

desatenção com a segurança (Anderson, 2001). Apesar disso, somos otimistas e acreditamos que, à medida que o comércio eletrônico se tornar mais difundido, as empresas irão depurar seus procedimentos operacionais, eliminando esse furo e trazendo os aspectos técnicos da segurança de volta à cena.

Com exceção da segurança na camada física, quase toda segurança se baseia em princípios criptográficos. Por essa razão, começaremos nosso estudo da segurança examinando em detalhes a criptografia. Na Seção 8.1, veremos alguns princípios básicos. Da Seção 8.2 até a Seção 8.5, examinaremos alguns algoritmos e estruturas de dados fundamentais usados em criptografia. Em seguida, examinaremos em detalhes como esses conceitos podem ser usados para se alcançar a segurança em redes. Concluiremos com alguns conceitos breves sobre tecnologia e sociedade.

Antes de começarmos, devemos chamar a atenção para o que não é abordado neste capítulo. Procuramos nos concentrar em questões de redes, e não em questões relacionadas ao sistema operacional e às aplicações, embora seja difícil traçar a linha que separa esses assuntos. Por exemplo, não há nada aqui sobre autenticação do usuário com a utilização da biometria, segurança de senhas, ataques de estouro de buffers, cavalos de Tróia, spoofing de login, bombas lógicas, vírus, vermes e temas semelhantes. Todos esses tópicos são abordados detalhadamente no Capítulo 9 do livro *Modern Operating Systems* (Tanenbaum, 2001). O leitor interessado deve consultar esse livro para conhecer os aspectos de segurança relacionados aos sistemas. Agora, vamos iniciar nossa jornada.

8.1 Criptografia

A palavra criptografia vem das palavras gregas que significam "escrita secreta". A criptografia tem uma longa e interessante história de milhares de anos. Nesta seção, vamos esquematizar alguns destaques, que serão usados como informações básicas para o que vem a seguir. Se desejar um histórico completo da criptografia, recomendamos a leitura do livro de Khan (1995). Para ver um tratamento completo do estado da arte atual em segurança e algoritmos criptográficos, protocolos e aplicações, consulte (Kaufman et al., 2002). Para uma abordagem mais matemática, consulte (Stinson, 2002). Se preferir uma abordagem menos matemática, consulte (Burnett e Paine, 2001). Os profissionais fazem distinção entre cifras e códigos. Uma cifra é uma transformação de caractere por caractere ou de bit por bit, sem levar em conta a estrutura lingüística da mensagem. Em contraste, um código substitui uma palavra por outra palavra ou símbolo. Os códigos não são mais utilizados, embora tenham uma história gloriosa. O código mais bem-sucedido já inventado foi usado pelas forças armadas dos Estados Unidos durante a Segunda Guerra Mundial no Pacífico. Eles simplesmente tinham índios Navajo que se comunicavam uns com os outros usando palavras Navajo específicas para termos militares como, por exemplo, chay-dagahi-nail-tsaidi (literalmente assassino de cágado) para indicar uma arma antitanque. A linguagem Navajo é altamente tonal, extremamente complexa, e não tem nenhuma forma escrita. Além disso, nem uma única pessoa no Japão conhecia nada sobre ela.

Em setembro de 1945, o San Diego Union descreveu o código da seguinte forma: "Por três anos, onde quer que os Marines aterrissassem, os japoneses recebiam uma enxurrada de estranhos ruídos gorgolejantes entremeados com outros sons que lembravam o clamor de um monge tibetano e o som de uma bolsa de água quente sendo esvaziada". Os japoneses nunca conseguiram romper o código e muitos índios Navajo receberam altas honras militares por serviço e bravura extraordinários. O fato dos Estados Unidos terem conseguido romper o código japonês e os japoneses nunca terem conseguido quebrar o código Navarro desempenhou um papel crucial nas vitórias americanas no Pacífico.

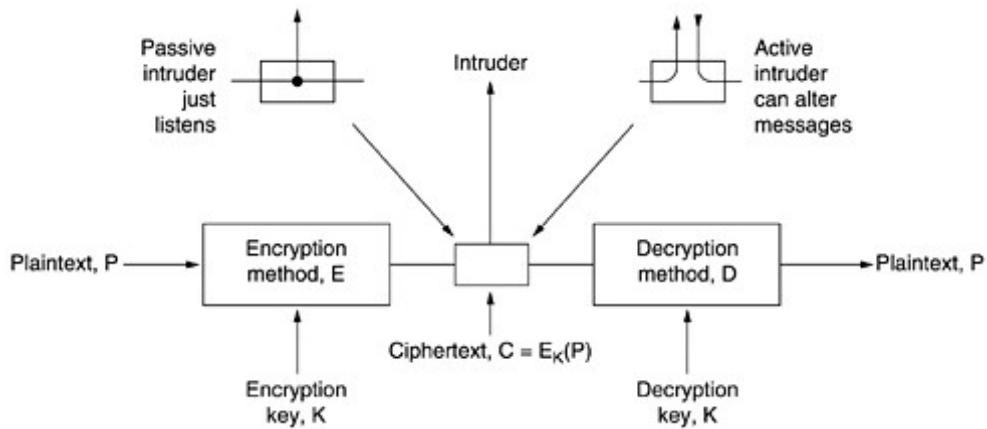
8.1.1 Introdução à criptografia

Historicamente, quatro grupos de pessoas utilizaram e contribuíram para a arte da criptografia: os militares, os diplomatas, as pessoas que gostam de guardar memórias e os amantes. Dentre eles, os militares tiveram o papel mais importante e definiram as bases para a tecnologia. Dentro das organizações militares, tradicionalmente as mensagens a serem criptografadas são entregues a auxiliares mal remunerados que se encarregam de criptografá-las e transmiti-las. O grande volume de mensagens impedia que esse trabalho fosse feito por alguns poucos especialistas.

Até o advento dos computadores, uma das principais restrições impostas à criptografia era a habilidade de auxiliar de criptografia fazer as transformações necessárias, em geral com poucos equipamentos e no campo de batalha. Uma outra restrição era a dificuldade de alternar os métodos criptográficos rapidamente, pois isso exigia a repetição do treinamento de um grande número de

pessoas. No entanto, o perigo de um auxiliar de criptografia ser capturado pelo inimigo tornou indispensável a possibilidade de se alterar o método criptográfico instantaneamente, se necessário. Essas necessidades conflitantes fizeram surgir o modelo da Figura 8.2.

Figura 8.2: O modelo de criptografia (para uma cifra de chave simétrica)



As mensagens a serem criptografadas, conhecidas como texto simples, são transformadas por uma função que é parametrizada por uma chave. Em seguida, a saída do processo de criptografia, conhecida como **texto cifrado**, é transmitida, normalmente através de um mensageiro ou por rádio. Presumimos que o inimigo, ou intruso, ouça e copie cuidadosamente o texto cifrado completo. No entanto, ao contrário do destinatário pretendido, ele não conhece a chave para descriptografar o texto e, portanto, não pode fazê-lo com muita facilidade. Às vezes, o intruso pode não só escutar o que se passa no canal de comunicação (intruso passivo), como também pode gravar mensagens e reproduzi-las mais tarde, injetar suas próprias mensagens ou modificar mensagens legítimas antes que elas cheguem ao receptor (intruso ativo). A arte de solucionar mensagens cifradas é chamada **criptoanálise**. A arte de criar mensagens cifradas (criptografia) e solucioná-las (criptoanálise) é chamada coletivamente **criptologia**.

Com freqüência, será útil e prático ter uma notação para estabelecer uma relação entre o texto simples, o texto cifrado e as chaves. Utilizaremos $C = E_K(P)$ para denotar que a criptografia do texto simples P usando a chave K gera o texto cifrado C . Da mesma forma, $P = D_K(C)$ representa a descriptografia de C para se obter o texto simples outra vez. Então, temos:

$$D_K(E_K(P)) = P$$

Essa notação sugere que E e D são simplesmente funções matemáticas, o que é verdade. A única parte complicada é que ambas são funções de dois parâmetros, e escrevemos um desses parâmetros (a chave) como um caractere subscrito, em vez de representá-lo como um argumento, para distingui-lo da mensagem.

Uma regra fundamental da criptografia é que se deve supor que o criptoanalista conhece os métodos genéricos de criptografia e descriptografia que são utilizados. Em outras palavras, o criptoanalista sabe como funciona o método de criptografia, E , e o método de descriptografia D da Figura 8.2. O esforço necessário para criar, testar e instalar um novo algoritmo toda vez que o antigo método (supostamente) é comprometido sempre dificultou a manutenção desse segredo. Imaginar que o algoritmo de criptografia é secreto quando ele não é resulta em mais prejuízo do que em benefícios.

É nesse ponto que entra a chave. A chave consiste em um string (relativamente) curto que seleciona uma das muitas formas possíveis de criptografia. Ao contrário do método genérico, que só pode ser modificado a cada período de alguns anos, a chave pode ser alterada sempre que necessário. Portanto, nosso modelo básico é um método genérico publicamente conhecido, parametrizado por uma chave secreta que pode ser alterada com facilidade. A idéia de que o criptoanalista conhece os algoritmos e que o segredo reside exclusivamente nas chaves é chamada **princípio Kerckhoff**, que recebeu esse nome em homenagem ao criptógrafo militar flamengo Auguste Kerckhoff que enunciou primeiramente em 1883 (Kerckhoff, 1883). Desse modo, temos:

Princípio de Kerckhoff: Todos os algoritmos devem ser públicos; apenas as chaves são secretas

Devemos enfatizar o caráter não sigiloso do algoritmo. Tentar manter o algoritmo secreto, uma estratégia conhecida no ramo como segurança pela obscuridade, nunca funciona. Além disso, ao tornar o algoritmo público, o especialista em criptografia se livra de te de consultar inúmeros criptólogos ansiosos por decodificar o sistema para poderem publicar artigos demonstrando sua esperteza e inteligência. Caso muitos especialistas tenham tentado decodificar o algoritmo durante cinco anos após sua publicação e nenhum tenha obtido sucesso, isso provavelmente significa que o algoritmo é sólido.

Na verdade, o sigilo está na chave, e seu tamanho é uma questão muito importante do projeto. Considere um bloco de combinação simples. Segundo o princípio geral, você insere dígitos em seqüência. Todo mundo sabe disso, mas a chave é secreta. Uma chave com um tamanho de dois dígitos significa que existem 100 possibilidades, uma chave de três dígitos significa mil possibilidades e uma chave de seis dígitos significa um milhão de possibilidades. Quanto maior for a chave, mais alto será o fator de trabalho com que o criptoanalista terá de lidar. O fator de trabalho para decodificar o sistema através de uma exaustiva pesquisa no espaço da chave é exponencial em relação ao tamanho da chave. O sigilo é decorrente da presença de um algoritmo forte (mas público) e de uma chave longa. Para impedir que o seu irmãozinho leia suas mensagens de correio eletrônico, serão necessárias chaves de 64 bits. Para uso comercial de rotina, devem ser usados pelo menos 128 bits. Para manter o governo de outros países à distância, são necessárias chaves de pelo menos 256 bits, de preferência maiores.

Do ponto de vista do criptoanalista, o problema da criptoanálise apresenta três variações principais. Quando tem um de terminado volume de texto cifrado mas nenhum texto simples, o analista é confrontado com o problema de haver somente texto cifrado. Os criptogramas da seção de palavras cruzadas do jornal são um exemplo desse tipo de problema. Quando há uma correspondência entre o texto cifrado e o texto simples, o problema passa a ser chamado texto simples conhecido. Por fim, quando o criptoanalista tem a possibilidade de codificar trechos do texto simples escolhidos por ele mesmo, temos o problema do texto simples escolhido. Os criptogramas dos jornais poderiam ser decodificados de forma trivial se o criptoanalista tivesse a permissão de fazer perguntas tais como: Qual é a criptografia de ABCDEFGHIJKLMNOP?

Com freqüência, os novatos na área de criptografia pressupõem que, se uma cifra puder resistir a uma estratégia de texto cifrado, isso significa que ela é segura. Essa suposição é muito ingênuca. Em muitos casos, o criptoanalista pode fazer uma estimativa com base em trechos do texto simples. Por exemplo, a primeira mensagem que muitos sistemas de tempo compartilhado emitem quando você os chama é "LOGIN:". Equipado com alguns pares de texto simples/texto cifrado, o trabalho do criptoanalista se torna muito mais fácil. Para obter segurança, o autor da criptografia deve ser conservador e se certificar de que o sistema seja inviolável, mesmo que seu oponente seja capaz de criptografar o texto simples escolhido.

Historicamente, os métodos de criptografia têm sido divididos em duas categorias: as cifras de substituição e as cifras de transposição. Em seguida, trataremos de cada uma dessas técnicas como informações básicas para a criptografia moderna.

8.1.2 Cifras de substituição

Em uma cifra de substituição, cada letra ou grupo de letras é substituído por outra letra ou grupo de letras, de modo a criar um "disfarce". Uma das cifras mais antigas é a cifra de César, atribuída a Júlio César. Nesse método, a se torna D, b se torna E, c se torna F,... e z se torna C. Por exemplo, ataque passaria a ser WDTXH. Nos exemplos, o texto simples é apresentado em letras minúsculas e o texto cifrado em letras maiúsculas.

Uma ligeira generalização da cifra de César permite que o alfabeto do texto cifrado seja deslocado k letras, em vez de 3. Nesse caso, k passa a ser uma chave para o método genérico dos alfabetos deslocados em forma circular. A cifra de César pode ter enganado os cartagineses, mas nunca mais enganou ninguém.

O próximo aprimoramento é fazer com que cada um dos símbolos do texto simples, digamos 26 letras, seja mapeado para alguma outra letra. Por exemplo,

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

Esse sistema geral é chamado **substituição monoalfabética**, sendo a chave o string de 26 letras correspondente ao alfabeto completo. Para a chave anterior, o texto simples ataque seria transformado no texto cifrado QZQJXT.

À primeira vista, talvez esse sistema pareça seguro, pois apesar de conhecer o sistema genérico (substituição de letra por letra), o criptoanalista não sabe qual das $26! \approx 4 \times 10^{26}$ chaves possíveis está em uso. Ao contrário do que acontece com a cifra de César, experimentar todas elas não é uma estratégia muito interessante. Mesmo a 1 ns por solução, um computador levaria 10^{10} anos para experimentar todas as chaves.

Todavia, com um volume de texto cifrado surpreendentemente pequeno, a cifra pode ser descoberta com facilidade. A estratégia básica se beneficia das propriedades estatísticas dos idiomas. Por exemplo, em inglês é a letra mais comum, seguida de t, o, a, n, i etc. As combinações de duas letras, ou digramas, mais comuns são th, in, er, re e an. As combinações de três letras, ou trigrama, mais comuns são the, ing, and e ion.

Um criptoanalista que esteja tentando decodificar uma cifra monoalfabética começaria contando as freqüências relativas de todas as letras do texto cifrado. Depois disso, através de tentativas, ele atribuiria e à letra mais comum e t à próxima letra mais comum. Em seguida, verificaria os trigramas para encontrar um no formato tXe, o que poderia sugerir que X é h. Da mesma forma, se o padrão thYt ocorrer com freqüência, provavelmente isso significará que Y representa a. Com essas informações, o criptoanalista poderá procurar por um trigrama com o formato aZW que ocorra com freqüência (muito provavelmente

and). Fazendo estimativas em relação a digramas, trigramas e letras mais comuns, e conhecendo os prováveis padrões de vogais e consoantes, o criptoanalista criaria um texto simples através de tentativas, letra por letra.

Outra estratégia é adivinhar uma palavra ou frase provável. Por exemplo, considere o seguinte texto cifrado de uma empresa de contabilidade (montado em grupos de cinco caracteres):

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

Nos Estados Unidos, uma palavra muito provável em uma mensagem de uma empresa de contabilidade é financial. Utilizando nosso conhecimento de que financial tem um caractere repetido (i), com quatro outras letras entre suas ocorrências, estamos procurando letras repetidas no texto cifrado com esse espaço entre elas. Encontramos 12 casos como esse nas posições 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 e 82. No entanto, apenas dois deles, 31 e 42, têm a letra seguinte (que corresponde a n no texto simples) repetida na localização adequada. Dessas duas, apenas 31 também tem a letra a corretamente posicionada; portanto, sabemos que financial começa na posição 30. Desse ponto em diante, fica fácil deduzir a chave utilizando a estatística de freqüência para o texto em inglês.

8.1.3 Cifras de transposição

As cifras de substituição preservam a ordem dos símbolos no texto simples, mas disfarçam esses símbolos. Por outro lado, as cifras de transposição reordenam as letras, mas não as disfarçam. A Figura 8.3 mostra uma cifra de transposição muito comum, a transposição de colunas. A cifra se baseia em uma chave que é uma palavra ou frase que não contém letras repetidas. Nesse exemplo, MEGABUCK é a chave. O objetivo da chave é numerar as colunas de modo que a coluna 1 fique abaixo da letra da chave mais próxima do início do alfabeto e assim por diante. O texto simples é escrito horizontalmente, em linhas. O texto cifrado é lido em colunas, a partir da coluna cuja letra da chave seja a mais baixa.

Figura 8.3: Uma cifra de transposição

M E G A B U C K	
7 4 5 1 2 8 3 6	
p l e a s e t r	Plaintext
a n s f e r o n	please transfer one million dollars to
e m i l l i o n	my swiss bank account six two two
d o l l a r s t	Ciphertext
o m y s w i s s	AFLLSKSOSELAWAIATO OSSCTCLNMOMANT
b a n k a c c o	ESILYNTWRNNTSOWDPAEDOBUEIRICXB
u n t s i x t w	
o t w o a b c d	

Para romper uma cifra de transposição, o criptoanalista deve primeiro estar ciente de que está lidando com uma cifra de transposição. Examinando a freqüência de E, T, A, O, I, N etc., fica fácil constatar se essas letras se encaixam no padrão normal para texto simples. Se houver correspondência, isso significa que a cifra é evidentemente uma cifra de transposição, pois nesse tipo de cifra cada letra é representada por ela mesma, mantendo intacta a distribuição de freqüências.

A próxima etapa é fazer uma estimativa do número de colunas. Em muitos casos, uma palavra ou frase provável pode ser deduzida a partir do contexto da mensagem. Por exemplo, suponha que o nosso criptoanalista tenha suspeitado de que a frase em texto simples milliondollars ocorre em algum lugar na mensagem. Observe que os digramas MO, IL, LL, LA, IR e OS ocorrem no texto cifrado como um resultado do desdobramento dessa frase. No texto cifrado, a letra O vem depois da letra M (ou seja, elas são verticalmente adjacentes na coluna 4), pois estão separadas na provável frase por uma distância igual ao tamanho da chave. Se tivesse sido usada uma chave de tamanho sete, teriam surgido os digramas MD, IO, LL, LL, IA, OR e NS. Na verdade, para cada tamanho de chave, é produzido um conjunto de digramas diferente no texto cifrado. Ao tentar encontrar diferentes possibilidades, muitas vezes o criptoanalista é capaz de determinar com facilidade o tamanho da chave.

A última etapa é ordenar as colunas. Quando o número de colunas k é pequeno, cada um dos $k(k - 1)$ pares de colunas pode ser examinado para que seja constatado se suas freqüências de digramas correspondem às do texto simples em inglês. O par que tiver a melhor correspondência será considerado na posição correta. Em seguida, cada uma das colunas restantes é experimentada como sucessora desse par. A coluna cujas freqüências de digramas e trigramas proporcione a melhor correspondência será experimentalmente considerada correta. O processo inteiro continua até ser encontrada uma ordenação potencial. O mais provável é que o texto simples seja reconhecido nesse ponto (por exemplo, se ocorrer milloin, ficará claro qual é o erro).

Algumas cifras de transposição aceitam um bloco de tamanho fixo como entrada e produzem um bloco de tamanho fixo como saída. Essas cifras podem ser completamente descritas fornecendo-se um a lista que informe a ordem na qual os caracteres devem sair. Por exemplo, a cifra da Figura 8.3 pode ser vista como uma cifra de blocos de 64 caracteres. Sua saída é 4, 12, 20, 28, 36, 44, 52, 60, 5, 13,..., 62. Em outras palavras, o quarto caractere de entrada, a, é o primeiro a sair, seguido pelo décimo segundo, f, e assim por diante.

8.1.4 Chave única

Na verdade, é fácil criar uma cifra inviolável; a técnica é conhecida há décadas. Primeiro, escolha como chave um string de bits aleatórios. Em seguida, converta o texto simples em um string de bits, utilizando por exemplo sua representação ASCII. Por fim, calcule o OR exclusivo (X OR) desses dois strings. O texto cifrado resultante não pode ser violado porque, em uma amostra suficientemente grande de texto cifrado, cada letra ocorrerá com a mesma freqüência, bem como digrama, cada trígrama e assim por diante. Esse método, conhecido como **chave única**, é imune a todos os ataques presentes e futuros, quanta capacidade computacional tenha o intruso. A razão deriva da teoria da informação: simplesmente não existe nenhuma informação na mensagem, todos os textos simples possíveis com o tamanho dado são igualmente prováveis.

Um exemplo de como as chaves únicas são usadas, é dado na Figura 8.4. Primeiro, a mensagem 1, "I love you", é convertida em ASCII de 7 bits. Em seguida, uma chave única chamada chave 1, é escolhida e sujeita à operação XOR com a mensagem para se obter o texto cifrado. Um criptoanalista poderia experimentar todas as chaves únicas possíveis para ver que texto resultou para cada uma. Por exemplo, a chave única listada como chave 2 na figura poderia ser

experimentada, resultando no texto simples 2, "Elvis lives", que pode ser ou não plausível (um assunto que está além do escopo deste livro). De fato, para cada texto simples ASCII de 11 caracteres, existe uma chave única que o gera. É isso que queremos dizer quando mencionamos que não existe nenhuma informação no texto cifrado: é possível obter qualquer mensagem com o tamanho correto a partir dele.

Figura 8.4: O uso de uma chave única para criptografia e a possibilidade de conseguir qualquer texto simples que seja possível a partir do texto cifrado pela utilização de alguma outra chave

Mensagem 1:	1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Chave 1:	1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Texto cifrado:	0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101
Chave 2:	1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Texto simples 2:	1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

As chaves únicas são ótimas na teoria, mas têm várias desvantagens na prática. Para começar, a chave não pode ser memorizada; então, tanto o remetente quanto o destinatário devem levar uma cópia escrita com eles. Se qualquer um dos dois estiver sujeito à possibilidade de captura, as chaves escritas sem dúvida serão indesejáveis. Além disso, a quantidade total de dados que podem ser transmitidos é limitada pelo tamanho da chave disponível. Se o espião tiver muita sorte e descobrir uma grande quantidade de dados, talvez ele seja incapaz de transmiti-los de volta para a matriz, porque a chave foi consumida. Outro problema é a sensibilidade do método para caracteres perdidos ou inseridos. Se o transmissor e o receptor ficarem fora de sincronismo, todos os dados a partir desse momento parecerão adulterados.

Com o advento dos computadores, a chave única se tornou potencialmente prática para algumas aplicações. A origem da chave poderia ser um DVD especial contendo vários gigabytes de informações que, se transportadas em uma caixa de filmes em DVD e tivessem no início alguns minutos de vídeo, nem sequer seriam suspeitas. É claro que, nas redes de gigabits, ter de inserir um novo DVD a cada 30 segundos seria algo tremendamente entediante. Além disso, os DVDs devem ser transportados pessoalmente do transmissor para o receptor, antes de ser possível enviar qualquer mensagem, o que reduz bastante sua utilidade prática.

Criptografia quântica

É interessante observar que talvez haja uma solução para o problema de como transmitir a chave única pela rede, e ela vem de uma fonte muito improvável: a mecânica quântica. Essa área ainda é experimental, mas os testes iniciais são promissores. Se eles puderem ser aperfeiçoados e se tornarem eficientes, quase toda a criptografia será realizada eventualmente com a utilização de chaves únicas, pois elas talvez sejam seguras. A seguir, explicaremos em linhas gerais como funciona esse método, denominado criptografia quântica. Em particular, descreveremos um protocolo chamado BB84 para indicar seus autores e o ano da publicação (Bennet e Brassard, 1984).

Uma usuária chamada Alice quer estabelecer uma chave única com um segundo usuário, Bob. Alice e Bob são chamados protagonistas, os personagens principais de nossa história. Por exemplo, Bob é um banqueiro com quem Alice gostaria de realizar negócios. Os nomes "Alice" e "Bob" foram usados como protagonistas em praticamente todos os ensaios e livros sobre criptografia na última década. Os criptógrafos amam a tradição. Se fôssemos usar "Andy" e "Barbara" como protagonistas, ninguém acreditaria em nada do que fosse explicado neste capítulo. Então, que seja!

Se Alice e Bob pudessem estabelecer uma chave única, eles teriam a possibilidade de empregá-la para se comunicarem com segurança. A pergunta é: como eles podem estabelecê-la sem anteriormente troca DVDs? Podemos supor que Alice e Bob estão em extremidades opostas de um cabo de fibra óptica pelo qual podem enviar e receber pulsos de luz. Porém, uma intrépida intrusa chamada Trudy pode cortar a fibra e criar um grampo ativo. Trudy pode ler todos os bits em ambos os sentidos. Ela também pode enviar falsas mensagens nos dois sentidos. A situação pode parecer desesperada para Alice e Bob, mas a criptografia quântica pode trazer uma nova luz sobre o assunto.

A criptografia quântica se baseia no fato de que a luz se propaga em pequenos pacotes chamados fôtons, que apresentam algumas propriedades peculiares. Além disso, a luz pode ser polarizada ao passar por um filtro de polarização, um fato bem conhecido para os usuários de óculos de sol e fotógrafos. Se um feixe de luz (isto é, um fluxo de fôtons) passar por um filtro de polarização, todos os fôtons que emergirem dele serão polarizados na direção do eixo do filtro (por exemplo, vertical).

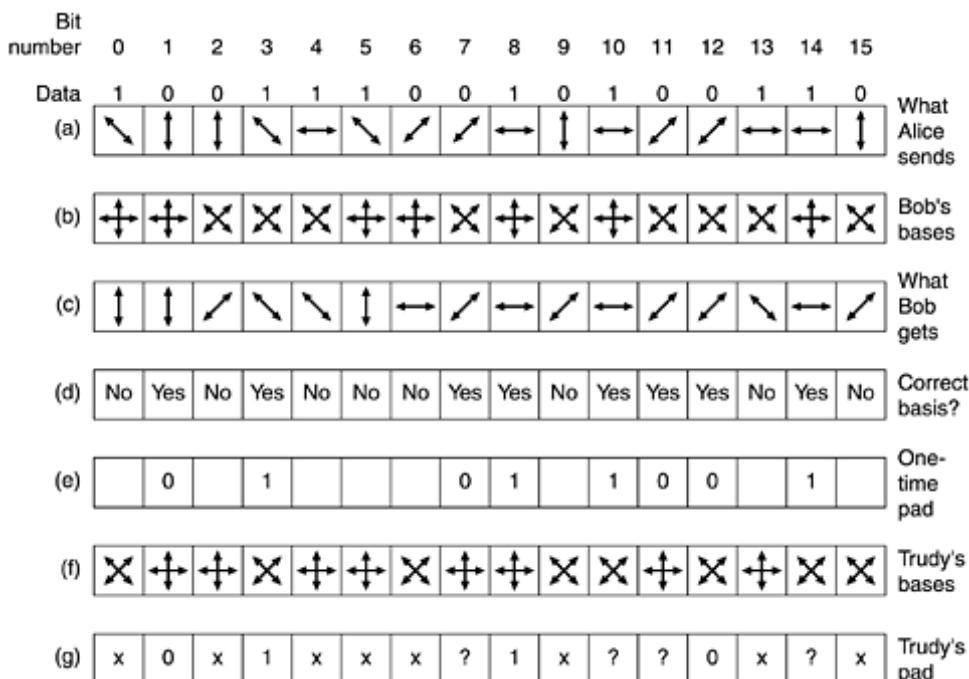
Se o feixe passar agora por um segundo filtro de polarização, a intensidade da luz que emergirá do segundo filtro será proporcional ao quadrado do cosseno do ângulo entre os eixos. Se os dois eixos forem perpendiculares, nenhum fóton passará pelo filtro. A orientação absoluta dos dois filtros não importa; só interessa o ângulo entre seus eixos.

Para gerar uma chave única, Alice precisa de dois conjuntos de filtros de polarização. O primeiro conjunto consiste em um filtro vertical e um filtro horizontal. Essa escolha é chamada **base retilínea**. Uma base é apenas um sistema de coordenadas. O segundo conjunto de filtros é idêntico, exceto por estar deslocado 45 graus, de forma que um filtro abrange desde o canto inferior esquerdo até o canto superior direito, e o outro filtro abrange desde o canto superior esquerdo até o canto inferior direito. Essa escolha é chamada **base diagonal**. Desse modo, Alice tem duas bases, que ela pode inserir rapidamente em seu feixe à vontade. Na realidade, Alice não tem quatro filtros separados, mas um cristal, cuja polarização pode ser trocada eletricamente para qualquer das quatro direções permitidas, em alta velocidade. Bob tem o mesmo equipamento de Alice. O fato de Alice e Bob terem cada um duas bases disponíveis é essencial para a criptografia quântica.

Para cada base, Alice atribui agora uma direção como 0 e a outra como 1. No exemplo apresentado a seguir, supomos que ela escolhe a direção vertical como 0 e a horizontal como 1. Independentemente, ela também escolhe do canto inferior esquerdo até o canto superior direito como 0, e do canto superior esquerdo até o canto inferior direito como 1. Alice envia essas escolhas a Bob como texto simples.

Agora, Alice escolhe uma chave única, por exemplo com base em um gerador de números aleatórios (um assunto por si só bastante complexo). Ela o transfere bit por bit para Bob, escolhendo uma de suas bases ao acaso para cada bit. Para enviar um bit, sua pistola de fôtons emite um fóton polarizado de maneira apropriada, conforme a base que ela está usando para esse bit. Por exemplo, ela poderia escolher as bases diagonal, retilínea, retilínea, diagonal, retilínea etc. Para enviar sua chave única igual a 1001110010100110 com essas bases, ela enviaria os fôtons mostrados na Figura 8.5(a). Dada a chave única e a seqüência de bases, a polarização a ser usada para cada bit é determinada de forma exclusiva. Bits enviados um fóton de cada vez são chamados qubits.

Figura 8.5: Um exemplo de criptografia quântica



Bob não sabe que base usar, e assim escolhe uma base ao acaso para cada fóton que chega e simplesmente o utiliza, como mostra a Figura 8.5(b). Se escolher a base correta, ele receberá o bit correto. Se escolher a base incorreta, ele receberá um bit aleatório porque, se um fóton acessar um filtro polarizado a 45 graus em relação à sua própria polarização, ele saltará ao acaso para a polarização do filtro ou para uma polarização perpendicular à do filtro, com igual probabilidade. Essa propriedade dos fôtons é fundamental para a mecânica quântica. Desse modo, alguns bits

estão corretos e alguns são aleatórios, mas Bob não consegue distingui-los. Os resultados de Bob estão representados na Figura 8.5(c).

De que maneira Bob pode descobrir quais são as bases corretas e quais são as erradas entre as que recebeu? Ele simplesmente diz a Alice que base usou para cada bit em texto simples, e elanhe diz quais são as bases corretas e quais são as erradas em texto simples, como mostra a Figura 8.5(d). A partir dessas informações, ambos podem construir um string de bits com os palpites corretos, como mostra a Figura 8.5(e). Em média, esse string de bits terá metade do comprimento do string de bits original mas, como ambas as partes o conhecem, elas poderão usá-lo como uma chave única. Tudo que Alice tem a fazer é transmitir um string de bits um pouco maior que o dobro do tamanho desejado, para que ela e Bob tenham uma chave única com o comprimento apropriado. Problema resolvido.

Porém, espere um minuto. Esquecemos de Trudy. Vamos supor que ela esteja curiosa para saber o que Alice tem a dizer e corte o cabo de fibra, inserindo seu próprio detector e transmissor. Infelizmente para Trudy, ela também não sabe que base usar para cada fóton. O melhor que ela pode fazer é escolher uma base ao acaso para cada um dos fótons, como fez Bob. Um exemplo de suas escolhas é mostrado na Figura 8.5(f). Quando mais tarde Bob informar (em texto simples) que bases usou e Alice disser a ele (em texto simples) quais delas estão corretas, Trudy saberá quando acertou e quando errou. Na Figura 8.5, ela acertou nos bits 0, 1, 2, 3, 4, 6, 8, 12 e 13. No entanto, ela sabe pela resposta de Alice na Figura 8.5(d) que só os bits 1, 3, 7, 8, 10, 11, 12 e 14 fazem parte da chave única. Em quatro desses bits (1, 3, 8 e 12), ela acertou seu palpite e captou o bit correto. Nos outros quatro (7, 10, 11 e 14), ela errou e não sabe qual bit foi transmitido. Desse modo, Bob sabe que a chave única começa com 01011001, a partir da Figura 8.5(e), mas tudo que Trudy tem é 01?1??0?, a partir da Figura 8.5(g).

É claro que Alice e Bob estão cientes de que Trudy talvez tenha captado parte de sua chave única, e assim gostariam de reduzir as informações que Trudy tem. Eles podem fazer isso executando uma transformação na chave. Por exemplo, poderiam dividir a chave única em blocos de 1024 bits e elevar ao quadrado cada uma para formar um número de 2048 bits, usando a concatenação desses números de 2048 bits como a chave única. Com seu conhecimento parcial do string de bits transmitido, Trudy não tem como gerar seu quadrado e, portanto, não tem nada. A transformação da chave única original em uma chave diferente que reduz o conhecimento de Trudy é chamada **amplificação da privacidade**. Na prática, são usadas transformações complexas em que todo bit de entrada depende de cada bit de saída em lugar da elevação ao quadrado.

Pobre Trudy. Ela não apenas não tem nenhuma idéia de qual é a chave única, mas sua presença não é mais secreta. Afinal, ela tem de retransmitir cada bit recebido para Bob, a fim de levá-lo a pensar que está se comunicando com Alice. Porém, o melhor que ela pode fazer é transmitir o qubit que recebeu, usando a mesma polarização que empregou para recebê-lo, e durante cerca de metade do tempo ela estará errada, provocando muitos erros na chave única de Bob.

Quando finalmente começar a transmitir dados, Alice os codificará usando um pesado código de correção antecipada de erros. Do ponto de vista de Bob, um erro de 1 bit na chave única é o mesmo que um erro de transmissão de 1 bit. De qualquer modo, ele receberá o bit errado. Se houver correção antecipada de erros suficiente, ele poderá recuperar a mensagem original apesar de todos os erros, mas poderá contar com facilidade quantos erros foram corrigidos. Se esse número for muito maior que a taxa de erros esperada do equipamento, ele saberá que Trudy grampeou a linha e poderá agir de acordo (por exemplo, informando a Alice que ela deve mudar para um canal de rádio, chamar a polícia etc.). Se Trudy tivesse um meio de clonar um fóton, de forma que ela tivesse um fóton para inspecionar e um fóton idêntico para enviar a Bob, ela poderia evitar a detecção mas, no momento, não se conhece nenhum modo perfeito de clonar um fóton. No entanto, mesmo que Trudy pudesse clonar fótons, o valor da criptografia quântica para estabelecer chaves únicas não seria reduzido.

Embora a criptografia quântica opere sobre distâncias de até 60 km de fibra, o equipamento é complexo e dispendioso. Ainda assim, a idéia é promissora. Para obter mais informações sobre a criptografia quântica, consulte (Mullins, 2002).

8.1.5 Dois princípios fundamentais da criptografia

Ainda estudaremos muitos sistemas criptográficos diferentes nas próximas páginas, mas é importante entender dois princípios básicos subjacentes a todos eles.

Redundância

O primeiro princípio é que todas as mensagens criptografadas devem conter alguma redundância, ou seja, informações que não são necessárias para a compreensão da mensagem. Talvez um exemplo esclareça por que isso é necessário. Considere uma empresa de encomendas postais, a The Couch Potato (TCP), com 60.000 produtos. Pensando que estavam sendo muito eficientes, os programadores da TCP decidiram que as mensagens de encomendas deveriam consistir no nome do cliente com 16 bytes, seguido por um campo de dados de 3 bytes (um para a quantidade e 2 para o número do produto). Os 3 últimos bytes devem ser criptografados por meio de uma chave muito longa conhecida apenas pelo cliente e pela TCP.

Em princípio, essa estratégia pode parecer segura, e até certo ponto isso acontece, porque os intrusos passivos não podem descriptografar as mensagens. Infelizmente, há uma falha fatal que a torna inútil. Suponha que uma funcionária recém-demitida queira punir a TCP por despedi-la. Antes de sair da empresa, ela leva consigo parte da lista de clientes e passa a noite acordada criando um programa para gerar encomendas fictícias utilizando nomes de clientes verdadeiros. Como não tem a lista das chaves, ela simplesmente inclui números aleatórios nos três últimos bytes e envia centenas de encomendas para a TCP.

Quando as mensagens chegam, o computador da TCP utiliza o nome do cliente para localizar a chave e descriptografar a mensagem. Infelizmente para a TCP, quase todas as mensagens de 3 bytes são válidas; portanto, o computador começa a imprimir as instruções de entrega. Apesar de parecer estranho um cliente encomendar 837 conjuntos de balões para crianças, ou 540 caixas de areia, para o computador, o cliente pode estar planejando abrir uma cadeia de parques de diversões franqueados. Portanto, um intruso ativo (a ex-funcionária) pode causar muitos problemas, mesmo que não seja capaz de entender as mensagens que seu computador está gerando.

Esse problema pode ser resolvido através da inclusão de informações redundantes em todas as mensagens. Por exemplo, se as mensagens de pedidos forem ampliadas para 12 bytes, os 9 primeiros deverão ser iguais a zero; assim, essa estratégia de ataque deixa de ser interessante, porque a ex-funcionária não é mais capaz de gerar um longo fluxo de mensagens válidas. A moral da história é que todas as mensagens devem conter informações redundantes suficientes para que os intrusos ativos sejam impedidos de transmitir dados inválidos que possam ser interpretados como uma mensagem válida.

No entanto, a inclusão de informações redundantes também facilita a ruptura de mensagens por parte dos criptoanalistas. Suponha que a empresa de encomenda postal seja muito competitiva e esteja na posição de principal concorrente da The Couch Potato. A Sofa Tuber adoraria saber quantas caixas de areia a TCP está vendendo. Portanto, a empresa resolve grampear a linha telefônica da TCP. No esquema original com mensagens de 3 bytes, a criptoanálise era praticamente impossível porque, após descobrir uma chave, o criptoanalista não era capaz de dizer se a mensagem estava correta. Afinal de contas, quase todas as mensagens são tecnicamente válidas. Com o novo esquema de 12 bytes, fica mais fácil para o criptoanalista distinguir uma mensagem válida de uma inválida. Desse modo, temos:

Princípio criptográfico 1 : as mensagens devem conter alguma redundância

Em outras palavras, ao decifrar uma mensagem, o destinatário deve ser capaz de saber se ela é válida, simplesmente inspecionando-a e talvez executando uma computação simples. Essa redundância é necessária para impedir que intrusos ativos enviem lixo e enganem o receptor, fazendo-o descriptografar o lixo e agir sobre o "texto simples". No entanto, essa mesma redundância permite que os intrusos passivos entrem no sistema com maior facilidade; portanto, há uma zona de tensão nessa situação. Além disso, a redundância nunca deverá ser criada sob a forma de n zeros no início ou no fim de uma mensagem, pois a submissão dessas mensagens a determinados algoritmos criptográficos proporciona resultados mais previsíveis, facilitando o trabalho do criptoanalista. Um polinômio de CRC é muito melhor que uma seqüência de valores 0, pois o receptor pode verificar facilmente, mas ele irá gerar mais trabalho para o criptoanalista. Seria muito melhor usar um hash criptográfico, um conceito que exploraremos mais adiante.

Voltando à criptografia quântica por um momento, também podemos ver como a redundância desempenha um papel importante. Devido à interceptação dos fótons por Trudy, alguns bits na chave única de Bob estarão errados. Bob precisa de alguma redundância nas mensagens de entrada para descobrir os erros presentes. Uma forma muito rudimentar de redundância é repetir a

mensagem duas vezes. Se as duas cópias não forem idênticas, Bob saberá que a fibra está muito ruidosa, ou que alguém está interferindo na transmissão. É claro que enviar tudo duas vezes é um exagero; um código de Hamming ou de Reed-Solomon é um modo mais eficiente de realizar a detecção e correção de erros. Porém, deve ficar claro que uma certa redundância é necessária para distinguir uma mensagem válida de uma mensagem inválida, em especial diante de um intruso ativo.

Atualidade

O segundo princípio criptográfico é tomar algumas medidas para assegurar que cada mensagem recebida possa ser confirmada como uma mensagem atual, isto é, enviada muito recentemente. Essa medida é necessária para impedir que intrusos ativos reutilizem mensagens antigas. Se tais medidas não fossem tomadas, nossa ex-funcionária poderia interceptar a linha telefônica da TCP e ficar simplesmente repetindo mensagens válidas já enviadas. Em outras palavras, essa idéia nos diz que:

Princípio criptográfico 2: algum método é necessário para anular ataques de repetição

Uma medida desse tipo seria incluir em cada mensagem um timbre de hora válido apenas por, digamos, 10 segundos. Em seguida, o receptor poderia manter as mensagens durante 10 segundos, a fim de comparar as mensagens recém-chegadas com as anteriores e filtrar duplicatas. As mensagens transmitidas há mais de 10 segundos poderiam ser descartadas, pois as repetições enviadas mais de 10 segundos depois da mensagem original serão rejeitadas por serem muito antigas. Outras medidas além dos timbres de hora serão discutidas mais adiante.

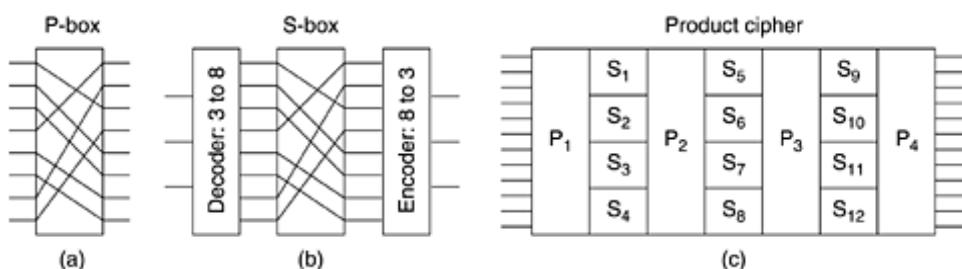
8.2 Algoritmos de chave simétrica

Embora a criptografia moderna utilize as mesmas idéias básicas da criptografia tradicional (transposição e substituição), sua ênfase é diferente. Tradicionalmente, as pessoas que criam a criptografia têm utilizado algoritmos simples. Hoje em dia, acontece o inverso: o objetivo é tornar o algoritmo de criptografia tão complexo e emaranhado que, mesmo que o criptoanalista adquira enormes volumes de texto cifrado de sua própria escolha, sem a chave ele não seja capaz de captar qualquer sentido em tudo que conseguir.

A primeira classe de algoritmos de criptografia que estudaremos neste capítulo é a dos **algoritmos de chave simétrica**, porque utilizam a mesma chave para codificação e decodificação. A Figura 8.2 ilustra o uso de um algoritmo de chave simétrica. Em particular, vamos nos concentrar nas **cifras de bloco**, que obtêm um bloco de n bits de texto simples como entrada e o transformam usando a chave em um bloco de n bits de texto cifrado.

Os algoritmos criptográficos podem ser implementados em hardware (para se obter velocidade) ou em software (para se obter flexibilidade). Embora a maior parte de nosso tratamento esteja relacionado aos algoritmos e protocolos, que são independentes da implementação real, algumas palavras sobre a construção de hardware criptográfico podem ser interessantes. As transposições e substituições podem ser implementadas com circuitos elétricos simples. A Figura 8.6(a) mostra um dispositivo, conhecido como **caixa P** (onde P significa permutação), usado para efetuar uma transposição em uma entrada de 8 bits. Se os 8 bits forem designados de cima para baixo como 01234567, a saída dessa caixa P específica será 36071245. Com uma fiação interna adequada, pode-se criar uma caixa P para executar qualquer transposição praticamente na velocidade da luz, pois nenhuma computação é envolvida, apenas a propagação e sinais. Esse projeto segue o princípio de Kerckhoff: o atacante sabe que o método geral é permutar os bits. O que ele não sabe é qual bit fica em cada posição, e isso é a chave.

Figura 8.6: Elementos básicos de cifras de produtos. (a) Caixa P. (b) Caixa S. (c) Produto



As substituições são realizadas por **caixas S**, como mostra a Figura 8.6(b). Nesse exemplo, é introduzido um texto simples de 3 bits, e a saída é um texto cifrado de 3 bits. A entrada de 3 bits seleciona uma das oito linhas de saída do primeiro estágio e a define como 1; todas as outras são iguais a 0. O segundo estágio é uma caixa P. O terceiro estágio codifica a linha selecionada novamente em binário. Com a fiação mostrada, se os oito números octais 01234567 fossem introduzidos um após o outro, a seqüência de saída seria 24506713. Em outras palavras, 0 foi substituído por 2, 1 foi substituído por 4 etc. Mais uma vez, com a fiação apropriada da caixa P dentro da caixa S, qualquer substituição pode ser realizada. Além disso, tal dispositivo pode ser construído em hardware e pode alcançar grande velocidade, pois os codificadores e os decodificadores têm apenas um ou dois retardos de porta (subnanosegundo) e o tempo de propagação pela caixa P pode ser menor que 1 picosegundo.

A capacidade real desses elementos básicos se torna aparente quando dispomos uma série inteira de caixas em cascata para formar uma **cifra de produto**, como mostra a Figura 8.6(c). Nesse exemplo, 12 linhas de entrada são transpostas (isto é, permutadas) pelo primeiro estágio (P1). Teoricamente, seria possível fazer com que o segundo estágio fosse uma caixa S que mapeasse um número de 12 bits em outro número de 12 bits. No entanto, tal dispositivo necessitaria de $2^{12} = 4096$ fios cruzados em seu estágio intermediário. Em vez disso, a entrada é dividida em quatro grupos de 3 bits, sendo que cada um deles é substituído de forma independente dos outros. Apesar de ser menos genérico, esse método ainda é mais eficiente. Através da inclusão de um número de estágios suficientemente grande na cifra de produto, a saída pode ser transformada em uma função excessivamente complicada da entrada.

As cifras de produto que operam sobre entradas de k bits para produzir saídas de k bits são muito comuns. Em geral, o valor de k varia de 64 a 256. Uma implementação de hardware normalmente tem pelo menos 18 estágios físicos, em vez de apenas sete, como na Figura 8.6(c). Uma implementação de software é programada como um loop com pelo menos 8 iterações, cada uma executando substituições semelhantes às de caixas S em sub-blocos do bloco de dados de 64 bits a 256 bits, seguidas por uma permutação que mistura as saídas das caixas S. Com frequência, existe uma permutação especial no início e também uma no fim. Na literatura, as repetições são chamadas rodadas.

8.2.1 DES — Data Encryption Standard

Em janeiro de 1977, o governo dos Estados Unidos adotou uma cifra de produto desenvolvida pela IBM como seu padrão oficial para informações não confidenciais. A cifra, **DES (Data Encryption Standard — padrão de criptografia de dados)**, foi amplamente adotada pelo setor de informática para uso em produtos de segurança. Em sua forma original, ela já não é mais segura; no entanto, em uma forma modificada ela ainda é útil. Agora, vamos explicar como o DES funciona.

Na Figura 8.7(a), é mostrado um esboço do DES. O texto simples é criptografado em blocos de 64 bits, produzindo 64 bits de texto cifrado. O algoritmo, parametrizado por uma chave de 56 bits, tem 19 estágios distintos. O primeiro deles é uma transposição independente da chave no texto simples de 64 bits. O último estágio é exatamente o inverso dessa transposição. O penúltimo estágio troca os 32 bits mais à esquerda pelos 32 bits mais à direita. Os 16 estados restantes são funcionalmente idênticos, mas são parametrizados por diferentes funções da chave. O algoritmo foi projetado para permitir que a decodificação fosse feita com a mesma chave da codificação, uma propriedade necessária em qualquer algoritmo de chave simétrica. As etapas são simplesmente executadas na ordem inversa.

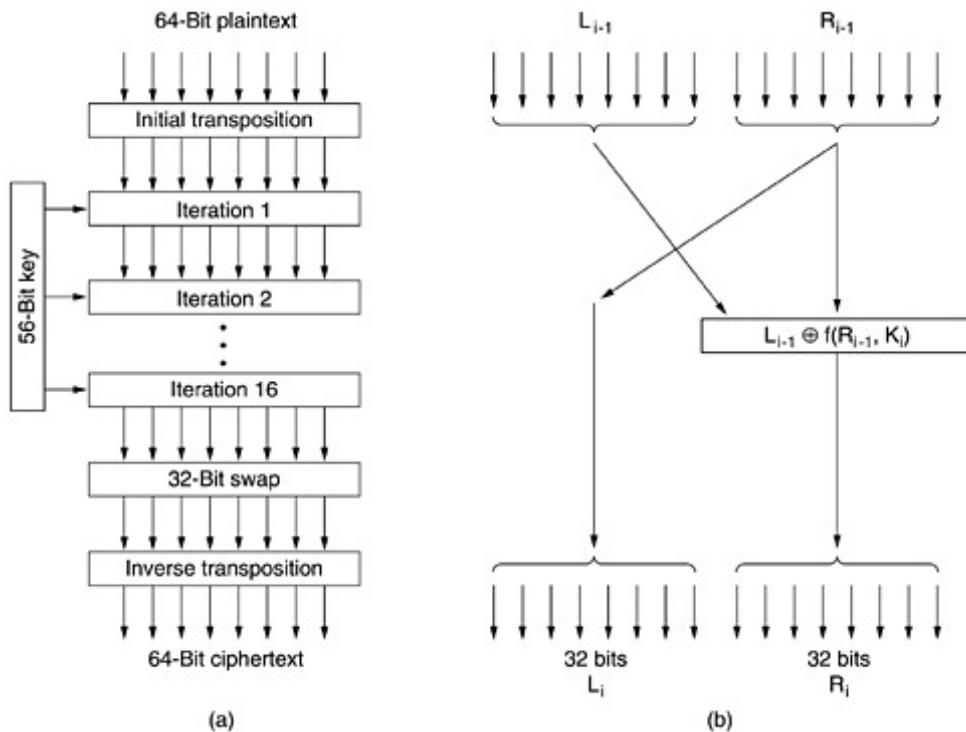
A operação desses estágios intermediários é ilustrada na Figura 8.7(b). Cada estágio utiliza duas entradas de 32 bits e produz duas saídas de 32 bits. A saída da esquerda é apenas uma cópia da saída da direita. A saída da direita é formada pelo resultado do OR exclusivo bit a bit aplicado à entrada da esquerda e a uma função da entrada da direita com a chave desse estágio, K_i . Toda a complexidade reside nessa função.

A função consiste em quatro etapas, executadas em seqüência. Primeiro, um número de 48 bits, E, é construído através da expansão do R_{i-1} de 32 bits, de acordo com uma regra fixa de transposição e duplicação. Em segundo lugar, E e K_i são submetidos a uma operação XOR. Em seguida, essa saída é particionada em oito grupos de 6 bits, sendo cada um deles entregue a uma caixa S diferente. Cada uma das 64 entradas possíveis para uma caixa S é mapeada em uma saída de 4 bits. Por fim, esses 4 bits passam por uma caixa P.

Em cada uma das 16 iterações, é utilizada uma chave diferente. Antes de se iniciar o algoritmo, é aplicada à chave uma transposição de 56 bits. Antes de cada iteração, a chave é particionada em duas unidades de 28 bits, sendo cada uma delas girada à esquerda um número de bits que depende do número da iteração.

Ki é derivada dessa chave girada, pela aplicação de mais uma transposição de 56 bits sobre ela. Em cada rodada, um subconjunto de 48 bits dos 56 bits é extraído e permutado.

Figura 8.5: O DES (Data Encryption Standard — padrão de criptografia de dados). (a) Esboço geral. (b) Detalhe de uma iteração. O sinal de adição dentro do círculo significa OR exclusivo (XOR)



Uma técnica às vezes utilizada para tornar o DES mais forte é chamada **branqueamento**. Ela consiste em operação XOR entre uma chave aleatória de 64 bits e cada bloco de texto simples, antes de sua entrega ao DES e depois uma operação XOR entre uma segunda chave de 64 bits e o texto cifrado resultante, antes de sua transmissão. O branqueamento pode ser removido com facilidade pela execução das operações inversas (se o receptor tiver as duas chaves de branqueamento). Tendo em vista que essa técnica acrescenta efetivamente mais bits ao tamanho da chave, ela torna uma pesquisa exaustiva do espaço de chaves muito mais demorada. Observe que a mesma chave de branqueamento é utilizada para cada bloco (isto é, só existe uma chave de branqueamento).

O DES esteve envolvido em controvérsias desde seu lançamento. Ele se baseia em uma cifra desenvolvida e patenteada pela IBM, cujo nome é Lucifer. A diferença é que a cifra da IBM utilizava uma chave de 128 bits, em vez de uma chave de 56 bits. Quando quis padronizar o uso de uma cifra para informações não confidenciais, o governo dos Estados Unidos "convidou" a IBM para "discutir" o problema com a NSA, o órgão do governo especializado em decifrar códigos, que é o maior empregador de matemáticos e criptólogos do mundo. A NSA é tão secreta que no setor existe a seguinte brincadeira com seu nome:

P: O que significa NSA?

R: No Such Agency (em português: não existe tal agência).

Na verdade, NSA significa National Security Agency.

Após essas discussões, a IBM reduziu a chave de 128 para 56 bits e decidiu manter em segredo o processo segundo o qual o DES foi projetado. Muita gente suspeitou de que o tamanho da chave foi reduzido para garantir que a NSA pudesse decifrar o DES, mas nenhuma organização com um orçamento menor foi de fazê-lo. Supostamente, o motivo para manter o projeto em segredo foi ocultar uma porta dos fundos (back door) que pudesse facilitar ainda mais a decifração do DES por parte da NSA. Quando um funcionário da NSA disse discretamente para o IEEE cancelaria uma

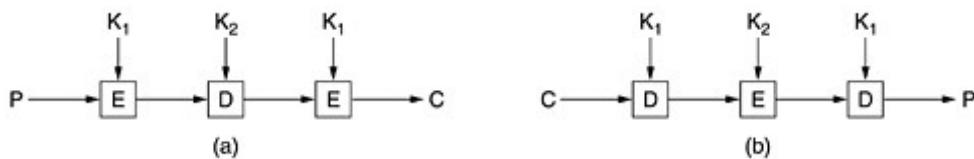
conferência sobre criptografia, as pessoas ficaram ainda mais desconfortáveis com a situação. A NSA negou tudo.

Em 1977, dois pesquisadores de criptografia de Stanford, Diffie e Hellman (1977), projetaram uma máquina para decifrar o DES e estimaram que ela poderia ser montada por um custo de 20 milhões de dólares. Com base em um pequeno trecho de texto simples e no texto cifrado correspondente, essa máquina poderia descobrir a chave através de uma pesquisa exaustiva do espaço de chaves de 2^{56} entradas em menos de 1 dia. Atualmente, essa máquina custaria bem menos de 1 milhão de dólares.

DES triplo

No início de 1979, a IBM percebeu que o tamanho da mensagem DES era muito pequeno e criou uma forma de aumentá-lo usando a criptografia tripla (Tuchman, 1979). O método escolhido, que desde então foi incorporado ao padrão internacional 8732, está ilustrado na Figura 8.8. Nesse caso, são usados três estágios e duas chaves. No primeiro estágio, o texto simples é criptografado com K_1 da maneira usual do DES. No segundo estágio, o DES é executado no modo de descriptografia, com o uso de K_2 como chave. Por fim, outra criptografia é feita com K_1 .

Figura 8.8: (a) Criptografia tripla usando o DES. (b) Descriptografia



Esse projeto levanta duas questões. Primeiro, por que são utilizadas apenas duas chaves em vez de três? Segundo, por que foi usado EDE (Encrypt Decrypt Encrypt), em vez de EEE (Encrypt Encrypt Encrypt)? São utilizadas duas chaves porque até mesmo os criptógrafos mais paranoides concordam que 112 bits serão suficientes para aplicações comerciais durante um bom tempo. (Entre os criptógrafos, a paranoia é considerada um recurso, não um bug.) O uso de 168 bits só criaria overhead desnecessário de gerenciar e transportar outra chave, com pouco ganho real.

O motivo para criptografar, descriptografar e criptografar mais uma vez é a compatibilidade retroativa com os sistemas DES de chave única existentes. Tanto as funções de criptografia quanto as de descriptografia são mapeamentos entre conjuntos de números de 64 bits. Do ponto de vista da criptografia, os dois mapeamentos são igualmente fortes. No entanto, ao usar EDE em vez de EEE, um computador que utiliza a criptografia triple pode se comunicar com outro que utiliza a criptografia apenas definindo $K_1 = K_2$. Essa propriedade permite que a criptografia tripla seja ajustada gradualmente, o que não interessa aos criptógrafos acadêmicos, mas que é de grande importância para a IBM e seus clientes.

8.2.2 AES — Advanced Encryption Standard

À medida que o DES começou a se aproximar do fim de sua vida útil, mesmo com o DES triplo, o NIST (National Institute of Standards and Technology), o órgão do departamento de comércio dos Estados Unidos encarregado de aprovar padrões para o Governo Federal dos Estados Unidos, decidiu que o governo precisava de um novo padrão criptográfico para uso não confidencial. O NIST estava ciente de toda a controvérsia que cercava o DES e sabia muito bem que, se anunciasse um novo padrão, todas as pessoas que soubessem algo sobre criptografia concluiriam automaticamente que a NSA havia criado uma porta dos fundos no DES, e assim a NSA poderia ler tudo que fosse criptografado com ele. Sob essas condições, talvez ninguém utilizasse o padrão e seria mais provável que ele desaparecesse.

Dessa forma, o NIST adotou uma estratégia diferente e surpreendente para um órgão do governo: patrocinou um concurso de criptografia. Em janeiro de 1997, pesquisadores do mundo inteiro foram convidados a submeter propostas para um novo padrão, a ser chamado AES (Advanced Encryption Standard). As regras do concurso eram:

1. O algoritmo teria de ser uma cifra de bloco simétrica.
2. Todo o projeto teria de ser público.

3. Deveriam ser admitidos tamanhos de chaves iguais a 128, 192 e 256 bits.
4. Teriam de ser possíveis implementações de software e de hardware.
5. O algoritmo teria de ser público ou licenciado em termos não discriminatórios.

Foram feitas quinze propostas sérias e foram organizadas conferências públicas nas quais essas propostas eram apresentadas, e os participantes eram encorajados ativamente a encontrar falhas em todas elas. Em agosto de 1998, o NIST selecionou cinco finalistas, baseado principalmente em seus requisitos de segurança, eficiência, simplicidade, flexibilidade e memória (importante para sistemas incorporados). Foram realizadas outras conferências e mais tentativas de encontrar falhas nos algoritmos. Na última conferência, houve uma votação sem compromisso. Os finalistas e suas pontuações foram:

1. Rijndael (de Joan Daemen e Vincent Rijmen, 86 votos).
2. Serpent (de Ross Anderson, Eli Biham e Lars Knudsen, 59 votos).
3. Twofish (de uma equipe liderada por Bruce Schneier, 31 votos).
4. RC6 (da RSA Laboratories, 23 votos).
5. MARS (da IBM, 13 votos).

Em outubro de 2000, o NIST anunciou que também votou no Rijndael e, em novembro de 2001, o Rijndael se tornou um padrão do Governo dos Estados Unidos publicado como Federal Information Processing Standard FIPS 197. Devido à extraordinária abertura da competição, às propriedades técnicas do Rijndael e ao fato de que a equipe premiada consistia em dois jovens criptógrafos belgas (com pouca probabilidade de terem criado uma porta dos fundos só para agradar a NSA), esperava-se que o Rijndael se tornasse o padrão criptográfico dominante no mundo por pelo menos uma década. O nome Rijndael deriva dos sobrenomes dos autores: Rijmen + Daemen.

O Rijndael admite tamanhos de chaves e tamanhos de blocos desde 128 bits até 256 bits em intervalos de 32 bits. O comprimento da chave e o do bloco podem ser escolhidos independentemente. Porém, o AES especifica que o tamanho do bloco deve ser 128 bits e o comprimento da chave deve ser 128, 192 ou 256 bits. É pouco provável que alguém utilize chaves de 192 bits; assim, de fato, o AES tem duas variantes: um bloco de 128 bits com uma chave de 128 bits e um bloco de 128 bits com uma chave de 256 bits.

Em nosso tratamento do algoritmo apresentado a seguir, examinaremos apenas o caso de 128/128, porque é provável que esse se torne a norma comercial. Uma chave de 128 bits oferece um espaço de chaves de $2^{128} \approx 3 \times 10^{38}$ chaves. Ainda que o NSA consiga construir uma máquina com 1 bilhão de processadores paralelos, cada um capaz de avaliar uma chave por picosegundo, tal máquina levaria cerca de 10^{10} anos para pesquisar o espaço de chaves. Nessa época, o sol já terá explodido e as pessoas que sobreviverem terão de ler os resultados a luz de vela.

Rijndael

Sob uma perspectiva matemática, o Rijndael se baseia na teoria de campo de Galois, o que proporciona ao algoritmo algumas propriedades de segurança demonstráveis. Porém, ele também pode ser visto como código em C, sem a necessidade de entrarmos nos detalhes matemáticos.

Como o DES, o Rijndael utiliza substituição e permutações, e também emprega várias rodadas. O número de rodadas depende do tamanho da chave e do tamanho do bloco, sendo 10 para chaves de 128 bits com blocos de 128 bits, passando para 14 no caso da maior chave ou do maior bloco. No entanto, diferente do DES, todas as operações envolvem bytes inteiros, a fim de permitir implementações eficientes, tanto em hardware quanto em software. Um esboço do código é apresentado na Figura 8.9.

Figura 8.9: Um esboço do Rijndael

```

#define LENGTH 16           /* # bytes in data block or key */
#define NROWS 4             /* number of rows in state */
#define NCOLS 4             /* number of columns in state */
#define ROUNDS 10            /* number of iterations */
typedef unsigned char byte;    /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
    int r;                  /* loop index */
    byte state[NROWS][NCOLS]; /* current state */
    struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* round keys */

    expand_key(key, rk);      /* construct the round keys */
    copy_plaintext_to_state(state, plaintext); /* init current state */
    xor_roundkey_into_state(state, rk[0]); /* XOR key into state */

    for (r = 1; r <= ROUNDS; r++) {
        substitute(state);      /* apply S-box to each byte */
        rotate_rows(state);     /* rotate row i by i bytes */
        if (r < ROUNDS) mix_columns(state); /* mix function */
        xor_roundkey_into_state(state, rk[r]); /* XOR key into state */
    }
    copy_state_to_ciphertext(ciphertext, state); /* return result */
}

```

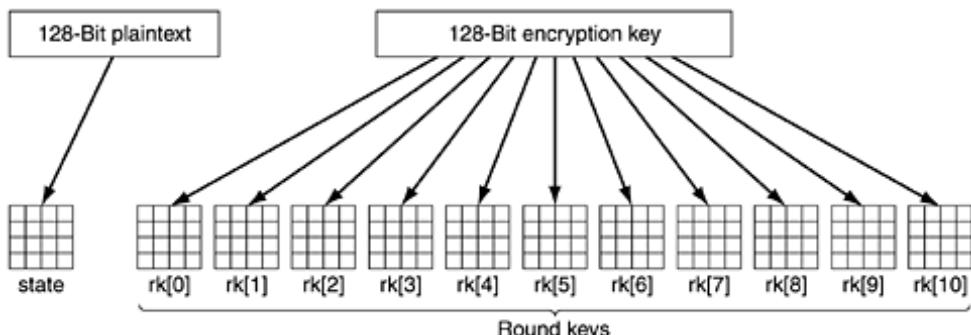
A função rijndael tem três parâmetros. Esses parâmetros são: plaintext, um array de 16 bytes contendo os dados de entrada, ciphertext, um array de 16 bytes no qual será retornada a saída cifrada e key, a chave de 16 bytes. Durante o cálculo, o estado atual dos dados é mantido no array de bytes state, cujo tamanho é NROWS NCOLS. No caso de blocos de 128 bits, esse array tem 4 4 bytes. Em 16 bytes, é possível armazenar todo o bloco de dados de 128 bits.

O array state é inicializado como o texto simples e modificado por cada etapa da computação. Em algumas etapas, é executada a substituição byte a byte. Em outras etapas, os bytes são permutados dentro do array. Outras transformações também são usadas. No final, o conteúdo do array state é retornado como texto cifrado.

O código começa expandindo a chave em 11 arrays do mesmo tamanho que o estado. Eles são armazenados em rk, um array de structs, cada uma contendo um array de estado. Um desses arrays será utilizado no início do cálculo, e os outros 10 serão usados durante as 10 rodadas, um por rodada. O cálculo das chaves de rodadas a partir da chave de criptografia é muito complicado para ser examinado aqui. Basta saber que as chaves de rodadas são produzidas pela rotação e pela operação XOR repetida de vários grupos de bits da chave. Para ver todos os detalhes, consulte (Daemen e Rijmen, 2002).

A próxima etapa é copiar o texto simples no array state de forma que ele possa ser processado durante as rodadas. O texto é copiado em ordem de colunas, com os quatro primeiros bytes na coluna 0, os quatro bytes seguintes na coluna 1 e assim por diante. Tanto as colunas quanto as linhas são numeradas a partir de 0, embora as rodadas se iniciem em 1. Essa configuração inicial dos 12 arrays de bytes com tamanho 4 4 é ilustrada na Figura 8.10.

Figura 8.10: Criação dos arrays state e rk



Há mais uma etapa antes do início da principal computação: rk[0] é submetido a uma operação XOR em state byte por byte. Em outras palavras, cada um dos 16 bytes em state é substituído pelo XOR do próprio valor com o byte correspondente em rk[0].

Agora chegou a hora da atração principal. O loop executa 10 repetições, uma por rodada, transformando state em cada repetição. O conteúdo de cada redonda consiste em quatro etapas. A etapa 1 efetua uma substituição byte a byte em state. Por sua vez, cada byte é usado como um índice para uma caixa S, a fim de substituir seu valor pelo conteúdo dessa entrada da caixa S. Essa etapa é uma cifra de substituição monoalfabética. Diferente do DES, que tem diversas caixas S, o Rijndael tem apenas uma caixa S.

A etapa 2 gira cada uma das quatro linhas para a esquerda. A linha 0 é girada 0 bytes (isto é, não se altera), a linha 1 é girada 1 byte, a linha 2 é girada 2 bytes e a linha 3 é girada 3 bytes. Essa etapa difunde o conteúdo dos dados atuais em torno do bloco, de modo análogo às permutações da Figura 8.6.

A etapa 3 mistura cada coluna independentemente das outras. A mistura é realizada com o uso da multiplicação de matrizes, na qual a nova coluna é o produto da coluna antiga por uma matriz constante, sendo a multiplicação feita com o campo finito de Galois, GF(28). Embora isso possa parecer complicado, existe um algoritmo que permite calcular cada elemento da nova coluna usando duas pesquisas de tabelas e três operações XOR (Daemen e Rijmen, 2002, Apêndice E).

Finalmente, a etapa 4 efetua o XOR da chave correspondente a essa rodada no array state.

Tendo em vista que cada etapa é reversível, a decodificação pode ser feita simplesmente executando-se o algoritmo no sentido inverso. Porém, também existe um artifício, pelo qual a decodificação pode ser realizada executando-se o algoritmo de criptografia com a utilização de tabelas diferentes.

O algoritmo foi projetado não só por segurança, mas também para aumentar a velocidade. Uma boa implementação de software em uma máquina de 2 GHz deve ser capaz de alcançar uma taxa de criptografia de 700 Mbps, que é rápida o suficiente para codificar mais de 100 vídeos MPEG-2 em tempo real. As implementações de hardware são ainda mais rápidas.

8.2.3 Modos de cifra

Apesar de toda essa complexidade, o AES (ou o DES, ou ainda qualquer cifra de bloco) é basicamente uma cifra de substituição monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para AES, e caracteres de 64 bits para DES). Sempre que o mesmo bloco de texto simples chega ao front end, o mesmo bloco de texto cifrado sai pelo back end. Se codificar o texto simples abcdefgh 100 vezes com a mesma chave DES, você obterá o mesmo texto cifrado 100 vezes. Um intruso pode explorar essa propriedade para ajudar a subverter a cifra.

O modo Electronic Code Book

Para ver como essa propriedade das cifras de substituição monoalfabética podem ser usadas para anular parcialmente a cifra, usaremos o DES (triplo), por ser mais fácil representar blocos de 64 bits que blocos de 128 bits, mas o AES tem exatamente o mesmo problema. A maneira direta de usar o DES para codificar um longo fragmento de texto simples é dividi-lo em blocos consecutivos de 8 bytes (64 bits) e codificá-los uns após outros com a mesma chave. O último fragmento de texto simples é completado até 64 bits, se necessário. Essa técnica é conhecida como **modo ECB** (modo Electronic Code Book) em analogia aos antigos livros de código em que cada palavra de texto simples era listada, seguida por seu texto cifrado (em geral, um número decimal de cinco dígitos).

Na Figura 8.11, temos o início de um arquivo de computador listando as gratificações anuais que uma empresa decidiu oferecer a seus funcionários. Esse arquivo consiste em registros de 32 bytes consecutivos, um por funcionário, no formato mostrado: 16 bytes para o nome, 8 bytes para o cargo e 8 bytes para a gratificação. Cada um dos dezesseis blocos de 8 bytes (numerados de 0 até 15) é codificado pelo DES (triplo).

Figura 8.11: O texto simples de um arquivo codificado como 16 blocos DES

Name	Position	Bonus
Adams, Leslie	Clerk	\$100,000
Black, Robin	Boss	\$500,000
Collins, Kim	Manager	\$100,000
Davis, Bobbie	Janitor	\$5

Bytes ← 16 → 8 → 8 →

Leslie acabou de ter uma briga com o chefe e sabe que não deve esperar uma grande gratificação. Em contraste, Kim é a favorito do chefe e todo mundo sabe disso. Leslie pode acessar o arquivo após a codificação, mas antes do arquivo ser enviado ao banco. Leslie pode retificar essa situação injusta, tendo apenas o arquivo criptografado?

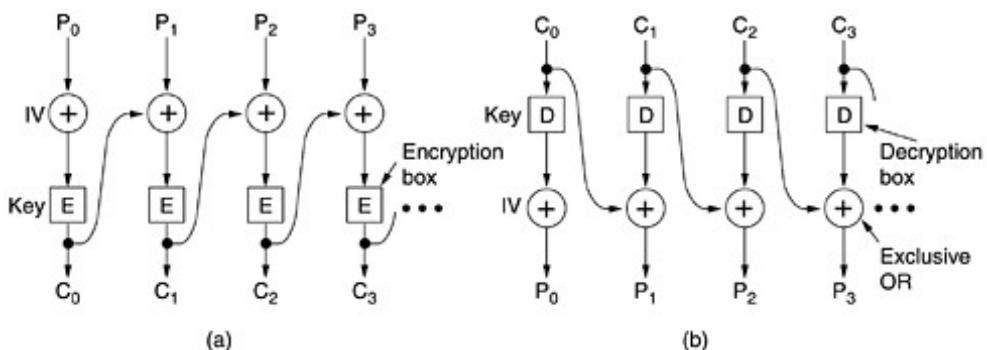
Não há problema. Leslie só precisa fazer uma cópia do 12º bloco de texto cifrado (que contém a gratificação de Kim) e usá-lo para substituir o 4º bloco de texto cifrado (que contém a gratificação de Leslie). Mesmo sem saber o que contém o 12º bloco, Leslie pode esperar ter um Natal muito mais feliz este ano. (Copiar o 8º bloco de texto cifrado também é uma possibilidade, mas a probabilidade de ser descoberto é maior; além disso, Leslie não é uma pessoa gananciosa.)

Modo de encadeamento de blocos de cífras

Para contrariar esse tipo de ataque, todos os cífras de blocos podem ser encadeadas de várias maneiras, para que a substituição de um bloco como o que Leslie fez transforme o texto simples decodificado em lixo, a partir do bloco substituído. Uma forma de encadeamento é o encadeamento de blocos de cífras. Nesse método, mostrado na Figura 8.12, cada bloco de texto simples é submetido a uma operação XOR com o bloco de texto cifrado anterior, antes de ser codificado. Consequentemente, o mesmo bloco de texto simples não é mais mapeado para o mesmo bloco de texto cifrado, e a criptografia não é mais uma grande cifra de substituição monoalfabética. O primeiro bloco é submetido a uma operação XOR com um IV (Initialization Vector — vetor de inicialização), escolhido ao acaso, que é transmitido (e m texto simples) juntamente com o texto cifrado.

Podemos ver como funciona o modo de encadeamento de blocos de cífras examinando o exemplo da Figura 8.12. Começamos calculando $C_0 = E(P_0 \text{ XOR } IV)$. Em seguida, calculamos $C_1 = E(P_1 \text{ XOR } C_0)$ e assim por diante. A decodificação também utiliza XOR para inverter o processo, com $P_0 = IV \text{ XOR } D(C_0)$ e assim por diante. Observe que a criptografia de bloco é uma função de todo o texto simples contidos nos blocos 0 a $i - 1$, e assim o mesmo texto simples gera um texto cifrado diferente, dependendo de onde ele ocorre. Uma transformação do tipo que Leslie fez resultará em texto sem sentido para dois blocos a partir do campo de gratificação de Leslie. Para um funcionário de segurança astuto, essa peculiaridade pode sugerir onde iniciar a investigação legal.

Figura 8.12: Encadeamento e blocos de cífras. (a) Codificação. (b) Decodificação



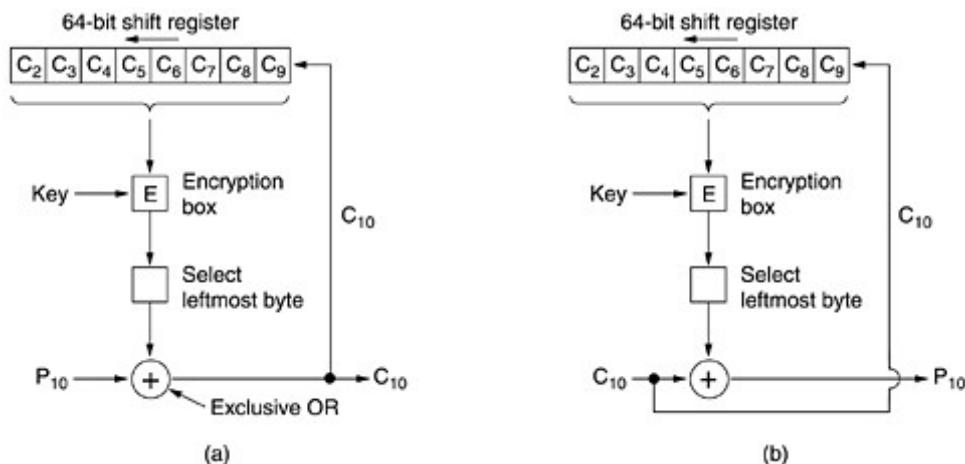
O encadeamento de blocos de cífras também tem uma vantagem: o mesmo bloco de texto simples não resultará no mesmo bloco de texto cifrado. Assim, a criptoanálise será difícil. De fato, essa é a principal razão de seu uso.

Modo de feedback de cífras

No entanto, o encadeamento de blocos de cifras tem a desvantagem de exigir a chegada de um bloco de 64 bits inteiro para poder iniciar a decodificação. Quando é utilizado em terminais interativo s, nos quais as pessoas podem digitar linhas com menos de oito caracteres e parar à espera de uma resposta, esse modo é inadequado. No caso da codificação byte a byte, é usado o modo de feedback de cifra, empregando o DES (triplo), como mostra a Figura 8.13. Para o AES, a idéia é exatamente a mesma, sendo usado um registrador de deslocamento de 128 bits. Na figura, o estado da máquina de criptografia é mostrado após os bytes 0 a 9 terem sido codificados e enviados. Ao chegar o byte 10 do texto simples, conforme ilustra a Figura 8.13(a), o algoritmo DES opera sobre o registrador de deslocamento de 64 bits para gerar um texto cifrado de 64 bits. O byte mais à esquerda desse texto cifrado é extraído e submetido a uma operação XOR com P_{10} . Depois, esse byte é encaminhado à linha de transmissão. Além disso, o registrador de deslocamento (shift register) é deslocado 8 bits à esquerda, fazendo C_2 ficar fora da extremidade esquerda, e C_{10} é inserido na posição que acabou de ficar vaga na extremidade direita, logo depois de C_9 . Observe que o conteúdo do registrador de deslocamento depende de todo o histórico anterior do texto simples; assim, um padrão que se repetir várias vezes no texto simples será criptografado de maneira diferente do texto cifrado a cada repetição. Como ocorre no encadeamento de blocos de cifras, é necessário um vetor de inicialização para dar início ao processo.

A decodificação com o modo de feedback de cifra funciona exatamente como a codificação. Em particular, o conteúdo do registrador de deslocamento é codificado e não decodificado, e assim o byte selecionado que é submetido à operação XOR com C_{10} para se obter P_{10} é o mesmo que sofreu a operação XOR com P_{10} para gerar C_{10} na primeira vez. Desde que os dois registradores de deslocamento permaneçam idênticos, a de codificação funcionar corretamente. Ela é ilustrada na Figura 8.13(b).

Figura 8.13: Modo de feedback de cifra. (a) Codificação. (b) Decodificação



O modo de feedback de cifra apresenta um problema: se um bit do texto cifrado for invertido acidentalmente durante a transmissão, os 8 bytes decodificados enquanto o byte defeituoso estiver no registrador de deslocamento serão danificados. Depois que o byte defeituoso é empurrado para fora do registrador de deslocamento, o texto simples correto será gerado mais uma vez. Desse modo, os efeitos de um único bit invertido são relativamente localizados e não arruinam o restante da mensagem, mas aruinam uma quantidade de bits igual à largura do registrador de deslocamento.

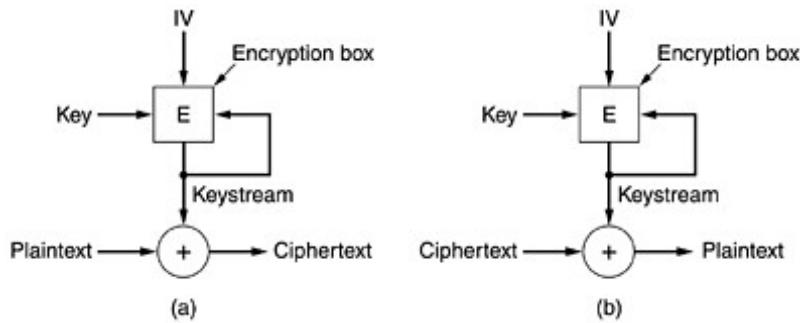
Modo de cifra de fluxo

Apesar disso, existem aplicações em que um erro de transmissão de 1 bit alterando 64 bits de texto simples provoca um impacto grande demais. Para essas aplicações, há uma quarta opção, o **modo de cifra de fluxo**. Ele funciona codificando um vetor de inicialização, com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para se obter um segundo bloco de saída. Em seguida, esse bloco é codificado para se obter um terceiro bloco e assim por diante. A seqüência (arbitrariamente grande) de blocos de saída, chamada **fluxo de chaves**, é tratada como uma chave única e submetida a uma operação XOR com o texto simples para se obter o texto cifrado, como mostra a Figura 8.14(a). Observe que o vetor de inicialização só é usado na primeira etapa. Depois disso, a saída é codificada. Observe também que o fluxo de chaves é independente dos dados, e portanto pode ser calculado com antecedência, se necessário,

e é completamente insensível a erros de transmissão. A decodificação é mostrada na Figura 8.14(b).

A decodificação ocorre gerando-se o mesmo fluxo de chaves no lado receptor. Como o fluxo de chaves só depende do vetor de inicialização e da chave, ele não é afetado por erros de transmissão no texto cifrado. Desse modo, um erro de 1 bit no texto cifrado transmitido gera apenas um erro de 1 bit no texto simples decodificado.

Figura 8.14: Uma cifra de fluxo. (a) Codificação. (b) Decodificação



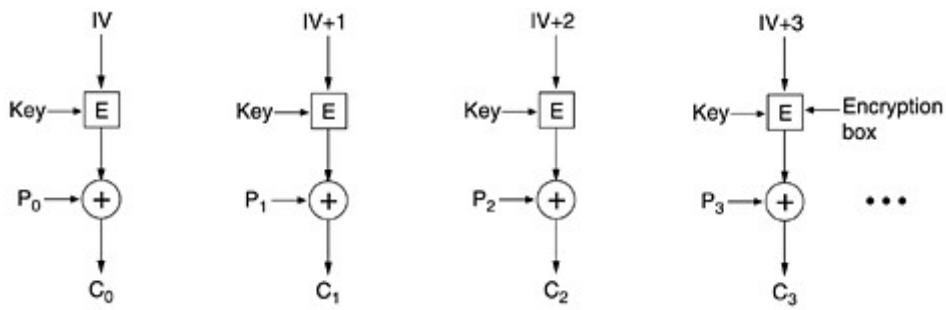
É essencial nunca utilizar o mesmo par (chave, IV) duas vezes com uma cifra de fluxo, porque isso irá gerar o mesmo fluxo de chaves o tempo todo. O uso de um mesmo fluxo de chaves duas vezes expõe o texto cifrado a um ataque de **reutilização de fluxo de chaves**. Imagine que o bloco de texto simples P0 seja codificado com o fluxo de chaves para se obter P0 XOR K0. Mais tarde, um segundo bloco de texto simples Q0 é codificado com o mesmo fluxo de chaves para se obter Q0 XOR K0. Um intruso que capturar ambos os blocos de texto cifrado poderá simplesmente efetuar um a operação XOR dos dois juntos para obter P0 XOR Q0, o que eliminará a chave. Agora, o intruso tem o XOR dos dois blocos de texto simples. Se um deles for conhecido ou puder ser encontrado, o outro também poderá ser encontrado. Em todo caso, o XOR de dois fluxos de texto simples poderá ser atacado com a utilização de propriedades estatísticas da mensagem. Por exemplo, no caso de text o em inglês, o caractere mais comum no fluxo provavelmente será o XOR de dois espaços, seguido pelo XOR de espaço e da letra "e" etc. Em resumo, equipado com o XOR de dois textos simples, o criptoanalista tem uma excelente chance de deduzi-los.

Modo de contador

Um problema apresentado por todos os modos, com exceção do modo de livro de código eletrônico, é a impossibilidade de conseguir acesso aleatório a dados codificados. Por exemplo, suponha que um arquivo seja transmitido por uma rede e depois armazenado em disco em forma codificada. Isso poderia ser um meio razoável de operação, se o computador receptor fosse um notebook que pudesse ser roubado. Armazenar todos os arquivos críticos em forma codificada reduz muito os danos causados pelo vazamento de informações secretas na eventualidade do computador cair em mãos erradas.

Porém, com freqüência os arquivos de disco são acessados em ordem não seqüencial, especialmente arquivos de bancos de dados. No caso de um arquivo codificado pela utilização do encadeamento de blocos de cifras, o acesso a um bloco aleatório exige primeiro a decodificação de todos os blocos situados à frente dele, uma proposta dispendiosa. Por essa razão, foi criado mais um modo, o **modo de contador**, como ilustra a Figura 8.15. Aqui, o texto simples não é codificado diretamente. Em vez disso, o vetor de inicialização somado a uma constante é codificado, e o texto cifrado resultante é submetido a um XOR com o texto simples. Aumentar o vetor de inicialização em 1 unidade a cada novo bloco, facilita a decodificação de um bloco em qualquer lugar no arquivo sem que primeiro seja preciso decodificar todos os seus predecessores.

Figura 8.15: Codificação com a utilização do modo de contador



Embora seja útil, o modo de contador tem um ponto fraco que vale a pena assinalar. Suponha que a mesma chave K seja usada novamente no futuro (com um texto simples diferente, mas com o mesmo IV) e um atacante adquira todo o texto cifrado de ambas as execuções. O fluxos de chaves são nos dois casos, expondo a cifra a um ataque de reutilização de fluxo de chaves do mesmo tipo que vimos no caso de cifras de fluxo. O criptoanalista tem de efetuar a operação XOR dos dois textos cifrados juntos, a fim de eliminar toda a proteção criptográfica e simplesmente obter o XOR dos textos simples. Essa debilidade não significa que o modo de contador é má idéia. Ela simplesmente quer dizer que tanto as chaves quanto os vetores de inicialização devem ser escolhidos de forma independente e ao acaso. Ainda que a mesma chave seja utilizada duas vezes por acidente, se o IV for diferente em cada utilização, o texto simples estará seguro.

8.2.4 Outras cifras

DES e Rijndael são os algoritmos criptográficos de chave simétrica mais conhecidos. Porém, vale a pena mencionar que foram criados várias outras cifras de chave simétrica. Algumas delas estão incorporadas a vários produtos. A Figura 8.16 mostra algumas dentre as cifras de chave simétrica mais comuns.

Figura 8.16: Alguns algoritmos criptográficos de chave simétrica comuns

Cifra	Autor	Comprimento da chave	Comentários
Blowfish	Bruce Schneier	1 a 448 bits	Velho e lento
DES	IBM	56 bits	Muito fraco para usar agora
IDEA	Massey e Xuejia	128 bits	Bom, mas patenteado
RC4	Ronald Rivest	1 a 2048 bits	Atenção: algumas chaves são fracas
RC5	Ronald Rivest	128 a 256 bits	Bom, mas patenteado
Rijndael	Daemen e Rijmen	128 a 256 bits	Melhor escolha
Serpent	Anderson, Biham, Knudsen	128 a 256 bits	Muito forte
DES triplo	IBM	168 bits	Segunda melhor escolha
Twofish	Bruce Schneier	128 a 256 bits	Muito forte; amplamente utilizado

8.2.5 Criptoanálise

Antes de deixar o assunto de criptografia de chave simétrica, vale a pena mencionar pelo menos quatro desenvolvimentos em criptoanálise. O primeiro desenvolvimento é a criptoanálise diferencial (Biham e Shamir, 1993). Essa técnica pode ser usada para atacar qualquer cifra de bloco. Ela funciona a partir de um par de blocos e texto simples que diferem apenas por um pequeno número de bits e pela observação cuidadosa do que acontece em cada iteração interna à medida que a codificação prossegue. Em muitos casos, alguns padrões de bits são muito mais comuns que outros, e essa observação leva a um ataque probabilístico.

O segundo desenvolvimento que vale a pena notar é a criptoanálise linear (Matsui, 1994). Ela pode romper o DES com apenas 2^{43} textos simples conhecidos. Essa técnica funciona efetuando o XOR de certos bits no texto simples e no texto cifrado, e examinando o resultado em busca de padrões. Quando isso é feito repetidamente, metade dos bits deve ter o valor 0 e metade deve ter o valor 1. Porém, com frequência as cifras introduzem uma inclinação em um sentido ou no outro, e essa inclinação, embora pequena, pode ser explorada para reduzir o fator de trabalho. Para ver os detalhes, consulte o ensaio de Matsui.

O terceiro desenvolvimento é o uso da análise do consumo de energia elétrica para encontrar chaves secretas. Em geral, os computadores utilizam 3 volts para representar um 1 bit e 0 volts para representar um bit 0. Desse modo, o processamento de um bit 1 exige mais energia elétrica

que o processamento de um 0. Se um algoritmo criptográfico cons istir em um loop no qual os bits da chave são processados em ordem, um atacante que substituir o clock principal de n GHz por um clock lento (por exemplo, 100 Hz) e prender pinças dentadas (pinças jacaré) nos pinos de energia da CPU e de terra poderá monitorar com precisão a energia consumida por cada instrução de máquina. A partir desses dados, será surpreendentemente fácil deduzir a chave. Esse tipo de criptoanálise só pode ser anulado por codificação cuidadosa do algoritmo em linguagem assembly para ter certeza de que o consumo de energia será independente da chave e também independente de todas as chaves de rodadas individuais.

O quarto desenvolvimento é a análise de sincronismo. Os algoritmos criptográficos estão repletos de instruções *if* que testam bits nas chaves de rodadas. Se as partes *then* e *else* demoram períodos de tempo diferentes, tornando mais lento o clock e verificando quanto tempo demoram diversas etapas, talvez seja possível deduzir as chaves de rodadas. Uma vez que todas as chaves de rodadas sejam conhecidas, em geral será possível calcular a chave original. A análise da energia e da sincronização também podem ser empregadas simultaneamente para facilitar o trabalho. Embora a análise da energia e da sincronização possam parecer exóticas, na realidade são técnicas eficientes que podem romper qualquer cifra não projetada de forma específica para resistir a elas.

8.3 Algoritmos de chave pública

Historicamente, o problema da distribuição de chaves sempre foi o elo mais fraco da maioria dos sistemas de criptografia. Independente de quanto um sistema de criptografia fosse sólido, se um intruso conseguisse roubar a chave, o sistema acabava sendo inútil. Como todos os criptólogos sempre presumem que a chave de criptografia e a chave de descriptografia são iguais (ou facilmente derivadas uma da outra) e que a chave é distribuída a todos os usuários do sistema, tinha-se a impressão de que havia um problema inerente ao sistema: as chaves tinham de ser protegidas contra roubo, mas também tinham de ser distribuídas; portanto, elas não podiam ser simplesmente trancadas na caixa-forte de um banco.

Em 1976, dois pesquisadores da University of Stanford, Diffie e Hellman (1976), propuseram um sistema de criptografia radicalmente novo, no qual as chaves de criptografia e de descriptografia eram diferentes, e a chave de descriptografia não podia ser derivada da chave de criptografia. Em sua proposta, o algoritmo de criptografia (chaveado) E e o algoritmo de descriptografia (chaveado) D tinham de atender a três requisitos, que podem ser declarados da seguinte forma:

1. $D(E(P)) = P$.
2. É extremamente difícil deduzir D a partir de E.
3. E não pode ser decifrado por um ataque de texto simples escolhido.

O primeiro requisito diz que, se aplicarmos D a uma mensagem criptografada, $E(P)$, obteremos outra vez a mensagem de texto simples original P . Sem essa propriedade, o destinatário legítimo não poderia decodificar o texto cifrado. O segundo é auto-explicativo. O terceiro é necessário porque, como veremos em um minuto, os intrusos podem experimentar o algoritmo até se cansarem. Sob essas condições, não há razão para a chave criptográfica não se tornar pública.

O método funciona da seguinte forma: uma pessoa, digamos Alice, desejando receber mensagens secretas, primeiro cria dois algoritmos que atendem aos requisitos anteriores. O algoritmo de criptografia e a chave de Alice se tornam públicos, daí o nome criptografia de chave pública. Por exemplo, Alice poderia colocar sua chave pública na home page que ela tem na Web. Usaremos a notação E_A para indicar o algoritmo de criptografia parametrizado pela chave pública de Alice. De modo semelhante, o algoritmo de descriptografia (segredo) parametrizado pela chave privada de Alice é D_A . Bob faz o mesmo, publicando E_B , mas mantendo secreta a chave D_B .

Agora vamos ver se podemos resolver o problema de estabelecer um canal seguro entre Alice e Bob, que nunca haviam tido um contato anterior. Supondo que tanto a chave de criptografia de Alice, E_A , quanto a chave de criptografia de Bob, E_B , estejam em arquivos de leitura pública. Agora, Alice pega sua primeira mensagem P , calcula $E_B(P)$ e a envia para Bob. Em seguida, Bob a descriptografa aplicando sua chave secreta D_B [ou seja, ele calcula $D_B(E_B(P)) = P$]. Ninguém mais pode ler a mensagem criptografada $E_B(P)$, porque o sistema de criptografia é considerado sólido e porque é muito difícil derivar D_B da chave E_B publicamente conhecida. Para enviar uma resposta R , Bob transmite $E_A(R)$. Agora, Alice e Bob podem se comunicar com segurança.