no protocolo, Alice enviará nonces diferentes para cada sessão TCP, modificando as chaves de criptografia nos dois dias. Portanto, quando Alice recebe os registros TLS repetidos de Trudy, eles falharão na verificação de integridade, e a transação do site de comércio

eletrônico falso não acontecerá. Em resumo, no TLS, os nonces são usados para proteger o “ataque de repetição de conexão”, e os números de sequência são usados para defender a repetição de pacotes individuais durante uma sessão em andamento.

Encerramento de conexão

Em algum momento, Bob ou Alice desejarão finalizar a sessão TLS. Um método seria deixar Bob finalizar a sessão TLS apenas encerrando a conexão TCP subjacente – ou seja, enviando um segmento TCP FIN para Alice. Mas esse plano ingênuo prepara o terreno para o *ataque por truncamento* pelo qual Trudy, mais uma vez, se localiza no meio de uma sessão TLS em andamento e a finaliza antes da hora com um TCP FIN. Se Trudy fizesse isso, Alice acharia que recebeu todos os dados de Bob, quando, na verdade, recebeu só uma parte deles. A solução para esse problema é indicar, no campo de tipo, se o registro serve para encerrar a sessão TLS. (Embora o tipo TLS seja enviado em aberto, ele é autenticado no receptor usando o HMAC do registro.) Ao incluir esse campo, se Alice fosse receber um TCP FIN antes de receber um registro TLS de encerramento, ela saberia que algo estranho estava acontecendo.

Isso conclui nossa introdução ao TLS. Vimos que ele usa muitos dos princípios da criptografia discutidos nas Seções 8.2 e 8.3. Os leitores que querem explorar o TLS mais profundamente podem consultar o livro de Rescorla sobre o SSL/TLS (Rescorla, 2001).

8.7 SEGURANÇA NA CAMADA DE REDE: IPSEC E REDES PRIVADAS VIRTUAIS

O protocolo IP de segurança, mais conhecido como **IPsec**, provê segurança na camada de rede. O IPsec protege os datagramas IP entre quaisquer entidades da camada de rede, incluindo hospedeiros e roteadores. Como descreveremos em breve, muitas instituições (corporações, órgãos do governo, organizações sem fins lucrativos etc.) usam o IPsec para criar **redes privadas virtuais (VPNs)** (do inglês *virtual private networks*) que trabalham em cima da Internet pública.

Antes de falar sobre o IPsec, vamos voltar e considerar o que significa prover sigilo na camada de rede. Com o sigilo da camada de rede entre um par de entidades da rede (p. ex., entre dois roteadores, entre dois hospedeiros, ou entre um roteador e um hospedeiro), a entidade remetente cifra as cargas úteis de todos os datagramas que envia à entidade destinatária. A carga útil cifrada poderia ser um segmento TCP, um segmento UDP (do inglês *User Datagram Protocol* – Protocolo de Datagrama de Usuário) ou uma mensagem ICMP (do inglês *Internet Control Message Protocol* – Protocolo de Mensagens de Controle da Internet), etc. Se esse serviço estivesse em funcionamento, todos os dados enviados de uma entidade a outra – incluindo e-mail, páginas Web, mensagens de apresentação TCP e mensagens de gerenciamento (como ICMP e SNMP [do inglês *Simple Network Management Protocol* – Protocolo Simples de Gerenciamento de Rede]) – ficariam ocultos de qualquer intruso que estivesse investigando a rede. Por essa razão, a segurança na camada de rede é conhecida por prover “cobertura total”.

Além do sigilo, um protocolo de segurança da camada de rede poderia prover outros serviços de segurança. Por exemplo, fornecer autenticação da origem, de modo que a entidade destinatária possa verificar a origem do datagrama protegido. Um protocolo de segurança da camada de rede poderia oferecer integridade dos dados, para que a entidade destinatária verificasse qualquer adulteração do datagrama que pudesse ter ocorrido enquanto o datagrama estava em trânsito. Um serviço de segurança da camada de rede também poderia oferecer a prevenção do ataque de repetição, querendo dizer que Bob conseguiria detectar quaisquer datagramas duplicados que um atacante pudesse inserir. Veremos em breve que o IPsec provê mecanismos para todos esses serviços de segurança, ou seja, para sigilo, autenticação da origem, integridade dos dados e prevenção do ataque de repetição.

8.7.1 IPsec e redes privadas virtuais (VPNs)

Uma instituição que se estende por diversas regiões geográficas muitas vezes deseja ter sua própria rede IP, para que seus hospedeiros e servidores consigam enviar dados um ao outro de uma maneira segura e sigilosa. Para alcançar essa meta, a instituição poderia, na verdade, empregar uma rede física independente – incluindo roteadores, enlaces e uma infraestrutura de DNS – que é completamente separada da Internet pública. Essa rede disjunta, reservada a uma instituição particular, é chamada de **rede privada**. Como é de se esperar, uma rede privada pode ser muito cara, já que a instituição precisa comprar, instalar e manter sua própria infraestrutura de rede física.

Em vez de implementar e manter uma rede privada, hoje muitas instituições criam VPNs em cima da Internet pública. Com uma VPN, o tráfego interdepartamental é enviado por meio da Internet pública, e não por meio de uma rede fisicamente independente. Mas, para prover sigilo, esse tráfego é criptografado antes de entrar na Internet pública. Um exemplo simples de VPN é mostrado na Figura 8.27. Aqui, a instituição consiste em uma matriz, uma filial e vendedores viajantes, que normalmente acessam a Internet do seu quarto de hotel. (A figura mostra só um vendedor.) Nesta VPN, quando dois hospedeiros dentro da matriz enviam datagramas IP um ao outro ou quando dois hospedeiros dentro de uma filial querem se comunicar, eles usam o bom e velho IPv4 (i.e., sem os serviços IPsec). Entretanto, quando dois dos hospedeiros da instituição se comunicam por um caminho que cruza a Internet pública, o tráfego é codificado antes de entrar na Internet.

Para entender como a VPN funciona, vamos examinar um exemplo simples no contexto da Figura 8.27. Quando um hospedeiro na matriz envia um datagrama IP a um vendedor no hotel, o roteador de borda na matriz converte o datagrama IPv4 em um IPsec e o encaminha à Internet. Esse datagrama IPsec, na verdade, possui um cabeçalho IPv4 tradicional, de modo que os roteadores na Internet pública processam o datagrama como se ele fosse um IPv4 comum – para eles, o datagrama é perfeitamente comum. Mas, conforme ilustrado na Figura 8.27, a carga útil do datagrama IPsec inclui um cabeçalho IPsec, que é utilizado

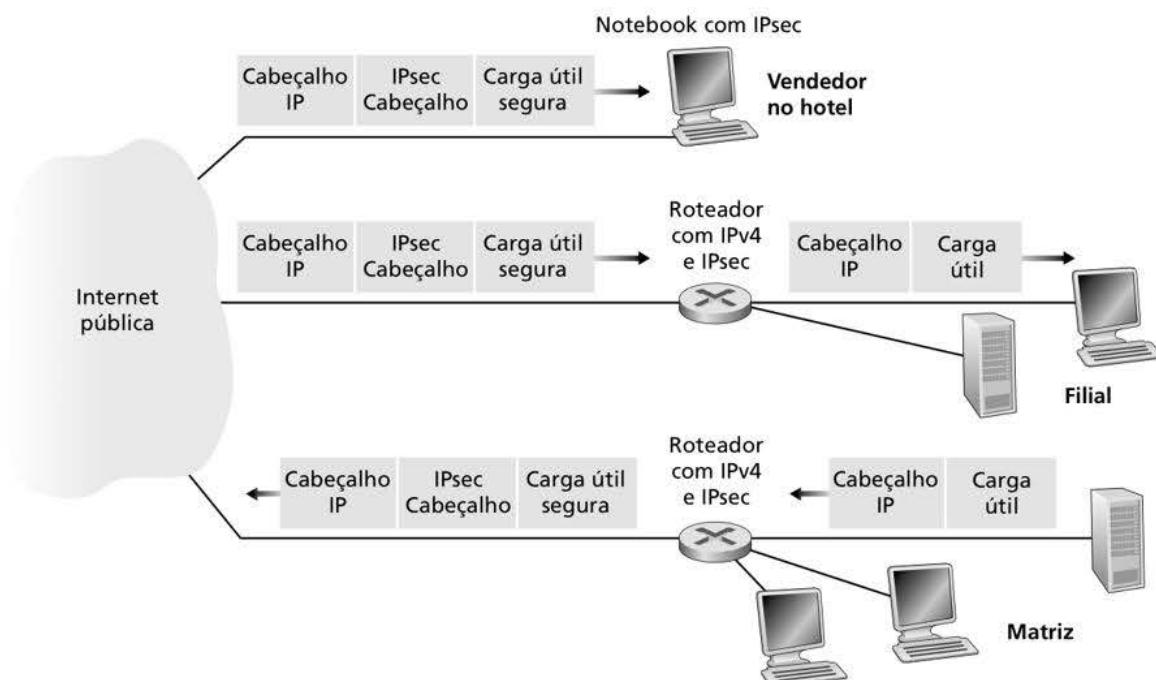


Figura 8.27 Rede privada virtual (VPN).

para o processamento do IPsec; além disso, a carga útil do datagrama IPsec está codificada. Quando o datagrama IPsec chegar ao notebook do vendedor, o sistema operacional no notebook decodifica a carga útil (e provê outros serviços de segurança, como a verificação da integridade dos dados) e passa a carga útil não codificada para o protocolo da camada superior (p. ex., para o TCP ou UDP).

Acabamos de dar uma visão geral de alto nível de como uma instituição pode implementar o IPsec para criar uma VPN. Para ver a floresta por entre as árvores, deixamos de lado muitos detalhes importantes. Vamos agora dar uma olhada neles mais de perto.

8.7.2 Os protocolos AH e ESP

O IPsec é um material um tanto complexo – ele é definido em mais de uma dúzia de RFCs. Dois RFCs importantes são RFC 4301, o qual descreve a arquitetura de segurança IP geral, e RFC 6071, que fornece uma visão geral dos protocolos IPsec. Nossa meta neste livro, como de costume, não é apenas reprocessar os RFCs secos e complexos, mas fazer uma abordagem mais operacional e pedagógica para descrever os protocolos.

No conjunto dos protocolos IPsec, existem dois principais: o protocolo **Cabeçalho de Autenticação (AH**, do inglês *Authentication Header*) e o protocolo **Carga de Segurança de Encapsulamento (ESP**, do inglês *Encapsulation Security Payload*). Quando uma entidade remetente IPsec (em geral, um hospedeiro ou um roteador) envia datagramas seguros a uma entidade destinatária (também um hospedeiro ou um roteador), ela utiliza o protocolo AH ou o protocolo ESP. O protocolo AH provê autenticação da origem e integridade dos dados, mas *não* provê sigilo. O protocolo ESP provê autenticação da origem, integridade dos dados *e* sigilo. Visto que o sigilo é muitas vezes essencial às VPNs e outras aplicações IPsec, o protocolo ESP é muito mais utilizado do que o protocolo AH. Para desmistificar o IPsec e evitar suas complexidades, a partir de agora estaremos focados exclusivamente no protocolo ESP. Os leitores que desejam aprender também sobre o protocolo AH devem explorar os RFCs e outros recursos online.

8.7.3 Associações de segurança

Os datagramas IPsec são enviados entre pares de entidades da rede, tal como entre dois hospedeiros, entre dois roteadores, ou entre um hospedeiro e um roteador. Antes de enviar datagramas IPsec da entidade remetente à destinatária, essas entidades criam uma conexão lógica da camada de rede, denominada **associação de segurança (SA**, do inglês *security association*). Uma SA é uma conexão lógica simples; ou seja, ela é unidirecional do remetente ao destinatário. Se as duas entidades querem enviar datagramas seguros entre si, então duas SAs (i.e., duas conexões lógicas) precisam ser estabelecidas, uma em cada direção.

Por exemplo, considere mais uma vez uma VPN institucional na Figura 8.27. Essa instituição consiste em uma filial, uma matriz e, digamos, n vendedores viajantes. Por exemplo, vamos supor que haja tráfego IPsec bidirecional entre a matriz e a filial e tráfego IPsec bidirecional entre a matriz e os vendedores viajantes. Quantas SAs existem nessa VPN? Para responder a essa questão, observe que há duas SAs entre o roteador de borda da matriz e o de borda da filial (uma em cada direção); para cada notebook do vendedor, há duas SAs entre o roteador de borda da matriz e o notebook (de novo, uma em cada direção). Portanto, no total, há $(2 + 2n)$ SAs. *Mantenha em mente, entretanto, que nem todo o tráfego enviado à Internet pelos roteadores de borda ou pelos notebooks será protegido por IPsec.* Podemos citar como exemplo um hospedeiro na matriz que quer acessar ao servidor Web (como Amazon ou Google) na Internet pública. Assim, o roteador de borda (e os notebooks) emitirão na Internet tanto datagramas IPv4 comuns como datagramas IPsec seguros.

Vamos agora olhar “por dentro” de uma SA. Para tornar essa discussão tangível e concreta, utilizaremos o contexto de uma SA do roteador R1 ao roteador R2 na Figura 8.28. (Você pode pensar no Roteador R1 como o roteador de borda da matriz, e o Roteador R2

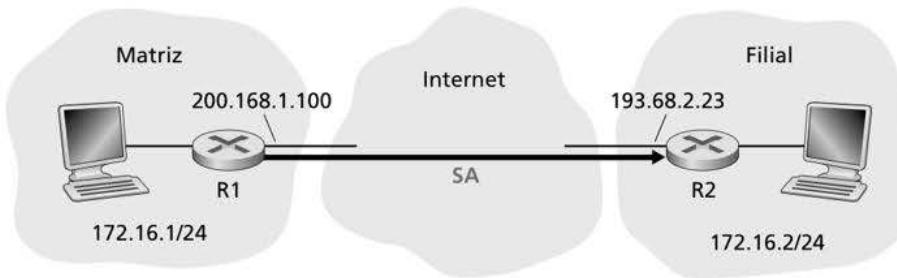


Figura 8.28 Associação de segurança (SA) de R1 a R2.

como o roteador de borda da filial, conforme a Figura 8.27.) O Roteador R1 manterá as informações de estado sobre essa SA, as quais incluirão:

- Um identificador de 32 bits para a SA, denominado **Índice de Parâmetro de Segurança (SPI, do inglês Security Parameter Index)**.
- A interface remetente da SA (neste caso, 200.168.1.100) e a interface destinatária da SA (neste caso, 193.68.2.23).
- O tipo de criptografia a ser usada (p. ex., 3DES com CBC).
- A chave de criptografia.
- O tipo de verificação de integridade (p. ex., HMAC com MD5).
- A chave de autenticação.

Quando o roteador R1 precisa construir um datagrama IPsec para encaminhar por essa SA, ele acessa as informações de estado para determinar como deveria autenticar e criptografar o datagrama. De maneira semelhante, o roteador R2 manterá as mesmas informações de estado sobre essa SA e as usará para autenticar e decodificar qualquer datagrama IPsec que chegar da SA.

Uma entidade IPsec (roteador ou hospedeiro) muitas vezes guarda essas informações de estado para muitas SAs. No exemplo sobre a VPN na Figura 8.27 com n vendedores, o roteador de borda guarda informações de estado para $(2 + 2n)$ SAs. Uma entidade IPsec armazena as informações de estado para todas as suas SAs em seu **Banco de Dados de Associação de Segurança (SAD, do inglês Security Association Database)**, o qual é uma estrutura de dados do núcleo do sistema operacional.

8.7.4 O datagrama IPsec

Após descrever sobre as SAs, podemos agora descrever o verdadeiro datagrama IPsec. Ele possui duas formas diferentes de pacotes, uma para o **modo túnel** e outra para o **modo transporte**. O modo túnel, o mais apropriado para as VPNs, é mais implementado do que o modo transporte. Para desmistificar o IPsec e evitar suas complexidades, a partir de agora vamos nos concentrar exclusivamente no modo túnel. Uma vez que você tiver uma compreensão sólida do modo túnel, poderá com facilidade aprender, por sua conta, o modo transporte.

O formato do pacote do datagrama IPsec é ilustrado na Figura 8.29. Você pode pensar que os formatos do pacote são maçantes e sem graça, mas logo veremos que o datagrama IPsec na verdade parece e tem gosto de uma iguaria mexicana! Vamos examinar os campos do IPsec no contexto da Figura 8.28. Suponha que o roteador R1 receba um datagrama IPv4 comum do hospedeiro 172.16.1.17 (na rede da matriz) destinado ao hospedeiro 172.16.2.48 (na rede da filial). R1 utiliza a seguinte receita para converter esse “datagrama IPv4 original” em um datagrama IPsec:

- Anexa ao final do datagrama IPv4 original (que inclui os campos de cabeçalho originais) um campo de “trailer ESP”.
- Codifica o resultado utilizando o algoritmo e a chave especificados pela SA.

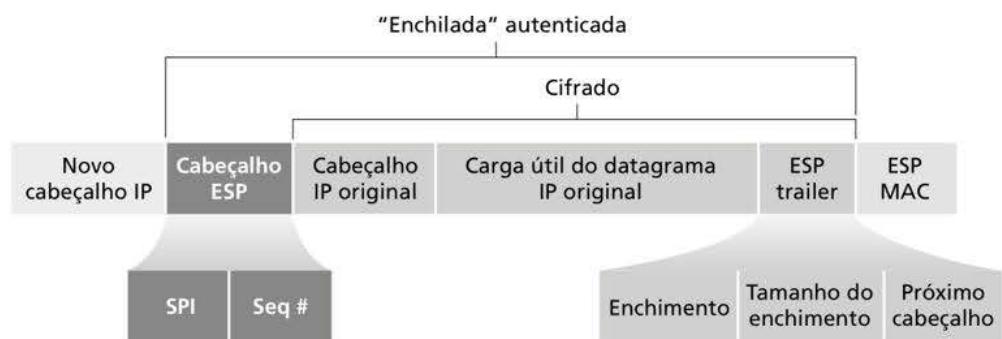


Figura 8.29 Formato do datagrama IPsec.

- Anexa na frente dessa quantidade codificada um campo chamado “cabeçalho ESP”; o pacote resultante é chamado de “enchilada”.
- Cria uma autenticação MAC sobre a *enchilada inteira*, usando o algoritmo e a chave especificados na SA.
- Anexa a MAC atrás da enchilada, formando a *carga útil*.
- Por fim, cria um cabeçalho IP novo com todos os campos de cabeçalho IPv4 clássicos (juntos, normalmente com 20 bytes de comprimento), o qual se anexa antes da carga útil.

Observe que o datagrama IP resultante é um autêntico datagrama IPv4, com os tradicionais campos de cabeçalho IPv4 acompanhados por uma carga útil. Mas, neste caso, a carga útil contém um cabeçalho ESP, o datagrama IP original, um trailer ESP e um campo de autenticação ESP (com o datagrama original e o trailer ESP cifrados). O datagrama IP original possui o endereço IP remetente 172.16.1.17 e o endereço IP destinatário 172.16.2.48. Em razão de o datagrama IPsec incluir o IP original, esses endereços são inclusos (e cifrados) como parte da carga útil do pacote IPsec. Mas e os endereços IP remetente e destinatário que estão no novo cabeçalho IP, ou seja, o cabeçalho localizado à esquerda do datagrama IPsec? Como você pode esperar, eles são estabelecidos para as interfaces do roteador remetente e destinatário nas duas extremidades dos túneis, isto é, 200.168.1.100 e 193.68.2.23. Além disso, o número do protocolo nesse novo campo de cabeçalho IPv4 não será o número do TCP, UDP ou SMTP, mas igual a 50, determinando que esse é um datagrama IPsec que está utilizando o protocolo ESP.

Após R1 enviar um datagrama IPsec à Internet pública, ele passará por diversos roteadores antes de chegar ao R2. Cada um desses roteadores processará o datagrama como se fosse um datagrama comum – eles são inconscientes do fato de que o datagrama está transportando dados cifrados pelo IPsec. Para esses roteadores da Internet pública, já que o endereço IP remetente no cabeçalho externo é R2, o destino final do datagrama é R2.

Depois de acompanhar um exemplo de como o datagrama IPsec é construído, vamos olhar de perto os ingredientes da enchilada. Vemos na Figura 8.29 que o trailer ESP consiste em três campos: enchimento, tamanho do enchimento e próximo cabeçalho. Lembre-se de que cifras de bloco exigem que a mensagem a ser cifrada seja um múltiplo inteiro do comprimento de bloco. O enchimento (que consiste em bytes sem significado) é usado de modo que, quando adicionada ao datagrama original (junto com o tamanho do enchimento e o próximo cabeçalho), a “mensagem” resultante tenha um número inteiro de blocos. O campo tamanho do enchimento indica à entidade destinatária quanto enchimento foi inserido (e, portanto, precisa ser removido). O próximo cabeçalho identifica o tipo (p. ex., UDP) de dados contidos no campo de dados da carga útil. Os dados da carga útil (em geral, o datagrama IP original) e o trailer ESP são concatenados e, então, cifrados.

Anexado na frente dessa unidade cifrada está o cabeçalho ESP, o qual é enviado em aberto e consiste em dois campos: o SPI e o campo de número de sequência. O SPI indica à entidade destinatária a SA à qual o datagrama pertence; essa entidade pode, então, indexar seu SAD com o SPI para determinar os algoritmos e chaves apropriados

de autenticação/decriptação. O campo de número de sequência é usado para a proteção contra ataques de repetição.

A entidade remetente também anexa uma autenticação MAC. Como já dissemos, a entidade remetente calcula um MAC por toda a enchilada (que consiste em um cabeçalho ESP, o datagrama IP original e o trailer ESP – com a criptografia do datagrama e do trailer). Lembre-se de que, para calcular um MAC, o remetente anexa uma chave MAC secreta à enchilada e calcula um hash de tamanho fixo do resultado.

Quando R2 recebe o datagrama IPsec, ele observa que o endereço IP destinatário do datagrama é o próprio R2, que, então, processa o datagrama. Como o campo de protocolo (no cabeçalho IP à esquerda) é 50, R2 entende que deve aplicar o processamento IPsec ESP ao datagrama. Primeiro, alinhando na enchilada, R2 usa o SPI para determinar à qual SA o datagrama pertence. Segundo, ele calcula o MAC da enchilada e verifica se o MAC é compatível com o valor do campo MAC ESP. Se for, ele sabe que a enchilada vem de R1 e que não foi adulterada. Terceiro, verifica o campo número de sequência para ver se o datagrama é novo (e não repetido). Quarto, ele decodifica a unidade cifrada utilizando o algoritmo e a chave de criptografia associados à SA. Quinto, ele remove o enchimento e extrai o datagrama IP original básico. E, por fim, sexto, ele encaminha o datagrama original à rede da filial em direção a seu destino final. Ufa, que receita complicada, não é? Ninguém disse que era fácil preparar e desvendar uma enchilada!

Na verdade, existe outra importante sutileza que precisa ser abordada. Ela se baseia na seguinte questão: quando R1 recebe um datagrama (inseguro) de um hospedeiro na rede da matriz, e esse datagrama é destinado a algum endereço IP destinatário fora da matriz, como R1 sabe se ele deve ser convertido em um datagrama IPsec? E se ele vai ser processado por um IPsec, como R1 sabe qual SA (de muitas SAs em seu SAD) deve ser usada para construir o datagrama IPsec? O problema é resolvido da seguinte maneira. Junto com um SAD, a entidade IPsec também mantém outra estrutura de dados denominada **Banco de Dados de Política de Segurança (SPD**, do inglês *Security Policy Database*). Este indica quais tipos de datagramas (como uma função do endereço IP remetente, endereço IP destinatário e tipo do protocolo) serão processados pelo IPsec; e para aqueles que serão processados pelo IPsec, qual SA deve ser usada. De certa forma, as informações em um SPD indicam “o que” fazer com um datagrama que está chegando; as informações no SAD indicam “como” fazer isso.

Resumo dos serviços IPsec

Quais serviços o IPsec provê, exatamente? Vamos examinar tais serviços do ponto de vista de um atacante, digamos Trudy, que é uma *woman-in-the-middle*, situada entre R1 e R2 na Figura 8.28. Suponha por toda essa discussão que Trudy não conhece as chaves de autenticação e criptografia usadas pela SA. O que Trudy pode e não pode fazer? Primeiro, ela não pode ver o datagrama original. Na verdade, não só os dados do datagrama original estão ocultos de Trudy, mas também o número de protocolo, o endereço IP remetente e o endereço IP destinatário. Em relação aos datagramas enviados através da SA, Trudy sabe apenas que eles vieram de 200.168.1.100 e são destinados a 193.68.2.23. Ela não sabe se está carregando dados TCP, UDP ou ICMP; ela não sabe que está carregando HTTP, SMTP ou quaisquer outros tipos de dados de aplicação. Esse sigilo, portanto, vai além do TLS. Segundo, suponha que Trudy tente adulterar um datagrama na SA alterando alguns de seus bits. Quando o datagrama alterado chegar a R2, a verificação de integridade falhará (usando um MAC), impedindo a tentativa maliciosa de Trudy mais uma vez. Terceiro, imagine que Trudy tente se passar por R1, criando um datagrama IPsec com remetente 200.168.1.100 e destinatário 193.68.2.23. O ataque será inútil, pois a verificação da integridade do datagrama em R2 falhará novamente. Por fim, em razão de o IPsec incluir números de sequência, Trudy não poderá criar um ataque de repetição bem-sucedido. Em resumo, conforme afirmado no início desta seção, o IPsec oferece – entre qualquer par de dispositivos que processam pacotes através da camada de rede – sigilo, autenticação da origem, integridade dos dados e proteção contra ataque de repetição.

8.7.5 IKE: gerenciamento de chave no IPsec

Quando uma VPN possui um número pequeno de pontos finais (p. ex., apenas dois roteadores como na Figura 8.28), o administrador de rede pode inserir manualmente informações sobre a SA (algoritmos e chaves de criptografia/autenticação e os SPIs) nos SADs dos pontos de chegada. Essa “teclagem manual” é claramente impraticável para uma VPN grande, a qual pode consistir em centenas ou mesmo milhares de roteadores e hospedeiros IPsec. Implementações grandes e geograficamente distribuídas exigem um mecanismo automático para a criação das SAs. O IPsec o faz com o protocolo de Troca de Chave (IKE, do inglês *Internet Key Exchange*), especificado em RFC 5996.

O IKE tem semelhanças com a apresentação (*handshake*) em TLS (consulte a Seção 8.6). Cada entidade IPsec possui um certificado, o qual inclui a chave pública da entidade. Da mesma forma que o TLS, o protocolo IKE tem os dois certificados de troca de entidades, autenticação de negociação e algoritmos de criptografia, e troca informações de chave com segurança para criar chaves de sessão nas SAs IPsec. Diferentemente do TLS, o IKE emprega duas fases para realizar essas tarefas.

Vamos investigar essas duas fases no contexto de dois roteadores, R1 e R2, na Figura 8.28. A primeira fase consiste em duas trocas de pares de mensagens entre R1 e R2:

- Durante a primeira troca de mensagens, os dois lados usam Diffie-Hellman (consulte os Problemas no final do capítulo) para criar um **IKE SA** bidirecional entre os roteadores. Para nos confundir, esse IKE SA bidirecional é totalmente diferente da SA IPsec discutida nas Seções 8.6.3 e 8.6.4. O IKE SA provê um canal cifrado e autenticado entre os dois roteadores. Durante a primeira troca de par de mensagens, as chaves são estabelecidas para a criptografia e autenticação para o IKE SA. Também é estabelecido um segredo mestre que será usado para calcular chaves SA IPsec mais adiante na fase 2. Observe que, durante a primeira etapa, as chaves públicas e privadas RSA não são usadas. Em particular, nem R1 nem R2 revelam sua identidade assinando uma mensagem com sua chave privada.
- Durante a segunda troca de mensagens, ambos os lados revelam sua identidade assinando suas mensagens. Entretanto, as identidades não são reveladas a um analisador passivo, pois são enviadas por um canal seguro IKE SA. Também nessa fase, os dois lados negociam os algoritmos de autenticação e criptografia que serão empregados pelas SAs IPsec.

Na fase 2 do IKE, os dois lados criam uma SA em cada direção. Ao fim da fase 2, as chaves de sessão de criptografia e autenticação são estabelecidas em ambos os lados para as duas SAs. Eles podem, então, usar as SAs para enviar datagramas seguros, como descrito nas Seções 8.7.3 e 8.7.4. A principal motivação de ter duas fases no IKE é o custo computacional – visto que a segunda fase não envolve qualquer chave pública de criptografia, o IKE pode criar um grande número de SAs entre as duas entidades IPsec com um custo computacional relativamente pequeno.

8.8 SEGURANÇA DE LANS SEM FIO E REDES CELULARES 4G/5G

A segurança é uma preocupação particularmente importante nas redes sem fio, nas quais o atacante pode analisar quadros simplesmente posicionando um dispositivo receptor em qualquer local ao alcance das transmissões do remetente, o que vale tanto para as LANs sem fio 802.11 quanto para as redes celulares 4G/5G. Em ambos os contextos, veremos o amplo uso das técnicas de segurança fundamentais estudadas anteriormente neste capítulo,

incluindo o uso de nonces para a autenticação, hashes criptográficas para a integridade da mensagem, derivação de chaves simétricas compartilhadas para a criptografia dos dados da sessão do usuário e amplo uso do padrão de criptografia AES. Veremos também, como no caso das configurações de Internet com fio, que os protocolos de segurança sem fio estão em evolução constante, pois pesquisadores e hackers estão sempre descobrindo falhas e pontos fracos nos protocolos de segurança existentes.

Nesta seção, apresentamos uma breve introdução à segurança sem fio nas redes 802.11(WiFi) e 4G/5G. Para um tratamento mais aprofundado sobre o tema, recomendamos a leitura dos livros sobre a segurança das redes 802.11 (Edney, 2003; Wright, 2015), de fácil leitura; a excelente cobertura sobre o tema da segurança em redes 3G/4G/5G em (Sauter, 2014); e estudos recentes (Zou, 2016; Kohlios, 2018).

8.8.1 Autenticação e acordo de chaves nas redes LAN sem fio 802.11

Começaremos nossa discussão sobre a segurança nas redes 802.11 com a identificação de duas (entre muitas [Zou, 2016]) questões de segurança críticas com as quais queremos que uma rede 802.11 saiba lidar:

- *Autenticação mútua.* Antes que um dispositivo móvel possa se ligar completamente a um ponto de acesso e enviar datagramas a hospedeiros remotos, a rede normalmente primeiro autentica o dispositivo para confirmar a sua identidade e verificar os seus privilégios de acesso. Da mesma forma, o dispositivo móvel quer autenticar a rede à qual está se ligando, para garantir que esta é realmente a rede à qual deseja se ligar. Essa autenticação bidirecional é conhecida pelo nome de **autenticação mútua**.
- *Criptografia.* Como quadros 802.11 são trocados por um canal sem fio que pode ser analisado e manipulado por intrusos em potencial, é importante criptografar os quadros do nível de enlace que transportam os dados do nível do usuário trocados entre o dispositivo móvel e o ponto de acesso (AP, do inglês *access point*). A criptografia de chaves simétricas é usada na prática, pois o processo de criptografar e decriptar deve ocorrer em altas velocidades. O dispositivo móvel e o AP precisam derivar as chaves simétricas de criptografia e decriptografia a serem usadas.

A Figura 8.30 ilustra o cenário no qual um dispositivo móvel deseja se ligar a uma rede 802.11. Vemos os dois componentes de rede tradicionais que encontramos em nosso estudo anterior sobre as redes 802.11 na Seção 7.3: o dispositivo móvel e o AP. Vemos também um novo componente da arquitetura, o **servidor de autenticação** (AS, do inglês *authentication server*), responsável por autenticar o dispositivo móvel. O servidor de autenticação pode estar localizado juntamente com o AP, mas a situação mais comum, mostrada na Figura 8.30, é que o AS é implementado como um servidor independente que presta serviços de autenticação. Para a autenticação, o AP serve como dispositivo de passagem, repassando mensagens de autenticação e de derivação de chaves entre o dispositivo móvel e o servidor de autenticação. Esse tipo de servidor de autenticação normalmente presta serviços de autenticação para todos os APs na sua rede.

É possível identificar quatro fases distintas do processo de autenticação mútua e derivação e uso de chaves criptográficas na Figura 8.30:

1. *Descoberta.* Na fase de descoberta, o AP anuncia sua presença e as formas de autenticação e criptografia que podem ser fornecidas ao dispositivo móvel. Este, então, solicita as formas específicas de autenticação e criptografia que deseja. Apesar de o dispositivo e o AP já estarem trocando mensagens, o dispositivo ainda não foi autenticado, nem tem uma chave criptografada para a transmissão de quadros através do enlace sem fio, então vários passos ainda serão necessários antes que o dispositivo possa se comunicar com segurança através do AP.

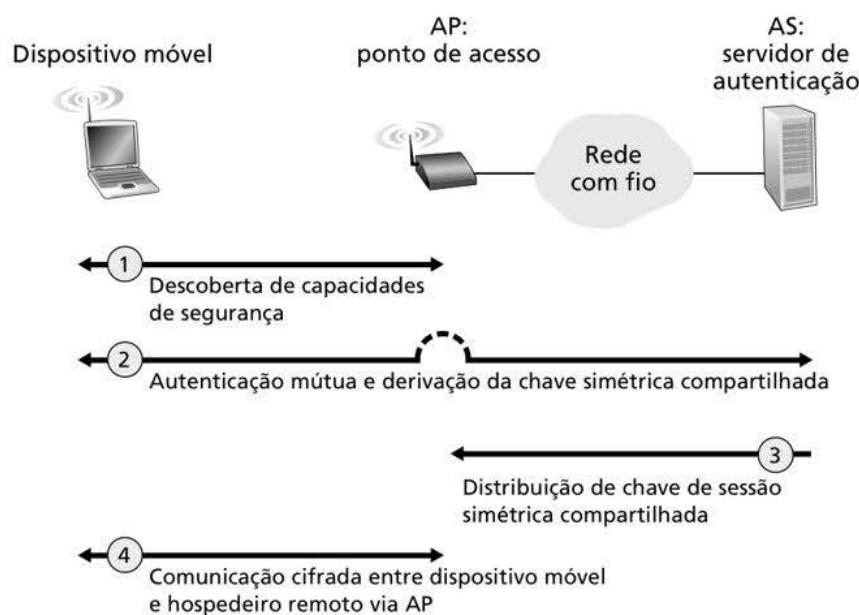


Figura 8.30 Autenticação mútua e derivação de chaves criptográficas no WPA.

2. *Autenticação mútua e derivação de chaves simétricas compartilhadas.* Este é o passo mais crítico para determinar a segurança do canal 802.11. Como veremos, o passo é bastante facilitado pela pressuposição (verdadeira na prática tanto para as redes 802.11 quanto para as 4G/5G) de que o servidor de autenticação e o dispositivo móvel já têm um **segredo compartilhado** antes de começarem a autenticação mútua. Nesse passo, o dispositivo e o servidor de autenticação usam esse segredo compartilhado, junto com nonces (para prevenir ataques de retransmissão) e hashes criptográficas (para garantir a integridade da mensagem), para autenticar um ao outro. Eles também derivam a chave da sessão compartilhada a ser usada pelo dispositivo móvel e o AP para criptografar quadros transmitidos pelo enlace sem fio 802.11.
3. *Distribuição das chaves de sessão simétricas compartilhadas.* Como a chave criptográfica simétrica é derivada no dispositivo móvel e no servidor de autenticação, será necessário um protocolo para que o servidor de autenticação informe o AP sobre a chave de sessão simétrica compartilhada. É algo relativamente simples, mas ainda um passo necessário.
4. *Comunicação criptografada entre o dispositivo móvel e um hospedeiro remoto através do AP.* Essa comunicação acontece da maneira discutida anteriormente na Seção 7.3.2, com os quadros da camada de enlace enviados entre o dispositivo móvel e o AP criptografados usando a chave de sessão compartilhada criada e distribuída nos Passos 2 e 3. A criptografia de chaves simétricas AES, discutida anteriormente na Seção 8.2.1, normalmente é usada na prática para criptografar/decriptar dados de quadros 802.11.

Autenticação mútua e derivação da chave de sessão simétrica compartilhada

Os temas da autenticação mútua e da derivação de chaves de sessão simétricas compartilhadas são os componentes centrais da segurança das redes 802.11. Como é aqui que as falhas de segurança de diversas versões anteriores da segurança nas redes 802.11 foram descobertas, trabalharemos esses desafios primeiro.

O tema de segurança em 802.11 tem atraído muita atenção em círculos técnicos e na mídia. Apesar de discussões consideráveis terem sido feitas, poucos debates aconteceram – existe um entendimento universal de que a especificação 802.11 original, conhecida

coletivamente pelo nome WEP (*Wired Equivalent Privacy* – Privacidade Equivalente à de Rede com fios), continha uma série de falhas graves na segurança (Fluhrer, 2001; Stubblefield, 2002). Depois que as falhas foram descobertas, logo havia disponível software de domínio público capaz de explorá-las, tornando os usuários de WLANs 802.11 protegidas por WEP tão vulneráveis a ataques de segurança quanto usuários que não utilizavam nenhum recurso de segurança. Os leitores interessados em aprender mais sobre a WEP podem consultar as referências, assim como edições anteriores deste livro-texto, que discutiam a WEP. Como sempre, materiais eliminados das novas edições estão disponíveis no site do livro.

O Acesso Protegido WiFi (WPA1, do inglês *WiFi Protected Access*) foi desenvolvido em 2003 pela WiFi Alliance (WiFi, 2020) para superar as falhas de segurança da WEP. A versão inicial do WPA1 introduzia verificações de integridade da mensagem e evitava ataques que permitiam que um usuário deduzisse chaves criptográficas por meio da observação do fluxo de mensagens criptografadas durante um determinado período, o que representava melhorias em relação à WEP. O WPA1 logo foi substituído pelo WPA2, que exigia o uso de criptografia de chaves simétricas AES.

No cerne do WPA temos um protocolo de apresentação de quatro vias que executa a autenticação mútua e a derivação de chaves de sessão simétricas compartilhadas. A Figura 8.31 mostra o protocolo de apresentação em forma simplificada. Observe que o dispositivo móvel (M) e o AS começam conhecendo uma chave secreta compartilhada K_{AS-M} (p. ex., uma senha). Uma das suas tarefas é derivar uma chave de sessão simétrica compartilhada K_{M-AP} que será utilizada para criptografar/decriptar quadros transmitidos posteriormente entre o dispositivo móvel (M) e o AP.

A autenticação mútua e a derivação de chaves de sessão simétricas compartilhadas são realizadas nos dois primeiros passos, a e b, da apresentação em quatro vias mostrada na Figura 8.31. Os passos c e d são usados para derivar uma segunda chave, usada para a comunicação em grupo; para mais detalhes, consulte (Kohlios, 2018; Zou, 2016).

- Neste primeiro passo, o AS gera um nonce, o $Nonce_{AS}$, e envia-o para o dispositivo móvel. Voltando à Seção 8.4, lembre-se que os nonces são usados para evitar ataques de reprodução e provar que o outro lado sendo autenticado está “ao vivo”.
- O dispositivo móvel, M, recebe o nonce, $Nonce_{AS}$, do AS e gera o seu próprio nonce, $Nonce_M$. O dispositivo móvel então gera a chave de sessão compartilhada simétrica,

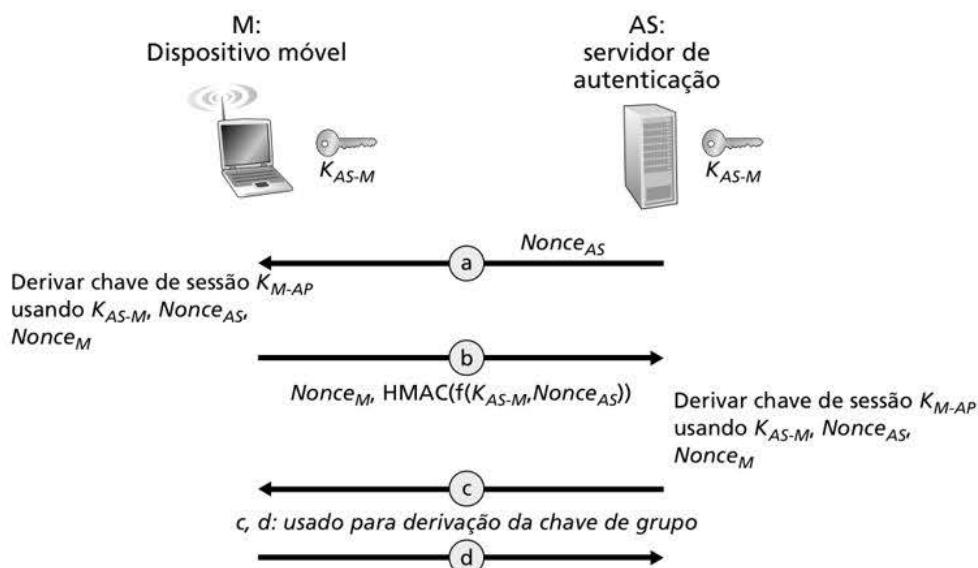


Figura 8.31 A apresentação de quatro vias do WPA2.

K_{M-AP} , usando o $Nonce_{AS}$, o $Nonce_M$, a chave secreta compartilhada inicial K_{AS-M} , seu endereço MAC e o endereço MAC do AS. A seguir, ele envia o seu nonce, o $Nonce_M$, e um valor com assinatura HMAC (ver Figura 8.9) que codifica o $Nonce_{AS}$ e o segredo compartilhado original.

O AS recebe esta mensagem de M. Analisando a versão do nonce com a assinatura HMAC que acaba de enviar, o $Nonce_{AS}$, o servidor de autenticação sabe que o dispositivo móvel está “vivo”, pois este foi capaz de criptografar usando a chave secreta compartilhada K_{AS-M} , e o AS também sabe que o dispositivo móvel é mesmo quem afirma ser (i.e., um dispositivo que conhece o segredo inicial compartilhado). Assim, o AS autenticou o dispositivo móvel! O AS também pode realizar exatamente o mesmo cálculo que o dispositivo móvel para derivar a chave de sessão simétrica compartilhada, K_{M-AP} , usando o $Nonce_M$ que recebeu, o $Nonce_{AS}$, a chave secreta compartilhada inicial K_{AS-M} , seu endereço MAC e o endereço MAC do dispositivo móvel. Nesse ponto, o dispositivo móvel e o servidor de autenticação ambos calcularam a mesma chave simétrica compartilhada K_{M-AP} , que será usada para criptografar/decriptar quadros transmitidos entre o dispositivo móvel e o AP. O AS informa o AP do valor dessa chave na Etapa 3 da Figura 8.30.

O WPA3 foi lançado em junho de 2018 para ser uma atualização do WPA2. A atualização trabalha um ataque do protocolo de apresentação de quatro vias que poderia induzir a reutilização de nonces usados anteriormente (Vanhoef, 2017), mas ainda permite o uso da apresentação de quatro vias como protocolo legado e inclui chaves mais longas, entre outras mudanças (WiFi, 2019).

Protocolos de mensagens de segurança 802.11

A Figura 8.32 mostra os protocolos usados para implementar o framework de segurança 802.11 discutido acima. O Protocolo de Autenticação Extensível (EAP, do inglês *Extensible Authentication Protocol*) (RFC 3748) define o formato da mensagem fim a fim usado em um modo simples de requisição/resposta de interação entre o dispositivo móvel e o servidor de autenticação, certificados sob o WPA2. Como mostrado na Figura 8.32, as mensagens EAP são encapsuladas usando um EAPoL (do inglês *EAP over LAN* – EAP através da LAN) e enviadas através de um enlace 802.11 sem fio. Então, estas mensagens EAP são desencapsuladas no ponto de acesso, e reencapsuladas usando um protocolo RADIUS para a transmissão por UDP/IP ao servidor de autenticação. Embora o protocolo e o servidor RADIUS (RFC 2865) não sejam obrigatórios, eles são componentes-padrão na prática. O protocolo DIAMETER (RFC 3588), padronizado recentemente, foi projetado para substituir o RADIUS no futuro.

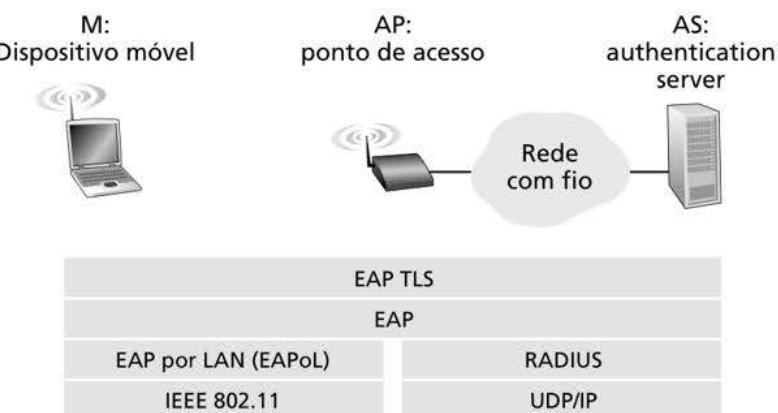


Figura 8.32 O EAP é um protocolo fim a fim. As mensagens EAP são encapsuladas usando um EAPoL através de um enlace sem fio entre o dispositivo móvel e o ponto de acesso, e usando RADIUS através de UDP/IP entre o ponto de acesso e o servidor de autenticação.

8.8.2 Autenticação e acordo de chaves nas redes celulares 4G/5G

Nesta seção, descreveremos os mecanismos de autenticação mútua e geração de chaves nas redes 4G/5G. Muitas das abordagens que encontraremos aqui têm paralelos com aquelas que acabamos de estudar para as redes 802.11, com a exceção importante de que, nas redes 4G/5G, os dispositivos móveis podem estar ligados à sua rede nativa (i.e., as redes das operadoras das quais são assinantes) ou podem estar em modo de roaming em uma rede visitada. No segundo caso, as redes visitadas e nativa precisam interagir ao autenticar um dispositivo móvel e gerar chaves criptográficas. Antes de continuarmos, talvez seja melhor que você releia as Seções 7.4 e 7.7.1 para se refamiliarizar com a arquitetura das redes 4G/5G.

Os objetivos de autenticação mútua e geração de chaves são os mesmos nos ambientes 4G/5G e 802.11. Para decriptar o conteúdo dos quadros transmitidos pelo canal sem fio, o dispositivo móvel e a estação-base precisam derivar uma chave criptográfica simétrica compartilhada. Além disso, a rede à qual o dispositivo móvel está se ligando precisa autenticar a identidade do dispositivo e verificar os seus privilégios de acesso. Da mesma forma, o dispositivo móvel também autentica a rede ao qual se liga. A necessidade da rede de autenticar um dispositivo móvel pode ser óbvia, mas a necessidade de autenticação no sentido inverso é menos clara. Contudo, existem casos documentados de criminosos que operavam estações-base celulares não autorizadas para atrair dispositivos móveis inocentes, ligá-los à rede criminosa e expô-los a diversas formas de ataque (Li, 2017). Então, assim como no caso das WLANs 802.11, o dispositivo móvel deve ser extremamente cauteloso quando se liga a uma rede celular.

A Figura 8.33 ilustra o cenário de um dispositivo móvel que se liga a uma rede celular 4G. No alto da Figura 8.33, vemos muitos dos componentes 4G que encontramos anteriormente na Seção 7.4: o dispositivo móvel (M), a estação-base (BS, do inglês *base station*), a entidade de gerenciamento móvel (MME, do inglês *Mobility Management Entity*) na rede à qual o dispositivo móvel deseja se ligar e o serviço de assinante doméstico (HSS, do inglês *Home Subscriber Server*) na rede nativa do dispositivo móvel. Uma comparação entre as Figuras 8.30 e 8.33 mostra as semelhanças e diferenças entre os contextos de segurança 802.11 e 4G. Mais uma vez, vemos um dispositivo móvel e uma estação-base; a chave de sessão do usuário derivada durante a ligação à rede, K_{BS-M} , será usada para criptografar e decriptar os quadros transmitidos pelo enlace sem fio. A MME e o HSS 4G juntos têm um

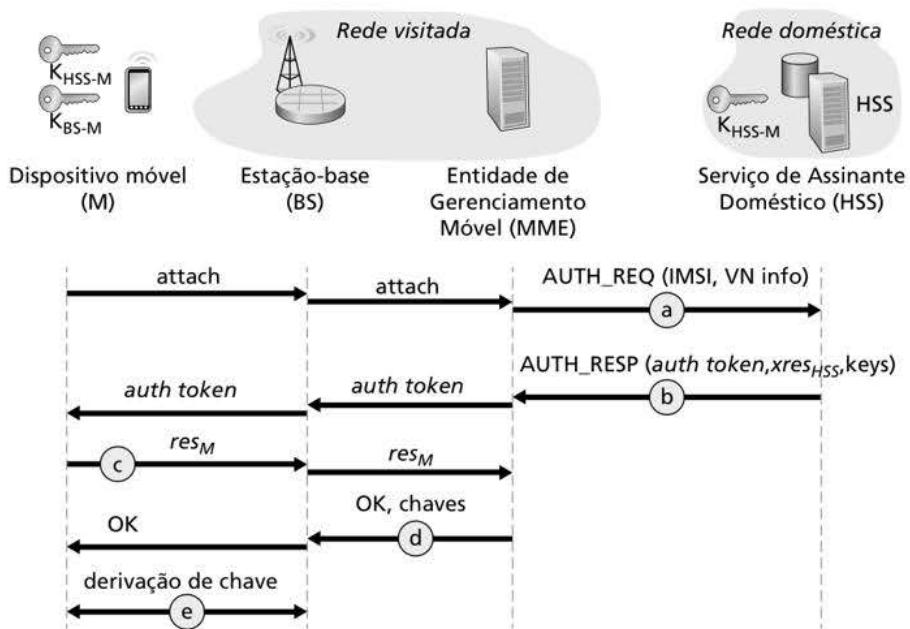


Figura 8.33 Autenticação mútua e acordo de chaves em uma rede celular 4G LTE.

papel semelhante ao do servidor de autenticação na rede 802.11. Observe que o HSS e o dispositivo móvel terão um segredo compartilhado em comum, K_{HSS-M} , conhecido por ambas as entidades antes que a autenticação inicie. Essa chave fica armazenada no cartão SIM (do inglês *Subscriber Identity Module* – módulo de identificação do assinante) do dispositivo móvel e no banco de dados do HSS na rede nativa do dispositivo móvel.

O protocolo de Autenticação e Acordo de Chaves (AKA, do inglês *Authentication and Key Agreement*) é composto pelos seguintes passos:

- a. *Pedido de autenticação para HSS.* Quando solicita pela primeira vez, através de uma estação-base, para se ligar à rede, o dispositivo móvel envia uma mensagem de ligação que contém a sua identidade internacional de assinante móvel (IMSI, do inglês *International Mobile Subscriber Identity*), que é repassada para a MME. A MME então envia a IMSI e as informações sobre a rede visitada (mostrada como “VN info” na Figura 8.33) para o HSS na rede nativa do dispositivo. Na Seção 7.4, descrevemos como a MME consegue se comunicar com o HSS através de uma rede global toda em IP de redes celulares interconectadas.
- b. *Resposta de autenticação do HSS.* O HSS realiza operações criptográficas usando a chave secreta compartilhada de antemão, K_{HSS-M} , para derivar um token de autenticação, $auth_token$, e um token de resposta de autenticação esperado, $xres_{HSS}$. O $auth_token$ contém informações criptografadas pelo HSS usando a K_{HSS-M} que permitirão que o dispositivo móvel saiba que quem calculou o $auth_token$ conhece a chave secreta. Por exemplo, suponha que o HSS calcula $K_{HSS-M}(\text{IMSI})$, ou seja, criptografa a IMSI do dispositivo móvel usando K_{HSS-M} e envia o valor como $auth_token$. Quando o dispositivo móvel recebe esse valor criptografado e usa a sua chave secreta para descriptografiá-lo, ou seja, para calcular $K_{HSS-M}(K_{HSS-M}(\text{IMSI})) = \text{IMSI}$, ele sabe que o HSS que gerou $auth_token$ conhece a sua chave secreta. Assim, o dispositivo móvel consegue autenticar o HSS.

O token de resposta de autenticação esperado, $xres_{HSS}$, contém um valor que o dispositivo móvel precisará conseguir calcular (usando K_{HSS-M}) e devolver à MME para provar que *ele* (o dispositivo móvel) conhece a chave secreta, autenticando, assim, o dispositivo móvel para a MME.

- Observe que a MME tem aqui apenas um papel intermediário, recebendo a mensagem de resposta de autenticação, guardando $xres_{HSS}$ para uso posterior, extraíndo o token de autenticação e repassando-o para o dispositivo móvel. Em especial, ele não precisa saber, e não descobre, a chave secreta, K_{HSS-M} .
- c. *Resposta de autenticação do dispositivo móvel.* O dispositivo móvel recebe o $auth_token$ e calcula $K_{HSS-M}(K_{HSS-M}(\text{IMSI})) = \text{IMSI}$, autenticando, assim, o HSS. A seguir, o dispositivo móvel calcula um valor res_M (usando a sua chave secreta para realizar exatamente o mesmo cálculo criptográfico que o HSS usara para calcular $xres_{HSS}$) e envia o valor para a MME.
 - d. *Autenticação do dispositivo móvel.* O MMS compara o valor de res_M calculado pelo dispositivo móvel com o valor de $xres_{HSS}$ calculado pelo HSS. Se houver correspondência, o dispositivo móvel é autenticado, pois provou para a MME que ele e o HSS conhecem a chave secreta em comum. O MMS informa à estação-base e ao dispositivo móvel que a autenticação mútua está completa e envia as chaves da estação-base que serão utilizadas no passo e.
 - e. *Derivação de chaves do plano de dados e do plano de controle.* O dispositivo móvel e a estação-base determinam as chaves usadas para criptografar/decriptar suas transmissões de quadros através do canal sem fio. Chaves independentes serão derivadas para as transmissões de quadros do plano de dados e do plano de controle. O algoritmo criptográfico AES que vimos em uso nas redes 802.11 também é usado nas redes 4G/5G.

A discussão acima se concentrou na autenticação e no acordo de chaves nas redes 4G. Apesar de boa parte da segurança das redes 4G ser reutilizada nas 5G, ocorrem algumas alterações importantes:

- Primeiro, observe na discussão acima que é a MME na rede visitada que toma as decisões sobre autenticação. Uma mudança significativa que está sendo adotada na segurança das redes 5G é permitir que serviços de autenticação sejam prestados pela rede nativa, com a rede visitada tendo um papel intermediário ainda menor. A rede visitada ainda pode rejeitar a autenticação de um dispositivo móvel, mas é a rede nativa que aceita o pedido de autenticação nesse novo cenário 5G.
- As redes 5G suportarão o protocolo de AKA descrito acima, assim como dois novos protocolos adicionais de autenticação e acordo de chaves. Um deles, chamado AKA', é bastante próximo do protocolo AKA 4G. Ele também usa a chave secreta compartilhada de antemão, K_{HSS-M} , mas como usa o protocolo EAP que encontramos anteriormente na Figura 8.33 no contexto da autenticação nas redes 802.11, o AKA' das redes 5G têm fluxos de mensagem diferentes do AKA do 4G. O segundo novo protocolo do 5G foi desenvolvido para um ambiente Internet das Coisas (IoT, do inglês *Internet of Things*) e não precisa de uma chave secreta compartilhada de antemão.
- Uma mudança adicional no 5G é o uso de técnicas de criptografia de chaves públicas para criptografar a identidade permanente de um dispositivo (i.e., sua IMSI) de modo que nunca seja transmitida em texto aberto.

Nesta seção, apresentamos apenas um breve resumo da autenticação mútua e do acordo de chaves nas redes 4G/5G. Como vimos, elas aplicam as técnicas de segurança estudadas nas seções anteriores deste capítulo. Para mais detalhes sobre a segurança 4G/5G, consulte (3GPP SAE, 2019; Cable Labs, 2019; Cichonski, 2017).

8.9 SEGURANÇA OPERACIONAL: FIREWALLS E SISTEMAS DE DETECÇÃO DE INVASÃO

Vimos, em todo este capítulo, que a Internet não é um lugar muito seguro – os delinquentes estão por toda parte, criando todo tipo de destruição. Sabendo da natureza hostil da Internet, vamos considerar a rede de uma organização e um administrador de rede que a administra. Do ponto de vista de um administrador, o mundo está dividido claramente em dois campos – os mocinhos (que pertencem à organização que administra a rede e que deveriam poder acessar recursos dentro da rede que eles administram de um modo relativamente livre de restrições) e os bandidos (todo o resto, cujo acesso aos recursos da rede deve ser cuidadosamente inspecionado). Em muitas organizações, que vão de castelos medievais a modernos escritórios de empresas, há um único ponto de entrada/saída onde ambos, mocinhos e bandidos que entram e saem, passam por inspeção de segurança. Em castelos medievais, essa inspeção era feita em um portão, na extremidade de uma ponte levadiça; em escritórios empresariais, ela é feita na central de segurança. Em redes de computadores, quando o tráfego que entra/sai de uma rede passa por inspeção de segurança, é registrado, descartado ou transmitido; isso é feito por mecanismos operacionais conhecidos como firewalls, sistemas de detecção de invasão (IDSs, do inglês *intrusion detection systems*) e sistemas de prevenção de invasão (IPSs, do inglês *intrusion prevention systems*).

8.9.1 Firewalls

Um **firewall** é uma combinação de hardware e software que isola a rede interna de uma organização da Internet em geral, permitindo que alguns pacotes passem e bloqueando outros. O firewall permite a um administrador de rede controlar o acesso entre o mundo externo e os recursos da rede que ele administra, gerenciando o fluxo de tráfego de e para esses recursos. Um firewall possui três objetivos:

- Todo o tráfego de fora para dentro, e vice-versa, passa por um firewall. A Figura 8.34 mostra um firewall, situado diretamente no limite entre a rede administrada e o resto da Internet. Embora grandes organizações possam usar diversos níveis de firewalls ou firewalls distribuídos (Skoudis, 2006), alocar um firewall em um único ponto de acesso à rede, conforme mostrado na Figura 8.34, facilita o gerenciamento e a execução de uma política de acesso seguro.
- Somente o tráfego autorizado, como definido pela política de segurança local, poderá passar. Com todo o tráfego que entra e sai da rede institucional passando pelo firewall, este pode limitar o acesso ao tráfego autorizado.
- O próprio firewall é imune à penetração. O próprio firewall é um mecanismo conectado à rede. Se não projetado ou instalado de modo adequado, pode ser comprometedor, oferecendo apenas uma falsa sensação de segurança (pior do que não ter nenhum firewall!).

Cisco e Check Point são dois dos principais fornecedores atuais de firewall. Você pode criar um firewall (filtro de pacotes) facilmente a partir de um sistema Linux usando iptables (software de domínio público que em geral acompanha o Linux). Além disso, como discutido nos Capítulos 4 e 5, os firewalls hoje frequentemente são implementados nos roteadores e controlados remotamente usando redes definidas por software (SDN, do inglês *software-defined networking*).

Os firewalls podem ser classificados em três categorias: **filtros de pacotes tradicionais**, **filtros de estado** e **gateways de aplicação**. Abordaremos cada um nas subseções seguintes.

Filtros de pacotes tradicionais

Como ilustra a Figura 8.34, uma organização normalmente tem um roteador de borda que conecta sua rede interna com seu Provedor de Serviços de Internet (ISP, do inglês *Internet Service Provider*) (e dali com a Internet pública, mais ampla). Todo o tráfego que sai ou que entra na rede interna passa por esse roteador, e é nele que ocorre a **filtragem de pacotes**. Um filtro de pacotes examina cada datagrama que está sozinho, determinando se deve passar ou ficar baseado nas regras específicas definidas pelo administrador. As decisões de filtragem costumam ser baseadas em:

- Endereço IP de origem e de destino
- Tipo de protocolo no campo do datagrama IP: TCP, UDP, ICMP, OSPF, etc.
- Porta TCP ou UDP de origem e de destino

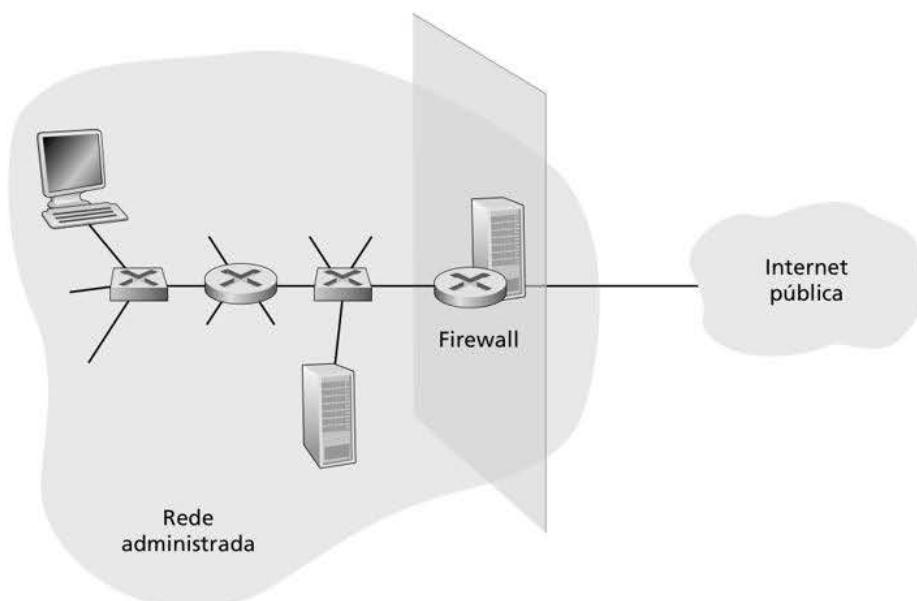


Figura 8.34 Posição do firewall entre a rede administrada e o mundo exterior.

- Bits de flag do TCP: SYN, ACK etc.
- Tipo de mensagem ICMP
- Regras diferentes para datagramas que entram e saem da rede
- Regras diferentes para diferentes interfaces do roteador

Um administrador de rede configura o firewall com base na política da organização. A política pode considerar a produtividade do usuário e o uso de largura de banda, bem como as preocupações com segurança da organização. A Tabela 8.5 lista diversas políticas que uma organização pode ter, e como elas seriam endereçadas com um filtro de pacotes. Por exemplo, se a organização não quer nenhuma conexão TCP de entrada, exceto aquelas para o servidor Web público, ela pode bloquear todos os segmentos TCP SYN de entrada, com exceção dos segmentos TCP SYN com porta de destino 80 e endereço IP de destino correspondente ao servidor Web. Se ela não quer que seus usuários monopolizem a largura de banda de acesso com aplicações de rádio via Internet, pode bloquear todo o tráfego UDP não importante (já que o rádio via Internet é em geral enviado por UDP). Se não quer que sua rede interna seja mapeada (por traceroute) por um estranho, pode bloquear todas as mensagens ICMP TTL expiradas que saem da rede da organização.

Uma política de filtragem também pode ser baseada na combinação de endereços e números de porta. Por exemplo, o roteador de filtragem poderia bloquear todos os datagramas Telnet (os que têm número de porta 23), exceto os que estão vindo ou indo de ou para uma lista de endereços IP específicos. Essa política permite conexões Telnet de e para os hóspedes que estão na lista. Infelizmente, basear a política em endereços externos não oferece nenhuma proteção contra datagramas cujo endereço de origem foi falsificado.

A filtragem pode também ser baseada em o bit TCP ACK estar ou não marcado. Esse truque é bastante útil quando uma organização quer permitir que seus clientes internos se conectem com servidores externos, mas impedir que clientes externos se conectem com servidores internos. Lembre-se de que o primeiro segmento de todas as conexões TCP (veja a Seção 3.5) tem o bit ACK com valor 0, ao passo que todos os outros segmentos da conexão têm o bit ACK com valor 1. Assim, se uma organização quiser impedir que clientes externos iniciem uma conexão com servidores internos, ela apenas filtrará todos os segmentos que entram que tenham o bit ACK definido como 0. Essa política elimina todas as conexões TCP originadas do exterior, mas permite conexões que se originam internamente.

As regras do firewall são executadas em roteadores com listas de controle de acesso, tendo cada interface do roteador sua própria lista. Um exemplo de uma lista de controle

TABELA 8.5 Políticas e regras de filtragem correspondentes para uma rede da organização 130.207.16 com servidor Web 130.207.244.203

Política	Configuração de firewall
Não há acesso exterior à Web	Descartar todos os pacotes de saída para qualquer endereço IP, porta 80
Não há conexões TCP de entrada, exceto aquelas apenas para o servidor Web público da organização	Descartar todos os pacotes TCP SYN para qualquer IP, exceto 130.207.244.203, porta 80
Impedir que rádios Web devorem a largura de banda disponível	Descartar todos os pacotes UDP de entrada – exceto pacotes DNS
Impedir que sua rede seja usada por um ataque DoS smurf	Descartar todos os pacotes ping que estão indo para um endereço de difusão (p. ex., 130.207.255.255)
Impedir que a rota de sua rede seja rastreada	Descartar todo o tráfego de saída ICMP com TTL expirado

TABELA 8.6 Lista de controle de acesso para uma interface do roteador

Ação	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Bit de flag
Permitir	222.22/16	Fora de 222.22/16	TCP	> 1023	80	Qualquer um
Permitir	Fora de 222.22/16	222.22/16	TCP	80	> 1023	ACK
Permitir	222.22/16	Fora de 222.22/16	UDP	> 1023	53	–
Permitir	Fora de 222.22/16	222.22/16	UDP	53	> 1023	–
Negar	Todos	Todos	Todos	Todos	Todos	Todos

de acesso para uma organização 222.22/16 é ilustrado na Tabela 8.6. Essa lista é para uma interface que conecta o roteador aos ISPs externos da organização. As regras são aplicadas a cada datagrama que atravessa a interface de cima para baixo. As primeiras duas regras juntas permitem que usuários internos naveguem na Web: a primeira permite que qualquer pacote TCP com porta de destino 80 saia da rede da organização; a segunda autoriza que qualquer pacote TCP com porta de origem 80 e o bit ACK marcado entrem na rede. Observe que, se uma fonte externa tentar estabelecer uma conexão TCP com um hospedeiro interno, a conexão será bloqueada, mesmo que a porta de origem ou de destino seja 80. As duas regras juntas permitem que pacotes DNS entrem e saiam da rede da organização. Em resumo, essa lista de controle de acesso limitada bloqueia todo o tráfego, exceto o tráfego Web iniciado de dentro da organização e do tráfego DNS. (CERT Filtering, 2012) apresenta uma lista de portas/protocolos para a filtragem de pacotes para evitar diversas brechas de segurança conhecidas nas aplicações de rede existentes.

Os leitores com boa memória lembrarão que encontramos listas de controle de acesso semelhantes à Tabela 8.6 quando estudamos o repasse generalizado na Seção 4.4.3 do Capítulo 4. Na verdade, nela, oferecemos um exemplo de como as regras de repasse generalizado podem ser utilizadas para construir um firewall de filtragem de pacotes.

Filtros de pacote com controle de estado

Em um filtro de pacotes tradicional, as decisões de filtragem são feitas em cada pacote isolado. Os filtros de estado rastreiam conexões TCP e usam esse conhecimento para tomar decisões sobre filtragem.

Para entender esses filtros de estado, vamos reexaminar a lista de controle de acesso da Tabela 8.6. Embora um tanto restritiva, essa lista permite que qualquer pacote que chegue de fora com um ACK = 1 e porta de origem 80 atravesse o filtro. Esses pacotes poderiam ser usados em tentativas de destruir o sistema interno com pacotes defeituosos, realizar ataques de recusa de serviço, ou mapear a rede interna. A solução ingênuia é bloquear pacotes TCP ACK também, mas tal método impediria que os usuários internos da organização navegassem na Web.

Os filtros de estado resolvem esse problema rastreando todas as conexões TCP de entrada em uma tabela de conexão. Isso é possível porque o firewall pode notar o início de uma nova conexão observando uma apresentação de três vias (SYN, SYNACK e ACK); ele pode observar o fim de uma conexão ao ver um pacote FIN para a conexão. O firewall também consegue (de forma conservadora) admitir que a conexão está finalizada quando não observou nenhuma atividade no decorrer da conexão, digamos, por 60 segundos. Um exemplo de tabela de conexão para um firewall é mostrado na Tabela 8.7. Essa tabela indica que, no momento, há três conexões TCP em andamento, as quais foram iniciadas dentro da organização. Ademais, o filtro de estado inclui uma nova coluna, “verificar conexão”, em sua lista de controle de acesso, como mostrado na Tabela 8.8. Observe que essa tabela é idêntica à lista de controle de acesso da Tabela 8.6, exceto por ela indicar que a conexão deve ser verificada para duas das regras.

TABELA 8.7 Tabela de conexão para o filtro de estado

Endereço de origem	Endereço de destino	Porta de origem	Porta de destino
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

TABELA 8.8 Lista de controle de acesso para filtro de estado

Ação	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Bit de flag	Verificar conexão
Permitir	222.22/16	Fora de 222.22/16	TCP	> 1023	80	Qualquer um	
Permitir	Fora de 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
Permitir	222.22/16	Fora de 222.22/16	UDP	> 1023	53	-	
Permitir	Fora de 222.22/16	222.22/16	UDP	53	> 1023	-	X
Negar	Todos	Todos	Todos	Todos	Todos	Todos	

Vamos analisar alguns exemplos e ver como a tabela de conexão e a lista de controle de acesso funcionam em conjunto. Suponha que um atacante tente enviar um pacote defeituoso para a rede da organização por meio de um datagrama com porta de origem TCP 80 e com o flag ACK marcado. Suponha ainda que ele possua um número de porta de origem 12543 e endereço IP remetente 150.23.23.155. Quando o pacote chega ao firewall, este verifica a lista de controle de acesso da Tabela 8.7, que indica que a tabela de conexão deve também ser verificada antes de permitir que esse pacote entre na rede da organização. O firewall verifica devidamente a tabela de conexão e observa que esse pacote não faz parte de uma conexão TCP em andamento, e o rejeita. Como segundo exemplo, imagine que um usuário interno queira navegar em um site externo. Como o usuário primeiro envia um segmento TCP SYN, sua conexão TCP é registrada na tabela de conexão. Quando o servidor envia pacotes de volta (com o bit ACK necessariamente definido), o firewall verifica a tabela e observa que uma conexão correspondente está em andamento. O firewall, então, deixará esses pacotes passarem, sem interferir na navegação do usuário interno.

Gateway de aplicação

Nos exemplos que acabamos de mostrar, vimos que a filtragem de pacotes permite que uma organização faça uma filtragem grosseira de conteúdos de cabeçalhos IP e TCP/UDP, incluindo endereços IP, números de porta e bits de reconhecimento. Mas e se a organização quiser fornecer o serviço Telnet a um conjunto restrito de usuários internos (em vez de a endereços IP)? E se quiser que esses usuários privilegiados se autentiquem antes de obter permissão para criar sessões Telnet com o mundo externo? Essas tarefas estão além das capacidades de um filtro. Na verdade, informações sobre a identidade de usuários internos não estão incluídas nos cabeçalhos IP/TCP/UDP; elas estão nos dados da camada de aplicação.

Para assegurar um nível mais refinado de segurança, os firewalls têm de combinar filtro de pacotes com gateways de aplicação. Gateways de aplicação fazem mais do que examinar cabeçalhos IP/TCP/UDP e tomam decisões com base em dados da aplicação. Um **gateway de aplicação** é um servidor específico de aplicação, através do qual todos os dados da aplicação (que entram e que saem) devem passar. Vários gateways de aplicação podem executar no mesmo hospedeiro, mas cada gateway é um servidor separado, com seus próprios processos.

Para termos uma ideia melhor desses gateways, vamos projetar um firewall que permite a apenas um conjunto restrito de usuários executar Telnet para o exterior e impede que todos os clientes externos executem Telnet para o interior. Essa política pode ser aplicada pela execução da combinação de um filtro de pacotes (em um roteador) com um gateway de aplicação de Telnet, como mostra a Figura 8.35. O filtro do roteador está configurado para bloquear todas as conexões Telnet, exceto as que se originam do endereço IP do gateway de aplicação. Essa configuração de filtro força todas as conexões Telnet de saída a passarem pelo gateway de aplicação. Considere agora um usuário interno que quer executar Telnet com o mundo exterior. Primeiro, ele tem de estabelecer uma sessão Telnet com o gateway de aplicação. Uma aplicação que está executando no gateway – e que fica à escuta de sessões Telnet que entram – solicita ao usuário sua identificação e senha. Quando o usuário fornece essas informações, o gateway de aplicação verifica se ele tem permissão para executar Telnet com o mundo exterior. Se não tiver, a conexão Telnet do usuário interno ao gateway será encerrada pelo gateway. Se o usuário tiver permissão, o gateway (1) pedirá ao usuário o nome do computador externo com o qual ele quer se conectar, (2) estabelecerá uma sessão Telnet entre o gateway e o hospedeiro externo, e (3) repassará ao hospedeiro externo todos os dados que chegam do usuário, e, ao usuário, todos os dados que chegam do hospedeiro externo. Assim, o gateway de aplicação Telnet não só autoriza o usuário, mas também atua como um servidor Telnet e um cliente Telnet, repassando informações entre o usuário e o servidor Telnet remoto. Note que o filtro permitirá a etapa 2, porque é o gateway que inicia a conexão Telnet com o mundo exterior.

Redes internas frequentemente têm vários gateways de aplicação, como gateways para Telnet, HTTP, FTP e e-mail. De fato, o servidor de correio (veja a Seção 2.3) e o cache Web de uma organização são gateways de aplicação.

Gateways de aplicação não estão isentos de desvantagens. Primeiro, é preciso um gateway de aplicação diferente para cada aplicação. Segundo, há um preço a pagar em termos de desempenho, visto que todos os dados serão repassados por meio do gateway. Isso se torna uma preocupação em particular quando vários usuários ou aplicações estão utilizando o mesmo gateway. Por fim, o software cliente deve saber como entrar em contato com o gateway quando o usuário fizer uma solicitação, e deve saber como dizer ao gateway de aplicação a qual servidor se conectar.

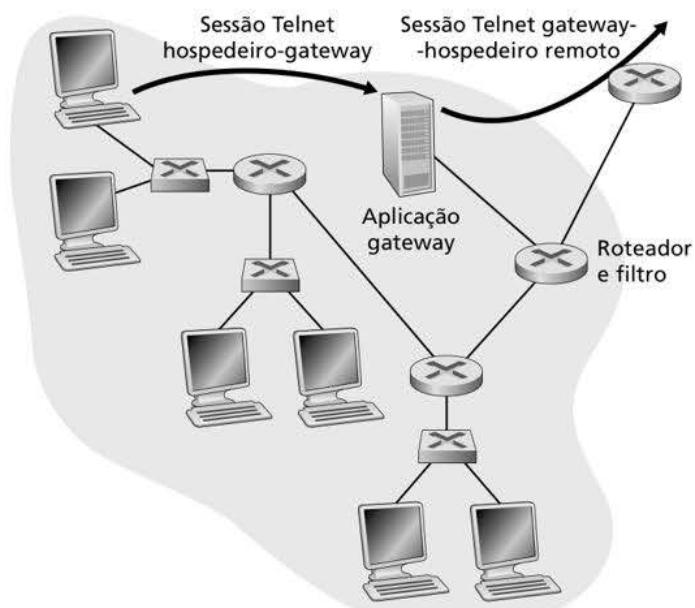


Figura 8.35 Firewall composto por um gateway de aplicação e um filtro.

HISTÓRICO DO CASO

ANONIMATO E PRIVACIDADE

Suponha que você queira visitar aquela polêmica página Web (p. ex., o site de um ativista político) e você (1) não quer revelar seu endereço IP à página Web, (2) não quer que o seu ISP (que pode ser o de sua casa ou do escritório) saiba que você está visitando esse site, e (3) não quer que o seu IP local veja os dados que você está compartilhando com o site. Se você usar o método tradicional de conexão direta à página Web sem nenhuma criptografia, então falhará em seus três objetivos. Mesmo que use SSL, você falhará nas duas primeiras questões: seu endereço IP de origem é apresentado à página Web em todo datagrama enviado; e o endereço de destino de cada pacote enviado pode facilmente ser analisado pelo seu ISP local.

Para obter anonimato e privacidade, você pode usar uma combinação de um servidor proxy confiável e SSL, como mostrado na Figura 8.36. Com essa técnica, você faz uma conexão SSL com o proxy confiável. Depois envia, nessa conexão SSL, uma solicitação HTTP para o site desejado. Quando o proxy receber essa solicitação HTTP criptografada por SSL, ele a decodificará e encaminhará o texto aberto da solicitação HTTP à página Web. Esta responde ao proxy, que encaminha a resposta a você pelo SSL. Como a página Web só vê o endereço IP do proxy, e não o do seu cliente, você está de fato obtendo um acesso anônimo à página Web. E devido ao tráfego entre você e o proxy ser criptografado, seu ISP local não pode invadir sua privacidade ao logar no site que você visitou ou gravar os dados que estavam sendo compartilhados. Hoje, muitas empresas disponibilizam tais serviços proxy (como a proxify.com).

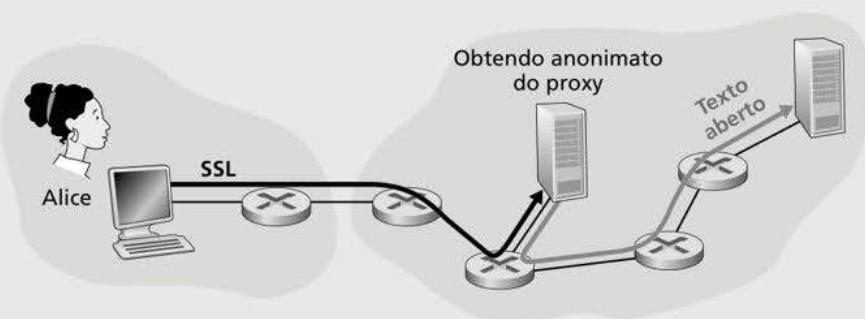


Figura 8.36 Fornecendo anonimato e privacidade com um proxy.

É claro que, ao usar esta solução, seu proxy saberá tudo: seu endereço IP e o endereço IP do site que você está visitando; pode ver todo o tráfego em texto aberto compartilhado entre você e a página. Uma técnica melhor, adotada pelo serviço de anonimato e privacidade TOR, é sequenciar seu tráfego através de uma série de servidores proxys que não compartilham informações entre si (TOR, 2020). Em particular, o TOR permite que indivíduos independentes contribuam com proxys para seu acervo. Quando um usuário se conecta a um

servidor usando o TOR, ele escolhe aleatoriamente (de seu acervo de proxys) uma corrente de três proxys e sequencia todo o tráfego entre cliente e servidor por essa corrente. Dessa maneira, supondo que os proxys não trocam informações entre si, ninguém percebe que ocorreu uma comunicação entre seu endereço IP e a página da Web desejada. Além disso, apesar de o texto aberto ser enviado entre o último proxy e o servidor, o último proxy não sabe qual endereço IP está enviando ou recebendo o texto aberto.

8.9.2 Sistemas de detecção de invasão

Acabamos de ver que um filtro de pacotes (tradicional e de estado) inspeciona campos de cabeçalho IP, TCP, UDP e ICMP quando está decidindo quais pacotes deixará passar através do firewall. No entanto, para detectar muitos tipos de ataque, precisamos executar uma **inspeção profunda de pacote**, ou seja, precisamos olhar através dos campos de cabeçalho e

dentro dos dados da aplicação que o pacote carrega. Como vimos na Seção 8.9.1, gateways de aplicação frequentemente fazem inspeções profundas de pacote. Mas um gateway de aplicação só executa isso para uma aplicação específica.

Decerto existe espaço para mais um dispositivo – um dispositivo que não só examina os cabeçalhos de todos os pacotes ao passar por eles (como um filtro de pacotes), mas também executa uma inspeção profunda de pacote (diferente do filtro de pacotes). Quando tal dispositivo observa o pacote suspeito, ou uma série de pacotes suspeitos, ele impede que tais pacotes entrem na rede organizacional. Ou, quando a atividade só é vista como suspeita, o dispositivo pode deixar os pacotes passarem, mas envia um alerta ao administrador de rede, que pode examinar o tráfego minuciosamente e tomar as ações necessárias. Um dispositivo que gera alertas quando observa tráfegos potencialmente mal-intencionados é chamado de **IDS**. Um dispositivo que filtra o tráfego suspeito é chamado de **IPS**. Nesta seção, estudaremos ambos os sistemas – IDS e IPS –, já que o mais interessante aspecto técnico desses sistemas é como eles detectam tráfego suspeito (em vez de enviarem alertas ou abandonarem pacotes). Daqui para a frente, vamos nos referir ao sistema IDS e ao sistema IPS coletivamente como sistema IDS.

Um IDS pode ser usado para detectar uma série de tipos de ataques, incluindo mapeamento de rede (p. ex., provindo de nmap), varreduras de porta, varreduras de pilha TCP, ataques de inundação de largura de banda de DoS, worms e vírus, ataques de vulnerabilidade de OS e ataques de vulnerabilidade de aplicações. (Veja, na Seção 1.6, um tutorial sobre ataques de rede.) Hoje, milhares de organizações empregam sistemas de IDS. Muitos desses sistemas são patenteados, comercializados pela Cisco, Check Point e por outros fornecedores de equipamentos de segurança. Mas muitos dos sistemas de IDS implementados são de domínio público, como o extremamente popular Snort IDS (o qual discutiremos em breve).

Uma organização pode pôr em prática um ou mais sensores IDS em sua rede organizacional. A Figura 8.37 mostra uma organização que tem três sensores IDS. Quando

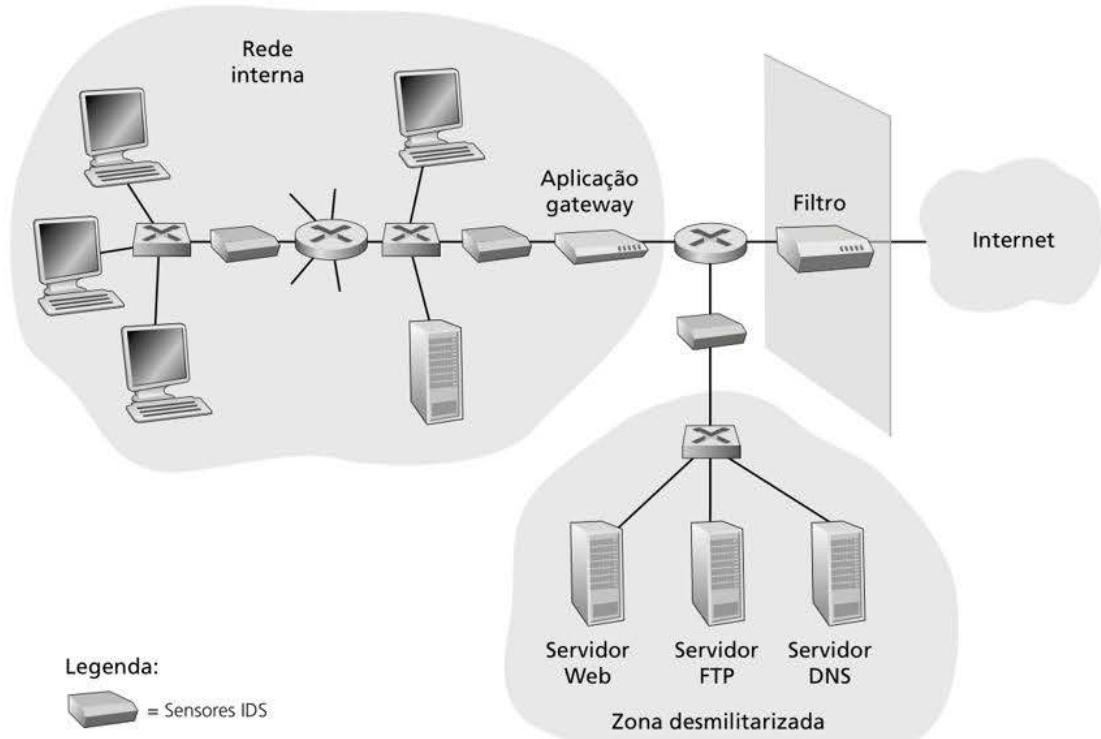


Figura 8.37 Uma organização implementando um filtro, uma aplicação gateway e sensores IDS.

múltiplos sistemas são executados, eles costumam trabalhar em harmonia, enviando informações sobre atividades de tráfegos suspeitos ao processador IDS central, que as coleta e integra e envia alarmes aos administradores da rede quando acharem apropriado. Na Figura 8.37, a organização dividiu sua rede em duas regiões: uma de segurança máxima, protegida por um filtro de pacotes e um gateway de aplicação e monitorada por sensores IDS; e uma região de segurança baixa – referida como **zona desmilitarizada** (DMZ, do inglês *demilitarized zone*) – protegida apenas por um filtro de pacotes, mas também monitorada por sensores IDS. Observe que a DMZ inclui os servidores da organização que precisam se comunicar com o mundo externo, como um servidor Web e seus servidores autoritativos.

Neste ponto, você deve estar imaginando: mas por que sensores IDS? Por que não colocar um sensor IDS logo atrás do filtro de pacotes (ou até integrá-lo ao filtro de pacotes) da Figura 8.37? Logo veremos que um IDS não só precisa fazer uma inspeção profunda do pacote, como também comparar cada pacote que passa com milhares de “assinaturas”; isso pode ser um volume de processamento significativo, em particular se a organização recebe gigabits/s de tráfego da Internet. Ao colocar sensores IDS mais à frente, cada sensor só vê uma fração do tráfego da organização, e pode facilmente acompanhar o ritmo. No entanto, hoje existem sistemas IDS e IPS de alto desempenho, e muitas organizações podem acompanhar com apenas um sensor localizado próximo ao roteador de acesso.

Sistemas IDS são classificados de modo geral como **sistemas baseados em assinatura** ou **sistemas baseados em anomalia**. Um IDS baseado em assinatura mantém um banco de dados extenso de ataques de assinaturas. Cada assinatura é um conjunto de regras relacionadas a uma atividade de invasão. Uma assinatura pode ser uma lista de características sobre um único pacote (p. ex., números de portas de origem e destino, tipo de protocolo, e uma sequência de bits em uma carga útil de um pacote), ou pode estar relacionada a uma série de pacotes. As assinaturas são normalmente criadas por engenheiros habilidosos em segurança de rede que tenham pesquisado ataques conhecidos. O administrador de rede de uma organização pode personalizar as assinaturas ou inserir suas próprias no banco de dados.

Operacionalmente, uma IDS baseada em assinatura analisa cada pacote que passa, comparando cada um com as assinaturas no banco de dados. Se um pacote (ou uma série deles) corresponder a uma assinatura no banco de dados, o IDS gera um alerta. O alerta pode ser enviado ao administrador da rede por uma mensagem de correio eletrônico, pode ser enviado ao sistema de gerenciamento da rede, ou pode simplesmente ser registrado para futuras inspeções.

Apesar de os sistemas IDS baseados em assinaturas serem amplamente executados, eles têm uma série de limitações. Acima de tudo, eles requerem conhecimento prévio do ataque para gerar uma assinatura precisa. Ou seja, um IDS baseado em assinatura é completamente cego a novos ataques que ainda não foram registrados. Outra desvantagem é que, mesmo que uma assinatura combine, isso pode não ser o resultado de um ataque mas mesmo assim um alarme é gerado. Por fim, pelo fato de cada pacote ser comparado com uma extensa coleção de assinaturas, o IDS fica atarefado com o processamento e deixa de detectar muitos pacotes malignos.

Um IDS baseado em anomalias cria um perfil de tráfego enquanto observa o tráfego em operação normal. Ele procura então por fluxos de pacotes que são estatisticamente incomuns, por exemplo, uma porcentagem irregular de pacotes ICMP ou um crescimento exponencial de análises de porta e varreduras de ping. O mais interessante sobre sistemas de IDS baseados em anomalias é que eles não recorrem a conhecimentos prévios de outros ataques – ou seja, potencialmente, eles conseguem detectar novos ataques, que não foram documentados. Por outro lado, é um problema extremamente desafiador distinguir o tráfego normal de tráfegos estatisticamente incomuns. Até hoje, a maioria das implementações de IDS são principalmente baseadas em assinaturas, apesar de algumas terem alguns recursos baseados em anomalias.

Snort

Snort é um IDS de código aberto, de domínio público, com centenas de milhares de execuções (Snort, 2012; Koziol, 2003). Ele pode ser executado em plataformas Linux, UNIX e Windows. Ele usa uma interface libpcap de análise genérica, que também é empregada pelo Wireshark e muitas outras ferramentas de análises de pacotes. Pode facilmente lidar com 100 Mbits/s de tráfego; para instalações com velocidades de tráfego de gigabit/s, múltiplos sensores Snort serão necessários.

Para termos uma ideia melhor do Snort, vamos observar o exemplo de uma assinatura Snort:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
  (msg:"ICMP PING NMAP"; dsiz: 0; itype: 8;)
```

Esta assinatura é compatível com qualquer pacote ICMP que entre na rede da organização (\$HOME _ NET) e que venha do exterior (\$EXTERNAL _ NET), seja do tipo 8 (ping ICMP), e tenha uma carga útil vazia (dsiz=0). Já que o nmap (veja a Seção 1.6) gera pacotes ping com características específicas, essa assinatura é projetada para detectar varreduras de ping usadas pelo nmap. Quando um pacote corresponde a essa assinatura, o Snort gera um alerta que inclui a mensagem “ICMP PING NMAP”.

Talvez o mais impressionante sobre o Snort seja a vasta comunidade de usuários e especialistas em segurança que mantêm sua base de dados de assinaturas. Normalmente, em algumas horas do novo ataque, a comunidade escreve e lança uma assinatura de ataque, que então é transferida via download pelas centenas de milhares de execuções de Snort ao redor do mundo. Além disso, ao usarem a sintaxe da assinatura do Snort, os administradores de rede podem criar suas próprias assinaturas a fim de satisfazer as necessidades da organização, modificando assinaturas existentes ou criando novas.

8.10 RESUMO

Neste capítulo, examinamos os diversos mecanismos que Bob e Alice, os amantes secretos, podem usar para se comunicar com segurança. Vimos que estão interessados em confidencialidade (para que somente eles possam entender o conteúdo de uma mensagem transmitida), autenticação do ponto final (para terem a certeza de que estão falando um com o outro) e integridade de mensagem (para terem certeza de que suas mensagens não sejam alteradas em trânsito). É claro que a necessidade de comunicação segura não está limitada a amantes secretos. Na verdade, vimos nas Seções 8.5 a 8.8 que a segurança é necessária em várias camadas de uma arquitetura de rede para proteção contra bandidos que têm à mão um grande arsenal de ataques possíveis.

Na primeira parte deste capítulo, apresentamos vários princípios subjacentes à comunicação segura. Na Seção 8.2, examinamos as técnicas para criptografar e decriptar dados, incluindo criptografia de chaves simétricas e criptografia de chaves públicas. O DES e o RSA foram examinados como estudos de caso específicos dessas duas classes mais importantes de técnicas de criptografia em uso nas redes de hoje.

Na Seção 8.3, examinamos duas técnicas para fornecer a integridade da mensagem: códigos de autenticação de mensagem (MACs) e assinaturas digitais. As duas têm uma série de paralelos. Ambas usam funções hash criptografadas e ambas permitem que verifiquemos a origem, assim como a integridade da própria mensagem. Uma diferença importante é que os MACs não recorrem à criptografia, ao passo que as assinaturas digitais necessitam de uma infraestrutura de chave pública. Ambas as técnicas são muito usadas na prática, como vimos nas Seções 8.5 a 8.8. Além disso, as assinaturas digitais são usadas para criar certificados digitais, os quais são importantes para a verificação da validade de uma chave pública.

Na Seção 8.4, vimos também a autenticação do ponto final e como nonces podem ser usados para impedir ataques de repetição.

Nas Seções 8.5 a 8.8, examinamos diversos protocolos de segurança de rede que são usados extensivamente na prática. Vimos que uma criptografia de chaves simétricas está no núcleo do PGP, TLS, IPsec e segurança sem fio. Vimos que uma criptografia pública é crucial para ambos PGP e TLS. Estudamos que o PGP usa assinaturas digitais para a integridade da mensagem, ao passo que TLS e IPsec usam MACs. Também exploramos a segurança nas redes sem fio, incluindo as redes WiFi e as redes celulares 4G/5G. Agora que tem entendimento dos princípios básicos da criptografia, e tendo estudado como esses princípios são usados, você deve estar pronto para projetar seus próprios protocolos de segurança de rede!

Munidos das técnicas abordadas nas Seções 8.2 a 8.8, Bob e Alice podem se comunicar com segurança. Porém, a confiabilidade é apenas uma pequena parte do quadro da segurança na rede. Como vimos na Seção 8.9, o foco está se concentrando cada vez mais em garantir a segurança da infraestrutura da rede contra o ataque potencial dos bandidos. Assim, na última parte deste capítulo, estudamos firewalls e sistemas IDS que inspecionam pacotes que entram e saem da rede de uma organização.

Exercícios de fixação e perguntas

Questões de revisão do Capítulo 8

SEÇÃO 8.1

- R1. Quais são as diferenças entre confidencialidade de mensagem e integridade de mensagem? É possível ter confidencialidade sem integridade? É possível ter integridade sem confidencialidade? Justifique sua resposta.
- R2. Equipamentos da Internet (roteadores, comutadores, servidores DNS, servidores Web, sistemas do usuário final, etc.) frequentemente precisam se comunicar com segurança. Dê três exemplos específicos de pares de equipamentos da Internet que precisem de uma comunicação segura.

SEÇÃO 8.2

- R3. Da perspectiva de um serviço, qual é uma diferença importante entre um sistema de chave simétrica e um sistema de chave pública?
- R4. Suponha que um intruso tenha uma mensagem criptografada, bem como a versão decodificada dessa mensagem. Ele pode montar um ataque somente com texto cifrado, um ataque com texto aberto conhecido ou um ataque com texto aberto escolhido?
- R5. Considere uma cifra de 8 blocos. Quantos blocos de entrada possíveis uma cifra tem? Quantos mapeamentos possíveis existiriam? Se analisarmos cada mapeamento como uma chave, então quantas chaves essa cifra teria?
- R6. Suponha que N pessoas queiram se comunicar com cada uma das outras $N - 1$ pessoas usando criptografia de chaves simétricas. Todas as comunicações entre quaisquer duas pessoas, i e j , são visíveis para todas as outras do grupo de N , e nenhuma outra pessoa desse grupo pode decodificar suas comunicações. O sistema, como um todo, requer quantas chaves? Agora, suponha que seja usada criptografia de chaves públicas. Quantas chaves serão necessárias nesse caso?
- R7. Suponha que $n = 10.000$, $a = 10.023$ e $b = 10.004$. Use uma identidade da aritmética modular para calcular $(a \cdot b) \bmod n$ de cabeça.

- R8. Suponha que você queira criptografar a mensagem 10101111 criptografando um número decimal que corresponda a essa mensagem. Qual seria esse número decimal?

SEÇÕES 8.3–8.4

- R9. De que maneira um hash fornece um melhor controle de integridade da mensagem do que uma soma de verificação (como a soma de verificação da Internet)?
- R10. Você pode decodificar o hash de uma mensagem a fim de obter a mensagem original? Explique sua resposta.
- R11. Considere a variação de um algoritmo MAC (Figura 8.9), em que o transmissor envia $(m, H(m) + s)$, sendo $H(m) + s$ a concatenação de $H(m)$ e s . Essa variação é falha? Por que ou por que não?
- R12. O que significa afirmar que um documento é verificável e não falsificável?
- R13. De que modo um resumo de mensagem criptografado por chave pública proporciona uma assinatura digital melhor do que utilizar a mensagem criptografada com chave pública?
- R14. Suponha que a certificador.com crie um certificado para alguém.com. Normalmente, o certificado inteiro seria criptografado com a chave pública de certificador.com. Verdadeiro ou falso?
- R15. Suponha que Alice tenha uma mensagem pronta para enviar para qualquer pessoa que pedir. Milhares de pessoas querem ter a mensagem da Alice, mas cada uma quer ter certeza da integridade da mensagem. Nesse contexto, você acha que é mais apropriado um esquema baseado em MAC ou um baseado em assinatura digital? Por quê?
- R16. Qual é a finalidade de um nonce em um protocolo de identificação de ponto final?
- R17. O que significa dizer que um nonce é um valor usado uma vez por toda a vida? Pelo tempo de vida de quem?
- R18. O esquema de integridade de mensagem baseado no HMAC é suscetível a ataques de repetição? Se for, como um nonce pode ser incorporado ao esquema para remover essa suscetibilidade?

SEÇÕES 8.5–8.8

- R19. Suponha que Bob receba uma mensagem PGP de Alice. Como Bob sabe com certeza que Alice criou a mensagem (e não Trudy, p. ex.)? O PGP usa um MAC para integridade da mensagem?
- R20. Em um registro TLS, há um campo para uma sequência de números TLS. Verdadeiro ou falso?
- R21. Qual é a finalidade de um nonce em um protocolo de autenticação em uma apresentação TLS?
- R22. Suponha que uma sessão TLS utilize uma cifra de bloco com CBC. Verdadeiro ou falso: o servidor envia o IV ao cliente de forma aberta.
- R23. Suponha que Bob inicie uma conexão TCP com Trudy, que está fingindo ser Alice. Durante a apresentação, Trudy envia a Bob um certificado de Alice. Em qual etapa do algoritmo de apresentação TLS Bob descobrirá que não está se comunicando com Alice?
- R24. Considere uma cadeia de pacotes do Hospedeiro A ao Hospedeiro B usando IPsec. Em geral, um novo SA será estabelecido para cada pacote enviado na cadeia. Verdadeiro ou falso?
- R25. Suponha que o TCP esteja sendo executado por IPsec entre a matriz e a filial na Figura 8.28. Se o TCP retransmitir o mesmo pacote, então os dois pacotes correspondentes

enviados por pacotes R1 terão o mesmo número sequencial no cabeçalho ESP. Verdadeiro ou falso?

- R26. Um SA IKE e um SA IPsec são a mesma coisa. Verdadeiro ou falso?
- R27. Considere um WEP para 802.11. Suponha que a informação seja 10101100 e a sequência de bits da chave seja 1111000. Qual é o texto cifrado resultante?

SEÇÃO 8.9

- R28. Um filtro de pacotes com estado mantém duas estruturas de dados. Nomeie-as e descreva resumidamente o que elas fazem.
- R29. Considere um filtro de pacotes tradicional (sem estado). Ele pode filtrar pacotes baseado em bits de flag TCP assim como em outros campos de cabeçalho. Verdadeiro ou falso?
- R30. Em um filtro de pacotes tradicional, cada interface pode ter sua própria lista de controle de acesso. Verdadeiro ou falso?
- R31. Por que uma aplicação de gateway deve trabalhar junto com o filtro de roteador para ser eficaz?
- R32. IDSs baseados em assinaturas e IPSs inspecionam a carga útil de segmentos TCP e UDP. Verdadeiro ou falso?

Problemas

- P1. Usando a cifra monoalfabética da Figura 8.3, codifique a mensagem “This is an easy problem” (este é um problema fácil). Decodifique a mensagem “rmij’u uamu xyj”.
- P2. Mostre que o ataque com texto aberto conhecido de Trudy, em que ela conhece os pares de tradução (texto cifrado, texto aberto) para sete letras, reduz em aproximadamente 10^9 o número de possíveis substituições a verificar no exemplo apresentado na Seção 8.2.1.
- P3. Considere o sistema polialfabético mostrado na Figura 8.4. Um ataque com texto aberto escolhido que consiga obter a codificação da mensagem “The quick brown fox jumps over the lazy dog” é suficiente para decifrar todas as mensagens? Explique sua resposta.
- P4. Considere o bloco cifrado na Figura 8.5. Suponha que cada bloco cifrado T_i simplesmente inverta a ordem dos oito bits de entrada (então, p. ex., 11110000 se torna 00001111). Além disso, imagine que o misturador de 64 bits não modifique qualquer bit (de modo que o valor de saída do m -ésimo bit seja igual ao valor de entrada do m -ésimo bit). (a) Sendo $n = 3$ e a entrada original de 64 bits igual a 10100000 repetidos oito vezes, qual é o valor da saída? (b) Repita a parte (a), mas agora troque o último bit da entrada original de 64 bits de 0 para 1. (c) Repita as partes (a) e (b), mas agora suponha que o misturador de 64 bits inverte a ordem de 64 bits.
- P5. Considere o bloco cifrado da Figura 8.5 Para uma determinada “chave”, Alice e Bob precisariam ter 8 tabelas, cada uma com 8 bits por 8 bits. Para Alice (ou Bob) armazenar todas as oito tabelas, quantos bits de armazenamento são necessários? Como esse número se compara com o número de bits necessários para um bloco cifrado de tabela cheia de 64 bits?
- P6. Considere o bloco cifrado de 3 bits da Tabela 8.1. Suponha que o texto aberto seja 100100100. (a) Inicialmente suponha que o CBC não seja usado. Qual é o texto cifrado resultante? (b) Imagine que Trudy analise o texto cifrado. Supondo que ela saiba que

um bloco cifrado de bits está sendo usado sem o CBC (mas ela não sabe a cifra específica), o que ela pode suspeitar? (c) Agora suponha que o CBC é usado com $IV = 111$. Qual é o texto cifrado resultante?

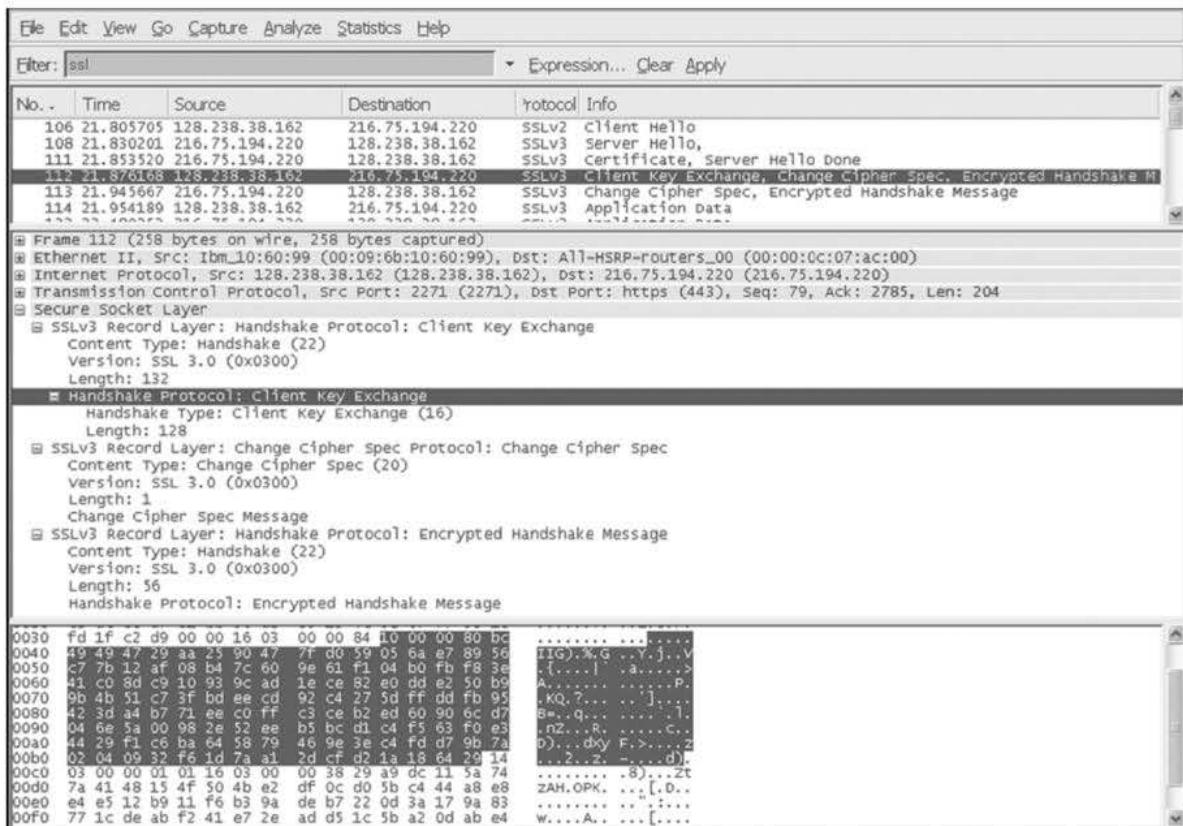
- P7. (a) Usando RSA, escolha $p = 3$ e $q = 11$ e codifique a palavra “dog”, criptografando cada letra em separado. Aplique o algoritmo de decriptação à versão criptografada para recuperar a mensagem original em texto aberto. (b) Repita a parte (a), mas agora criptografe “dog” como uma mensagem m .
- P8. Considere o RSA com $p = 5$ e $q = 11$.
- Quais são n e z ?
 - Seja e igual a 3. Por que esta é uma escolha aceitável para e ?
 - Encontre d tal que $de \equiv 1 \pmod{z}$ e $d < 160$.
 - Criptografe a mensagem $m = 8$ usando a chave (n, e) . Seja c o texto cifrado correspondente. Mostre todo o processo. *Dica:* para simplificar os cálculos, use este fato:

$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

- P9. Neste problema, exploraremos o algoritmo criptografado Diffie-Hellman (DH) de chave pública, o qual permite que duas entidades concordem em uma chave compartilhada. O algoritmo DH faz uso de um número primo grande p e outro número grande g , menor que p . Ambos, p e g , são públicos (de modo que um atacante os conheça). Em DH, Alice e Bob escolhem, independentemente, chaves secretas, S_A e S_B . Alice então calcula sua chave pública, T_A , elevando g a S_A e tomando o mod p . Bob, similarmente, calcula sua própria chave pública, T_B , elevando g a S_B e tomando o módulo p . Então, Alice e Bob trocam suas chaves públicas pela Internet. Alice calcula a chave secreta compartilhada S ao elevar T_B a S_A e tomando o módulo p . Bob, de modo semelhante, calcula a chave compartilhada S' ao elevar T_A a S_B e tomando o módulo p .
- Prove que, no geral, Alice e Bob obtêm a mesma chave simétrica, ou seja, prove que $S = S'$.
 - Com $p = 11$ e $g = 2$, suponha que Alice e Bob escolham chaves privadas $S_A = 5$ e $S_B = 12$, respectivamente. Calcule as chaves públicas de Alice e Bob, T_A e T_B . Mostre todo o processo.
 - Seguindo a parte (b), calcule S como a chave simétrica compartilhada. Mostre todo o processo.
 - Forneça um diagrama de tempo que mostre como o Diffie-Hellman pode sofrer um ataque *man-in-the-middle*. O diagrama de tempo deve ter três linhas verticais, uma para Alice, uma para Bob e uma para a atacante Trudy.
- P10. Suponha que Alice queira se comunicar com Bob usando uma chave simétrica criptografada usando uma chave de sessão K_S . Na Seção 8.2, aprendemos que uma chave pública criptografada pode ser usada para distribuir a chave de sessão de Alice para Bob. Neste problema, exploraremos como uma chave de sessão pode ser distribuída – sem uma chave pública criptografada – usando um centro de distribuição de chaves (KDC, do inglês *key distribution center*). O KDC é um servidor que compartilha uma única chave simétrica secreta com cada usuário registrado. Para Alice e Bob, indique essas chaves como K_{A-KDC} e K_{B-KDC} . Projete um esquema que use o KDC para distribuir K_S a Alice e Bob. Seu esquema deverá usar três mensagens para distribuir uma chave de sessão: a de Alice ao KDC; a do KDC a Alice; e, por fim, a de Alice para Bob. A primeira mensagem é $K_{A-KDC}(A, B)$. Usando a notação, K_{A-KDC} , K_{B-KDC} , S , A e B , responda às seguintes perguntas.
- Qual é a segunda mensagem?
 - Qual é a terceira mensagem?

- P11. Calcule uma terceira mensagem, diferente das duas mensagens na Figura 8.8, que tenha a mesma soma de verificação dessas duas mensagens na figura citada.
- P12. Suponha que Alice e Bob compartilhem duas chaves secretas: uma chave de autenticação S_1 e uma chave criptografada simétrica S_2 . Aumente a Figura 8.9 para que ambas, integridade e confidencialidade, sejam fornecidas.
- P13. No protocolo de distribuição de arquivo P2P BitTorrent (veja o Capítulo 2), a semente quebra o arquivo em blocos, e os pares redistribuem os blocos uns aos outros. Sem proteção alguma, um atacante pode facilmente causar destruição em um torrent ao se mascarar como se fosse um par benevolente e enviar blocos falsos aos outros conjuntos de pares no torrent. Esse pares que não são suspeitos redistribuem os blocos falsos a mais outros pares, que distribuem os blocos falsos aos outros pares. Sendo assim, é importante para o BitTorrent ter um mecanismo que permita aos pares verificar a integridade de um bloco, para que não distribuam blocos falsos. Suponha que, quando um par se junta a um torrent, de início pega um arquivo .torrent de uma fonte *inteiramente* confiável. Descreva um esquema simples que permita que os pares verifiquem a integridade dos blocos.
- P14. O protocolo de roteamento OSPF usa um MAC em vez de assinaturas digitais para fornecer a integridade da mensagem. Por que você acha que foi escolhido o MAC em vez de assinaturas digitais?
- P15. Considere nosso protocolo de autenticação da Figura 8.18, no qual Alice se autentica para Bob, e que vimos que funciona bem (i.e., não encontramos nenhuma falha nele). Agora suponha que, enquanto Alice está se autenticando para Bob, este deve se autenticar para Alice. Apresente um cenário no qual Trudy, fazendo-se passar por Alice, agora pode se autenticar para Bob como se fosse Alice. (*Dica:* considere que a sequência de operações do protocolo, uma com Trudy e outra com Bob iniciando a sequência, pode ser intercalada arbitrariamente. Preste atenção particular ao fato de que Bob e Alice usarão um nonce e que, caso não se tome cuidado, o mesmo nonce pode ser utilizado de forma maliciosa.)
- P16. Uma questão natural é se podemos usar um nonce e a criptografia de chave pública para resolver o problema de autenticação do ponto final na Seção 8.4. Considere o seguinte protocolo natural: (1) Alice envia a mensagem “Eu sou Alice” a Bob. (2) Bob escolhe um nonce, R , e o envia a Alice. (3) Alice usa sua chave *privada* para criptografar o nonce e envia o valor resultante a Bob. (4) Bob aplica a chave pública de Alice à mensagem recebida. Assim, Bob calcula R e autentica Alice.
- Faça um diagrama desse protocolo, usando a notação para chaves públicas e privadas empregada neste livro.
 - Suponha que os certificados não sejam usados. Descreva como Trudy pode se tornar uma “woman-in-the-middle”, interceptando as mensagens de Alice e depois se passando por Alice para Bob.
- P17. A Figura 8.21 mostra as operações que Alice deve realizar com PGP para fornecer confidencialidade, autenticação e integridade. Faça um diagrama das operações correspondentes que Bob precisa executar no pacote recebido por Alice.
- P18. Suponha que Alice queira enviar um e-mail a Bob. Bob tem um par de chave pública-privada (K_B^+, K_B^-) , e Alice tem o certificado de Bob. Mas Alice não tem um par de chave pública, privada. Alice e Bob (e o mundo inteiro) compartilham a mesma função de hash $H(\cdot)$.
- Nessa situação, é possível projetar um esquema para que Bob possa verificar que Alice criou a mensagem? Se sim, mostre como, com um diagrama de bloco para Alice e Bob.
 - É possível projetar um esquema que forneça confidencialidade para enviar a mensagem de Alice a Bob? Se sim, mostre como com um diagrama de bloco para Alice e Bob.

- P19. Considere a saída Wireshark a seguir para uma parte de uma sessão TLS.
- O pacote Wireshark 112 é enviado pelo cliente ou pelo servidor?
 - Qual o número IP do servidor e seu número de porta?
 - Considerando que nenhuma perda ou retransmissão ocorreram, qual será a sequência de números do próximo segmento TCP enviado pelo cliente?
 - Quantos registros TLS existem no pacote Wireshark 112?
 - O pacote 112 contém um Segredo Mestre ou um Segredo Mestre Criptografado ou nenhum?
 - Supondo que o campo do tipo na apresentação é de 1 byte e cada campo de comprimento é de 3 bytes, quais são os valores do primeiro e do último bytes do Segredo Mestre (ou do Segredo Mestre Criptografado)?
 - A mensagem de apresentação criptografada do cliente leva em conta o número de registros TLS?
 - A mensagem de apresentação criptografada do servidor leva em conta o número de registros TLS?



(Captura de tela do Wireshark reimpressa com permissão da Wireshark Foundation.)

- P20. Na Seção 8.6.1, mostramos que, sem os números de sequência, Trudy (a “woman-in-the-middle”) pode causar destruição em uma sessão TLS ao trocar os segmentos TCP. Trudy pode fazer algo semelhante ao excluir um segmento TCP? O que ela precisa fazer para ter sucesso em seu ataque de exclusão? Quais serão os seus efeitos?

- P21. Suponha que Alice e Bob estão se comunicando por uma sessão TLS. Imagine que um atacante, que não tem nenhuma das chaves compartilhadas, insira um segmento TCP falso em um fluxo de pacotes com a soma de verificação TCP e números sequenciais corretos (e endereços IP e números de porta corretos). A TLS do lado receptor aceitará o pacote falso e passará a carga útil à aplicação receptora? Por que ou por que não?
- P22. As seguintes questões de Verdadeiro/Falso se referem à Figura 8.28.
- Quando um hospedeiro em 172.16.1/24 envia um datagrama ao servidor Amazon.com, o roteador R1 vai criptografar o datagrama usando IPsec.
 - Quando um hospedeiro em 172.16.1/24 envia um datagrama a um servidor 172.16.2/24, o roteador R1 mudará os endereços de origem e destino do datagrama IP.
 - Suponha que um hospedeiro 172.16.1/24 inicie uma conexão TCP com um servidor Web em 172.16.2/24. Como parte dessa conexão, todos os datagramas enviados pelo R1 terão o número de protocolo 50 no campo esquerdo do cabeçalho IPv4.
 - Considere o envio de um segmento TCP de um hospedeiro em 172.16.1/24 a um hospedeiro em 172.16.2/24. Suponha que o reconhecimento desse segmento se perde, então um TCP reenvia o segmento.
- P23. Considere o exemplo na Figura 8.28. Suponha que Trudy é a “woman-in-the-middle”, que insere datagramas no fluxo de datagramas indo de R1 a R2. Como parte do ataque de repetição, Trudy envia uma cópia duplicada de um dos datagramas enviados de R1 a R2. R2 decriptografará o datagrama duplicado e o encaminhará à rede da filial? Se não, descreva em detalhes como R2 detecta o datagrama duplicado.
- P24. Forneça uma tabela de filtro e uma tabela de conexão para um firewall de estado que seja tão restrito quanto possível, mas que efetue o seguinte:
- Permita que todos os usuários internos estabeleçam sessões Telnet com hospedeiros externos.
 - Permita que usuários externos naveguem pela página Web da empresa em 222.22.0.12.
 - Mas, em qualquer outro caso, bloqueie todo o tráfego interno e externo.
- A rede interna é 222.22/16. Em sua solução, suponha que a tabela de conexão esteja normalmente ocultando três conexões, de dentro para fora. Você precisará inventar um endereço IP e números de portas apropriados.
- P25. Suponha que Alice queira visitar a página Web activist.com usando um serviço do tipo TOR. Esse serviço usa dois servidores proxy colaboradores, Proxyl e Proxy2. Alice primeiro obtém os certificados (cada um contendo uma chave pública) para Proxyl e Proxy2 de um servidor central. Indique com $K_1^+()$, $K_1^-()$, $K_2^+()$ e $K_2^-()$ para a criptografia/decriptografia com chaves RSA pública e privada.
- Usando um diagrama de tempo, forneça um protocolo (o mais simples possível) que permita a Alice estabelecer uma sessão compartilhada de chave S_1 com Proxyl. Indique com $S_1(m)$ a criptografia/decriptografia do dado m com a chave compartilhada S_1 .
 - Usando um diagrama de tempo, forneça um protocolo (o mais simples possível) que permita a Alice estabelecer uma sessão compartilhada da chave S_2 com Proxy2, *sem revelar seu endereço IP ao Proxy2*.
 - Suponha que as chaves compartilhadas S_1 e S_2 estejam determinadas. Usando um diagrama de tempo, forneça um protocolo (o mais simples possível e *não usando uma criptografia de chave pública*) que permita a Alice requisitar uma página HTML de activist.com *sem revelar seu endereço IP ao Proxy2 e sem revelar ao Proxyl qual site ela está visitando*. Seu diagrama deve terminar com uma requisição HTTP chegando a activist.com.

Wireshark Lab: TLS

Neste laboratório (disponível no site de apoio), investigamos o protocolo TLS. Lembre-se de que na Seção 8.6 o TLS é usado para a segurança de uma conexão TCP, e é extensivamente usado na prática para a segurança de transações pela Internet. Neste laboratório, vamos focalizar os registros TLS enviados por uma conexão TCP. Tentaremos delinear e classificar cada registro, focando no entendimento do porquê e como de cada registro. Investigamos os vários tipos de registro TLS, assim como os campos nas mensagens TLS, analisando um relatório dos registros TLS enviados entre seu hospedeiro e um servidor de comércio eletrônico.

IPsec Lab

Neste laboratório (disponível no site de apoio), exploraremos como criar SAs IPsec entre caixas Linux. Você pode fazer a primeira parte com duas caixas Linux simples, cada uma com um adaptador Ethernet. Mas, na segunda parte do laboratório, você precisará de quatro caixas Linux, duas tendo dois adaptadores Ethernet. Na segunda metade do laboratório, você criará SAs IPsec usando protocolos ESP no modo túnel. Primeiro você criará as SAs manualmente, e depois fazendo o IKE criar as SAs.

ENTREVISTA

Steven M. Bellovin

Steven M. Bellovin ingressou no corpo docente da Universidade de Columbia depois de muitos anos no Network Services Research Lab do AT&T Labs Research, em Florham Park, Nova Jersey. Ele trabalha especificamente com redes e segurança e com as causas que as tornam incompatíveis. Em 1995, Steven recebeu o Usenix Lifetime Achievement Award por seu trabalho na criação da Usenet, a primeira rede de troca de grupos de bate-papo que ligava dois ou mais computadores e permitia que os usuários compartilhassem informações e se juntassem em discussões. Steve também foi eleito membro da National Academy of Engineering. Obteve bacharelado na Columbia University e doutorado na University of North Carolina, em Chapel Hill.

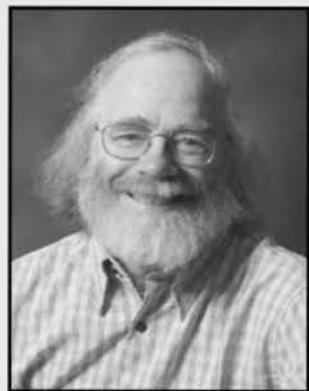


Imagem cortesia de Steven Bellovin

O que o fez se decidir pela especialização na área de segurança de redes?

O que vou dizer parecerá estranho, mas a resposta é simples. Estudei programação de sistemas e administração de sistemas, o que levou naturalmente à segurança. E sempre me interessei por comunicações, desde a época em que trabalhava em parte do tempo com programação de sistemas, quando ainda estava na universidade.

Meu trabalho na área de segurança continua a ser motivado por duas coisas – um desejo de manter os computadores úteis, o que significa impedir que sua função seja corrompida por atacantes, e um desejo de proteger a privacidade.

Qual era sua visão sobre a Usenet na época em que você a estava desenvolvendo? Qual é sua visão agora?

No início, considerávamos a Usenet como um meio para discutir ciência da computação e programação de computadores em todo o país e para utilizar em questões administrativas locais, como recurso de vendas, e assim por diante. Na verdade, minha previsão original era de uma ou duas mensagens por dia, de 50 a 100 sites no máximo. Mas o crescimento real ocorreu em tópicos relacionados a pessoas, incluindo – mas não se limitando a – interações de seres humanos com computadores. Meus grupos de bate-papo preferidos foram, durante muitos anos, coisas como rec.woodworking (marcenaria), bem como sci.crypt (criptografia).

Até certo ponto, as redes de notícias foram substituídas pela Web. Se eu fosse iniciar o projeto hoje, ele seria muito diferente. Mas ainda é um excelente meio para alcançar um público muito amplo interessado no assunto, sem ter de passar por sites Web particulares.

Quais pessoas o inspiraram profissionalmente? De que modo?

O professor Fred Brooks – fundador e primeiro chefe do departamento de ciência da computação da University of North Carolina, em Chapel Hill, gerente da equipe que desenvolveu o IBM S/360 e o OS/360, e autor do livro *The Mythical Man-Month* – exerceu uma tremenda influência sobre minha carreira. Acima de tudo, ele me ensinou observação e compromissos – como considerar problemas no contexto do mundo real (e como o mundo real é muito mais confuso do que qualquer teórico gostaria que fosse), e como equilibrar interesses conflitantes ao elaborar uma solução. Grande parte do trabalho com computadores é engenharia – a arte de fazer os compromissos certos de modo a satisfazer muitos objetivos contraditórios.

Em sua opinião, qual é o futuro das redes e da segurança?

Até agora, grande parte da segurança que temos vem do isolamento. Um firewall, por exemplo, funciona impedindo o acesso a certas máquinas e serviços. Mas vivemos em uma época na qual a

conectividade é cada vez maior – ficou mais difícil isolar coisas. Pior ainda, nossos sistemas de produção requerem um número muito maior de componentes isolados, interconectados por redes. Garantir a segurança de tudo isso é um de nossos maiores desafios.

Em sua opinião, tem havido grandes avanços na área de segurança? Até onde teremos de ir?

Ao menos do ponto de vista científico, sabemos como fazer criptografia. E isso é uma grande ajuda. Mas a maioria dos problemas de segurança se deve a códigos defeituosos, e este é um problema muito mais sério. Na verdade, é o problema mais antigo da ciência da computação que ainda não foi resolvido – e acho que continuará sendo. O desafio é descobrir como manter a segurança de sistemas quando temos de construí-los com componentes inseguros. Hoje já podemos fazer isso no que tange às falhas de hardware; mas podemos fazer o mesmo em relação à segurança?

O senhor pode dar algum conselho aos estudantes sobre a Internet e segurança em redes?

Aprender os mecanismos é a parte fácil. Aprender a “pensar como um paranoico” é mais difícil. Você tem de lembrar que as distribuições de probabilidade não se aplicam – os atacantes podem encontrar e encontrarão condições improváveis. E os detalhes são importantes – e muito!