

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.

Talvez seja interessante fazer uma observação sobre a terminologia. A criptografia de chave pública exige que cada usuário tenha duas chaves: uma chave pública, usada pelo mundo inteiro para criptografar as mensagens a serem enviadas para esse usuário, e uma chave privada, que o usuário utiliza para descriptografar mensagens. Faremos referência a essas chaves como chave pública e chave privada, respectivamente, e vamos distingui-las das chaves secretas (também chamadas chaves simétricas) usadas na criptografia de chave simétrica convencional.

### 8.3.1 RSA

O único problema é que temos de encontrar algoritmos que realmente satisfaçam a todos os três requisitos. Devido às vantagens potenciais da criptografia de chave pública, muitos pesquisadores estão se dedicando integralmente a seu estudo, e alguns algoritmos já foram publicados. Um método muito interessante foi descoberto por um grupo de pesquisadores do MIT (Rivest et al., 1978) e é conhecido pelas iniciais dos três estudiosos que o criaram (Rivest, Shamir, Adleman): RSA. Ele sobreviveu a todas as tentativas de rompimento por mais de um quarto de século e é considerado um algoritmo muito forte. Grande parte da segurança prática se baseia nele. Sua principal desvantagem é exigir chaves de pelo menos 1024 bits para manter um bom nível de segurança (em comparação com 128 bits para os algoritmos de chave simétrica), e isso o torna bastante lento.

O método RSA se baseia em alguns princípios da teoria dos números. Agora vamos mostrar de forma resumida como usar o método; para obter mais detalhes, consulte o documento original.

1. Escolha dois números primos extensos,  $p$  e  $q$  (geralmente, de 1024 bits).
2. Calcule  $n = p q$  e  $z = (p - 1)(q - 1)$ .
3. Escolha um número  $d$  tal que  $z$  e  $d$  sejam primos entre si.
4. Encontre  $e$  de forma que  $e d \equiv 1 \pmod{z}$ .

Com esses parâmetros calculados com antecedência, estamos prontos para começar a criptografia. Divida o texto simples (considerado um string de bits) em blocos, de modo que cada mensagem de texto simples  $P$  fique no intervalo  $0 \leq P < n$ . Isso pode ser feito agrupando-se o texto simples em blocos de  $k$  bits, onde  $k$  é o maior inteiro para o qual a desigualdade  $2^k < n$  é verdadeira.

Para criptografar a mensagem  $P$ , calcule  $C = P^e \pmod{n}$ . Para descriptografar  $C$ , calcule  $P = C^d \pmod{n}$ . É possível provar que, para todo  $P$  na faixa especificada, as funções de criptografia e descriptografia são inversas entre si. Para realizar a criptografia, você precisa de  $e$  e  $n$ , enquanto para a descriptografia, são necessários  $d$  e  $n$ . Portanto, a chave pública consiste no par  $(e, n)$  e a chave privada consiste em  $(d, n)$ .

A segurança do método se baseia na dificuldade de fatorar números extensos. Se pudesse fatorar o valor  $n$  (publicamente conhecido), o criptoanalista poderia então encontrar  $p$  e  $q$  e, a partir desses, encontrar  $z$ . Com o conhecimento de  $z$  e  $e$ , é possível encontrar  $d$  utilizando-se o algoritmo de Euclides. Felizmente, os matemáticos têm tentado fatorar números extensos por pelo menos 300 anos, e o conhecimento acumulado sugere que o problema é extremamente difícil.

De acordo com Rivest e seus colegas, a fatoração de um número de 500 dígitos requer  $10^{25}$  anos, usando-se a força bruta. Nesse caso, eles pressupõem o melhor algoritmo conhecido e um computador com um tempo por instrução de 1 s. Mesmo que os computadores continuem a se tornar cada vez mais rápidos na proporção de uma ordem de magnitude por década, ainda se passarão séculos até que a fatoração de um número de 500 dígitos se torne viável e, nesse tempo, nossos descendentes poderão simplesmente escolher  $p$  e  $q$  ainda maiores. Um exemplo didático muito comum do algoritmo RSA é mostrado na Figura 8.17. Para esse exemplo, escolhemos  $p = 3$  e  $q = 11$ , o que gera  $n = 33$  e  $z = 20$ . Um valor adequado para  $d$  é  $d = 7$ , visto que 7 e 20 não têm fatores comuns. Com essas opções, pode ser encontrado resolvendo-se a equação  $7 e \equiv 1 \pmod{20}$ , que produz  $e = 3$ . O texto cifrado  $C$  para uma mensagem de texto simples  $P$  é dado por  $C = P^3 \pmod{33}$ . O texto cifrado é decodificado pelo receptor usando a regra  $P = C^7 \pmod{33}$ . A figura mostra a codificação do texto simples "SUZANNE" como exemplo.

**Figura 8.17:** Um exemplo do algoritmo RSA

Plaintext (P)		Ciphertext (C)		After decryption	
Symbolic	Numeric	$P^3$	$P^3 \pmod{33}$	$C^7$	Symbolic
S	19	6859	28	13492928512	19 S
U	21	9261	21	1801088541	21 U
Z	26	17576	20	1280000000	26 Z
A	01	1	1	1	01 A
N	14	2744	5	78125	14 N
N	14	2744	5	78125	14 N
E	05	125	26	8031810176	05 E

Sender's computation                          Receiver's computation

Como os números primos escolhidos para esse exemplo são muito pequenos, P tem de ser menor que 33; portanto, cada bloco do texto simples só pode conter um caractere isolado. O resultado é uma cifra de substituição monoalfabética, que não impressiona muito. Se em vez disso, tivéssemos escolhido  $p$  e  $q \approx 2^{512}$ , teríamos  $n \approx 2^{1024}$  e assim cada bloco poderia ter até 1024 bits ou 128 caracteres de 8 bits, em comparação com 8 caracteres para o DES e 16 caracteres para o AES.

Devemos destacar que o uso do RSA da forma como descrevemos é semelhante ao uso de um algoritmo simétrico no modo ECB — o mesmo bloco de entrada gera o mesmo bloco de saída. Portanto, é necessário algum tipo de encadeamento para a criptografia de dados. No entanto, na prática, a maior parte dos sistemas baseados no RSA utiliza a criptografia de chave pública principalmente para distribuir chaves únicas de sessão que, por sua vez, são empregadas com algum algoritmo de chave simétrica, como AES ou DES triplo. O RSA é lento demais para codificar grandes volumes de dados, mas é amplamente utilizado para a distribuição de chaves.

### 8.3.2 Outros algoritmos de chave pública

Apesar de ser amplamente utilizado, o RSA não é de forma alguma o único algoritmo de chave pública conhecido. O primeiro algoritmo de chave pública foi o algoritmo da mochila (Merkle e Hellman, 1978). A idéia aqui é que alguém possui um grande número de objetos, cada objeto com um peso diferente. O dono dos objetos codifica a mensagem selecionando secretamente um subconjunto dos objetos e colocando-os na mochila. O peso total dos objetos contidos na mochila torna-se público, e o mesmo acontece com a lista de todos os objetos possíveis. A lista de objetos contidos na mochila é mantida em segredo. Com outras restrições específicas, o problema de descobrir uma lista de objetos possíveis com o peso fornecido foi considerado computacionalmente inviável e formou a base do algoritmo de chave pública.

O inventor do algoritmo, Ralph Merkle, estava bastante seguro de que esse algoritmo não poderia ser decifrado; portanto, ofereceu um prêmio de 100 dólares a quem conseguisse fazê-lo. Adi Shamir (o "S" do RSA) se prontificou a decifrar o algoritmo e ganhou o prêmio. In dignado, Merkle reforçou o algoritmo e ofereceu um prêmio de 1.000 dólares a quem pudesse decifrá-lo. Ronald Rivest (o "R" do RSA) também conseguiu decifrar o novo algoritmo e ganhou o prêmio. Merkle não ousou oferecer 10.000 dólares pela nova versão revisada; portanto, "A" (Leonard Adleman) não teve sorte. Apesar de ter sido refeito, o algoritmo da mochila não é mais considerado seguro e não é usado.

Outros esquemas de chave pública se baseiam na dificuldade de calcular logaritmos discretos. Os algoritmos que utilizam esse princípio foram criados por El Gamal (1985) e Schnorr (1991).

Existem alguns outros esquemas, como os que se baseiam em curvas elípticas (Menezes e Vanstone, 1993), mas as duas principais categorias são aquelas que se baseiam na dificuldade de fatorar números extensos e no cálculo de logaritmos discretos cuja base é um número primo extenso. Esses problemas são considerados genuinamente difíceis de resolver — os matemáticos estão estudando os algoritmos há anos sem grandes resultados.

## 8.4 Assinaturas digitais

A autenticidade de muitos documentos legais, financeiros e outros documentos é determinada pela presença de uma assinatura autorizada. Isso não vale para as fotocópias. Para que os sistemas de mensagens computadorizadas possam substituir o transporte físico de documentos em papel e