ALGUNS ASPECTOS DA CRIPTOGRAFIA COMPUTACIONAL

Este exemplar corresponde a redação da tese defendida pelo Sr. RICARDO DAHAB e aprovada pela comissão julgadora.

Campinas, 03 de Agosto de 1984.

Prof. Dr. CLÁUDIO LEONARDO LUCCHESI
Orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Julho – 1984

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Cláudio Lucchesi, pelas idéias, pela objetividade, e pela crítica agressiva, que me ensinaram mais que simplesmente fazer uma Tese de Mestrado.

Ao amigo, Cláudio pela força, pelo inúmeros papos, pela ânsia cúmplice pelo fim da Tese, e pelos vários e vários cafezinhos madrugada adentro.

Ao Joni e à Sueli, meus companheiros de Sala e de longas conversas, por me ajudarem a me manter vivo.

À Vera, pelos desenhos, pela montagem do caderno final da Tese, pela força, pela proximidade, pelo carinho, pela infinita paciência nos últimos meses.

Ao Otero, pelas letras nos desenhos.

À Elda pela dedicação e paciência na excelente datilografia: à Lourdes e à Bel pela ajuda final.

Aos meus pais pelo constante incentivo e apoio.

A todos amigos que vivem perguntando: E a Tese?

A meus pais e

à Vera.

Í N D I C E

CAPÍTULO 1

1.1. NECESSIDADE DE CRIPTOGRAFIA

O uso de computadores e mais especificamente de redes de computadores em um número cada vez maior de atividades, trouxe também seus problemas; o uso adequado dessas redes depende diretamente da confiabilidade que elas oferecem. Iniciaremos descrevendo as ameaças a que um ambiente de computação moderno está sujeito e como pode-se frustrá-las, criptografando ou cifrando (usaremos os dois termos indistintamente) informações, estejam elas armazenadas em algum dispositivo de memória secundária ou em trânsito pela rede.

Criptografia se apresenta hoje como o meio mais prático de se garantir *sigilo* de informações, sejam em trânsito ou armazenadas, e *autenticidade* dessas informações, e de quem as transmite ou recebe.

Mantendo-se o *segredo* ou *sigilo* de textos através do cifra-mento, eles se tornam inteligíveis apenas aos possuidores da chave de decifra-mento, assim autorizados a decifrar os textos. Previne-se com isso a chamada interferência *passiva*, que se carateriza pela aquisição de informações por pessoas não autorizadas, com diversos fins:

a) Se um usuário de uma rede puder "ler" um arquivo contendo

as senhas dos usuários, ele poderá ter acesso aos arquivos destes. Este ataque pode ser frustrado cifrando os arquivos.

b) Outros tipos de informações confidenciais estão sujeitos a essa ameaça: arquivos contendo saldos bancários de clientes, informações médicas ou mesmo software, e todos podem ser protegidos usando criptograia.

c) Através do "grampeamento" de linhas que interligam os computadores de uma rede, uma pessoa também pode adquirir informações confidenciais como as descritas em (a) e (b). Esse "grampeamento" pode ser físico, ligando fios às linhas ou "escutando" canais de microondas, rádio, etc.

A passividade dessa interferência se refere, portanto, à atitude do espião que se limita a "ler" as informações e produzir ou não uma cópia.

Falar em *autenticidade* ou *autenticação* de mensagens significa assegurar

• integridade da mensagem: certeza de que ela não foi modificada, intencionalmente ou não, durante sua transmissão ou mesmo durante o tempo em que esteve armazenada. .

• identidade correta dos usuários em conversaçãò; quando falarmos em usuários estaremos designando de maneira geral os dois extremos que estão se comunicando: duas pessoas, dois programas,

enfim dois processos fazendo uso de uma linha de comunicação, que suporemos vulnerável a ataques de terceiros. Portanto, autenticação de uma mensagem, quanto à identidade do transmissor/receptor, significa assegurar, a um e outro, que a mensagem tem origem/destino que se supõe ter.

Quando falamos em assegurar autenticidade da mensagem, estamos querendo evitar interferência *ativa*, onde uma terceira pessoa não se limita apenas a obter informações, mas também forja mensagens, modifica de maneira imperceptível mensagens legais, faz-se passar por outra pessoa, ou até destrói informações.

Pode-se imaginar várias situações de perigo:

d) Um arquivo armazenado em algum disco ou fita pode ser modificado com diversas finalidades; por exemplo, um cliente de banco pode querer modificar seu saldo de conta corrente.

e) Um aluno pode querer mudar suas notas, modificando o arquivo que as contém.

f) A interferência nas linhas de transmissão pode ser mais sutil: um usuário A inicia uma conversação com outro usuário B, após o que um terceiro usuário C toma o lugar de B sem que A se aperceba disso e passe informações confidencias a C. Ou ainda, A e B conversam sem se dar conta que C está intermediando a conversa de maneira imperceptível aos dois.

g) Podem acontecer situações mais graves: A envia uma mensagem a B e posteriormente nega o envio; ou o inverso, B pode mais tarde negar o recebimento. Se B puder ter em mãos uma mensagem assinada por A: "Compre 1.000.000 de ações da companhia M, Data: XX/XX/XX, assinado: A", não há como A negar o envio, um dia após uma queda espetacular das cotações.

O simples ciframento de textos pode não impedir certas ameaças:

h) se um usuário C consegue gravar todo o diálogo entre um caixa automático e o computador central de um banco, C pode repetir a dose mais tarde, fazendo-se passar por computador central e enviar as mensagens que causem nova liberação de dinheiro pelo caixa automático. Se essas mensagens estão cifradas ou não pouco importa; basta que C repita a sequência correta de bits. De maneira geral ciframento não impede a simples repetição de mensagens por pessoas não autorizadas; situações análogas podem ocorrer em arquivos armazenados: C pode não conseguir gerar o criptograma correspondente a um salário maior que o seu, mas pode copiar o criptograma correspondente a um funcionário que ganha mais do que ele.

i) Criptografia também não oferece proteção contra a simples destruição de informações armazenadas ou em trânsito; destruição pode ser impedida exercendo controle de acesso a esses arquivos, assunto esse que não é objetivo deste trabalho.

j) Análises do fluxo de tráfego de informações numa rede podem ser úteis em casos de atividades militares, por exemplo. O aumento súbito de tráfego numa certa região da rede pode ser prenúncio de atividades nessa região. Esta ameaça pode ser neutralizada mantendo o tráfego constante em todas as direções, sem que os "espiões" saibam do conteúdo das mensagens trocadas.

Em resumo, criptografia não resolve todos os problemas; é apenas mais um meio de aumentar a segurança dos dados.

1.2. CRIPTOSSISTEMAS

Um *criptossistema bidirecional* consiste de:

a) um conjunto de *mensagens* M e outro de *criptogramas* C.

b) Dois conjuntos de chaves, um de *cifra*mento, KC, e outro de *decifra*mento, KD.

c) Algoritmos E e D, de *cifra*mento e *decifra*mento, respectivamente, onde

$$E : KC \times M \rightarrow C$$

$$D : KD \times C \rightarrow M,$$

de forma que para toda chave de ciframento k em KC existe pelo menos uma chave de deciframento ℓ, em KD, tal que $D(\ell,E(k,m)) = m$, para toda mensagem m em M. A razão de chamarmos o criptossistema de bidirecional, é explicada pelo fato de que o processo de ciframento é reversível: o deciframento restaura a mensagem m, dado o criptograma c e a chave ℓ. Como veremos adiante, estaremos interessados também em criptossistemas unidirecionais.

Detalhando melhor este processo, o algoritmo E, dada a chave k em KC, transforma a mensagem m em M num criptograma c em C:

$$c := E(k,m).$$

O algoritmo D, dada uma chave ℓ em KD correspondente à chave k, recupera a mensagem m:

$$m := D(\ell, c).$$

É possível assim estabelecer uma conversação entre dois usuários A e B que já tenham concordado em usar algoritmos E e D e chaves $(kA, \ell A)$ e $(kB, \ell B)$:

Para enviar uma mensagem m_1 para B, A cifra m_1 usando o algoritmo E e a chave k_B, obtendo c_1, que é enviado a B:

$$A \rightarrow B : c_1 := E(kB, m_1).$$

Para decifrar c_1, B usa o algoritmo D e a chave ℓB, recuperando m_1:

$$B : m_1 := D(\ell B, c_1).$$

Analogamente, B envia para A uma mensagem m_2 cifrada, usando E e a chave kA:

$$B \rightarrow A : c_2 := E(kA, m_2).$$

A recupera m_2 usando D com chave ℓA e o criptograma c2

$$A : m_2 := E(\ell A, c_2).$$

A Figura 101 ilustra a conversação acima:

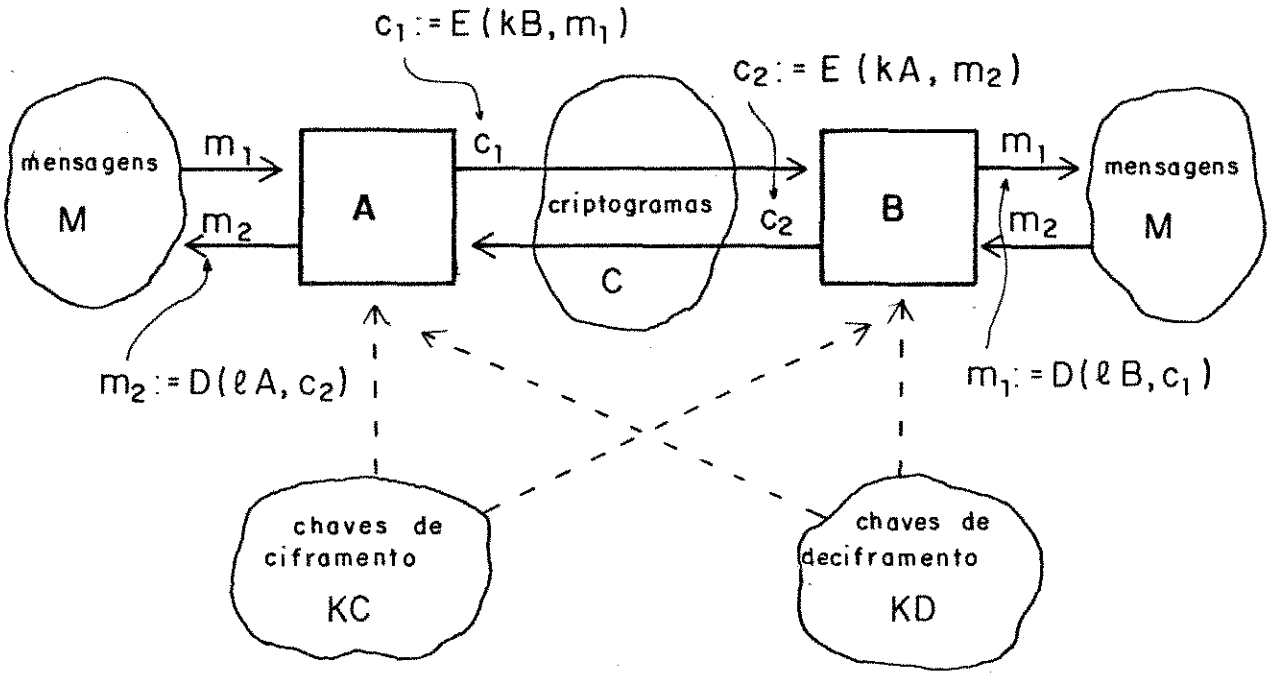


Figura 101: CRIPTOSSISTEMA BIDIRECIONAL

Convém notar que A conhece kA, kB e ℓA, mas não necessariamente ℓB. Analogamente, B conhece kA, kB e ℓB, mas pode não saber ℓA.

Quando é viável computar ℓ a partir de k (em particular no caso em que k = ℓ), o criptossistema é dito *simétrico* ou de *chave secreta;* pode-se então usar apenas uma chave k para a conversação só inteligível para os dois usuários A e B que conhecem

k. Os quatro passos da conversação ficam então:

$$A \rightarrow B : c_1 = E(k,m_1)$$

$$B : m_1 = D(k,c_1)$$

$$B \rightarrow A : c_2 = E(k,m_2)$$

$$A : m_2 = D(k,c_2).$$

A Figura 102 ilustra um criptossistema simétrico.

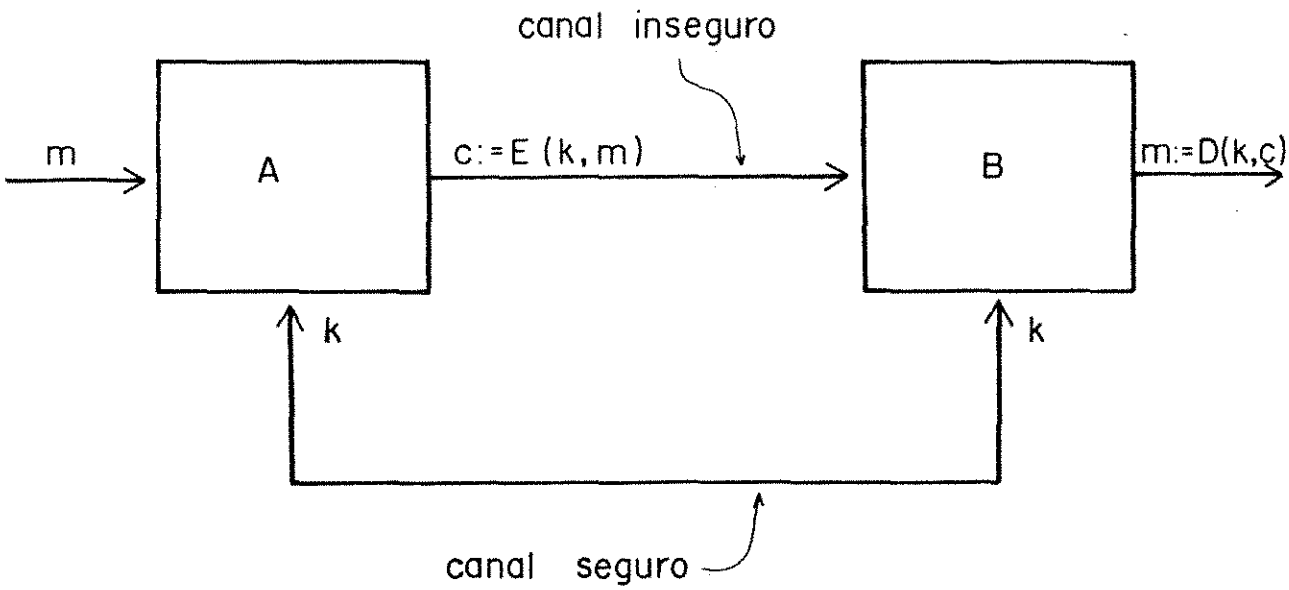


Figura 102: CRIPTOSSISTEMA BIDIRECIONAL SIMÉTRICO

Se por outro lado for inviável computar ℓ a partir de k o criptossistema é dito *assimétrico* ou de *chave pública*, onde o conhecimento que A e B tem é o originalmente descrito: A conhece kA, ℓA e kB; B conhece kB, ℓB e kA. Essa distinção entre pública e secreta e a razão desses nomes é vista com detalhe mais à frente.

• Exemplos

1) Substituição de César

Suponhamos que o algoritmo de ciframento encara as mensagens a serem cifradas como seqüências de inteiros compreendidos entre 0 e 25, correspondentes às 26 letras do alfabeto, e realiza a seguinte operação sobre esses inteiros: some uma constante módulo 26 a cada inteiro, substituindo a "letra" original pela "letra" correspondente ao resultado da soma.

Dois usuários A e B combinam previamente o valor da constante a que nos referimos acima, como sendo por exemplo igual a 15, e agora podem trocar mensagens.

Com este algoritmo de ciframento, a mensagem

ACABOU O CAFÉ. FAÇA MAIS.,

que corresponde aos inteiros:

01 03 01 02 15 21 15 03 01 06 05 06 01 03 01 13 01 09 19,

se transforma na mensagem:

P R P Q D J D R P U T U P R P B P X H

que corresponde aos inteiros:

16 18 16 17 04 10 04 18 16 21 20 21 16 18 16 02 16 24 08.

O deciframento é feito de maneira análoga, apenas subtraindo a constante 15 módulo 26.

O criptossistema descrito acima é chamado *cifra de César* ou *substituição de César*, cuja invenção ou perpetuação é atribuída a Júlio César, que usou-o com constante igual a 3 [KAHN-67]. Este método faz parte dos métodos de ciframento conhecidos como de *substituição simples*, onde tudo o que se faz é substituir as letras do alfabeto original por outras. Obviamente, este método é muito fraco.

Em resumo, a cifra de César consiste de:

a) um conjunto de mensagens M e outro de criptogramas C, ambos consistindo de textos formados com as 26 letras do alfabeto;

b) um conjunto de chaves de ciframento KC e outro de chaves de deciframento KD, ambos consistindo dos inteiros em [0,25];

c) algoritmos E e D de ciframento e de deciframento, respectivamente; o ciframento consiste em somar uma constante,

módulo 26, às "letras" das mensagens e o deciframento em subtrair essa constante módulo 26 das "letras" do criptograma.

Note-se que o criptossistema é simétrico, onde a mesma chave é usada para ciframento e deciframento.

2) One-time pad

O *one-time pad* é semelhante à substituição de César pois cada letra sofre um deslocamento. E por outro lado é diferente na medida em que o deslocamento da letra depende da sua posição no texto. Assim, a chave é uma sequência que consiste de tantos deslocamentos quantos forem os caracteres do texto.

A característica fundamental deste sistema é que essa seqüência de deslocamentos é gerada aleatoriamente para cada novo texto a ser cifrado, de maneira que todas as seqüências de mesmo comprimento tenham a mesma probabilidade de ocorrer.

Por exemplo, vamos supor o alfabeto {0,1}:

mensagem m_1 = 1 0 1 1 0 1 1 0 0 0 1

+ módulo 2

chave k = 0 0 1 1 1 1 1 0 1 1 0

criptograma c_1 = 1 0 0 0 1 0 0 0 1 1 1

Analogamente:

criptograma c_1 = 1 0 0 0 1 0 0 0 1 1 1

- módulo 2

chave k = 0 0 1 1 1 1 1 0 1 1 0

mensagem m_1 = 1 0 1 1 0 1 1 0 0 0 1

Assim, o one-time pad é também um criptossistema simétrico.

É possível provar que este sistema tem segurança perfeita, ou seja, que se um espião tem acesso ao criptograma c_1, ele não consegue deduzir absolutamente nada sobre a mensagem ou chave, pois esta é aleatória e cada caracter da mensagem foi transformado aleatoriamente portanto; mais precisamente, a distribuição de caracteres no criptograma é uniforme: todos os criptogramas de um mesmo comprimento têm a mesma probabilidade de ocorrer.

Se por outro lado, a mesma chave k for usada para cifrar uma outra mensagem m_2, produzindo portanto $c_2 = m_2 + k$, um espião que tenha acesso a c_2 computa:

$d := c_1 + c_2$, o que equivale a dizer que

$d = m_1 + k + m_2 + k$, e como $k + k = 0$,

$d = m_1 + m_2$, de onde conclue-se que d

é um criptograma gerado a partir de m_1 com chave m_2, que não é uma seqüência aleatória de bits, mas uma mensagem, e portanto com características que o espião pode conhecer e usar para decifrar

m_1 . Além disso, se o espião tiver acesso a m_1 , o que é uma situação nem tão rara, ele computa facilmente a chave:

$$k := c_1 + m_1$$

e usa k para decifrar m_2 .

O one-time pad não é utilizado em redes por razões de ordem prática: o volume de informações em trânsito numa rede é muito grande, e o trabalho de gerar a mesma chave de maneira efetivamente aleatória, e a posterior combinação dessa chave pelos dois usuários A e B, antes da transmissão, é algo impraticável.

3) O RSA

Suponhamos aqui que as mensagens sejam inteiros, só que num certo intervalo [1,n - 1]. O ciframento e o deciframento consistem de exponenciações módulo um número. No ciframento, o expoente é público, no deciframento ele é secreto. Por exemplo, o usuário A querendo enviar a mensagem 20 para B, computa

$$20^{17} \bmod 33 = 26;$$

ou seja, a mensagem é elevada à potência 17, e o resultado dividido por 33; o resto dessa divisão é o criptograma que será enviado para B.

Prosseguindo, B decifra 26 elevando à potência 13, módulo 33, recuperando a mensagem 20 original:

$$26^{13} \bmod 33 = 20.$$

O expoente de ciframento e de deciframento são distintos; as chaves kB e ℓB são respectivamente, (17,33) e (13,33). O criptossistema exemplificado acima é conhecido como RSA (iniciais dos criadores), e será descrito com detalhe em capítulo posterior.

O RSA consiste de:

a) Conjunto de mensagens M e conjunto de criptogramas C, ambos consistindo de inteiros no intervalo $[1, n-1]$, onde $n = p \cdot q$, p e q primos "grandes".

b) Conjunto de chaves de ciframento KC e de deciframento KD, ambos consistindo de pares (k,n), tal que $mdc(k,(p-1)(q-1)) = 1$.

c) Algoritmos de ciframento e de deciframento, E e D respectivamente, onde E = D, tal que:

Dada uma mensagem m (criptograma c) e uma chave de ciframento k (deciframento ℓ), computa-se:

$$m^k \bmod n \quad (c^{\ell} \bmod n).$$

A propriedade fundamental do RSA é que dados (k,n) e (ℓ,n),

$m^{k\ell} \bmod n = m$, para todo m em [1,n - 1].

Este criptossistema é o mais conhecido dentre os assimétricos ou de chave pública. Ainda não foi descoberta uma maneira eficiente de quebrá-lo.

Em resumo, nos criptossistemas de chave secreta, é fundamental que apenas A e B conheçam a chave k, caso contrário é possível interceptar um criptograma e decifrá-lo pois os algoritmos E e D são públicos.

Já nos casos de criptossistemas de chave pública, basta que a chave secreta seja de conhecimento exclusivo de seu dono; isso implica que um terceiro usuário conhecendo apenas a chave pública é incapaz de deduzir a secreta. Por outro lado, é possível comunicar-se em segredo com um usuário, bastando obter sua chave pública num arquivo público, e enviar suas mensagens sem necessidade de acordo prévio.

1.3. CRIPTOANÁLISE

Um criptoanalista pode ter acesso a vários componentes dos sistemas descritos, desde uma quantidade qualquer de mensagens cifradas ou informações sobre a natureza das mensagens, até pares mensagem-criptograma à sua escolha. Os algoritmos que implementam o ciframento/deciframento devem oferecer segurança, mesmo sob a hipótese de criptoanalista ter à sua disposição muita informação. Resta como único segredo a chave de deciframento. Há três tipos básicos de ataques:

1) SOMENTE CRIPTOGRAMAS - um espião (criptoanalista) pode obter uma quantidade razoável de criptogramas utilizando-se de escuta em linhas de transmissão ou acesso a arquivos criptografados.

2) SOMENTE PARES MENSAGEM-CRIPTOGRAMA - o criptoanalista consegue obter as mensagens que deram origem aos criptogramas em seu poder.

3) MENSAGENS ESCOLHIDAS - mesmo não conhecendo a chave, o espião tem acesso ao algoritmo de ciframento para obter o criptograma correspondente a uma certa mensagem escolhida por ele.

É importante fazer distinção entre o conceito de "quebrar" o criptossistema e o de descobrir uma mensagem que deu origem a

um criptograma. Às vezes tudo que se quer é descobrir uma mensagem; outras vezes se deseja deduzir a chave para decifrar futuros criptogramas. É possível que após muito esforço o criptoanalista descubra a chave, mas todo o esforço de nada lhe sirva se a chave for mudada, a não ser para decifrar mensagens antigas.. Pode ser por outro lado, que se obtenha um algoritmo que deduza a chave em tempo hábil, seja ela qual for; a isso chamaremos *quebrar o criptossistema*. As alternativas são muitas, mas os extremos são bem delimitados: algoritmos de ciframento que são seguros apenas contra ataques baseados somente em criptogramas (tipo 1) são fracos e não proporcionam segurança. Algoritmos que resistem a ataques baseados em mensagem escolhida (tipo 3) são os mais desejáveis.

Se de um lado foi possível idealizar algoritmos cada vez mais complexos para criptografar mensagens com o advento dos computadores, pelo mesmo motivo o trabalho de criptoanálise se tornou mais ameaçador. A busca exaustiva de todas as chaves possíveis, deixou de ser um parâmetro absoluto na avaliação de eficiência de criptossistemas, para se tornar algo relativo à vida útil das informações que se quer manter em segredo. O sigilo de um arquivo contendo informações confidenciais pode estar totalmente comprometido se a busca exaustiva de chaves durar um mês usando computadores, ou ainda mais dependendo da importância dos dados. Outras informações somente são úteis num curto espaço de tempo e busca exaustiva pode ser muito demorada. Portanto um criptossistema será

considerado seguro, quando o volume de recursos computacionais (tempo, espaço, dinheiro) necessário para deduzir uma simples mensagem, for grande o suficiente para tornar a tentativa inviável, dada a importância dos dados.

1.4. FUNÇÕES UNIDIRECIONAIS

• Funções Unidirecionais - alçapão.

Retornando à definição de criptossistemas, fixada uma chave k, o algoritmo de ciframento E pode ser visto como uma função do espaço de mensagens no espaço de criptogramas. Para ser útil do ponto de vista criptográfico, essa função deve satisfazer os seguintes requisitos:

1) Para quaisquer $m \in M$ e $k \in KC$, $E(k,m)$ deve ser eficientemente computável.

2) Para quase todos os m, dado um certo k, $D(\ell,E(k,m))$ deve ser difícil de computar sem conhecer ℓ.

3) Para quaisquer $c \in C$ e $\ell \in KD$, $D(\ell,c)$ deve ser eficientemente computável, dado ℓ correto.

4) Os espaços de chaves KC e KD, e de mensagens M, devem ser grandes o suficiente para inviabilizar busca exaustiva de quaisquer dos três espaços.

5) Deve ser computacionalmente difícil deduzir ℓ, sistematicamente, a partir de qualquer quantidade de pares mensagem-criptograma.

Por (1) e (2) pode-se dizer que E é uma *função unidirecional*, pois ela é facilmente computável em um sentido (ciframento), mas muito difícil de se computar no sentido inverso (deciframento). Por (3), E é dita *função unidirecional-alçapão* no sentido de que o conhecimento de ℓ viabiliza o cálculo da inversa; fica caracterizado o efeito alçapão, que se abre facilmente num dos sentidos (de baixo para cima), mas não no sentido inverso (a menos que se saiba onde se esconde a alavanca - a chave ℓ); em inglês essas funções são chamadas "trapdoor one-way functions".

· Funções Unidirecionais

Em algumas situações é desejável que a propriedade (c) da definição de criptossistemas bidirecionais *não* seja satisfeita: não é necessário, em algumas aplicações, que exista ℓ tal que $D(\ell,E(k,m)) = m$. Nem mesmo é necessário que, fixado k, E seja injetora. Definiremos então *criptossistemas unidirecionais* como consistindo de:

a) um conjunto de mensagens M e outro de criptogramas C.

b) Um conjunto de chaves de ciframento KC .

c) Algoritmo de ciframento E, onde

$$E : KC \times M \rightarrow C.$$

Como tal, a função E é chamada *função unidirecional* ("one-way function") e para ser útil nas aplicações em que estaremos interessados, deve satisfazer às seguintes propriedades:

1) Para quaisquer $m \in M$ e $k \in KC$, $E(k,m)$ deve ser eficientemente computável.

2) Para quase todos os $c \in C$, dado um certo $k \in KC$, deve ser difícil computar ao menos um $m \in M$ tal que $c = E(k,m)$.

3) O espaço de chaves, KC, e de mensagens M, devem ser

grandes o suficiente para inviabilizar busca exaustiva de quaisquer dos dois espaços.

4) Deve ser computacionalmente difícil deduzir k a partir de qualquer quantidade de pares mensagem-criptograma.

Essas funções foram usadas pela primeira vez num contexto criptográfico por Needham [EVAN-74], para resolver problemas de autenticação de usuários frente ao sistema operacional: ao invés de armazenar as senhas dos usuários no sistema, armazena-se a versão cifrada dessas senhas, usando uma função unidirecional; descreveremos esta particular aplicação, com mais detalhes, no último capítulo.

Daremos a seguir um exemplo de funções unidirecionais:

Sejam:

- p um primo ímpar, "grande".
- α uma raiz primitiva módulo p, isto é, p - 1 é o menor inteiro positivo tal que $\alpha^{p-1} \equiv 1 \bmod p$.
- x um inteiro, $1 \leq x \leq p - 1$.

Seja

$$E(\alpha,x) : \alpha^x \bmod p.$$

Assim, dizemos que x é o logaritmo de $\alpha^x \bmod p$ na base α, módulo p.

O cálculo de $E(\alpha,x)$ é muito rápido e requer no máximo $2 \log_2 x$ multiplicações [KNUT 2-69]. O cálculo do logaritmo é extremamente demorado para p muito grande (600 bits). O melhor algoritmo é de autoria de Adleman [ADLE1-79] e leva tempo proporcional a $e^{\sqrt{\ell n\, p \ell n\, \ell n\, p}}$.

Se $\log_2 p \approx 600$, então a expressão acima vale $1{,}2 \times 10^{23}$; se cada operação levar 1μs para ser executada, e desprezando constantes de proporcionalidade, teremos a fantástica quantia de 10^{12} dias ou 2,7 bilhões de anos para calcular a inversa de E. A adaptação dessa função para o esquema de senhas é imediata, bastando tomar x como o valor da senha.

1.5. CRIPTOSSISTEMAS DE CHAVE PÚBLICA versus DE CHAVE SECRETA

Até 1976, quando Diffie & Hellman [DIFF1-76] lançaram o conceito de sistemas criptográficos de chave pública o procedimento para cifrar e decifrar mensagens era simétrico [SIMM1-79], o que exigia que dois usuários querendo se comunicar em sigilo, devessem estabelecer previamente uma chave k. O problema de armazenamento dessas chaves previamente combinadas, tem duas soluções:

• ou cada usuário mantem um arquivo com as chaves previamente combinadas com os outros usuários,

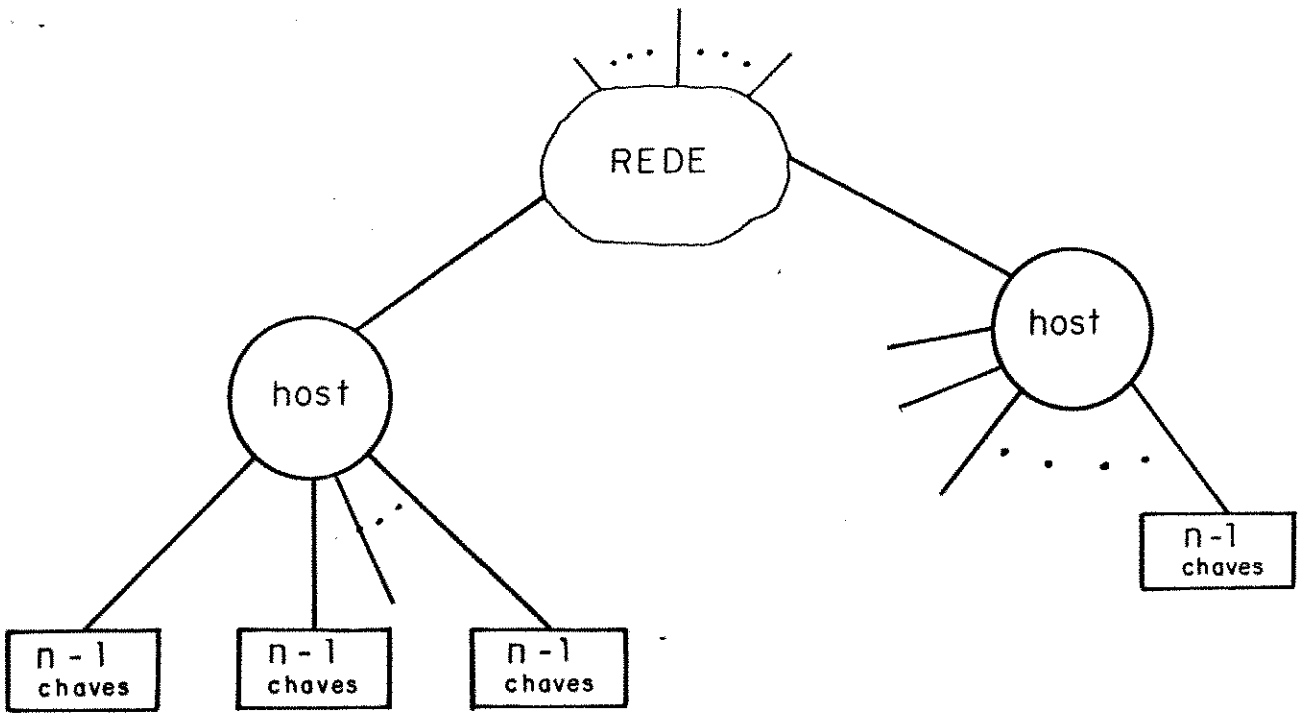


Figura 103: TOTAL DE $\binom{n}{2}$ CHAVES PREVIAMENTE ESTABELECIDAS, UMA PARA CADA PAR DE USUÁRIOS. TOTAL DE $2\binom{n}{2}$ CHAVES ARMAZENADAS.

• ou estabelece-se um sistema hierarquizado, onde todos os usuários estão ligados a um computador hospedeiro, que mantém uma chave diferente para se comunicar em sigilo com cada usuário. Para um usuário se comunicar com outro, inicialmente se comunica com o hospedeiro usando a chave previamente combinada - *chave permanente* -, e este gera uma chave a ser usada somente naquela comunicação, que chamaremos *chave transitória*. A chave transitória é distribuida pelo hospedeiro para ambos, cifrando-a com as respectivas chaves permanentes. Ao fim da comunicação, a chave transitória é descartada.

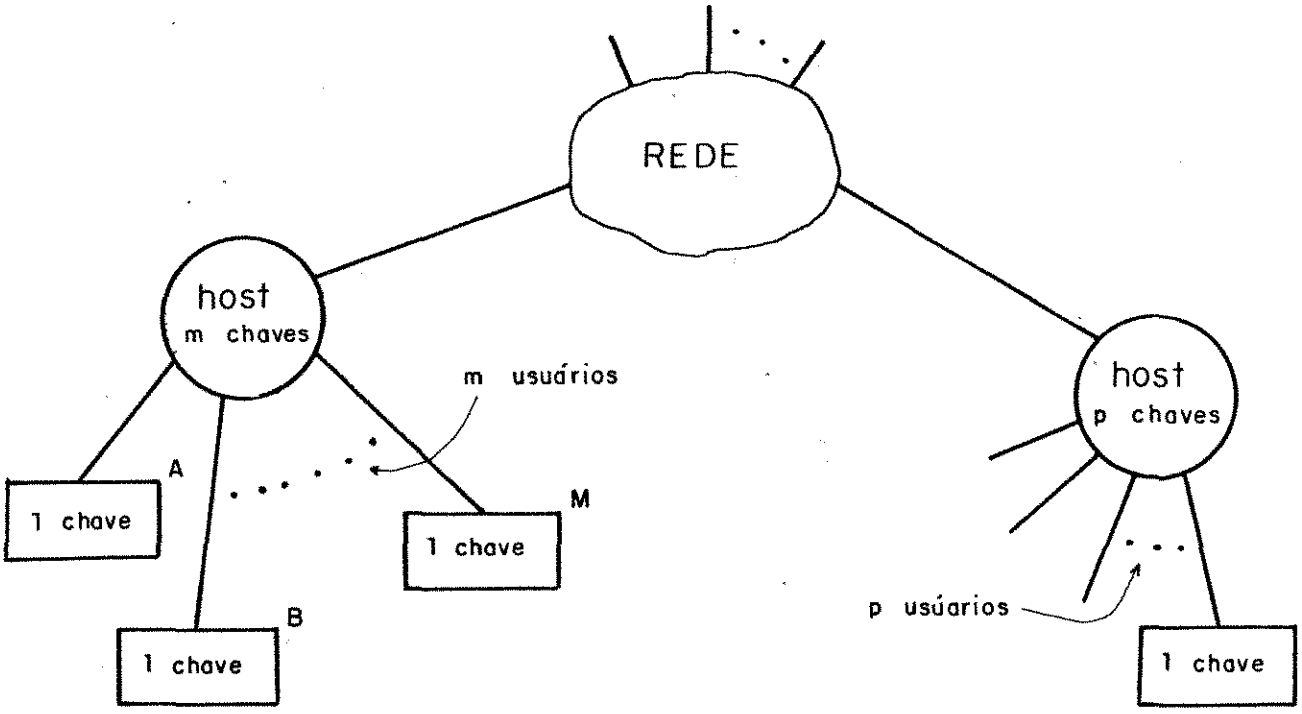


Figura 104: TOTAL DE n(m + p + ...) CHAVES, PREVIAMENTE ESTABELECIDAS ENTRE CADA USUÁRIO E O RESPECTIVO HOST:

Na primeira opção são necessários, numa rede de n usuários, n arquivos com $n-1$ chaves cada, num total de $n(n-1) = 2\binom{n}{2}$ chaves armazenadas na rede. Na segunda opção, apenas n chaves diferentes são armazenadas no hospedeiro.

A troca de chaves (recomendável afim de não expor muitos criptogramas cifrados com mesma chave) entre os usuários do primeiro esquema pode ser cara e lenta. No segundo esquema as chaves transitórias têm tempo de vida curto e as permanentes são pouco usuadas; além disso, se precisarem ser trocadas, basta modificar dois arquivos, o do usuário e o do hospedeiro. O hierarquizado, todavia, ainda apresenta um problema: o comprometimento de uma chave permanente põe em perigo toda a comunicação com o hospedeiro e portanto com os outros usuários.

A proposta de Diffie & Hellman, de assimetrização, parece mudar mas não muda esse cenário: ainda que a chave de ciframento seja pública, temos as mesmas duas situaçãoes:

• ou cada usuário mantém um catálogo e portanto teremos no total as mesmas $2\binom{n}{2}$ chaves distribuídas, além das n secretas.

• ou o catálogo fica centralizado num hospedeiro e cada usuário se comunica com ele sem necessidade de sigilo.

Neste caso não está muito claro qual das duas alternativas é a melhor; embora a primeira opção ofereça mais flexibilidade

e independência do hospedeiro, surge o problema da confiabilidade do catálogo que deve ser atualizado periodicamente, o que pode ser problemático; o segundo esquema elimina porém a vantagem mais flagrante dos criptossistemas assimétricos sobre os simétricos: a possibilidade de um usuário iniciar imediatamente uma comunicação com outro sem entendimento prévio.

Existem ainda outras maneiras de dois usuários chegarem a uma chave secreta comum; duas das propostas mais conhecidas são a de Diffie & Hellman [DIFF1-76] e Merkle [MERK1-78]. Vamos descrever a de Diffie & Hellman, que usa uma função unidirecional descrita anteriormente, a saber, exponenciação modular:

Dois usuários A e B compartilham os inteiros α e p, onde p é um primo "grande" e α uma raiz primitiva módulo p. Ao iniciarem uma sessão, A e B geram, ao acaso, inteiros x e y respectivamente; a seguir A computa

$$c_1 := \alpha^x \bmod p, \quad \text{e envia } c_1 \text{ a B;}$$

analogamente, B computa

$$c_2 := \alpha^y \bmod p, \quad \text{e envia } c_2 \text{ para A.}$$

Neste ponto, A conhece c_1, c_2, α, p e x;

B conhece c_1, c_2, α, p, y e um possível

espião conhece c_1, c_2, p, α.

Agora, A e B podem computar o mesmo valor

$$\alpha^{xy} \bmod p;$$

para A, isso corresponde a fazer

$$(c_2)^x \bmod p = (\alpha^y \bmod p)^x \bmod p = \alpha^{xy} \bmod p;$$

para B analogamente

$$(c_1)^y \bmod p = (\alpha^x \bmod p)^y \bmod p = \alpha^{xy} \bmod p.$$

Fica estabelecido o valor comum $k = \alpha^{xy} \bmod p$ para ser usado diretamente ou não, num criptossistema simétrico conveniente. O espião não consegue calcular k pois não conhece nem x nem y, mas apenas $\alpha^x \bmod p$ e $\alpha^y \bmod p$.

1.6. CIFRAMENTO ENCADEADO

Como veremos nos capítulos 2 e 3, os algoritmos de ciframento/deciframento manipulam mensagens de tamanho fixo; isto é, um algoritmo cifra um bloco de tamanho t a cada vez; o que significa que se a mensagem tiver tamanho maior que t, ela terá que ser "quebrada" em blocos de tamanho t, que serão cifrados com a mesma chave. Se por outro lado a mensagem tiver tamanho menor que t, deve-se completá-la com o número necessário de caracteres, se possível não fixos.

Isto significa que se um texto muito estruturado (um programa em linguagem de montagem, por exemplo) for cifrado, o resultado do ciframento também será estruturado. Trechos iguais do texto levarão a trechos iguais de criptograma, pois a chave que está sendo usada nos vários blocos é a mesma. Um criptoanalista pode obter assim informações adicionais que à primeira vista, supunha-se, ele não conseguia obter.

Isto pode ser resolvido, fazendo-so *ciframento encadeado*; isto é, cada bloco do criptograma será dependente de blocos anteriores, usando-se a mesma chave para cifrar todo o texto.

Pode-se fazer ciframento encadeado, de duas formas, entre outras:

- *ciframento encadeado com realimentação do criptograma*, e

• *ciframento encadeado com realimentação da mensagem e do criptograma.*

O primeiro método, com realimentação do criptograma, é feito como segue:

• Seja m a mensagem dividida em blocos de t caracteres

$$m_1 , m_2 , \ldots , m_n .$$

• o criptograma $c = c_1 , c_2 , \ldots , c_n$ é igual a

$$c_i := E(k, [m_i + c_{i-1}]), \quad 1 \leq i \leq n,$$

onde c_o é um valor inicial, público ou não.

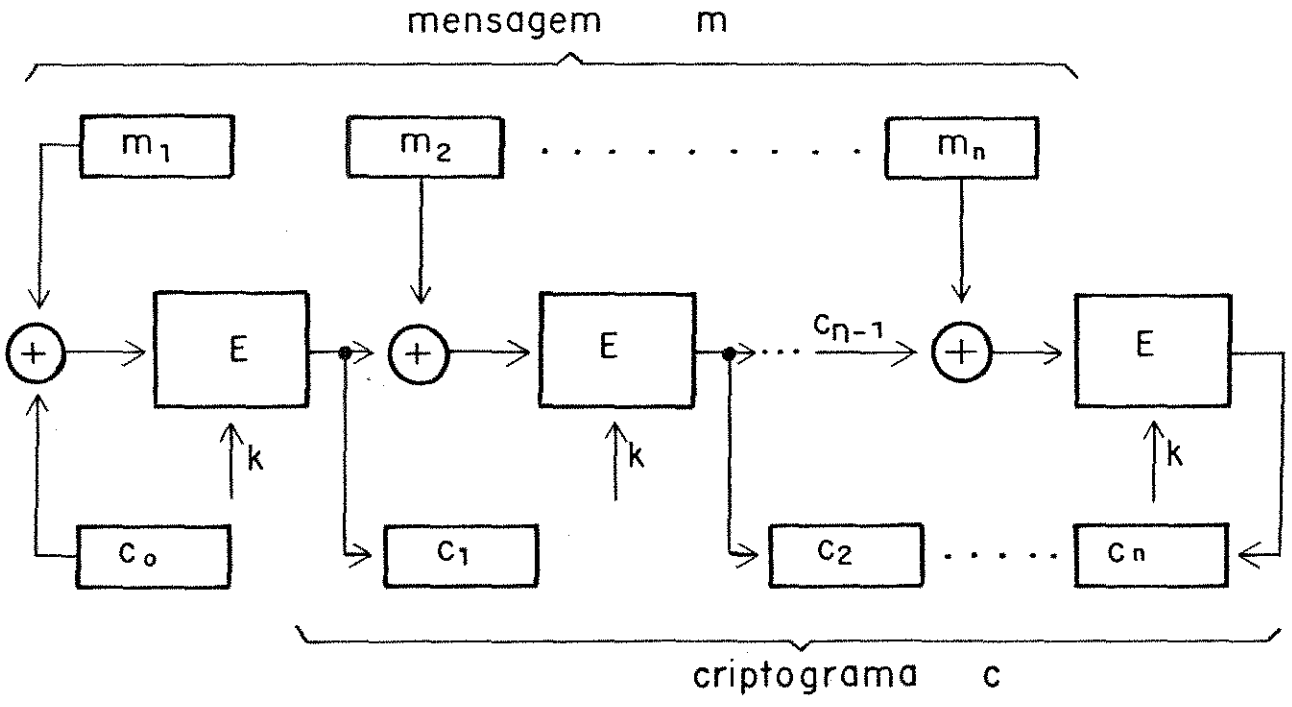


Figura 105: ENCADEAMENTO ("CHAINING") COM REALIMENTAÇÃO DO CRIPTOGRAMA.

• Decifrar c corresponde a computar

$$m_i := D(k,c_i) + c_{i-1}, \quad 1 \leq i \leq n.$$

Note-se que o resultado do ciframento de m é diferente se, para a mesma chave, trocarmos o valor de c_o. Observa-se também, que embora c_n seja dependente de todos os c_i's anteriores, um erro na transmissão de um desses c_i's *não* se propaga além do próximo bloco, pois:

Supondo que houve um erro apenas na transmissão do bloco c_j, verifica-se que

$m_j := D(k,c_j) + c_{j-1}$ será decifrado com erro pois c_j errado,

$m_{j+1} := D(k,c_{j+1}) + c_j$ será decifrado com erro pois c_j errado,

$m_{j+2} := D(k,c_{j+2}) + c_{j+1}$ será decifrado corretamente pois c_{j+2} e c_{j+1} corretos.

O segundo método, com realimentação da mensagem e do criptograma, é como segue:

• A mensagem m é como antes

• O criptograma $c = c_1, c_2, \ldots, c_n$ é igual a

$$c_i := E(k, [m_i + c_{i-1} + m_{i-1}]), \quad 1 \leq i \leq n,$$

onde c_o e m_o são valores iniciais públicos ou não.

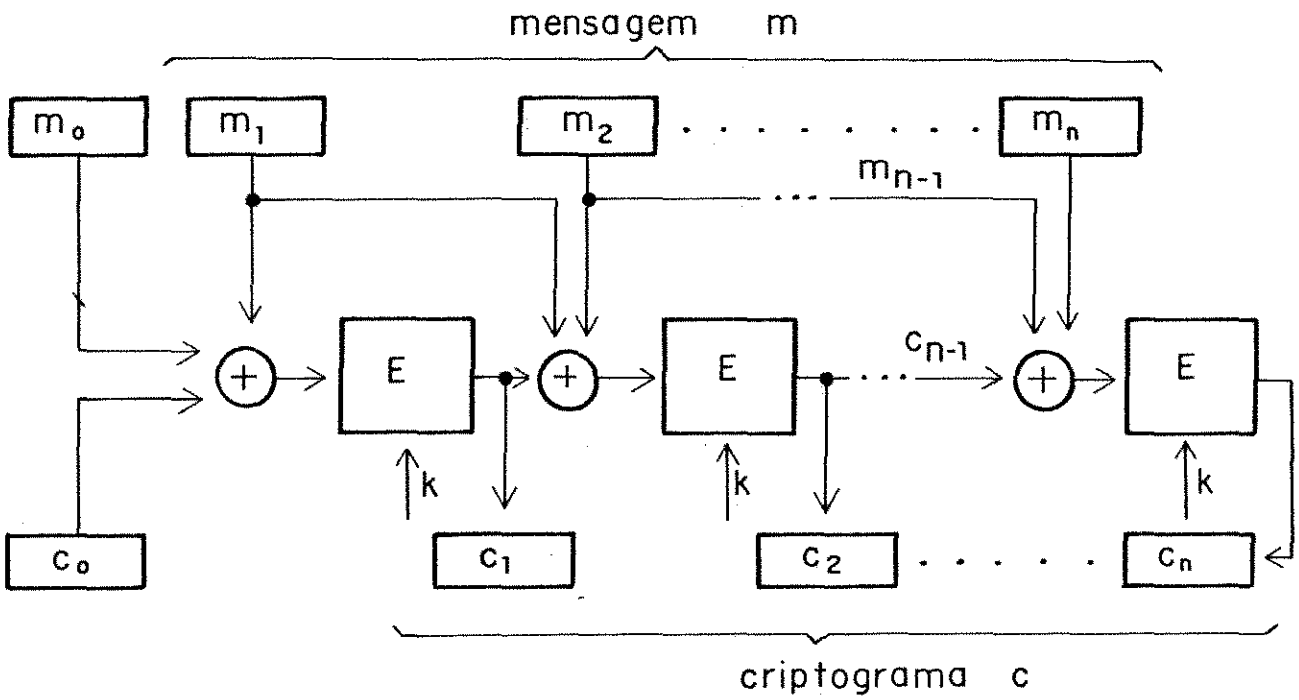


Figura 106: ENCADEAMENTO COM REALIMENTAÇÃO DO CRIPTOGRAMA E DA MENSAGEM.

• O deciframento corresponde a:

$$m_i := D(k, c_i) + m_{i-1} + c_{i-1}, \quad 1 \leq i \leq n.$$

Assim, o deciframento de cada bloco depende do deciframento dos blocos anteriores. É fácil ver que um erro na transmissão de algum bloco do criptograma se propaga até o último bloco, que pode ser usado como detector de erros, acidentais ou não. Dizemos assim que o primeiro método é *sem* propagação de erros e o segundo *com* propagação de erros.

1.7. RESUMO BIBLIOGRÁFICO

Quatro livros compõem hoje as referências em criptografia moderna: [KONH1-81], [DENN1-82], [MEYE1-82] e [LUCC1-84] com enfoques diferentes: o primeiro trata mais profundamente de criptoanálise, com argumentos baseados principalmente em teoria da informação e estatística, dando atenção especial aos métodos simétricos clássicos.

A segunda referência cobre de maneira mais ampla o aspecto da segurança de dados, tendo criptografia como uma das ferramentas, além de controle de acesso, controle de fluxo de informações e controles de inferência.

O terceiro livro, mais atualizado, descreve com grande riqueza os aspectos práticos da implementação de criptografia em redes de computadores, dando ênfase a criptossistemas simétricos.

O quarta referência, a única em português, além de descrever e analisar alguns dos algoritmos para ciframento/deciframento propostos nos últimos anos, trata das principais aplicações da criptografia computacional.

Um romance clássico em criptografia é [KAHN-67], em que o autor descreve com outro sabor a criptografia através dos séculos, desde o antigo Egito até a quebra dos métodos de ciframento usados na Segunda Guerra Mundial.

CAPÍTULO 2

CRIPTOSSISTEMAS SIMÉTRICOS - O DES

• RETROSPECTIVA

Sem dúvida, o criptossistema simétrico mais conhecido e usado atualmente é o chamado DES - DATA ENCRYPTION STANDARD - que surgiu da evolução de um outro criptossistema-LUCIFER [FEIS1-73], desenvolvido pela IBM por volta de 1970. O LUCIFER foi o primeiro produto criptográfico projetado para atender às necessidades modernas de criptografia - rapidez, segurança contra busca exaustiva e implementável em quantidade relativamente pequena de hardware; usava idéias de composição de métodos de ciframento - substituição e transposição - que sozinhos são criptograficamente fracos, mas cuja composição gera um algoritmo de ciframento bastante seguro; essas idéias foram lançadas por Shannon [SHAN1-49], num artigo fundamental da Teoria da Informação. O DES apresenta ligeiras modificações em relação ao LUCIFER (A principal delas, a diminuição do tamanho da chave à metade, é uma das pricinpais fontes de controvérsia sobre a segurança do algoritmo), e foi projetado também por uma equipe da IBM, e adotado como padrão pelo governo Norte-Americano [DES1-77] para uso pelos seus orgãos, em aplicações não ultra-secretas. O projeto do DES já se preocupava mais com a implementação do algoritmo, que deveria ser adaptável

ao uso em linhas de transmissão relativamente rápidas, além da necessidade de caber num único "chip". Desde a apresentação pública do algoritmo (1975), e posterior aprovação em 1977, até hoje, muita controvérsia surgiu sobre a efetiva segurança do algoritmo ("essa segurança não pode ser matematicamente *provada*"), mas o consenso atualmente é que ele é o melhor e mais rápido disponível no mercado.

No que segue, faremos uma descrição do algoritmo e seus critérios de construção, métodos de criptoanálise e controvérsias surgidas.

• DESCRIÇÃO DO ALGORITMO

A Figura 201, na página seguinte, ilustra os algoritmos de cifra mento e seleção de chaves internas usadas nas várias iterações por que passa uma mensagem m (64 bits), usando-se uma chave k(56 bits).

O DES, visto macroscopicamente, é o produto de 16 iterações do tipo: (+ é a operação de ou-exclusivo bit a bit)

• $r(i + 1) := \ell(i) + f(r(i), k(i))$

$\ell(i + 1) := r(i), \quad 1 \leq i \leq 15, \quad$ e

• $r(17) := r(16)$

$\ell(17) := \ell(16) + f(r(16), k(16));$

onde:

• $\ell(j)$, $r(j)$ são as metades (32 bits) da mensagem na j-ésima iteração;

• f é uma função *não* linear descrita adiante.

• k(j) é a *subchave* escolhida para a j-ésima iteração de acordo com um algoritmo também descrito adiante.

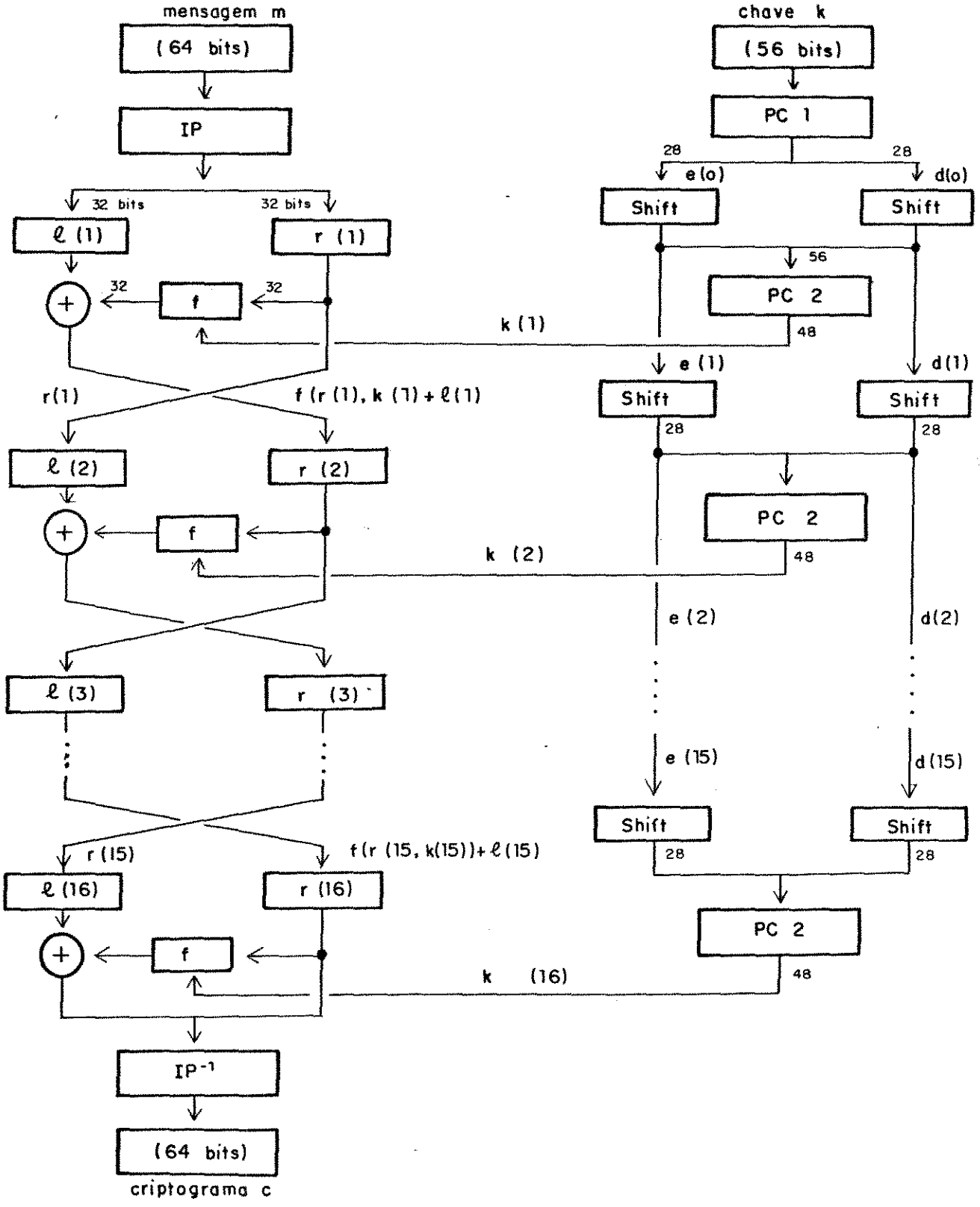


Figura 201: O DES

Cada iteração do DES é composta portanto, pelo produto de duas funções:

A primeira, chamaremos $T(\ell(i),r(i))$ consiste de:

$$T(\ell(i),r(i)) := (\ell(i) + f(r(i),k(i)),r(i)) =$$

$$= (\ell(i + 1),r(i + 1))$$

e a segunda, chamaremos $U(\ell(i + 1),r(i + 1))$ troca as metades:

$$U(\ell(i + 1),r(i + 1)) := (r(i + 1),\ell(i + 1)).$$

Uma iteração é portanto o produto:

$$U \cdot T(\ell(i),r(i)) := (r(i),\ell(i) + f(r(i),k(i))).$$

A Figura 202, a seguir, mostra uma iteração do DES:

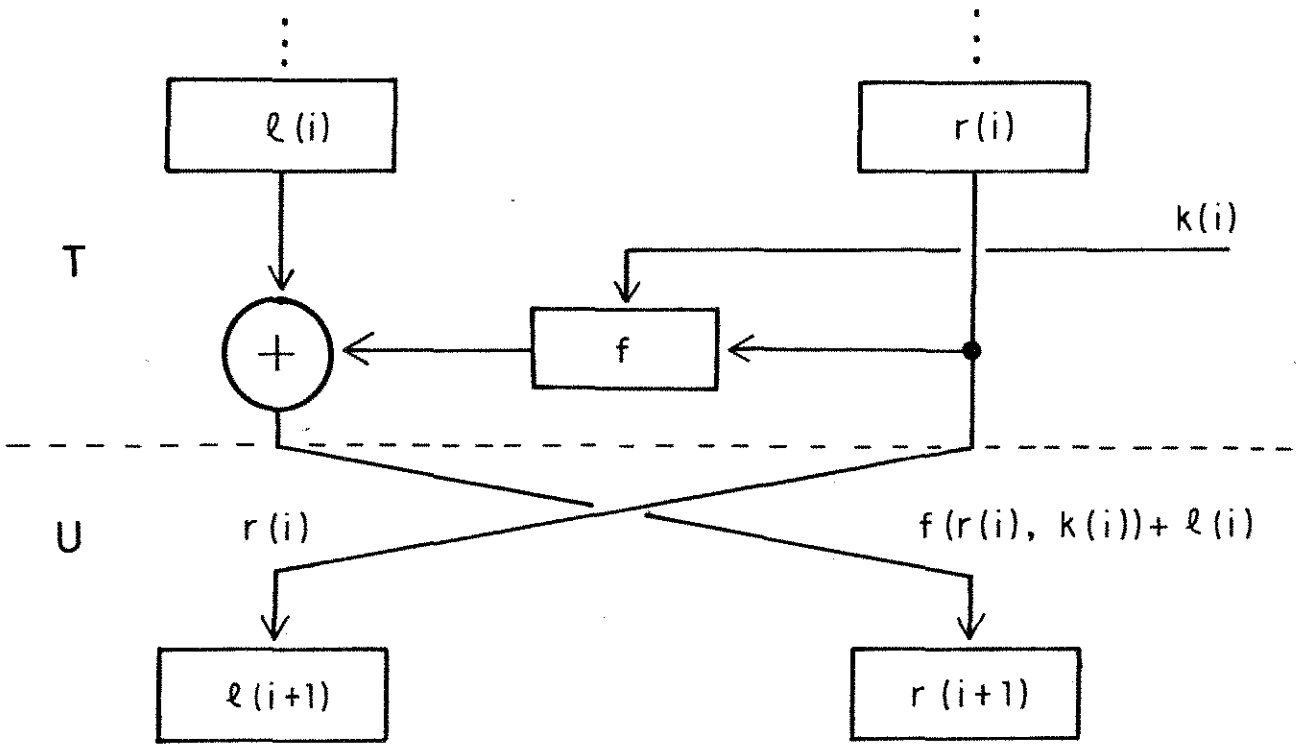


Figura 202: i-ésima iteração do DES.

Tanto a função T quanto a função U são *involuções*, isto é:

- $T(T(x)) = U(U(x)) = x, \qquad \forall x.$

De fato, no nosso caso:

- $T(T(\ell,r)) = T(\ell + f(r,k),\ r) = (L + \underbrace{f(r,k) + f(r,k)}_{0}, r) =$

$$= (\ell, r)$$

- $U(U(\ell,r)) = U(r,\ell) = (\ell,r).$

Chamando à aplicação de U e T na i-ésima iteração, de U_i e T_i, respectivamente, o DES se compõe do produto de 33 funções, a saber:

$$IP^{-1} \cdot T_{16} \cdot U_{15} \cdot T_{15} \cdot \cdot \cdot \quad U_1 \cdot T_1 \cdot IP$$

IP e IP^{-1} são, respectivamente, a permutação inicial e final dos 64 bits da mensagem. Observa-se que U_{16} (o que seria uma última troca) não é executado, havendo simplesmente uma concatenação das duas metades, cujos bits sofrem a permutação final, IP^{-1}, inversa de IP:

$$\cdot\ IP^{-1}(IP(x)) = x, \quad \forall x$$

O ciframento de uma mensagem $\underline{m}$ num criptograma é portanto:

$$c = IP^{-1} \cdot T_{16} \cdot U_{15} \cdot T_{15} \cdot \cdot \cdot \cdot \quad U_1 \cdot T_1 \cdot IP(m)$$

O deciframento é imediato:

$$m = IP^{-1} \cdot T_1 \cdot U_1 \cdot \cdot \cdot \cdot \quad \cdot U_{15} \cdot T_{16} \cdot IP \quad (c),$$

pois:

$$m = IP^{-1} \cdot T_1 \cdot U_1 \cdot \cdots \cdot U_{15} \cdot T_{16} \cdot IP \cdot IP^{-1} \cdot T_{16} \cdot U_{15} \cdot \cdots \cdot U_1 \cdot T_1 \cdot IP(m)$$

Como T e U são involuções, e IP e IP^{-1} são inversas, a expressão acima está correta. O que diferencia T_i de T_j é tão somente a subchave usada. Dado que $U_1 = U_2 = \ldots = U_{15}$, o deciframento consiste em aplicar o *mesmo* algoritmo utilizado para o cifamento tomando as subchaves k(i), i = 1,...,16 na ordem inversa.

Note-se que o processo descrito acima é *independente* da natureza da função f empregada, mas é exatamente nessa natureza que reside o força do algoritmo: se f(r,k) for uma função linear dos bits de r e k, todo o algoritmo será uma funçao linear dos bits da mensagem m e chave k. A operação de ou-exclusivo é linear, assim como a função U; portanto cada bit da saída pode ser escrito como função linear dos bits de m e k; mesmo que cada bit do criptograma dependa de todos os bits de m e k, teremos 64 equações a 110 incógnitas; com mais um criptograma teremos 128 equações lineares a 110 incógnitas, o que provavelmente torna o sistema linear determinado; é necessário portanto que a função f seja não linear: vamos à sua descrição:

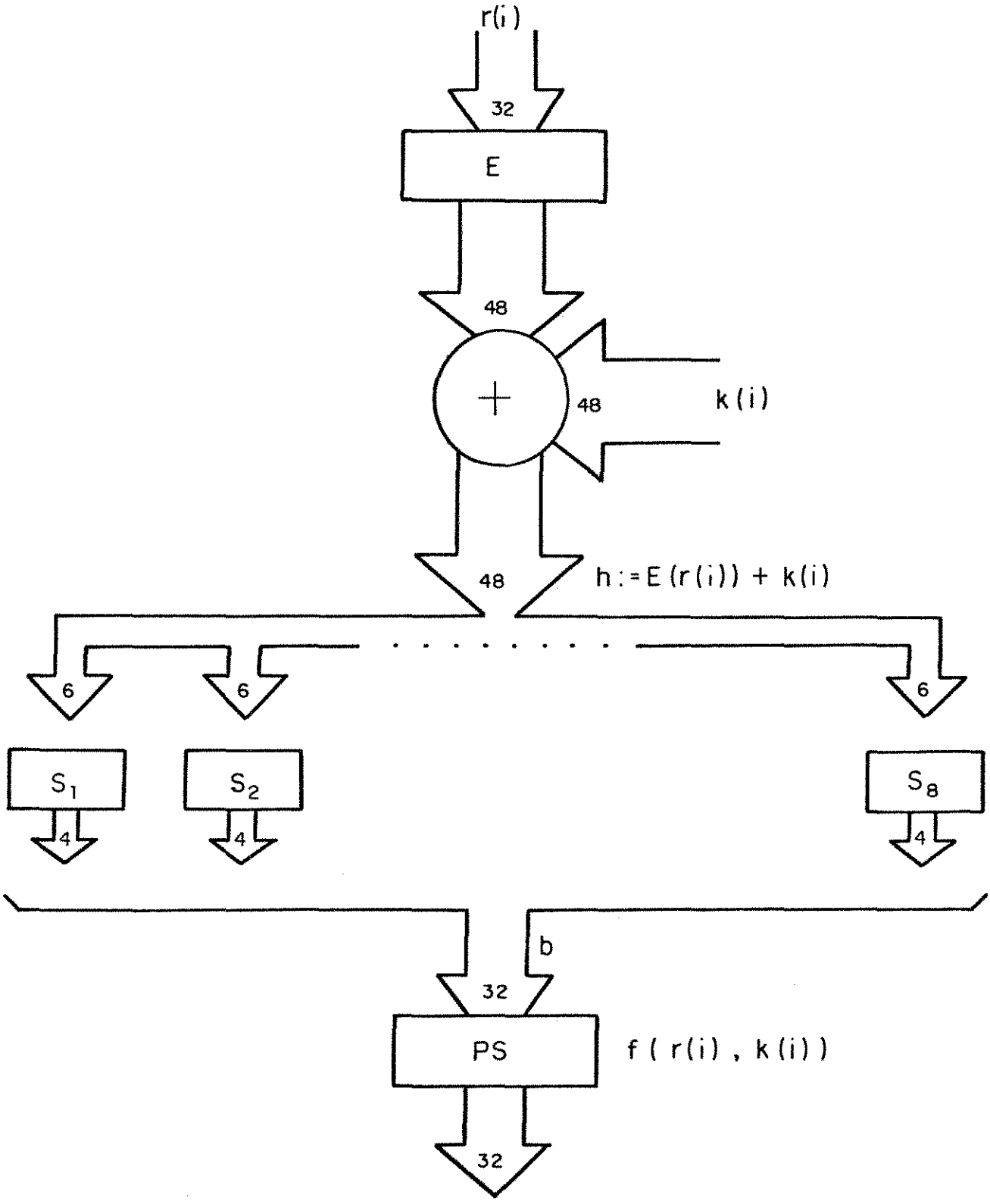


Figura 203: A FUNÇÃO f

De acordo com a figura acima, f age da seguinte maneira sobre r(i) e k(i):

a) r(i) sofre uma expansão E (cópia de determinados bits) de 32 para 48 bits.

b) O resultado E(r(i)) é somado, módulo 2, com k(i), a subchave selecionada para a i-ésima iteração, e que tem também 48 bits, gerando

$$h := E(r(i)) + k(i).$$

c) Os 48 bits de h alimentam, em 8 grupos de 6 bits, 8 funções S_t, $t = 1,\ldots,8$, que podem ser vistas como 8 matrizes diferentes de 4 linhas por 16 colunas, onde cada linha é uma permutação dos inteiros $\{0,\ldots,15\}$; assim cada grupo de 6 bits de h, digamos

$$h_1, h_2, h_3, h_4, h_5, h_6,$$

selecionam uma das 64 posições de S_1, neste caso, da seguinte maneira:

h_1h_6 selecionam uma de 4 linhas

$h_2h_3h_4h_5$ selecionam uma de 16 colunas,

e o conteúdo dessa posição de S_1 $(h_1h_6, h_2h_3h_4h_5)$ é a saída (4 bits) de S_1. Analogamente nas demais funções S_t, os dois bits externos determinan a linha e os quatro internos a coluna.

d) As 8 saídas de S_t são concatenadas, resultando 32 bits, que chamaremos b:

$$b := S_1(h_1 \ldots h_6)\ S_2(h_7 \ldots h_{12}) \ldots S_8(h_{43} \ldots h_{48}).$$

e) Finalmente b sofre uma permutação P, de seus bits, resultando

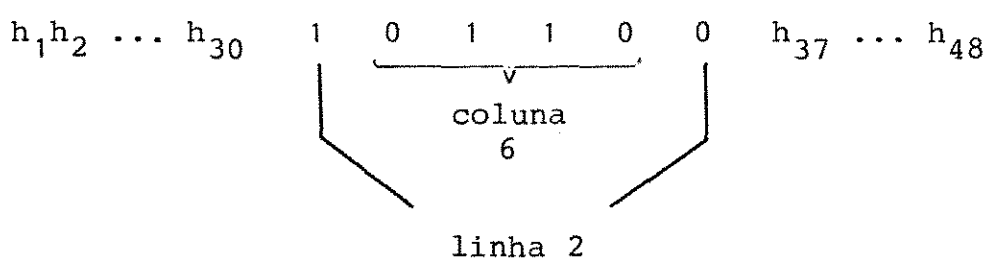
$$f(r(i), k(i)) := P(b).$$

Exemplo da função de seleção S: (neste caso S_5)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
$S_5 =$	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	0
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	1
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	2
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	3

Suponhamos que

$h = (E(r(i)) + k(i)) =$



Portando saída da função S_5 é igual a $S_5(2,6) = 7 = 0111$.

As tabelas descrevendo a permutação IP, as funções S, a expansão E e a permutação P, além das tabelas usadas pelo algoritmo de seleção de chaves descrito adiante, estão relacionadas a seguir.

PERMUTAÇÃO IP:

57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7
56	48	40	32	24	16	8	0
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6

FUNÇÕES S:

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

S_7 :

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8 :

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

EXPANSÃO E

31	0	1	2	3	4
3	4	5	6	7	8
7	8	9	10	11	12
11	12	13	14	15	16
15	16	17	18	19	20
19	20	21	22	23	24
23	24	25	26	27	28
27	28	29	30	31	0

PERMUTAÇÃO P

15	6	19	20
28	11	27	16
0	14	22	25
4	17	30	9
1	7	23	13
31	26	2	8
18	12	29	5
21	10	3	24

SHIFTS $\lambda(i)$

Iteração	nº de Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

PERMUTAÇÃO PC1

49	42	35	28	21	14	7
0	50	43	36	29	22	15
8	1	51	44	37	30	23
16	9	2	52	45	38	31
55	48	41	34	27	20	13
6	54	47	40	33	26	19
12	5	53	46	39	32	25
18	11	4	24	17	10	3

PERMUTAÇÃO PC2

13	16	10	23	0	4
2	27	14	5	20	9
22	18	11	3	25	7
15	6	26	19	12	1
40	51	30	36	46	54
29	39	50	44	32	47
43	48	38	55	33	52
45	41	49	35	28	31

Todas as tabelas, com exceção das caixas S e a da função $\lambda(i)$, devem ser lidas da esquerda para a direita, de cima para baixo; representam permutações dos índices dos bits da entrada. Por exemplo, na matriz da permutação IP, a posição (3,5) é igual a 29; isto significa que o bit 29, trigésimo bit, da esquerda para a direita, da entrada, será o 21º bit da saída, isto é, (2.8) + 5.

Vamos examinar com mais detalhes a constituição de f e em particular de S, a única componente não linear de f, pois P e E o são; aqui entra a idéia de Shannon, de composição das funções transposição e substituição: S é não linear e portanto introduz o que Shannon chama de efeito de *confusão*, mapeando de maneira não linear (confusa) os seus bits de entrada; esse efeito de confusão ficaria porém restrito aos bits de saída de cada função S, não fossem as funções P, que introduzem *difusão* da *confusão* causada por S. Não fossem as funções P, teríamos ao final de 16 iterações, grupos de bits na saída dependendo de grupos de bits da entrada e da chave, quando o ideal seria que cada bit dependesse de todos os bits da entrada e da chave, e de maneira não linear. As funções P cumprem esse papel, como veremos adiante com mais detalhe.

Voltando às funções S, ou *caixas* S como também são chamadas, elas se constituem de 4 substituições cada uma, perfazendo um total de 32 substituições; como cada substituição é selecionada independentemente, temos um total de $4^8 = 2^{16}$ substituições diferentes de 32 em 32 bits. Poder-se-ia pensar em ter todas as $(2^{32})!$ substituições possíveis de 32 em 32 bits. Mas isso exigiria um chave de $\log_2(2^{32})! \approx 32 \cdot 2^{32} = 2^{37}$ bits, o que é impraticável.

A solução foi usar pequenas *caixas* S, com a flexibilidade de poder-se selecionar uma em quatro substituições independentemente;

cada uma das quatro substituições é por sua vez não linear; a maneira de selecionar uma das quatro substituições pode ser mais ou menos simples. No caso do DES, a escolha, como vimos, é feita dependendo de bits da chave e da própria mensagem; esta característica é chamada de *autoclave*: a mensagem desempenhando o papel de chave também.

Seria conveniente fazer-se um resumo das premissas em que o DES baseia sua força:

a) funções S não lineares combinadas com funções P introduzindo o efeito de confusão e difusão;

b) dependência total de cada bit do criptograma de cada bit da entrada e da chave, de maneira "complicada", isto é, cada bit do criptograma é uma função complexa, não linear, dos bits da entrada (via autoclave e via a própria substituição em si) e da chave (idem, idem). A complexidade dessas relações é causada via caixa S e a difusão por todos os bits da entrada e da chave, via função P e as 16 iterações.

É conveniente, a título de ilustração, mostrar como a saída de S_5 está relacionada aos bits da entrada:

A saída terá, como vimos, quatro bits (y_1, y_2, y_3, y_4) e podem ser escritos como função (soma de produtos) dos seis bits da entrada $(x_1, x_2, \ldots, x_6)$, estes resultantes de operações de ou-exclusivo dos bits convenientes da mensagem e chave:

$$y1 = \overline{x1}\,\overline{x2}\,\overline{x4}\,\overline{x5}\,\overline{x6} + \overline{x1}\,\overline{x3}\,x4\,\overline{x5}\,\overline{x6} + x1\,x3\,\overline{x4}\,\overline{x5}\,x6 +$$

$$x1\,\overline{x2}\,x3\,\overline{x4} + x1\,x2\,\overline{x3}\,x6 + x2\,x4\,x5\,x6 + x2\,\overline{x3}\,\overline{x5}\,x6 +$$

$$\overline{x1}\,x2\,x5 + \overline{x1}\,\overline{x4}\,x6 + x4\,x5 + \overline{x3}\,x6 + \overline{x1}$$

$$y2 = \overline{x_1}\,x2\,\overline{x3}\,\overline{x4}\,\overline{x6} + x3\,\overline{x4}\,\overline{x5}\,x6 + x_1\,\overline{x4}\,x5\,x6 + x_1\,\overline{x3}\,x4\,x5 +$$

$$x2\,\overline{x4}\,x6 + x2\,\overline{x3}\,\overline{x4} + \overline{x_1}\,x2\,\overline{x4} + x_1 + x2 + x3 + x4 + x5 + x6$$

$$y3 = \overline{x_1}\,\overline{x2}\,\overline{x3}\,\overline{x5}\,\overline{x6} + \overline{x_1}\,x2\,\overline{x3}\,\overline{x4}\,\overline{x6} + \overline{x_1}\,x3\,\overline{x4}\,\overline{x5}\,x6 +$$

$$x_1\,x2\,\overline{x4}\,\overline{x5}\,x6 + x_1\,\overline{x2}\,x3\,\overline{x5} + x_1\,\overline{x4}\,\overline{x5} + \overline{x_1}\overline{x3}\,x6 + \overline{x2}\,\overline{x4}\,\overline{x5}$$

$$+ \overline{x3}\,x5 + x4\,x6 + x_1\,\overline{x6} + x2\,x5 + 1$$

$$y4 = \overline{x1}\,x3\,x4\,x5\,\overline{x6} + x1\,x2\,\overline{x3}\,\overline{x5}\,\overline{x6} + \overline{x1}\,\overline{x2}\,x4\,x5 + x1\,\overline{x2}\,\overline{x3}\,x4 +$$

$$x2\,\overline{x4}\,\overline{x5}\,\overline{x6} + \overline{x1}\,x3\,\overline{x6} + \overline{x3}\,x5\,\overline{x6} + x2\,\overline{x3}\,x5 + x3\,x4 +$$

$$x1\,x6 + \overline{x2}\,\overline{x5} + 1.$$

Se a escolha das substituições em cada caixa S fosse feita baseando-se somente em bits da chave, a relação ainda seria complicada e não linear mas não tanto como a implementada no DES, pois se perderia o efeito de autoclave.

Análises de interdependência entre os bits do criptograma e os bits da entrada e da chave mostram que [MEYE-82]:

a) Após 5 iterações *todos* os bits do criptograma dependem de todos os bits da mensagem via autoclave.

b) Após 10 iterações *todos* os bits do criptograma dependem de todos os bits da chave via autoclave.

Essa forte dependência é ilustrada pelos efeitos causados no criptograma pela mudança de um único bit da mensagem e/ou da chave, e que Feistel chamou de "efeito avalanche" [MEYE1-82] [KONH1-81]. A Tabela 1 a seguir mostra as mudanças, como função do números de bits diferentes entre os resultados de cada uma das 16 iterações do DES, quando se modifica apenas *um* bit de uma mensagem arbitrária, mantendo-se uma mesma chave, escolhida aleatoriamente. A Tabela 2 mostra essas mudanças quando se troca apenas *um* bit da chave e se mantém a mensagem fixa [KONH1-81]:

iteração	nº de bits diferentes
0	1
1	6
2	21
3	35
4	39
5	34
6	32
7	31
8	29
9	42
10	44
11	32
12	30
13	30
14	26
15	29
16	34

TABELA 1

iteração	nº de bits diferentes
0	0
1	2
2	14
3	28
4	32
5	30
6	32
7	35
8	34
9	40
10	38
11	31
12	33
13	28
14	26
15	34
16	35

TABELA 2

· Algoritmo de Seleção de Chaves

A escolha das subchaves $k(1), k(2), \ldots, k(16)$ se dá também iterativamente (ilustrada na Figura 201), da seguinte maneira:

a) os 56 bits de k sofrem uma permutação inicial PC1, seguida de uma divisão em duas metades $e(0)$, $d(0)$.

b) $e(0)$, $d(0)$ sofrem, separadamente, um shift circular à esquerda de $\lambda(1)$ posições, onde $\lambda(i)$, $i = 1, \ldots, 16$, vale 1 ou 2; assim, um "shift" de uma ou duas posições à esquerda é executado dependendo da iteração. Esses "shifts" dão origem a $e(1)$ e $d(1)$, que, concatenados ($28 + 28 = 56$ bits), alimentam a função PC2.

c) PC2 é uma seleção dos bits de $e(1)$, $d(1)$, em que 8 bits são retirados dando origem a $k(1)$, a subchave que alimentará f na primeira iteração. A descrição de PC1, PC2 e λ está feita juntamente com as outras tabelas do DES.

O algoritmo de seleção das subchaves $k(1), \ldots, k(16)$ é tal que, a partir da chave k, se os "shifts" forem executados à esquerda obtém-se $k(1), \ldots, k(16)$ sucessivamente; se forem porém executados à direita, e fazendo $\lambda(1) = 0$, obtém-se $k(16), \ldots, k(1)$ sucessivamente; completa-se assim o processo de deciframento, praticamente idêntico ao de ciframento, com exceção do sentido dos shifts $\lambda(i)$ e do valor de $\lambda(1)$.

Esse processo de escolha de chaves pode gerar $k(1) = k(2) = \ldots = k(16)$, se k for, o que se convencionou chamar, uma *chave fraca*:

Por exemplo, se

$k = 00 \ldots 0$ - 56 bits iguais a zero, ou

$k = 11 \ldots 1$ - 56 bits iguais a um, então

$k(1) = k(2) = \ldots = k(16)$; isto reduz o DES a

$$c := IP^{-1} . T . U . T . U \ldots\ldots U . T . IP\ (m),$$

pois as funções T_i são distinguidas pelas diferentes subchaves usadas nas 16 iterações; é possível assim construir um ataque, e descobrir qual é essa chave fraca em tempo $0(2^{32})$ como segue:

Desprezaremos momentâneamente as permutações IP e IP^{-1}; assim, uma mensagem m e o correspondente criptograma c serão vistos como a concatenação de suas duas metades:

$[\ell(1), r(1)]$, a mensagem m,

$[\ell(17), r(17)]$, o criptograma c, resultante das 16 iterações do DES.

Suponhamos que o criptoanalista tem acesso ao dispositivo

de ciframento que usa uma chave fraca, e assim consegue cifrar uma mensagem arbitrária m, obtendo c.

Conhecendo $[\ell(1), r(1)]$, o criptoanalista conhece também $\ell(2)$, a metade esquerda após a primeira iteração, pois $\ell(2) = r(1)$; não conhece porém $r(2)$, pois $r(2) = f(r(1),k) + \ell(1)$, onde k é secreta.

A seguir, o criptoanalista cifra as 2^{32} mensagens correspondentes a

$$[\ell(2),\ i\ ],\quad 0 \leq i \leq 2^{32} - 1,$$

que darão origem a 2^{32} criptogramas correspondentes a

$$[\ell_i(18),\ r_i(18)\ ],\quad 0 \leq i \leq 2^{32} - 1.$$

Quando

$$r_j(18) = r(17),\quad \text{para algum } j,\ \text{então}$$

j é um valor possível para $r(2)$; como

$$r(2) = f(r(1),k) + \ell(1),$$

basta resolver a equação acima em k; pode ser que o $r(2)$ testado seja um "alarme falso", e assim seja necessário buscar outro valor de j até encontrar o valor correto de $r(2)$, e portanto

de 48 bits da chave fraca.

• Outras considerações sobre chaves fracas e *semi-fracas* (geram dois tipos de subchaves apenas) estão em [MEYE1-82], com grande riqueza de detalhes.

É importante notar que a chave k tem na realidade 64 bits, 8 dos quais (um por bite) são usados como bit de paridade, para corrigir eventuais erros.

• CRIPTOANÁLISE DO DES

O trabalho de decifrar uma mensagem, cifrada com DES, pode ser resumido em três alternativas: busca exaustiva, análise estatística e formulação analítica.

A busca exaustiva pode, por sua vez, ser feita no espaço de chaves ou no espaço de mensagens. No caso de busca exaustiva de chaves, o criptoanalista tenta encontrar a chave que foi usada no ciframento de uma mensagem em seu poder; ou seja, dados m e c, um par mensagem-criptograma conhecido, ele produz todos os valores de k tal que $E(k,m) = c$.

Fazer análise estatística significa estabelecer relações de natureza estatística entre os bits do criptograma e os da mensagem; no DES estas relações são inviabilizadas pelo espalhamento causado pelas funções P, e pela alta interdependência entre os bits de c e de m e k, de forma que não há quase correlação entre grupos de poucos bits.

O trabalho de formulação analítica, ao contrário de método da força bruta, expressa os bits da saída (criptograma) como função dos bits da entrada, e assim resolve essas expressões para deduzir o valor da chave utilizada. Se essas relações forem simples, a solução pode ser fácil; se, porém, as relações forem não lineares e bastante complicadas, como no DES, a tarefa se torna quase impossível.

• Busca Exaustiva no Espaço de Chaves

"Embora grande, o espaço de chaves (2^{56} diferentes) é possível de esgotamento, ou busca exaustiva, usando-se paralelismo". Com essa afirmação iniciou-se uma longa polêmica [DIFF 2-76], [DIFF 3-77],[YASA1-76], [HELL1-79], [DAVI1-79],[TUCH1-79],[BRAN1-79] sobre a possibilidade de se realizar busca exaustiva da chave k, tendo em mãos um par mensagem-criptograma. A busca se faria com uso de paralelismo: 1 milhão de chips DES examinariam, em paralelo, trechos disjuntos do espaço de chaves, até encontrar k tal que E(k,m) = c. Cálculos [DIFF 3-77] estimam que uma máquina desse porte levaria, em média, meio-dia para realizar a busca e no máximo um dia. Os custos de construção, dizia-se em 1977, seriam de 20 milhões de dólares, que depreciados ao longo de 10 anos, e levando em conta a queda do custo do hardware (uma ordem de magnitude a cada cinco anos), resultariam num custo de 50 dólares por busca, em média. Outros argumentos [MEIS1-76] afirmam que uma máquina desse tipo não seria factível antes de 1990, e com custo de dezenas de milhões de dólares.

Os cálculos em [DIFF 3-77] foram reavaliados [DIFF 4-81] e subiram para 50 milhões de dólares de custo de construção, e média de dois dias para realizar uma busca. De qualquer maneira, prevê-se que por volta de 1990, a segurança do DES, se usado na forma atual, estará comprometida. Há duas maneiras de se melhorar o nível de segurança do DES:

a primeira consiste em aumentar o tamanho da chave, e a segunda em fazer múltiplos ciframentos usando chaves diferentes.

O aumento do tamanho da chave detemina mudanças no algoritmo, e consequente obsolescência dos equipamentos de criptografia baseados no atual DES. O custo da elevação do tamanho da chave, de 64 (56 na realidade) bits para 128 (112), seria mínimo, pois as partes dependentes do tamanho da chave perfazem um máximo de 10% dos componentes de um chip [DIFF 3-77, pag. 77]. Por outro lado, uma chave de 112 bits deve prevenir qualquer tentativa de busca exaustiva, pois o espaço de chaves cresceria para $2^{112} \approx 10^{34}$; nesse caso, usando o mesmo equipamento proposto, o tempo médio de busca aumentaria para 10^{17} dias, ou 10^{23} chips/dias, o que parece impraticável.

A segunda maneira consiste em fazer múltiplos ciframentos da mensagem usando chaves diferentes em cada um deles:

$$c = E(k_n, E(k_{n-1}, \ldots, E(k_1, m) \ldots));$$

nem sempre fazer cifrамento múltiplo aumenta a segurança do sistema usado: fazer duas substituições monoalfabéticas, por exemplo, corresponde a fazer uma apenas. No caso do DES, a composição de cifrамento é mais forte do que o cifrамento simples [KONH1-81]: por outro lado, não há provas de que o cifrамento duplo:

$E(k_2,E(k_1,m))$ produza $2^{56} \cdot 2^{56} = 2^{112}$

permutações diferentes; é possível que existam k_i, k_j e k_ℓ tais que

$$E(k_i,E(k_j,m)) = E(k_\ell,m), \quad \forall m .$$

O DES pode ser visto como uma grande função de substituição S, não linear, que permuta inteiros no intervalo $[0,2^{64}-1]$. Existem $2^{64}!$ permutações diferentes; como cada chave determina uma delas, para realizarmos todas as permutações, seria necessário uma chave com $\log(2^{64})! \approx 2^{64} \cdot \log 2^{64}$ bits, o que é impraticável. No DES temos apenas 2^{56} permutações possíveis; daí usar-se a composição de cifrамentos.

No esquema de duas chaves:

$$c = E(k_2 , E(k_1,m));$$

Diffie & Hellman [DIFF 3-77] mostraram que, com um compromisso entre tempo e espaço (ambos da ordem de 2^{56}), é possível construir um ataque, usando-se um par mensagem-criptograma conhecido (m,c), da seguinte maneira:

Para cada valor possível da chave k, é gerado $x := E(k,m)$; os 2^{56} valores assim obtidos são armazenados numa tabela que

conterá, para cada k, os valores x e k. Esta tabela é, a seguir, ordenada segundo x, usando um algoritmo linear.

Analogamente, uma outra tabela é gerada contendo os 2^{56} valores resultantes do deciframento de c sob todas as chaves possíveis; isto é $y := D(\ell,c)$, $0 \leq \ell \leq 2^{56} - 1$. Esta tabela é também ordenada segundo y.

É feita, a seguir, uma intercalação das duas tabelas, segundo os valores de x e y; quando x = y, teremos k e ℓ como candidato a k_1 e k_2 respectivamente.

Um outro esquema de ciframento múltiplo, usando duas chaves e três iterações, foi proposto por Tuchman [TUCH 2-78] e consiste de

$$c = E(k_1 , E^{-1}(k_2 , E(k_1,m)))$$

isto é, m é cifrada sob k_1, gerando m_1; esta é decifrada sob k_2 gerando m_2, que é cifrada novamente sob k_1 originando o criptograma c. É interessante notar que se $k_1 = k_2$ o esquema é compatível com o ciframento simples sob k_1. A Figura abaixo ilustra os sucessivos cifamentos:

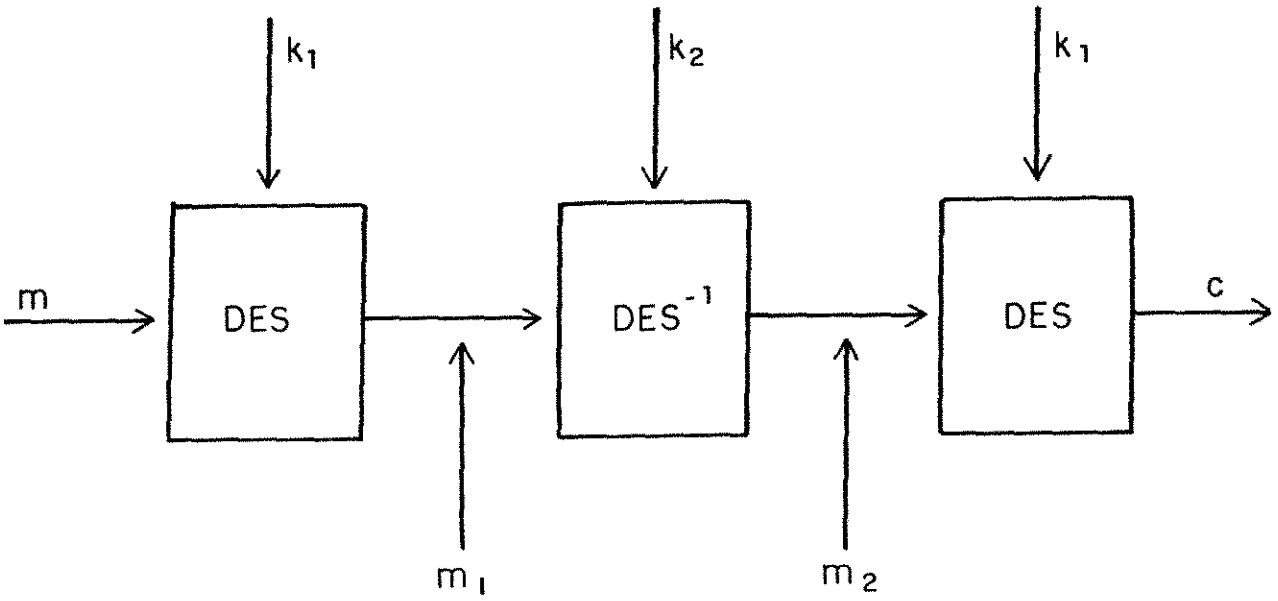


Figura 204: CIFRAMENTO TRIPLO SOB k_1 e k_2

Novamente, é possível construir um ataque de complexidade de tempo e espaço da ordem de 2^{56} para descobrir as duas chaves k_1 e k_2 [MERK 2-81]. A descrição deste ataque leva em conta a possibilidade de se usar o dispositivo de ciframento triplo para cifrar uma mensagem escolhida pelo criptoanalista:

• O criptoanalista, inicialmente, computa os 2^{56} valores

$$x := E^{-1}(k, m_1), \quad 0 \leq k \leq 2^{56} - 1:$$

isto é, calcula o resultado do deciframento de uma mensagem arbitrária m_1 sob todas as 2^{56} chaves possíveis; o valores x e k são armazenados numa tabela T_1 que é, a seguir, ordenada segundo x, usando um algoritmo linear.

• Usando os 2^{56} valores de x, o criptoanalista calcula os 2^{56} valores

$$y := E(k_1, E^{-1}(k_2, E(k_1, x)));$$

isto é, o dispositivo de ciframento triplo é usado para produzir os 2^{56} valores de y.

• A seguir, calcula-se os 2^{56} valores:

$$z := E^{-1}(\ell, y), \quad 0 \leq \ell \leq 2^{56} - 1,$$

que são armazenados, juntamente com ℓ, numa tabela T_2, ordenada, a seguir, segundo z.

Os valores de z são, portanto, o resultado de:

$$E^{-1}(\ell,E(k_1,E^{-1}(k_2,E(k_1,E^{-1}(\ell,m_1))))),\quad 0 \leq \ell \leq 2^{56} - 1;$$

quando $\ell = k_1$, temos que

$$z = E^{-1}(k_2,m_1).$$

• Basta aos criptoanalista fazer agora uma intercalação de T_1 e T_2, detectando as posições em que $x = z$; assim, ℓ será um candidato a chave k_1, e k será um candidato a chave k_2.

Embora seja impraticável realizar ataques como os descritos acima, em que é preciso armazenar uma tabela gigantesca, as análises feitas por Diffie & Hellman são indicativos de alguma fraqueza. A complexidade de tempo, $0(2^{56})$, não é grande como se esperava ($0(2^{112})$) e é passível de diminuição usando paralelismo; assim, é recomendável que o esquema de ciframento múltiplo seja usado com no mínimo 3 chaves, o que eleva para $0(2^{112})$ a complexidade de tempo e espaço dos ataques descritos. Por exemplo, o esquema de Tuchman, com k_3 no último ciframento, se torna:

$$c = E(k_3,E^{-1}(k_2,E(k_1,m)));$$

se $k = k_1 = k_2 = k_3$ conserva-se a equivalência com o ciframento simples sob k. A Figura 205, abaixo ilustra esses ciframento.

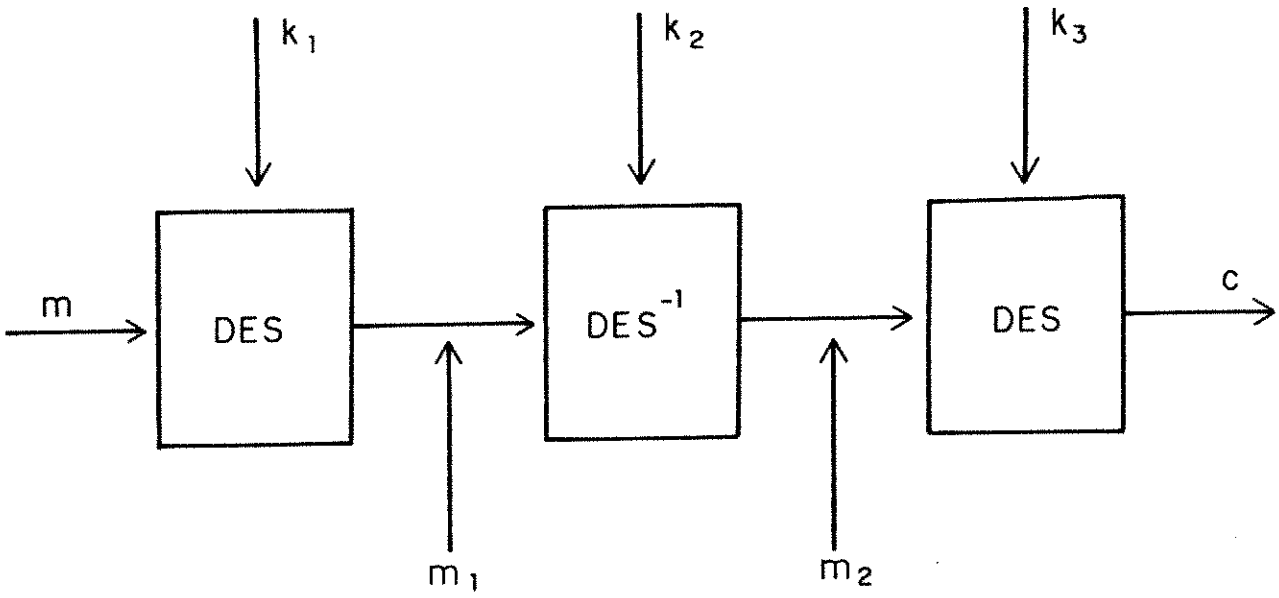


Figura 205: CIFRAMENTO TRIPLO SOB k_1, k_2 E k_3

• Análise Estatística

Em [KONH1-81], encontra-se muito bem detalhada, a parte publicada da análise estatística feita para validação do DES.

Concluiu-se que após 8 das 16 iterações desaparece a dependência estatística de bits do criptograma em relação a poucos bits da mensagem, para uma chave k fixa; esse teste *não* mostra que todos os subconjuntos dos 64 bits do criptograma são independentes de todos os subconjuntos dos bits da mensagem, mas que essa dependência é baixa o suficiente para não ter valia para um criptoanalista.

· Formulação Analítica

Por outro lado, pode-se estabelecer analiticamente a dependência de cada bit do criptograma com os bits da mensagem e da chave: é possível escrever cada bit do criptograma como uma soma de produtos dos bits da mensagem e chave, como feito para a caixa S_5 na descrição do DES. Para que essas relações sejam úteis a um criptoanalista, é necessário que ou as somas tenham poucos termos ou os produtos tenham poucos fatores.

A caixa S_5 já dá uma idéia da complexidade que teria a equação de apenas *um* bit do criptograma; como dissemos, na 10^{a} iteração cada bit do criptograma depende de maneira complexa (via autoclave e via substituição) de todos os bits da mensagem e chave. A função de substituição é a única componente não linear do sistema.

• CONTROVÉRSIAS

Quando a IBM expôs o algoritmo à comunidade científica e à aprovação do NBS, ela não explicou o critério de construção das caixas S por imposição da NSA - National Security Agency americana. Este orgão declarou que os critérios utilizados pela IBM reproduziam algumas idéais já desenvolvidas pela NSA, que queria obviamente mantê-las em segredo. Isso gerou o temor de que a NSA soubesse como decifrar mensagens cifradas com o DES; O DES seria assim suspeito demais para ser adotado como padrão governamental. Havia acusações também de que a NSA havia interferido no projeto do DES, introduzindo "atalhos" no algoritmo, de maneira a possibilitar que pessoas que não possuíssem a chave, mas que conhecessem esses atalhos, realizassem a criptoanálise mais facilmente. Até hoje não se sabe qual o envolvimento real da NSA [SEN-78], [SHAP1-78]. Após algumas discussões, a NSA revelou alguns dos critérios usados [KONH1-81, pag. 248, 249] e [MEYE1-82, pag. 163]:

1) nenhuma caixa S é uma função linear, ou afim, da sua entrada.

2) Mudando um bit da entrada de uma caixa S, mudam-se no mínimo dois bits da saída.

3) A complexidade das caixas S é medida pela quantidade mínima de componentes lógicos necessários para expressar as 4 saídas

em relação às 6 entradas. O que surpreendeu os projetistas, é que se cada uma das quatro linhas da matriz que representa a função S, fosse gerada a partir de uma permutação aleatória dos inteiros 0, 1, ... , 15, nem sempre a caixa S seria "complexa" o suficiente; daí a presença de algumas estruturas nas caixas S, detectadas em [HELL 3-76]. Diz-se que à época do projeto obedeceu-se a um compromisso: o maior número de componentes possível, mas com a limitação de que o DES todo deveria caber num único "chip"; mas não se explica porque aquelas estruturas e não outras.

Um trabalho de Gordon & Retkin [GORD1-82] analisa o tamanho da entrada e da saída das caixas S, e conclui que quanto maior a caixa S, menor a probabilidade dela conter alguma linearidade, ainda que os elementos das linhas sejam escolhidos aleatoriamente.

Análises independentes do DES [HELL 3-76] foram feitas, onde se constatou a presença de estruturas e até de redundâncias (a caixa S_4 é 75% redundante - três linhas podem ser deduzidas de uma quarta). Apesar dessas considerações e de outras análises, os autores do trabalho não indicam uma maneira efetiva de criptoanálise do DES, mas reforçam a suspeita de que as estruturas encontradas sejam indicativas de alguma fraqueza propositalmente introduzida. Por outro lado, recomendam que se dobre o tamanho da chave.

CAPÍTULO 3

CRIPTOSSISTEMAS ASSIMÉTRICOS

Uma fonte natural de funções unidirecionais-alçapão é a classe de problemas NP-difíceis. Essa constatação foi feita por Diffie & Hellman [DIFF1-76] e explica-se: se o algoritmo E de ciframento for uma função, tal que o cálculo de sua inversa seja equivalente a resolver um problema reconhecidamente difícil, teremos o efeito unidirecional desejado; fica claro que o cálculo da inversa pode ser possível à custa de alguma informação adicional: daí o efeito alçapão. Essa assimetria de dificuldades também está presente nos problemas NP-difíceis: dada uma proposta de solução x, de um problema S, NP-difícil, é possível verificar se x é ou não solução, em tempo polinomial; calcular, porém, uma solução desse problema S, acredita-se que seja uma tarefa difícil.

A grosso modo, o algoritmo E seria assim, o equivalente a produzir um "exemplo" de um problema de NP-difícil a partir de m e k, respectivamente a mensagem e chave usadas; equivalentemente, o algoritmo D de deciframento consistiria em resolver esse "exemplo" recuperando m. É importante notar que qualquer problema difícil, com as características acima, mesmo que não NP-difícil, é candidato a função unidirecional-alçapão.

Desde o artigo de Diffie & Hellman, um grande esforço tem

sido feito para se encontrar problemas NP-difíceis que admitam algum tipo de alçapão; a pergunta imediata é se esses problemas, ou a subclasse desses problemas que admitem alçapão, são tão difíceis quanto os problemas originais: disso depende toda a segurança do criptossistema. Além disso, a teoria da NP-completude se baseia na hipótese do pior caso: alguns exemplos do problema não têm solução polinomial conhecida. Se a maioria dos exemplos, por outro lado, for solúvel em tempo polinomial, esse problema não tem utilidade criptográfica. Um resultado mais recente de Lagaryas & Odlyzko [LAGA1-83] ilustra essas questões: no artigo, os dois autores mostram que o conhecido problema da mochila tem solução probabilística polinomial, para uma certa classe de exemplos, justamente a classe que era útil do ponto de vista criptográfico. Lempel [LEMP1-79] mostra um outro exemplo do uso do problema da mochila para gerar uma função unidirecional-alçapão, a ser usada num criptossistema simétrico; embora o deciframento, no exemplo de Lempel, seja equivalente a resolver o problema da mochila, a linearidade entre mensagem (solução do problema da mochila) e criptograma (exemplo do problema) destrói toda a força do algoritmo.

Essas considerações negativas não desmerecem o uso de problemas difíceis em funções unidirecionais-alçapão, mas ilustram o cuidado que se deve ter na escolha do problema.

Nos criptossistemas assimétricos, a chave de ciframento é pública, e o seu conhecimento não habilita ninguém a deduzir a

chave de deciframento. Veremos a seguir como essa situação pode ser conseguida, com o uso de problemas difíceis para gerar funções unidirecionais-alçapão. Dois exemplos serão apresentados: o primeiro usa o problema da fatoração de inteiros que é reconhecidamente difícil; o segundo, de Merkle & Hellman [MERK 3-78] usa uma certa subclasse de problemas da mochila.

3.1. O CRIPTOSSISTEMA DE RIVEST, SHAMIR E ADLEMAN (RSA)

O criptossistema bidirecional assimétrico denominado RSA (iniciais dos seus propositores: Rivest, Shamir e Adleman) [RIVE1-78], usa um algoritmo de ciframento que extrai sua força de um problema muito difícil: "Dado um inteiro n descobrir seus fatores primos". Esse problema tem origem na antigüidade, passando por Fermat e Legendre, e até hoje permanece difícil. O melhor algoritmo conhecido para fatoração de um número n leva tempo proporcional a $0(e^{\sqrt{\ell n\, n\, \ell n \ell n\, n}})$, o mesmo que o obtido para resolver logaritmos modulo p, p primo. O algoritmo é de autoria de Ricard Schroeppel e não foi publicado. Calcula-se que a fatoração de um número de 200 dígitos decimais, usando esse algoritmo, levaria cerca de 4 bilhões de anos. Outros algoritmos de fatoração de inteiros estão em [POLL1-74], [KNUT 2-69], [BREN1-80].

• GERAÇÃO DO CRIPTOSSISTEMA

Como vimos no Capítulo 1, um usuário A publica, no catalogo de chaves públicas, seu nome associado à sua chave pública, que consiste, neste caso, de um par (e,n) onde:

• n = pq com p e q primos "grandes" (100 dígitos decimais); n tem, portanto, 200 dígitos decimais.

• e é tal que mdc((p - 1)(q - 1),e) = 1.

A cada e corresponde d, tal que:

$$mdc(d,(p-1)(q-1)) = 1 \quad e$$

$$ed \equiv 1 \bmod (p-1)(q-1);$$

O par (e,n) é a chave pública, e (d,n) a chave mantida secreta (na verdade apenas d, obviamente).

• ALGORITMO DE CIFRAMENTO

Um usuário B querendo cifrar a mensagem m, $0 \leq m < n$, após obter (e,n), a chave pública de A, computa o criptograma c:

$$c = E((e,n),m) = m^e \bmod n,$$

e envia c para A.

• ALGORITMO DE DECIFRAMENTO

O usuário A, usando sua chave secreta (d,n), decifra c recebendo m:

$$m = D((d,n),c) = c^d \bmod n.$$

Resumindo

$$(m^e)^d \bmod n = m^{ed} \bmod n = m, \quad 0 \leq m < n.$$

Os resultados acima se justificam pois:

• A quantia $\varphi(n) = (p - 1)(q - 1)$, dita função de Euler [HARD1-38] se traduz por:

$$\varphi(n) = |\{k,\ 1 \leq k < n \ /\ mdc(k,n) = 1\}|;$$

em particular, para p primo, $\varphi(p) = p - 1$; se $n = p \cdot q$, p e q

primos, $\varphi(n) = (p-1)(q-1)$

• O teorema de Fermat [HARD1-38] diz que:

Se p é primo então

• $m^{p-1} \equiv 1 \bmod p$, $\quad mdc(m,p) = 1$.

• obviamente $m^{p-1} \equiv 0 \bmod p$, se $\quad mdc(m,p) = p$.

Em ambos os casos:

$$m^{\varphi(n)+1} \equiv m \bmod p, \ \forall m, \quad \text{e também}$$

$$m^{k\varphi(n)+1} \equiv m \bmod p, \ \forall m, \ k \geq 0 \quad \text{inteiro.}$$

Analogamente,

$$m^{k\varphi(n)+1} \equiv m \bmod q, \ \forall m, \ k \geq 0 \quad \text{inteiro, } q \quad \text{primo.}$$

Assim, se $n = p \cdot q$

$$m^{k\varphi(n)+1} \equiv m \bmod n, \quad \forall m, \quad k \geq 0.$$

Em particular

$$m^{k\varphi(n)+1} \equiv m \bmod n, \quad 0 \leq m < n, \quad k \geq 0.$$

Dado que

$ed \equiv 1 \mod (p-1)(q-1)$, reescrevemos

$ed = k\varphi(n) + 1$, para algum $k > 0$, e portanto

$m^{ed} \bmod n = m^{k\varphi(n)+1} \bmod n = m, \quad 0 \leq m < n.$

• GERAÇÃO DOS PRIMOS p E q

O usuário A, ao tentar gerar o módulo n, deve encontrar os primos "grandes" p e q. O problema de testar primalidade não é simples, e os algoritmos eficientes existentes, ou são probabilísticos [RABI 3-76], [SOLO1-77], isto é, determinam se um número é ou não primo com alta probabilidade de acerto, ou não tem sua complexidade efetivamente avaliada.

Para escolher p e q, pode-se gerar aleatoriamente números ímpares de 100 dígitos, e testar sua primalidade usando algum dos algoritmos acima. Pelo teorema do número primo [HARD1-38], aproximadamente 115 candidatos terão que ser testados, em média, até se encontrar um primo; descreveremos o teste de Solovay & Strassen [SOLO1-77] e o de Rabin [RABI 3-76] para teste de primalidade:

• ALGORITMO DE SOLOVAY & STRASSEN PARA TESTE DE PRIMALIDADE

1) Tome k, inteiro, aleatoriamente, no intervalo $[1, p-1]$, onde p é o número cuja primalidade será testada.

2) Calcule mdc(k,p);

• se $mdc(k,p) \neq 1$ então p não é primo; caso contrário calcule

• $a := k^{(p-1)/2} \bmod p$, e

• b := J(k,p), o símbolo de Jacobi, e que pode ser computado da seguinte maneira:

Se k = 1

então J(k,p) := 1

senão se k é par

então $J(k,p) := J(k/2,p)(-1)^{(p^2-1)/8}$

senão $J(k,p) := J(p \bmod k,p)(-1)^{(k-1)(p-1)/4}$

3) Se $a \neq b$ então p *não* é primo.

Se a = b então p é composto com probabilidade menor que ½.

Portanto, se para 100 valores de k diferentes, escolhidos aleatória e independentemente, o valor de a é igual ao de b, então p é primo com probabilidade de erro menor que $\frac{1}{2^{100}}$, o que, para efeitos práticos, é desprezível; se porém, a é diferente de b para algum valor de k, então p é composto. O tempo de execução do algoritmo é proporcional a $m \log_2 p$, onde m é o número de k's considerados, e p é o número testado.

· ALGORITMO DE RABIN PARA TESTE DE PRIMALIDADE

1) Tome k, inteiro, aleatoriamente, no intervalo $[1, p-1]$, onde p é o número cuja primalidade será testada.

2) Verifique se valem as seguintes afirmações:

(i) $k^{p-1} \not\equiv 1 \bmod p$,

(ii) $\exists n \;/\; \frac{p-1}{2^n}$ é inteiro, com $1 < mdc(k^{(\frac{p-1}{2^n})} - 1, p) < p$;

se valem (i) ou (ii) então p é composto.

Rabin cita o seguinte teorema [RABI 3-76]:

"Se p é composto então existem pelo menos $\frac{p-1}{2}$ valores de k, $1 \leq k < p$, tal que (i) ou (ii) se verificam".

Analogamente ao algoritmo de Solovay & Strassen, tomando aleatoriamente e independentemente m valores de k, e verificando (i) ou (ii), concluimos que, se as duas condições não se verificam para algum dos m valores de k, p é primo com probabilidade de erro menor que $\frac{1}{2^m}$.

O tempo de execução do algoritmo é proporcional a $m \log p$ pois os cálculos de k^{p-1}, $k^{(\frac{p-1}{2^n})}$, 2^n e máximo divisor comum são factíveis em tempo $0(\log p)$.

Miller [MILL1-75] afirma que se a hipótese generalizada de

Riemann for verdadeira, então existe uma constante c, tal que, se p é composto, existe k, $1 \leq k \leq c\,(\log p)^2$ em que (i) ou (ii) se verificam. Miller não especifica, porém, qual a magnitude dessa constante c. Bastaria, portanto, fazer uma busca seqüencial nos valores de k entre 1 e $c(\log p)^2$; se para algum deles valem (i) ou (ii), p é composto; senão p é primo.

• DETERMINAÇÃO DOS EXPOENTES d, e

Os expoentes d e e são determinados como segue:

1) Dado $\varphi(n) = (p-1)(q-1)$, escolhe-se e, inteiro, aleatoriamente, no intervalo $[1,\varphi(n))$, tal que $mdc(e,\varphi(n)) = 1$. Qualquer número primo maior que $max(p,q)$ é uma escolha válida.

2) Resolve-se $e \cdot d \equiv 1 \bmod \varphi(n)$ em d; este cálculo equivale ao algoritmo do máximo divisor comum [KNUT 2-69], e seu tempo de execução é, portanto, proporcional $\log_2(\varphi(n))$.

• OS ALGORITMOS DE CIFRAMENTO E DECIFRAMENTO

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

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

têm tempo de execução proporcional a $\log_2 e$ e $\log_2 d$, respectivamente; além disso, sempre que necessário, uma redução módulo n é feita, mantendo as quantias sempre menores que n.

• ATAQUES AO RSA

Um criptoanalista pode tentar deduzir o expoente d de deciframento, ou descobrir a mensagem que deu origem a um criptograma, mesmo sem conhecer d. Mostraremos que ambas as tarefas são difíceis.

• Dedução do expoente d.

Pode-se tentar calcular d a partir de:

• criptogramas somente, ou

• pares mensagem-criptograma; convém lembrar que estes pares podem ser escolhidos pelo criptoanalista pois a chave de ciframento é pública.

• além desses pares, o criptoanalista pode tentar usar a chave pública (e,n), de alguma outra maneira, para descobrir d.

Resumindo, m, c, n, e são conhecidos, onde

• $c \equiv m^e \bmod n$,

e quer-se descobrir d, onde

• $de \equiv 1 \bmod \varphi(n)$,

• $\varphi(n) = (p - 1)(q - 1)$, com

$p, q, \varphi(n)$ secretos.

Obviamente, se n for fatorável eficientemente, p e q são obtidos e portanto $\varphi(n)$ e d; assim, descobrir d é, no máximo, tão difícil quanto fatorar. Por outro lado, se, de algum modo, obtivermos d, descobriremos um múltiplo de $\varphi(n)$. Miller [MILL1-75] mostra que, se a hipótese estendida de Riemman for verdadeira, é possível fatorar n a partir de qualquer múltiplo de $\varphi(n)$. Assim, descobrir o expoente d é equivalente à fatoração de n (se valer a hipótese estendida de Riemman), que, acredita-se, é muito difícil.

O que não se conseguiu provar até hoje, é se o trabalho de inverter a função de ciframento, dados o criptograma e a chave pública utilizada, é equivalente a fatorar o módulo n; sabe-se porém da discussão acima que o conhecimento da chave pública (e,n) não leva, de maneira fácil, a descobrir d, a chave de decifra-mento.

• Inversão da função de decifra mento

Um criptoanalista poderia tentar descobrir a mensagem m, dados c, e, n, invertendo a função de ciframento, isto é, descobrindo a raiz m, que elevada ao expoente e, módulo n, dá origem a c. Esse problema não é suficientemente conhecido para

se estabelecer alguma relação com a fatoração de n.

Rabin [RABI 2-79] estabeleceu essa equivalência para um problema parecido:

Seja n = pq, p e q primos; então, descobrir m, dado c, tal que:

$$c = m^2 \bmod n,$$

equivale a fatorar n. Rabin mostra, portanto, que a extração de raízes quadradas módulo n, leva à fatoração de n e vice-versa. Este não é um algoritmo de ciframento, mas um problema correlato.

• OUTROS ATAQUES

Como vimos, o conhecimento da chave pública (e,n) não ajuda o criptoanalista a descobrir a chave secreta (d,n), a menos que ele saiba fatorar n; vimos também que inverter a função $m^e \bmod n$ é supostamente muito difícil. Há porém, outras maneiras de se obter m a partir de c e (e,n):

• Mensagens Raras

Se para algum número inteiro k, $1 \leq k < n$, tal que $mdc(k,n) \neq 1$, então descobre-se imediatamente a fatoração de n, pois $mdc(k,n) = p$ ou q. Se gerarmos tais números k, aleatoriamente, a probabilidade de um número desses ocorrer, se supusermos uma distribuição uniforme no intervalo considerado, e n tiver 200 digitos (p e q têm 100 dígitos cada), é:

$$\frac{p+q-2}{pq-1} = \frac{\text{nº de múltiplos de p e q}}{n-1} \approx \frac{10^{101}}{10^{200}} \approx \frac{1}{10^{99}}$$

o que em termos práticos é zero.

Assim, podemos supor daqui em diante que uma certa mensagem m, $1 \leq m < n$ é tal que $mdc(m,n) = 1$.

• Ciframentos Sucessivos do Criptograma

Consideremos a seqüência $c_o, c_1, c_2, \ldots,$ onde:

$$c_o := c = E((e,n),m) = m^e \bmod n$$

$$c_{i+1} := E((e,n),c_i) = c_i^e \bmod n, \quad (i = 0,1,\ldots).$$

Então $\exists s$ tal que

$$\exists j,\ 0 \leq j < s, \quad \text{tal que} \quad c_j = c_s. \tag{1}$$

Como vimos, o ciframento é bijetor no intervalo $[0,n)$. Assim, se $j > 0$ em (1) então

$$c_{j-1} = c_{s-1};$$

se $j = 0$ temos

$$m = c_{s-1}.$$

Assim, se interrompermos a seqüência na primeira repetição de c_o, o penúltimo elemento da seqüência será a mensagem m. Se s for relativamente pequeno, o criptoanalista poderá computar essa seqüência, recuperando m sem conhecer a chave secreta d.

Por exemplo [SIMM 2-77]:

$$n = 215629 = 383 \times 563$$

$$e = 49$$

$$d = 56957$$

e supondo a mensagem m = 123.456, calcula-se

$$c_o = c = 123.456^{49} \equiv 1603 \mod 123.456$$

$$c_1 = c_o^{49} = 1603^{49} \equiv 180.661 \mod 123.456$$

$$c_2 = c_1^{49} = 180.661^{49} \equiv 109.265 \mod 123.456$$

$$c_3 \equiv 131.172 \mod 123.456$$

$$c_4 \equiv 98.178 \mod 123.456$$

$$\vdots$$

$$c_9 \equiv 123.456 \mod 123.456 = m$$

$$c_{10} \equiv 1603 \mod 123.456 = c$$

Esse ataque foi proposto pela primeira vez por Simmons [SIMM 2-77] e respondido prontamente por Rivest [RIVE 2-77], que argumenta que, se o módulo n for escolhido convenientemente, a probabilidade de s ser pequeno suficiente, para permitir um criptoanálise como a sugerida, é menor que $1/10^{90}$. Rivest sugere que os primos p e q sejam da forma

p = a'p' + 1 onde p' e q' são primos também "grandes", e

q = b'p' + 1 onde a' e b' são fatores "pequenos".

Por sua vez,

$p' = a"p" + 1$ onde novamente p", q" primos "grandes"

$q' = b"q" + 1$ a" e b" "pequenos".

Outros artigos neste tópico são [HERL1-78], [RIVE 3-79], [WILL1-79], [WILL 2-80].

Em [WILL 2-80], Williams apresenta uma variação do RSA e prova a equivalência do seu esquema com o problema de fatorar n.

• CRITÉRIO DE ESCOLHA DOS PRIMOS p e q

No artigo de Rivest, Shamir e Adleman [RIVE1-78], os autores recomendam que p e q sejam escolhidos de tal forma que (p - 1) e (q - 1) tenham fatores primos igualmente grandes; isto para evitar que algoritmos sofisticados de fatoração [POLL1-74], [BREN1-80], fatorem n eficientemente, quando (p - 1) e (q - 1) forem um produto de fatores primos "pequenos".

Podemos juntar essa exigência às feitas por Blakley [BLAK1-78], [BLAK 2-79], onde o autor "mede" a eficiência do RSA.

A função de exponenciação $m^e \bmod n$, realiza uma permutação no espaço de mensagens M; seria desejável que essa permutação não tivesse pontos fixos, isto é, não existissem valores de m tal que

$$m^e \equiv m \bmod n. \qquad 0 \leq m < n.$$

Blakley verifica que quando $n = p \cdot q$, ao menos 9 mensagens permanecem inalteradas, pois:

Seja qual for o expoente e,

$$m^e \equiv m \bmod n$$

tem 9 soluções, a saber as combinações das soluções triviais de

(i) $m^e \equiv m \bmod p$ e

(ii) $m^e \equiv m \bmod q$;

Claramente 0, 1 e - 1 são soluções triviais de (i) e 0, 1 e - 1 são soluções triviais de (ii). Fora as três realmente triviais: 0, 1 e n - 1, as outras 6 soluções:

$$m_1 \equiv 0 \bmod p \quad e \quad m_1 \equiv -1 \bmod q$$

$$m_2 \equiv 0 \bmod p \quad e \quad m_2 \equiv 1 \bmod q$$

$$m_3 \equiv 1 \bmod p \quad e \quad m_3 \equiv -1 \bmod q$$

$$m_4 \equiv 1 \bmod p \quad e \quad m_4 \equiv 0 \bmod q$$

$$m_5 \equiv -1 \bmod p \quad e \quad m_5 \equiv 0 \bmod q$$

$$m_6 \equiv -1 \bmod p \quad e \quad m_6 \equiv 1 \bmod q,$$

determinam um de p e q, quando se calcula $mdc(m_i, n)$ ou $mdc(m_i - 1, n)$; a probabilidade delas ocorrerem é, porém, desprezível.

Além dessas 9 mensagens invariantes, outras podem se tornar pontos fixos, se a escolha dos primos for pobre. Blakley mostra exemplos em que até 50% das mensagens são pontos fixos, e recomenda assim o uso de *primos seguros* da forma:

$$\begin{aligned} p &= 2p' + 1 \\ q &= 2q' + 1 \end{aligned} \qquad \text{com } p', q' \text{ primos;}$$

p e q darão origem a expoentes de ciframento também seguros, isto é, o número de pontos fixos se restringem aos noves citados. Essa exigência, juntamente com a necessidade de que p - 1 e q - 1 tenham fatores primos grandes para evitar fatoração rápida, mais as imposições de Rivest sobre p' e q', nos leva a considerar seguros os primos que tenham a forma:

$p = 2p' + 1$, $p' = 2p'' + 1$, p, p', p" primos "grandes"

$q = 2q' + 1$, $q' = 2q'' + 1$, q, q', q" primos "grandes".

Os algoritmos para determinar expoentes de ciframento com alto grau de opacidade estão em [BLAK 2-79]. *Opacidade* foi definida por Blakley, como sendo o quociente entre o número de pontos fixos e o número total de mensagens para uma dada escolha de (e,n).

• OUTRAS CONSIDERAÇÕES SOBRE A SEGURANÇA DO RSA

Outras tentativas de se estabelecer relações entre o trabalho criptoanalítico e a segurança do RSA foram feitas. Ainda que não conheça *toda* a mensagem m, dado um certo criptograma c e a chave pública (e,n), se um criptoanalista souber como determinar sistematicamente um bit ou um grupo de bits de m, que ganho pode ter com isso ? Ou ainda, se o criptoanalista souber como determinar outras propriedades de m, como pode tirar vantagem disso ?

Shamir e outros [SHAM 6-83] respondem a algumas dessas perguntas. Mais claramente, provou-se que:

(i) Dado qualquer criptograma c1 correspondente a uma mensagem m 1, não revelada:

$$c1 := m^{e} \bmod n,$$

e é possível determinar, por algum meio sistemático, se m1 apesar de não conhecida, está ou não num intervalo qualquer I, de tamanho não desprezível ($\varepsilon < \frac{|I|}{n} < 1 - \varepsilon$), então é possível calcular m1 em "tempo polinomial aleatório", isto é:

Existe um algoritmo que, dado c1 qualquer, fazendo uso de um Oráculo (como sugerem os autores) para determinar se $m1 \in I$, calcula m1 com probabilidade maior que zero; esta probabilidade

não depende da entrada c1 , mas da natureza do algoritmo. É importante notar que este algoritmo, quando emite uma resposta, esta é correta; caso contrário, a saída do algoritmo é simplesmente: "não consegui determinar m1". Em contraposição aos algoritmos chamados Monte-Carlo, que fornecem uma resposta que tem alta probabilidade de ser correta, estes, chamados Las Vegas, quando fornecem uma resposta, ela é correta.

(ii) Se conseguir determinar, para um k fixo, e para qualquer c1, o k-ésimo bit de m1, sistematicamente, então, em tempo polinomial aleatório, é possível descobrir toda a mensagem m1. Isto é verdade para "quase" todos os módulos n.

(iii) Se puder determinar, com probabilidade de erro menor que ¼, o bit menos significativo de m1, dado qualquer c1, então é possível determinar m1 em tempo polinomial aleatório.

Estes resultados, acreditam os autores, aumentam a confiabilidade do RSA, pois o aproximam mais de um permutador aleatório de bits.

3.2. CRIPTOSSISTEMAS BASEADOS NO PROBLEMA DA MOCHILA

"Dados n números inteiros $a_1, a_2, \ldots, a_n$, e um inteiro s, todos positivos, determinar uma família $\{a_i \setminus i \in [1,n]\}$ cuja soma seja igual a s, se tal família existir".

O problema enunciado acima é o chamado *problema da mochila* e pode ser enunciado de outra maneira:

Dados os vetor $A = (a_1, a_2, \ldots, a_n)$, $a_i \in Z^+$, $s \in Z^+$, determinar o vetor $X = (x_1, x_2, \ldots, x_n)$, $x_i \in \{0,1\}$, tal que

$$\sum_{i=1}^{n} a_i x_i = s.$$

O problema é NP-difícil mas a complexidade de cada exemplo do problema, depende fortemente da escolha do vetor A:

(i) Se $A = (1,2,4,8,\ldots,2^{n-1})$, o problema se reduz a determinar a representação binária de s, que leva tempo proporcional a $\log_2 s$,

(ii) Se $A = (a_1, a_2, \ldots, a_n)$ tal que $a_i > \sum_{j=1}^{i-1} a_j$, $i = 1,2,\ldots,n$, temos uma *seqüência supercrescente* de a_i's e a resolução é também fácil, com tempo proporcional a n; o seguinte algoritmo resolve este caso especial:

Dados $(a_1,a_2,\ldots,a_n)$ e s, o algoritmo produz $(x_1,x_2,\ldots,x_n)$, se este existir:

```
começo
    x_i ← 0,  i = 1,n;
    i ← n;
  Repita
    se  a_i ≤ s
        então começo
                  x_i ← 1;
                  s ← s - a_i
              fim;
    i ← i - 1
até (s = 0)  ou  (i < 1);
se  s = 0  então  escreva  x_i , i = 1, n
          senão  escreva ('não existe X tal que AX = s')
fim.
```

(iii) se por outro lado os a_i's são escolhidos aleatoriamente, com distribuição uniforme, o problema é exponencial em n

e o melhor algoritmo conhecido para resolvê-lo é de autoria de Schroepple & Shamir [SCHR1-79], com complexidade de tempo $0(2^{n/2})$ e de espaço $0(2^{n/4})$.

Merkle & Hellman [MERK 3-78] usaram essa diferença de complexidade entre (ii) e (iii) para propor o seu criptossistema assimétrico.

Um usuário X determina e publica sua chave da seguinte maneira:

1) Escolhe-se um vetor A' como descrito em (ii), isto é,

$$A' = (a'_1, a'_2, a'_3, \ldots, a'_n), \quad a'_i > \sum_{j=1}^{i-1} a'_j, \quad i = 2,3,\ldots,n .$$

2) Escolhe-se um inteiro positivo z aleatoriamente tal que

$$z > \sum_{j=1}^{n} a'_j .$$

3) Escolhe-se u, aleatoriamente $1 < u < z$ tal que $mdc(u,z) = 1$; obtém-se a seguir o único w, $1 < w < z$, tal que $u \cdot w \equiv 1 \bmod z$.

4) Calcula-se o vetor $A = (a_1, a_2, \ldots, a_n)$ tal que $A = w \cdot A' \bmod z$

5) O usuário X públicа o vetor A como sendo sua chave pública.

6) Mantêm-se secretos A', z, u, w.

Por exemplo:

Para n = 5

A' = (3,4,12,25,53)

z = 101

u = 10

w = 91

$u \cdot w = 10.91 = 910 \equiv 1 \bmod 101$

portanto

A = (91.3,91.4,91.12,91.25,91.53) mod 101

A = (71,61,82,53,76).

Assim, tudo o que o usuário X fez foi mascarar a característica supercrescente de A', transformando-o em um vetor A, cujos elementos são aparentemente aleatórios. Os elementos usados para efetuar essa transformação, u, w, z, além do próprio vetor A', são mantidos secretos.

Vamos descrever como um outro usuário Y cifra uma mensagem m, usando a chave pública de X, e remete c, o criptograma correspondente:

· ALGORITMO DE CIFRAMENTO

A mensagem m será cifrada, como no RSA, em blocos de n bits, onde n é a dimensão do vetor A, a chave pública. Para simplificar, vamos supor que m consiste de apenas um bloco de n bits:

$$m = (m_1, m_2, \ldots, m_n), \quad m_i \in \{0,1\}, \quad \text{os bits de } m.$$

O usuário Y calcula:

$$c = E(k,m) = E(A,m) = \sum_{i=1}^{n} a_i m_i$$

e envia c para X. Observa-se que decifrar c corresponde a resolver este particular exemplo do problema da mochila.

· ALGORITMO DE DECIFRAMENTO

X recupera m como segue:

a) calcula c' tal que $c' = u \cdot c \bmod z$

b) resolve o exemplo fácil do problema da mochila

$$\sum_{i=1}^{n} a'_i \cdot m_i = c'$$

obtendo m.

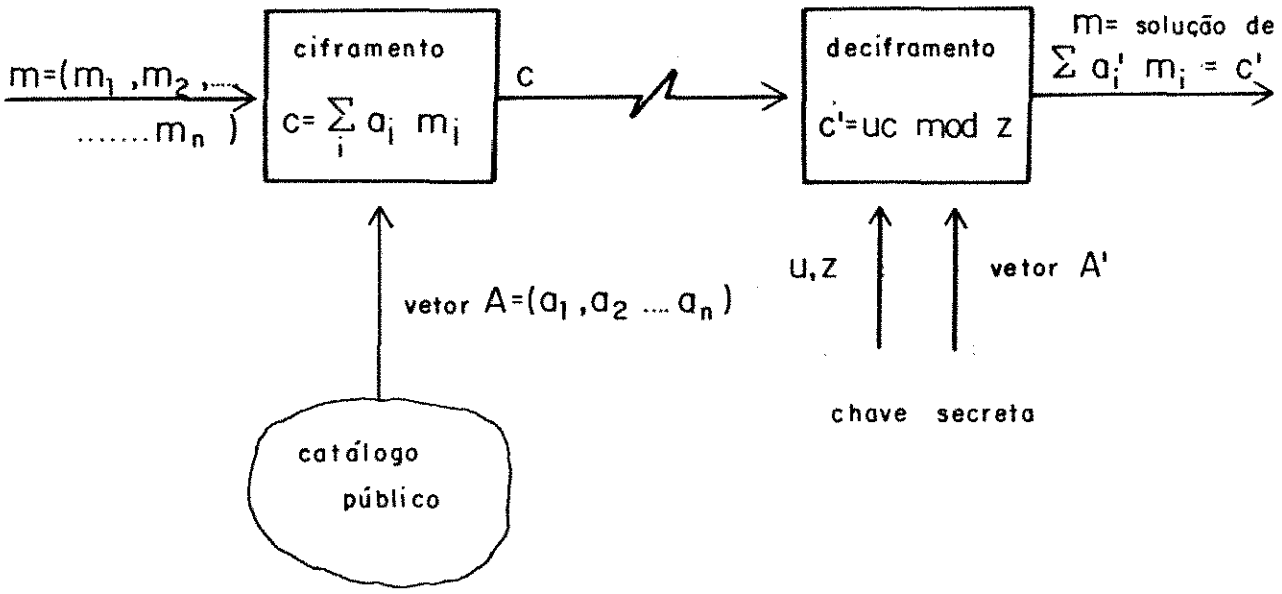


Figura 301: CRIPTOSSISTEMA BASEADO NO PROBLEMA DA MOCHILA (MERKLE & HELLMAN)

Fica caracterizado o efeito unidirecional com alçapão, pois o ciframento consiste em produzir, a partir de m e A, um exemplo do problema da mochila; um criptoanalista deve tentar resolver $\sum_{i=1}^{n} a_i m_i = c$ se quiser decifrar o criptograma c; mas isso corresponde a resolver um exemplo do problema da mochila pretensamente difícil. O usuário X, por outro lado, de posse de u, w, z, A', sua chave secreta, resolve o problema facilmente.

Por exemplo:

· Ciframento

Supondo $m = 01101$ e usando o vetor $A = (71,61,82,53,76)$ temos

$$c = \sum_{i=1}^{5} a_i m_i = 71.0 + 61.1 + 82.1 + 53.0 + 76.1 = 219$$

· Deciframento

Utilizando $A' = (3,4,12,25,53)$, $u = 10$, $w = 91$, $z = 101$, obtemos:

$$c' = u\,c \bmod z = 10.219 \bmod 101 = 69;$$

resolvendo

$$\sum_{i=1}^{5} a'_i m_i = 69$$

encontramos

$69 = 4 + 12 + 53$ e portanto $m = (0, 1, 1, 0, 1)$.

Os resultados se justificam, pois:

como $c' = u\,c \bmod z \Rightarrow c' \equiv u \cdot \sum_{i=1}^{n} a_i m_i \bmod z \Rightarrow$

$$\Rightarrow c' = u \cdot \sum_{i=1}^{n} a'_i \cdot w \cdot m_i \bmod z = u \cdot w \cdot \sum_{i=1}^{n} a'_i \cdot m_i \bmod z$$

como $u \cdot w \equiv 1 \bmod z$, a expressão acima se reduz a

$$c' = \sum_{i=1}^{n} a'_i \; m_i \bmod z$$

como $z > \sum_{i=1}^{n} a'_i$, segue que

$$c' = \sum_{i=1}^{n} a'_i \; m_i \; .$$

O algoritmo de deciframento é não ambíguo, pois, se valem os resultados acima, há apenas uma solução para a última equação $c = \Sigma a'_i \, m_i$, dada a característica supercrescente de A'.

• CRIPTOANÁLISE

• BUSCA EXAUSTIVA

Embora difícil, o problema pode ser resolvido pelo criptoanalista, enumerando todas as famílias de a_i's, até achar uma cuja soma seja igual a s. Isso é evitado escolhendo-se vetores de dimensão apropriada. Merkle & Hellman recomendam vetores A' dimensão igual a 100, e que cada a'_i seja escolhido aleatoriamente com distribuição uniforme nos seguintes intervalos:

$$a'_1 \quad \text{em} \quad [1,\ 2^{100}]$$

$$a'_2 \quad \text{em} \quad [2^{100}+1, 2^{101}]$$

$$a'_3 \quad \text{em} \quad [2^{101}+1, 2^{102}]$$

$$\vdots$$

$$a'_{100} \quad \text{em} \quad [2^{198}+\ ,\ 2^{199}];$$

o módulo z pode ser escolhido aleatoriamente com distribuição uniforme em $[2^{201}+1, 2^{202}-1]$; o multiplicador u é escolhido aleatoriamente em $[2, z-2]$ tal que $mdc(u,z) = 1$, e w é calculado a partir de u e z.

O critério de escolha acima garante que um criptoanalista, se quiser fazer busca exaustiva, terá que testar 2^{100} possibilidades

ou $2^{100}/2 = 2^{99}$, em média $(0(2^n))$, até achar a combinação que deu origem ao criptograma; é claro que com o algoritmo de Schroeppel, a complexidade baixa para $0(2^{n/2})$.

· DETERMINAÇÃO DE u E w SEM RESOLVER O PROBLEMA DA MOCHILA

A questão imediata é se o tipo particular de problemas da mochila, gerados pela multiplicação módulo z, de um vetor supercrescente por uma constante, também é NP-difícil; não se provou este fato, mas conseguiram-se algoritmo polinomiais para resolver essa particular subclasse de problemas da mochila, o que tornou o criptossistema de interesse apenas teórico.

Inicialmente, Shamir & Zippel [SHAM 3-80] mostraram que se o módulo z é conhecido, então há um algoritmo polinomial probabilístico para descobrir os multiplicadores u e w e portanto a seqüência $a'_1, a'_2, \ldots, a'_n$. O algoritmo tira partido da enorme diferença entre os valores de a'_1, a'_2 e o módulo z, que é da ordem de 2^{100}. Para obscurecer essa supercrescência inicial de $a'_1, a'_2, \ldots, a'_n$, Shamir & Zippel propõem um esquema variante conhecido como método Graham-Shamir [SHAM 3-80].

Posteriormente, Shamir [SHAM 2-82] mostrou que, mesmo não conhecendo o módulo z, é possível deduzir, a partir do vetor A publicado, um par w', z', tal que $a_i \equiv w'a'_i \bmod z'$, e assim "quebrar" o criptossistema; não é apresentada, porém, uma solução para a variante Graham-Shamir.

· CIFRAMENTO MULTIPLO

Ao transformar o vetor A' no vetor A, mascara-se a característica supercrescente de A'. Se o multiplicador w, tal que $a_i \equiv a'_i w \bmod z$, for pequeno o suficiente, $a_i = a'_i w$ para os elementos a'_i suficientemente pequenos.

Para remediar isso, Merkle & Hellman propuseram modificar mais de uma vez o vetor A', antes de chegar ao vetor A que seria publicado. Acreditava-se assim, que o criptoanalista teria em mãos um vetor "mais distante" do vetor original, dificultado seu trabalho.

Por exemplo:

Partiremos de um vetor supercrescente A" = (1,2,5,11,19) e duas triplas (u,w,z) a saber:

$$(u_1, w_1, z_1) = (10,4,39)$$

$$(u_2, w_2, z_2) = (13,7,90).$$

Após a primeira iteração, usando $(u_1, w_1, z_1) = (10,4,39)$, o vetor A" = (1,2,5,11,19) se torna:

$$A' = (4,8,20,5,37);$$

na segunda iteração, usando $(u_2, w_2, z_2) = (13,7,90)$, A' se torna:

A = (28,56,50,35,79), que é afinal publicado.

Poder-se-ia usar uma terceira tripla (u_3,w_3,z_3), para eliminar o que restou de supercrescimento entre 28 e 56.

O ciframento é feito como anteriormente, mas o deciframento consiste em determinar $c' := u_1(u_2c \bmod z_1) \bmod z_1$, e só então resolver o problema fácil final. Um criptoanalista não pode usar o método de Shamir mencionado na seção anterior, para descobrir o par (u,w), pois A" não foi gerado diretamente a partir de A, fato esse que Shamir usa no seu algoritmo.

Mas recentemente, Adleman [ADLE 2-83] mostrou como quebrar o esquema de Merkle - Hellman com múltiplas iterações e também a variante de Graham - Shamir; seu método é uma generalização do de Shamir.

• RESOLUÇÃO DE PROBLEMAS DA MOCHILA GERAIS

Os ataques de Shamir e Adleman, se baseiam, portanto, no fato de que é possível deduzir A', w e z a partir de A. Não atacam o problema da mochila em si, mas sim a relação que existe entre o vetor exposto e o vetor secreto, respectivamente A e A'; essa relação mostrou-se fraca e portanto inútil criptograficamente.

Outra maneira de atacar o criptossistema é resolver o problema da mochila geral, sem se importar se o vetor A foi gerado a partir de outro supercrescente ou não:

$$c = \sum_{i=1}^{n} a_i m_i , \quad \text{em} \quad m_i .$$

Em [SHAM3-79], Shamir analisa as relações entre densidade e segurança, de criptossistemas baseados em problemas da mochila; dado um vetor $A = (a_1, a_2, \ldots, a_n)$ define-se *densidade* como sendo

$$\frac{2^n}{\sum_i a_i} = \frac{\text{nº valores representados}}{\text{nº valores possíveis}};$$

por exemplo:

se $A = (1, 2, 4, \ldots, 2^{n-1})$ então

$$\text{densidade} = \frac{2^n}{2^n} = 1;$$

se $A = (3,1,4,10,20,50)$

$$\text{densidade} = \frac{2^6}{88} \approx 0.73;$$

existem outras definições próximas a essa.

Nesse trabalho, Shamir concluiu que, para serem seguros do ponto de vista criptográfico, os criptosssistemas do tipo Merkle & Hellman devem ser baseados em exemplos pouco densos do problema da mochila; caso contrário eles serão solúveis por algoritmos probabilísticos.

Por outro lado, Lagarias & Odlyzko [LAGA1-83] apresentaram um algoritmo que resolve "quase todos" os exemplos de problemas da mochila cuja densidade seja menor que 0.645 (a definição de densidade em [LAGA1-83] é diferente mas também válida aqui), o que soterrou de vez as chances de uso criptográfico desses problemas.

CAPITULO 4

AUTENTICAÇÃO, DISTRIBUIÇÃO DE CHAVES, ASSINATURAS DIGITAIS

No Capítulo 1 mencionamos autenticação de mensagens e de usuários para prevenir interferência ativa, e descrevemos algumas situações de fraude; vamos discutir aqui alguns protocolos criptográficos para autenticação mútua das partes em conversação, bem como das mensagens trocadas por elas. Examinaremos também métodos para autenticação da identidade de usuários frente ao sistema operacional; continuaremos considerando usuário todo processo que se comunica com outros, ou que usa recursos do sistema.

Nos dois capítulos anteriores descrevemos e discutimos alguns criptossistemas simétricos e assimétricos; em ambos os casos, surge porém o problema da obtenção da chave correta, pública no caso assimétrico, secreta no caso simétrico. Descreveremos alguns métodos para resolver esse problema. Como veremos, existe uma questão recorrente: a distribuição de chaves deve ser feita usando o mesmo meio inseguro por onde passarão as mensagens; deveremos então cifrá-las usando outras chaves e voltamos ao problema inicial; como obter estas outras chaves ? Examinaremos alguns protocolos criptográficos que resolvem o impasse.

Finalmente, verificaremos alguns protocolos para confecção de assinaturas digitais, que se constituem num dos meios mais versáteis de autenticação de usuários e mensagens conjuntamente.

4.1. AUTENTICAÇÃO DA IDENTIDADE DE USUÁRIOS

A solução clássica para o problema de permissão de acesso a recursos computacionais é o uso de senhas que identifiquem unicamente cada usuário perante o sistema gerenciador desses recursos. Esse problema é conhecido como autenticação de identidade de usuários ou simplesmente autenticação de usuários.

A cada usuário está associada uma senha, escolhida por ele ou pelo sistema, e que é conhecida somente por ambos. A cada tentativa de acesso ao sistema, o usuário deve apresentar sua identidade (geralmente um número), ao qual o sistema responde com um pedido de senha. Se a senha apresentada a seguir pelo usuário conferir com aquela armazenada em alguma tabela, o sistema permite o acesso deste usuário a um certo conjunto de recursos; caso a senha não confira, o pedido de acesso é rejeitado. É um protocolo simples e bem conhecido.

Um oponente, de posse da senha de outro usuário, pode se fazer passar pelo outro, já que o número de identidade é em geral público. Como normalmente o sistema não verifica características físicas do usuário, dois usuários com a mesma senha são indistinguíveis.

Descreveremos a seguir alguns dos métodos de que um oponente dispõe para se fazer passar por outro usuário; veremos também como a criptografia pode ajudar a prevenir alguns destes ataques.

• 1º ATAQUE: *Testar todas as senhas possíveis*

Suponhamos um usuário comum que dispõe, como tantos outros, de um microcomputador ligado a um computador central; ele pode fazer um programa simples que teste todas as senhas possíveis e descobrir assim a senha de um particular usuário. A probabilidade dele descobrir essa senha pode ser alta pois:

i) As senhas são seqüências não muito longas (6 - 8 caracteres) para facilitar a memorização.

ii) As senhas tem comumente relação com a vida do usuário: nome de parentes, conhecidos, números de placas de carros, RG, trechos de CIC, conta bancária, etc.; portanto, além de ter um significado, a senha é composta de caracteres tirados de um conjunto restrito de letras e números, ainda que possam ser escolhidos de um conjunto maior. Um programa mais sofisticado pode testar assim senhas típicas, com maior probabilidade de acerto.

Morris & Thompson [MORR 1-79] relatam a experiência vivida na implementação do UNIX nos laboratórios da BELL, quando foi feito um levantamento das senhas habitualmente escolhidas pelos usuários; aproximadamente 86% das senhas, tinham 6 caracteres ou menos e a maioria delas era constituída de letras apenas. Além disso, um dicionário, com cerca de 250.000 palavras de lingua inglesa, foi usado para busca de possíveis senhas; descobriu-se que um terço das senhas eram palavras desse dicionário. Há soluções

parciais para o problema da senha curta: guardar uma senha maior e mais complicada num cartão magnético; isso exige que exista uma leitora desses cartões por terminal, além de introduzir novos problemas como a perda ou roubo do cartão.

Embora seja difícil memorizar uma senha composta de uma longa seqüência de caracteres aleatórios, é fácil memorizar uma frase de tamanho qualquer que faça sentido. Há porém o problema de que o sistema armazena senhas de tamanho fixo. Pode-se compatibilizar essas duas exigências, criando-se uma maneira efetiva de dificultar o trabalho de um oponente.

Supondo que o terminal possa realizar ciframento, o usuário no momento do login fornece a frase f como senha. O terminal cifra f:

```
c := E(f, "texto padrão");
```

isto é, o algoritmo de ciframento E usa f como chave para cifrar um texto conhecido; a senha, previamente armazenada, a ser conferida pelo sistema, é c, que tem tamanho fixo, e não f.

Se E = DES, por exemplo, pode-se implementar E(f, "texto padrão") como segue:

- "quebra-se" f em blocos de 56 bits $f_1 f_2 \ldots f_t$, completando o último bloco com zeros se necessário:

• "texto padrão" é uma seqüência qualquer de 64 bits.

• $c := c_t$ de 64 bits, onde

$$c_i := DES(f_i , c_{i-1}), \quad i = 1,...,t$$

$$c_o := \text{texto padrão.}$$

A figura 401 abaixo ilustra este processo. A dificuldade maior do oponente é que ele não sabe ao menos o tamanho da frase f; deve assim tentar frases de tamanho aleatório que resultem no mesmo valor c. A probabilidade disso acontecer é muito pequena.

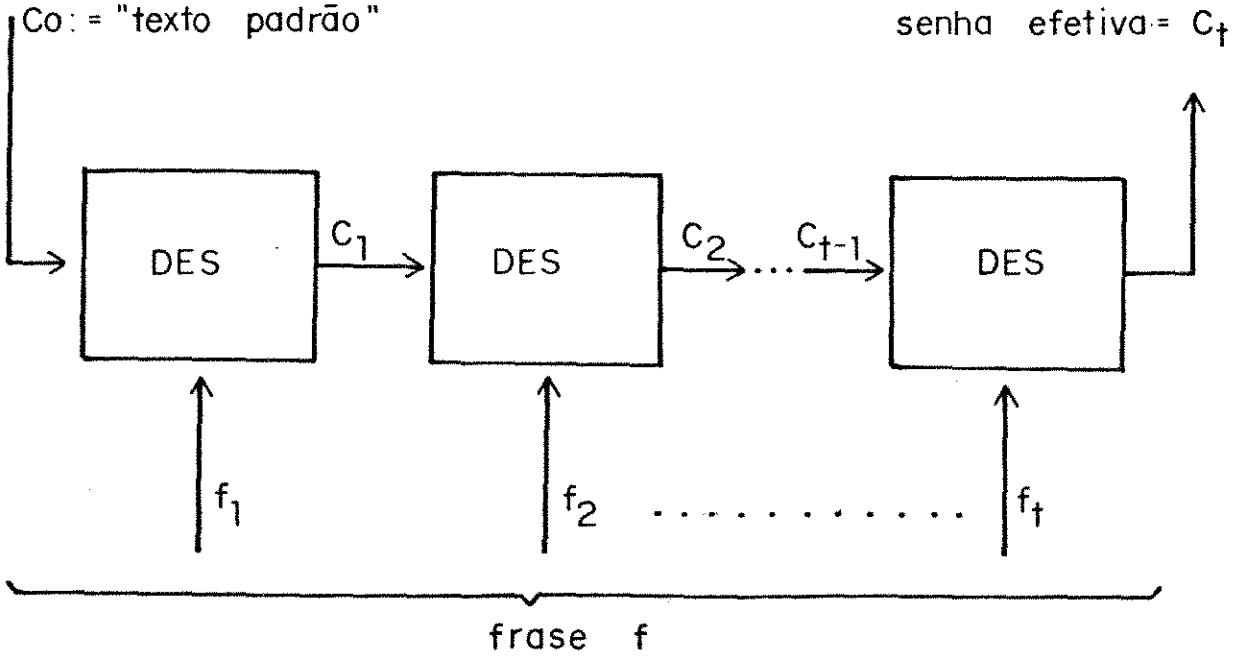


Figura 401: GERAÇÃO DE SENHAS A PARTIR DE FRASES LONGAS.

Esse método foi usado inicialmente para proteção de arquivos de usuários num produto chamado IPS (Information Protection System), implementado pela IBM no centro de pesquisa de Yorktown Heights; hoje várias instalações da IBM usam esse produto. Ao invés de senhas grandes, o IPS usa esse método para que usuários possam escolher chaves grandes, de fácil memorização, para cifrar seus arquivos. Por esse motivo o método tem o nome de "key crunching" [KONH1 - 81].

• 2º ATAQUE: *Acesso ao arquivo de senhas*

A premissa de que um intruso disponha apenas de uma linha conectada ao computador central parece pouco realista. Funcionários dos centros de computação têm normalmente acesso aos arquivos que contêm as senhas, ao menos para leitura delas; um "dump" acidental de memória pode revelar o arquivo de senhas [MORR1-79]; o próprio funcionário encarregado de atualizar o arquivo de senhas tem acesso livre a ele. Não é necessário que mesmo esse encarregado tenha conhecimento das senhas dos usuários; uma vez escolhida pelo usuário, é desejável que só ele saiba a senha e mais ninguém.

A solução mais conhecida para esse problema é aquela apresentada no Capítulo 1, na descrição de funções unidirecionais: o arquivo do sistema contém as senhas cifradas com uma função unidirecional. No Capítulo I usamos a função exponenciação módulo um primo; quaisquer outras podem ser usadas. No UNIX é usado o DES tendo a senha como chave, cifrando uma seqüência padrão de bits:

senha cifrada := DES (senha, "seqüência padrão");

como vimos, o DES tem a propriedade de que dado o criptograma e a mensagem correspondente, é muito difícil deduzir-se a chave usada no ciframento; essa a propriedade usada aqui. A Figura 402 ilustra este esquema.

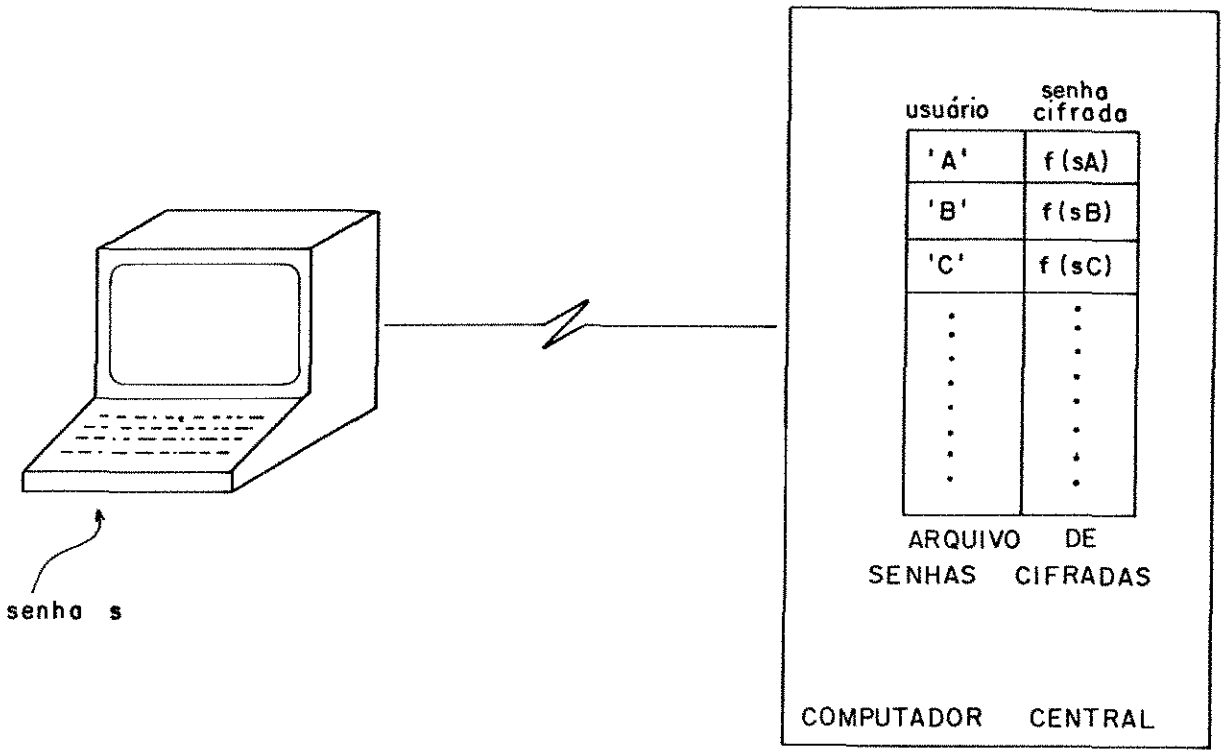


Figura 402: ARQUIVO DE SENHAS CIFRADO COM UMA FUNÇÃO UNI-DIRECIONAL f.

Um intruso mais sofisticado, B, pode modificar o arquivo de senhas cifradas e se fazer passar por um usuário A como segue:

i) B coloca uma senha cifrada, f(s), no lugar da senha cifrada f(sA) de A;

ii) B entra no sistema sob o número de identificação de A, geralmente público, e fornece a senha s. Obtém assim acesso a tudo que A está autorizado a ter acesso.

iii) Após efetuar as operações desejadas, B restaura a senha cifrada f(sA) de A no arquivo de senhas, para não deixar pistas.

Essa ameaça está ilustrada na Figura 403 abaixo. A solução para esse problema usa uma técnica conhecida como autenticação de dados armazenados que (quase) não variam com o tempo. Foi proposta em [LENN1- 82] e [MEYE1- 82] e está descrita a seguir.

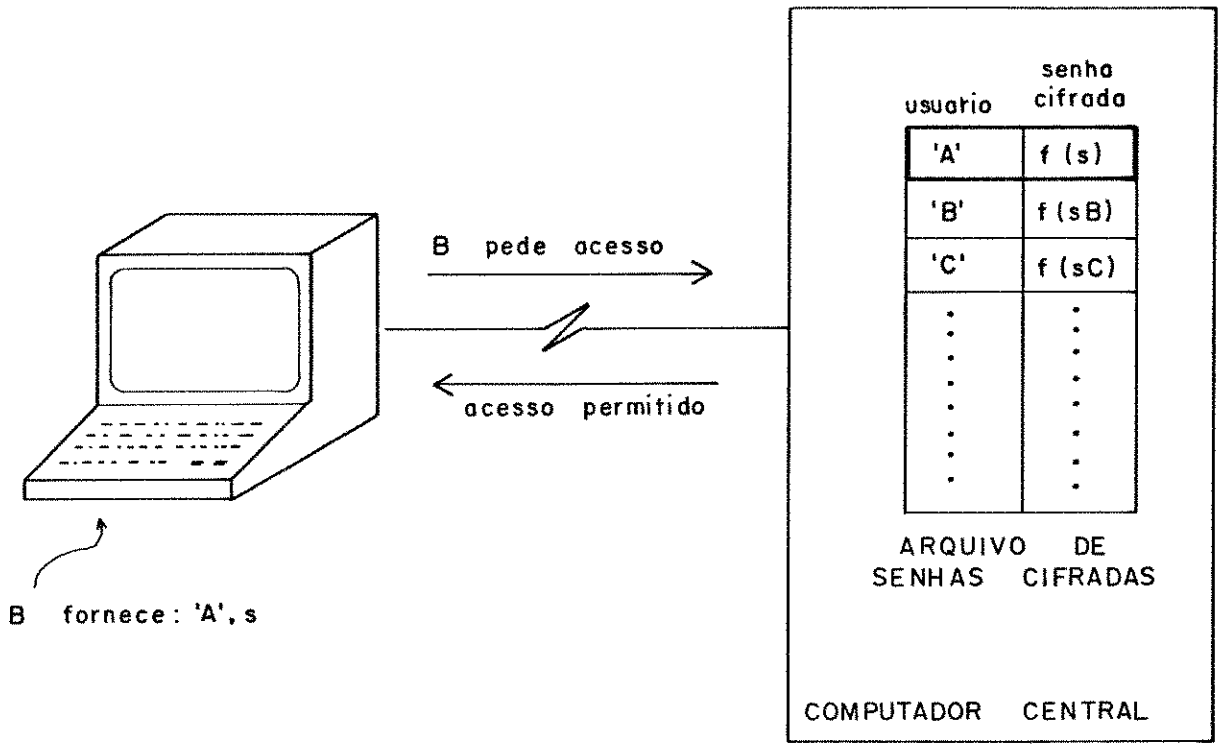


Figura 403: ALTERAÇÃO INDEVIDA DO ARQUIVO DE SENHAS CIFRADAS

Suponhamos que no momento em que uma informação x foi armazenada no endereço vx, calculou-se

$$tx := E(x,vx);$$

a essa operação chamaremos AR e se constitui no ciframento de

vx usando x como chave. O valor tx é armazenado numa tabela T, juntamente com vx.

Posteriormente, se quiser verificar se o conteúdo do endereço vx continua ou não sendo x, obtém-se tx da tabela T e executa-se:

$$vx' := D(x,tx),$$

isto é, o deciframento de tx usando x como chave. Se $vx' = vx$, o conteúdo de vx é o mesmo que foi armazenado inicialmente; estamos assim autenticando x, o conteúdo de vx. A essa segunda operação chamaremos AF.

Obviamente, se qualquer pessoa puder executar AR com chave x', pode-se gerar tx', modificar o conteúdo do endereço vx para x', modificar a posição correspondente em T, e assim a informação x' será considerada autêntica.

Impõe-se portanto que a execução de AR seja exclusiva de uma pessoa autorizada. Pode-se pensar em um dispositivo de cifra mento acionável somente mediante o uso de uma chave física. Há um problema claro: como o algoritmo E é público, qualquer pessoa pode reproduzir o efeito de AR no seu próprio equipamento. Basta usar a chave x' e o endereço vx, gerando tx'. É necessário assim que o dispositivo de cifra mento que implementa AR transforme x internamente antes de usá-la como chave; isto pode ser

feito cifrando x com uma chave interna ao dispositivo e não acessível a ninguém, nem mesmo o único usuário autorizado a executar AR. Isto pode ser conseguido usando "cryptographic facilities" descritas em [LENN1-81] e [MEYE1-82].

A operação AF pode ser implementada por um dispositivo análogo, igualmente inviolável, mas de acesso público; qualquer usuário pode executar AF mas não tem acesso à chave interna do dispositivo. A Figura 404 esquematiza as operações AR e AF.

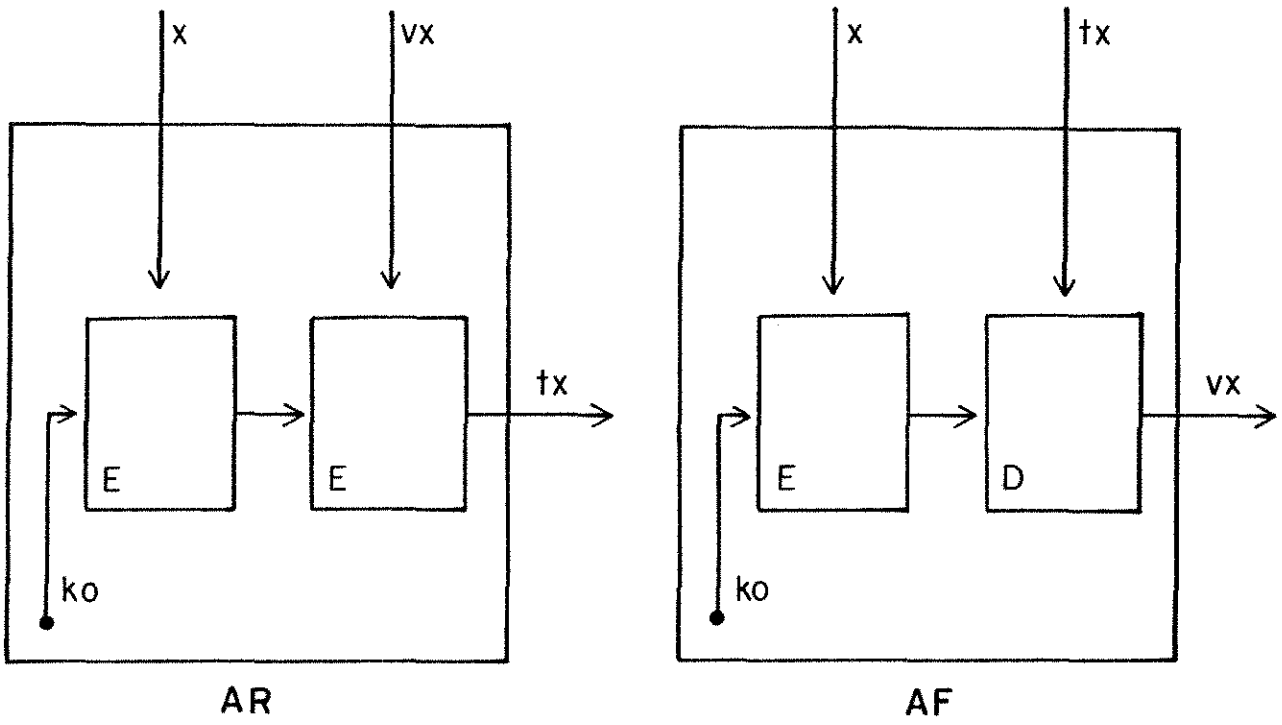


Figura 404: AS OPERAÇÕES AR : $E(E(k_o,x),vx)$ e
AF : $D(E(k_o,x),tx)$.

As operações AR e AF podem ser usadas para implementar um método para autenticar as senhas cinfradas armazenadas pelo sistema: ao invés de aceitar imediatamente um usuário cuja senha fornecida coincidiu com aquela armazenada, o sistema autentica antes essa senha armazenada como segue:

O usuário A quando se registra no sistema, fornece sua identidade iA e sua senha sA. O funcionário autorizado calcula f(sA), a senha de A cifrada usando uma função unidirecional f, e armazena iA e f(sA) na tabela de senhas cifradas. O funcionário calcula também vA := f(iA), a identidade de A cifrada usando f, e usa vA para executar AR:

tA := E(f(sA),vA); a seguir

armazena iA e tA numa tabela T. Convém lembrar que AR só é executável pelo usuário autorizado. A Figura 405 mostra este registro de senhas usando AR.

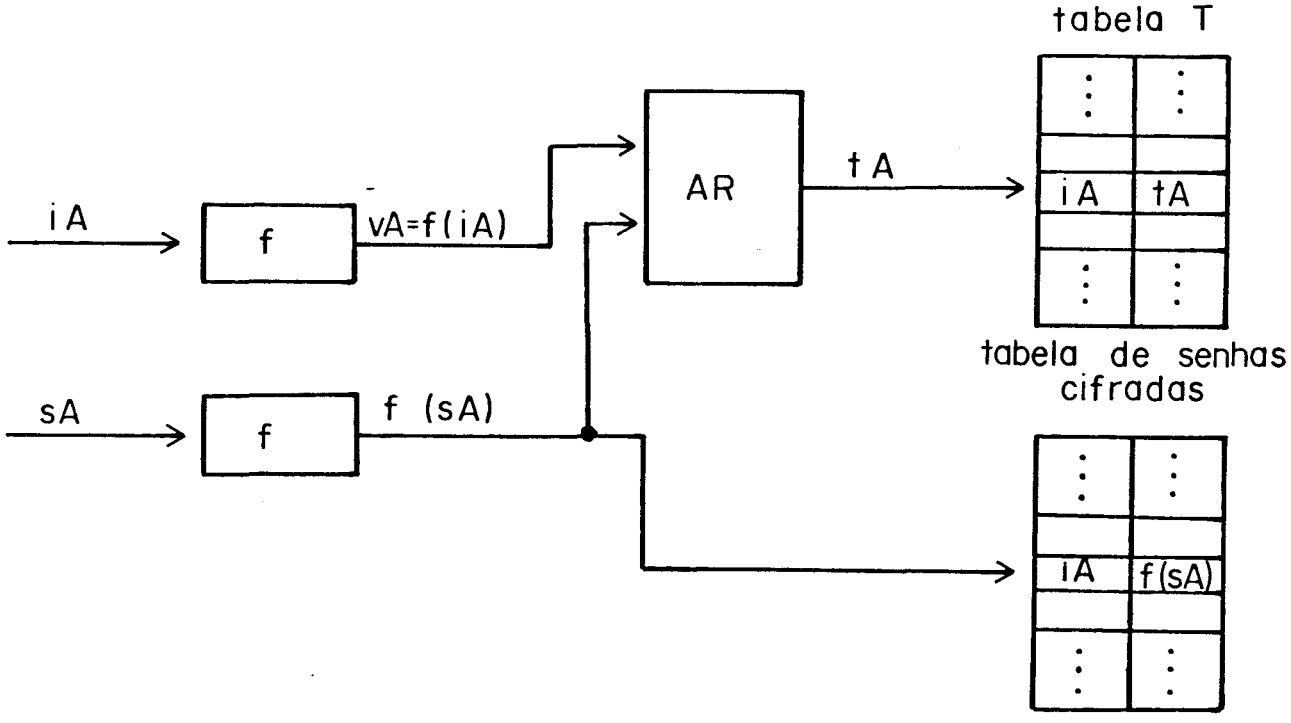


Figura 405: REGISTRO DE SENHAS USANDO AR.

No momento de um login, o usuário A fornece iA e sA; o sistema deve consultar inicialmente a tabela de senhas cifradas como já vimos; logo a seguir deve autenticar f(sA) como segue: Dados iA e f(sA), o sistema calcula primeiramente

$$vA' := f(iA);$$

a seguir, usando iA, obtém, da tabela T, o valor tA. Com tA e f(sA) executa AF:

$$vA := D(f(sA), tA).$$

Se vA' = vA o sistema autentica f(sA) aceitando o login; caso contrário rejeita. A Figura 406 ilustra este processo de autenticação.

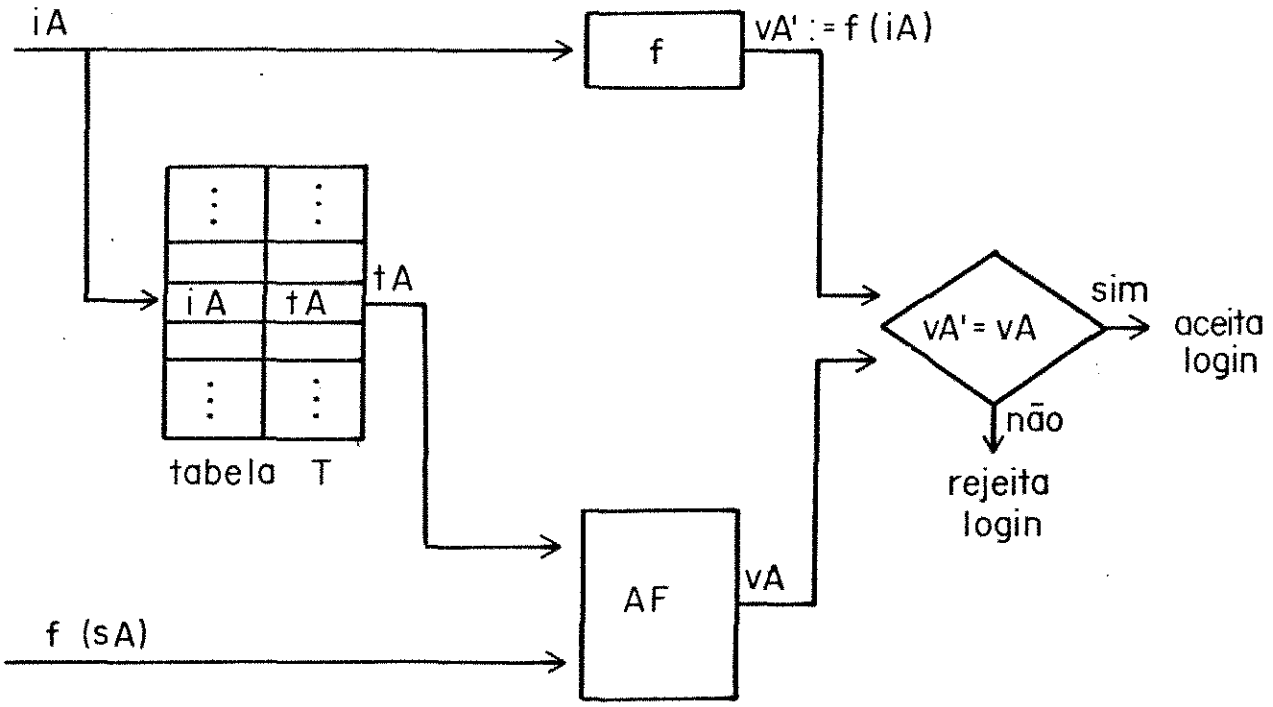


Figura 406: AUTENTICAÇÃO DA SENHA CIFRADA, f(sA), USANDO AF.

Um intruso B que queira se fazer passar por A teria que modificar a tabela T, trocando f(sA) por f(sB), onde sB é a senha escolhida por B e que será apresentada juntamente com iA no momento do login. Além disso teria que alterar a tabela T trocando tA por tB, gerada a partir de uma operação AR. Mas, como vimos, AR é de acesso exclusivo do funcionário autorizado.

B poderia fazer o caminho inverso: escolher tB e sB arbitrários e colocar tB no lugar de tA em T. Precisaria gerar porêm um iB tal que, vB' := f(iB), seja igual ao resultado de AF com argumentos f(sB) e tB. Para conseguir o iB correto, precisaria calcular $f^{-1}(vB')$, o que, por hipótese, é inviável.

Este esquema é o melhor que se conhece para realizar este tipo de autenticação, com a restrição de que a cada mudança de senha, o usuário deve se apresentar ao funcionário autorizado, que obviamente saberá sua senha.

• 3º ATAQUE: *Grampeamento do canal terminal-host*

Se um intruso conseguir "escutar" a linha de transmissão que liga o terminal à instalação central, no momento em que um usuário fornece sua senha, de nada adianta um arquivo de senhas cifradas. É preciso que o ciframento da senha s, f(s), seja realizada no terminal, que supomos aqui inviolável. A atitude do intruso pode ser mais agressiva: ele pode forçar na linha de transmissão uma senha já cifrada, que ele "escutou" previamente, e assim se fazer pelo usuário do qual ele escutou f(s).

Há duas soluções para este último problema. A primeira deve-se a Lamport [LAMP1-81], e propõe que o sistema, ao invés de armazenar f(s), armazene $f^{1000}(s)$, onde $f^{1000}(s)$ é f(f(f(f . . . f(s)) . . .), mil vezes. No i-ésimo login, o usuário fornece $f^{1000-i}(s)$ para o sistema, como sendo sua senha; o sistema executa por sua vez $f(f^{1000-i}(s))$ e confere com o valor armazenado; após a aceitação, o sistema substitue o valor armazenado por $f^{1000-i}(s)$. Assim, as sucessivas senhas fornecidas pelo usuário são $f^{999}(s)$, $f^{998}(s)$, ... ,f(s) enquanto que os sucessivos valores armazenados pelo sistema são $f^{1000}(s)$, $f^{999}(s)$, ... $f^{2}(s)$. Após mil operações de login é preciso trocar a senha.

Um espião que tenha gravado todas as senhas já fornecidas, não consegue gerar uma senha aceitável para o próximo login, pois isso implicaria em saber inverter f, o que, por hipótese, é difícil.

Enquanto resolve problemas de grampeamento de fios, este esquema não resolve o problema anterior, de modificação da tabela de senha; um usuário B poderia colocar um valor conveniente na tabela e personificar A:

Supondo que já foram feitos 20 "login's" e o espião B obteve

$$f^{999}(sA), \ldots, f^{980}(sA),$$

basta substituir o valor armazenado, $f^{980}(s)$, na tabela do sistema, por algum outro entre $f^{999}(sA), \ldots, f^{981}(sA)$, e fornecer o valor conveniente no momento do "login". Deve-se usá-lo portanto junto com o esquema de proteção do arquivo de senhas já descrito.

A outra solução para o grampeamento é da autoria de Feistel e outros [FEIS2 -75] e baseia-se num pequeno protocolo entre usuário e sistema:

Aqui, o terminal do usuário também deve ser capaz de realizar ciframento; vamos supor também que o usuário e o sistema podem conversar em sigilo usando um criptossistema simétrico; ambos usam uma chave kA, sendo que a cópia do usuário está armazenada num cartão magnético e a do sistema num arquivo cifrado com uma chave acessível somente a usuários privilegiados.

O protocolo segue da seguinte maneira:

• A envia para o sistema o par iA, E(kA,r), onde r é uma quantia gerada por A aleatoriamente a cada login.

• O sistema obtem r executando D(kA,E(kA,r)), e envia para A o par E(kA, r + 1), E(kA,r'), onde r' é uma quantia gerada pelo sistema aleatoriamente a cada login.

• A decifra r + 1 e se certifica de que a resposta do sistema é atual e não uma repetição de algum r antigo. A probabilidade de A ter mandado realmente um r antigo no primeiro passo é pequena, se r for grande. Além disso pode-se incluir em r alguma informação relacionada à data e hora do login. A decifra também r' e envia E(kA, r' + sA) para o sistema, onde r' + sA é a concatenação de r' com a senha de A.

• O sistema decifra r' + sA e se certifica de que a mensagem é atual, pois r' é atual, e além disso confere a senha sA, ou f(sA), armazenada no arquivo de senhas. A necessidade do envio da senha é para evitar que um usuário que perca seu cartão, fique totalmente indefeso.

Se o criptossistema em uso for assimétrico, o protocolo é facilmente adaptável, com a vantagem que o sistema não precisa mais guardar um arquivo secreto de chaves, mas somente sua chave secreta e um arquivo de chaves públicas dos usuários.

Há outras maneiras de se garantir a identificação correta de usuários além do uso de senhas: no último protocolo de Feistel vimos um exemplo de combinação de senhas com cartões magnéticos. Pode-se pensar ainda em características físicas do usuário: uma

leitora de impressões digitais ou assinaturas tradicionais, ou ainda um reconhecedor de voz [DAVS1- 79].

4.2. AUTENTICAÇÃO DE MENSAGENS

O conteúdo de uma mensagem, ainda que cifrada, pode ser alterado no seu caminho até o destinatário de várias formas:

· Um intruso pode substituir uma mensagem atual por outra antiga, cifrada com a mesma chave, se esta não for trocada a cada conversação.

· É comum em certos contextos, como transferência eletrônica de fundos por exemplo, que dois usuários troquem mensagens de tamanho e formato fixos; um oponente poderia fazer alterações em trechos adequados de mensagens atuais, substituindo-os por outros trechos pertencentes a mensagens previamente interceptadas, e que foram cifradas com a mesma chave.

As técnicas usuais de autenticação de mensagens (divisão por polinômios) são eficazes na detecção de erros acidentais; num contexto mais sofisticado, um espião poderia modificar a mensagem, produzir um novo autenticador e tudo se passaria como se nada houvesse acontecido. Tornou-se necessário criar meios de detectar modificações intencionais nas mensagens.

Descreveremos assim, maneiras de autenticar o *conteúdo* e a *atualidade* de mensagens; dar ao destinatário meios de verificar se a mensagem recebida é a mesma que foi enviada, e se ela ou parte dela não é a repetição de uma mensagem antiga.

· AUTENTICAÇÃO DO CONTEÚDO DE MENSAGENS

Dada uma mensagem m, o remetente calcula uma quantia a(m), que chamaremos *autenticador* de m, e remete m seguido de a(m); o destinatário só precisará recalcular o autenticador e verificar sua igualdade com a(m).

O autenticador deve ter a propriedade de que qualquer alteração na mensagem m é refletida com grande probabilidade em a(m). Além disso, deve ser muito difícil a um oponente calcular a(m'), dada uma mensagem arbitrária m' diferente de m.

Se m for transmitida cifrada, a(m) pode ser gerado como subproduto do ciframento: no Capítulo 2 vimos como usar o DES de maneira encadeada com propagação de erro:

$$c_i = E(k, m_i + c_{i-1} + m_{i-1}),$$

onde · m_i's são blocos de 64 bits da mensagem m.

· m_o e c_o são valores iniciais previamente combinados.

Se ao invés de cifrarmos m, cifrarmos (m, m_1), isto é duplicarmos o primeiro bloco no fim de m antes do ciframento, este último bloco, após o ciframento, será o autenticador a(m). A Figura 407, na página seguinte, ilustra esta técnica.

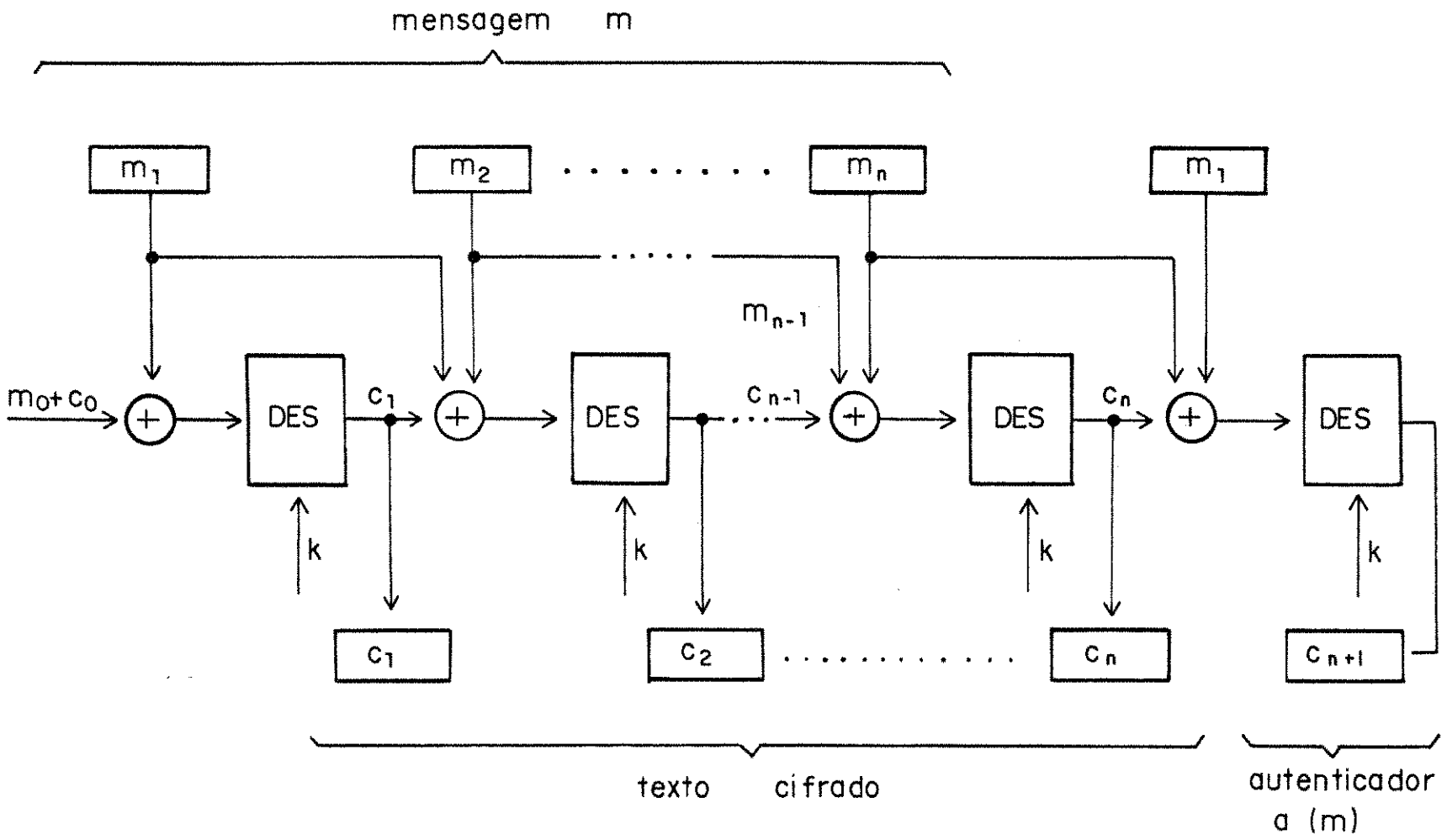


Figura 407: CÁLCULO DE a(m) USANDO CIFRAMENTO ENCADEADO COM PROPAGAÇÃO DE ERRO.

Esta maneira de calcular a(m) se justifica pois:

No deciframento de $c_1,c_2,\ldots,c_n,c_{n+1}$, o primeiro bloco deve ser igual ao último; como há propagação de erro, qualquer alteração em $c_1,c_2,\ldots,c_n$, inclusive alteração da ordem dos c_i's, será refletida no último bloco. Admitindo que cada bloco autenticador (64 bits) tem aproximadamente a mesma probabilidade de ocorrer, $(1/2^{64})$, a probabilidade de que duas mensagens diferentes tenham o mesmo autenticador é $1/2^{64}$.

Um intruso não tem como forjar um a(m) correto se quiser fazer alguma alteração nos c_i's; para isso ele precisaria ou conhecer a chave usada no cinframento, ou a modificação feita não deveria alterar o autenticador; novamente, a probabilidade disso acontecer é $1/2^{64}$.

Se for indesejável o encadeamento ou mesmo ciframento das mensagens, pode-se ainda usar criptografia para autenticação, da seguinte maneira:

• calcula-se E(k,m), o ciframento de m, usando encadeamento com propagação de erro,

• toma-se o último bloco resultante do ciframento E(k,m) como sendo a(m).

• Envia-se m, a(m) se não se quiser cifrar m ou, E(k,m), a(m) se o ciframento de m for necessário.

Nota-se que as técnicas para autenticação do conteúdo de mensagens aqui apresentadas são válidas somente se o criptossistema usado é simétrico; caso contrário qualquer pessoa poderia gerar um autenticador válido para uma mensagem arbitrária, pois a chave de ciframento em criptossistemas assimétricos é pública. Neste caso a autenticação é feita usando assinaturas digitais, de que trataremos adiante.

• AUTENTICAÇÃO DA ATUALIDADE DE MENSAGENS

Se uma quantia t, dependente do tempo, de conhecimento do remetente e destinatário, for concatenada a cada mensagem antes do ciframento, teremos uma maneira de verificar se a mensagem é atual ou antiga. O remetente A, calcula portanto E(k,(m,t)) e envia para o destinatário B, este decifra o criptograma descobrindo t, que é conferido com o valor esperado por ele. Este esquema exige sincronismo das duas partes, o que pode ser difícil. Num método alternativo, A envia uma quantia r, gerada aleatoriamente, ao invés de t:

$$A \rightarrow B : E(k,(m,r)); \qquad (1)$$

o destinatário B, aplica alguma transformação pública em r, por exemplo r + 1, e continua a conversação:

$$B \rightarrow A : E(k,(m', r+1));$$

A pode verificar assim que m' é atual e não a repetição de uma mensagem antiga cifrada com a mesma chave k; A pode continuar a conversação:

$$A \rightarrow B : E(k,(m'', r+2));$$

e assim, daqui em diante, B também pode se certificar da atualidade

das mensagens enviadas por A.

Já vimos o uso desta técnica no protocolo de Feistel para autenticação de usuários frente ao sistema (seção 4.1).

Convém lembrar que este esquema, tal qual descrito, somente é válido se usamos criptossistemas simétricos; isto porque a mensagem (1) poderia ter sido enviada a B por qualquer outro usuário que não A, se o criptossistema fosse assimétrico. É a certeza de que somente A conhece a chave k, além dele próprio, que garante a B que a mensagem (1) veio de A. Ainda assim esta garantia não é total, pois se a chave k é a mesma de comunicações anteriores, um intruso poderia substituir a mensagem de (1) por outra antiga e o fato passaria desapercebido de B. Resolve-se esta questão, impondo que as partes combinem a quantia r no momento em que estabelecem a chave k para a atual conversação; examinaremos a seguir questões como: estabelecimento de chaves entre duas partes (caso simétrico) e como dois usuários podem ter certeza de que as mensagens lhes chegaram realmente de quem diz ser o remetente (caso assimétrico).

4.3. AUTENTICAÇÃO DAS PARTES E DISTRIBUIÇÃO DE CHAVES

Nesta seção, iremos descrever e discutir respostas para as seguintes perguntas:

• Como podem o destinatário e o remetente certificar-se da identidade um do outro ?

• Quais as diferenças se usamos criptossistemas de chave pública ou de chave secreta, no que toca o problema acima ?

O problema de autenticação da identidade das partes numa conversação é um pouco diferente do de autenticação de usuários frente ao sistema operacional; não é mais possível usar senhas já que elas teriam que ser combinadas aos pares para cada par de usuários; isto poderia criar problemas sérios de segredo e atualização dessas senhas. Mas há um efeito parecido com o de senhas se as duas partes usam um criptossistema simétrico para trocar mensagens: como a chave secreta k é conhecida apenas pelas duas partes, o deciframento correto das mensagens recebidas garante que as partes são autênticas (supõe-se que as partes usem as técnicas de autenticação de conteúdo e atualidade de mensagens descritas). Se o criptossistema for assimétrico, não existe esta garantia pois uma das chaves é pública; veremos adiante como resolver esse problema. Examinaremos a seguir o caso de criptossistemas simétricos.

· DISTRIBUIÇÃO DE CHAVES - CRIPTOSSISTEMAS SIMÉTRICOS

Há duas maneiras básicas de dois usuários A e B estabelecerem uma chave comum k para uma conversação: *estática* ou *dinâmica*.

Diz-se que a chave é estabelecida estaticamente quando ela é determinada a priori por A e B; eles podem através de um meio seguro combinar k ou várias k's, uma para cada ocasião que forem trocar mensagens. É a visão não hierarquizada descrita no Capítulo 1; a chave, ou chaves, devem ser mantidas num local seguro, ou então cifradas com outra chave que deve ser mantida em local seguro, e temos aí um problema recorrente; não discutiremos as questões de segurança da chave para cifrar chaves (conhecida como chave mestra). Pode-se também armazenar estas chaves pré-combinadas em algum nó especial da rede, que chamaremos *distribuidor de chaves* que suporemos confiável. A cada conversação, um dos usuários se comunica com o distribuidor e obtém a chave correta; essa comunicação deve ser feita cifrando-se as chaves em trânsito, o que nos leva de volta ao problema inicial. Veremos abaixo como se comunicar com o distribuidor de maneira segura.

Independentemente de quem ficará responsável pelo armazenamento das chaves, esta abordagem estática obriga a existência de no mínimo $\binom{n}{2}$ chaves, onde n é o número de usuários da rede, todas muito bem protegidas.

Seria mais seguro e conveniente, que estas chaves pudessem

ser geradas à medida que fosse necessário; sempre que A e B quisessem se comunicar, estabeleceriam uma chave que deixa de existir após a conversação. No Capítulo 1 nos referimos a essa chave chamando-a de chave transitória. Eles podem estabelecer esta chave independentemente ou com a colaboração de um distribuidor de chaves que se encarregará de gerá-las. Isto elimina a necessidade de se guardar tantas informações sigilosas a priori: o distribuidor guarda apenas n (para conversação em sigilo com cada usuário).

Novamente, se a chave for obtida de um distribuidor de chaves, é necessário cifrá-la durante a transmissão. Descreveremos a seguir um protocolo para comunicação com o distribuidor, de autoria de Needham & Schroeder [NEED1-78]:

Supõe-se que cada usuário i da rede tenha preestabelecido uma chave ki para trocar mensagens cifradas com o distribuidor. Supõe-se também que cada par de usuários A, B tenham uma chave kAB (caso estático); no caso dinâmico, kAB denota a chave gerada pelo distribuidor.

Assim, se A quiser se comunicar com B, o estabelecimento de uma chave kAB no caso dinâmico, ou a obtenção de kAB no caso estático, segue os seguintes passos:

1) A envia para o distribuidor de chaves DC:

A, B, IA

onde IA é uma quantia aleatória gerada por A.

2) DC responde para A enviando:

E(kA, [IA, B, kAB, E(kB, [kAB, A])]) ;

como dissemos, kA e kB são as chaves de A e B respectivamente para comunicação com DC.

3) A decifra a mensagem acima obtendo

IA, B, kAB, E(kB, [kAB, A]) ;

observa-se que A não pode decifrar o último ítem, pois está cifrado com kB.

4) A envia para B

E(kB, [kAB,A]), que decifra obtendo kAB e A.

Está estabelecida a chave kAB para conversação entre A e B. A Figura 408 ilustra este protocolo.

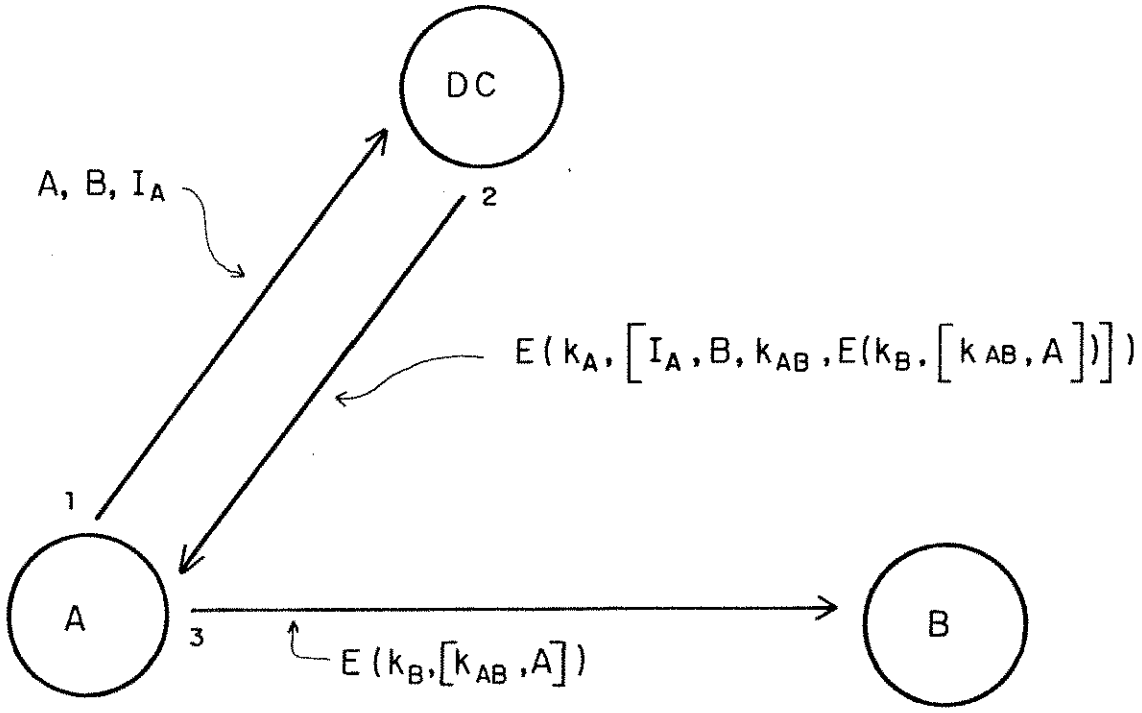


Figura 408: ESTABELECIMENTO DA CHAVE kAB ENTRE A E B USANDO UM DISTRIBUIDOR DE CHAVES DC - CASO SIMÉTRICO.

Examinemos os Passos do Protocolo:

• A mensagem do passo 1 pode ser enviada às claras, pois qualquer modificação intencional por um intruso não lhe é de nenhuma valia. O usuário A poderia enviar (B, IA) cifrados com kA, mas ao menos a sua identidade 'A', deve ser enviada às claras, caso contrário o distribuidor DC não saberia a quem responder.

• A presença de IA, cifrado, no passo 2, previne que a mensagem enviada por DC para A seja substituída por outra antiga. Portanto IA funciona como autenticador da atualidade da mensagem do passo 2. Além disso, a presença de 'B' nesta mensagem, garante que o distribuidor de chaves recebeu a mensagem do passo 1 intacta; isto é, um intruso C não poderá substituir 'B' por 'C' na mensagem 1, e assim fazer com que A se comunique com ele C, pensando se tratar de B.

• Como E(kB, [kAB, A]) não é decifrável por ninguém além de B, este tem certeza que A, e só A, possui kAB. O usuário A tem a mesma certeza com relação a B. O que um oponente C pode fazer é substituir a mensagem enviada no quarto passo, por E(kB, [kAB', A]), onde kAB' é uma chave usada numa conversação prévia por A e B, mas que C conseguiu, por algum meio, descobrir. Assim ele pode decifrar ao menos a primeira mensagem que B enviar para A, antes da conversação entre os dois, A e B, entrar em colapso. Uma modificação neste protocolo foi feita por Denning

& Sacco [DENN1-82], introduzindo o que se convencionou chamar de "timestamps", e que resolve este problema. O protocolo modificado é o seguinte:

1) A → DC:

A,B

2) DC → A:

E(kA, [B, kAB, T, E(kB, [A, kAB, T])]

onde T é a data e hora presentes, dito "timestamp".

3) A → B:

E(kB, [A, kAB, T]).

Este protocolo conserva as mesmas garantias do anterior e resolve o problema da atualidade da última mensagem; exige porém sincronismo entre os nós da rede, o que pode ser incômodo.

Para proteção adicional, pode-se acrescentar dois passos aos protocolos apresentados; esses passos se constituem numa espécie de apresentação entre A e B; em inglês "handshaking":

- Após obter kAB, B envia para A:

E(kAB, IB), onde IB é o análogo de IA para B.

- A decifra esta mensagem e responde para B:

$$E(kAB, IB + 1).$$

As duas partes podem se comunicar seguramente daqui em diante.

O estabelecimento dinâmico de kAB pode ser feito independentemente por A e B sem a ajuda do distribuidor de chaves. Há duas maneiras para isso:

• usar um criptossistema assimétrico que serve momentaneamente para que A e B estabeleçam kAB, a ser usada num criptossistema simétrico. Isto nos remete ao problema de obtenção das chaves públicas corretas de ambos, que será tratado adiante.

• Usar uma função unidirecional como descrito no Capítulo 1. Esta a única opção que oferece liberdade total a A e B; não há necessidade de se manter absolutamente nada em segredo antes do estabelecimento da chave transitória. Há porém um problema sério: qualquer pessoa pode iniciar o protocolo de troca dizendo ser, por exemplo, o usuário X, quando na realidade não o é. Não há como autenticar a identidade deste usuário pois supomos que não há informação alguma exclusiva de algum usuário. Pode-se solucionar esse problema com o uso de assinaturas digitais que examinaremos logo mais.

• DISTRIBUIÇÃO DE CHAVES - CRIPTOSSISTEMAS ASSIMÉTRICOS

Um usuário A, querendo se comunicar com B, pode obter a chave pública de B de três maneiras:

• De seu arquivo particular; a opção por um tipo de catálogo público não é viável, por problemas de atualização e praticidade.

• De um distribuidor de chaves análogo ao anterior.

• A e B trocam suas chaves públicas diretamente usando a linha insegura de comunicações.

Examinaremos as duas últimas alternativas:

Needham & Schroeder [NEED1 - 78] apresentam uma versão do protocolo anterior para chaves públicas:

Supõe-se aqui que as chaves públicas de todos os usuários estejam armazenadas no distribuidor de chaves; são elas kA, kB,... etc. Além disso, o distribuidor tem a sua chave pública kDC conhecida por todos os usuários, que a mantêm armazenada em local seguro; se um intruso conseguir modificá-la, ele pode personificar o distribuidor como veremos. Analogamente, as chaves secretas dos usuários serão $\ell A, \ell B, \ldots$ etc. e ℓDC.

Suporemos também que as funções de ciframento e deciframento, E e D respectivamente, são comutativas. Isto é, para qualquer mensagem m e pares de chaves (k, ℓ), vale a seguinte propriedade:

$$D(\ell,E(k,m)) = E(k,D(\ell,m)) = m.$$

A propriedade, $D(\ell,E(k,m)) = m$, é a exigida de todos os criptossistemas assimétricos vistos até aqui: toda mensagem cifrada deve ser passível de deciframento usando a chave correta. A segunda propriedade, $E(k,D(\ell,m)) = m$, apenas um criptossistema conhecido tem: o RSA; com ele pode-se "decifrar" uma mensagem, e posteriormente "cifrar" o resultado do deciframento obtendo a mensagem original. Esta propriedade é fundamental na geração de assinaturas digitais, como veremos.

O protocolo segue:

1) A envia para DC:

$$A,B$$

2) DC responde:

$$D(\ell DC, [kB,B]);$$

isto é o distribuidor envia a mensagem [kB,B], onde kB é a chave secreta de B, "decifrada" sob ℓDC, sua chave secreta.

3) A executa

$E(kDC, D(\ell DC, [kB,B]))$ obtendo kB e B, e envia para B:

$$E(k,B,[I_A,A]),$$

onde IA é uma quantia aleatória gerada por A.

4) B decifra a mensagem acima obtendo 'A' e envia para DC:

B,A

5) DC responde, enviando para B:

$D(\ell DC, [kA, A])$,

e B obtém kA executando o análogo do passo 3:

$E(kDC, D(\ell DC, [kA, A]))$.

Neste ponto A e B se garantem de que:

• As chaves que eles obtiveram, kB e kA respectivamente, são corretas pois resultantes do deciframento de uma mensagem cifrada com uma chave de posse exclusiva do distribuidor, a saber ℓDC. Isto identifica unicamente o originador das mensagens enviadas nos passos 2 e 5. Além disso as chaves vieram acompanhadas da identidade dos seus donos, o que garante que o distribuidor recebeu as mensagens 1 e 4 sem alterações.

Novamente, B não tem certeza se a mensagem recebida no passo 3 é atual e gerada realmente por A. Mais dois passos são necessários para que A e B se apresentem:

6) B envia para A

$$E(kA, [I_A, I_B])$$

onde IB é uma quantia aleatória gerada por B.

7) A responde enviando

$$E(kB, I_B).$$

Este protocolo está ilustrado na Figura 409 abaixo:

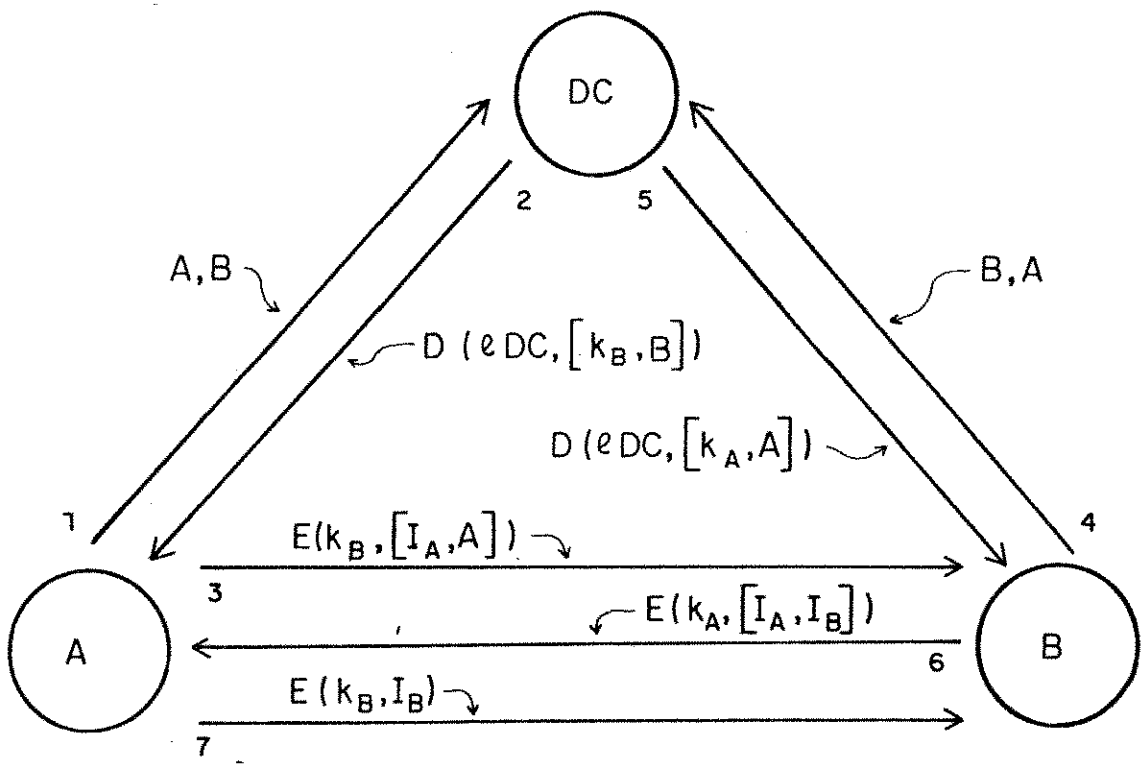


Figura 409: OBTENÇÃO DAS CHAVES PÚBLICAS DE A E B USANDO UM DISTRIBUIDOR DE CHAVES DC - CASO ASSIMÉTRICO.

Analogamente ao que foi feito no caso de criptossistemas simétricos, "timestamps" podem ser incluidos no protocolo [DENN1-82],

com o uso de certificados que serão descritos na seção que trata de assinaturas digitais.

Convém notar que mesmo após este protocolo, qualquer pessoa poderia mandar mensagens válidas para A ou B se fazendo passar pelo outro. Isto pode ser solucionado exigindo que todas as mensagens, ou blocos de bits na qual a mensagem foi dividida, sejam cifrados junto com algum serializador (pode-se usar os valores de I_A e I_B como base para a serialização, dado que são desconhecidos de outros usuários que não A e B).

Assim, as mensagens m1,m2,... etc. de A para B seriam cifradas como segue:

$$c1 = E(kB, [m1, I_B])$$

$$c2 = E(kB, [m1, I_B + 1])$$

.
.
.

O mesmo acontece com as mensagens de B para A, mas com I_A em lugar de I_B.

Observa-se que I_A e I_B não só servem de autenticadores de atualidade de c1,c2,..., mas também identificam o usuário que as está mandando; acabaram tendo o mesmo efeito de uma senha pré-combinada.

A e B podem querer fornecer diretamente, pela linha, um ao

outro, as suas chaves públicas. Há uma ameaça óbvia: um terceiro usuário C poderia interromper a ligação no momento da troca de chaves e fornecer a ambos a sua chave pública; durante a conversação ele poderia assim "ouvir" toda a troca de mensagens transparentemente. Rivest & Shamir [RIVE4-84] mostram como evitar essa ameaça, ainda que A e B troquem suas chaves às claras, sem cuidados adicionais.

4.4. ASSINATURAS DIGITAIS

Os protocolos apresentados acima garantem que as mensagens trocadas entre A e B são corretas no que diz respecito à sua origem, destino, atualidade e conteúdo. É necessário que exista também um instrumento com a mesma força de uma assinatura para evitar as seguintes situações:

• A envia uma mensagem a B e posteriormente nega o envio; como B poderia provar a uma terceira parte, um juiz por exemplo, que A realmente lhe enviou aquela mensagem ? um exemplo dessa situação: B é um corretor de ações e A seu cliente; A envia para B a mensagem: "Compre 10.000 ações da companhia X"; no dia seguinte as cotações de X sofrem uma queda muito grande e A, querendo se livrar do prejuizo, nega o envio do pedido de compra. B deve estar munido de provas deste envio para poder se defender.

• B pode receber uma mensagem e negar posteriormente o fato. Deveria existir portanto uma espécie de recibo assinado.

A existência de uma assinatura digital, que pudesse ser transmitida de A para B ou vice-versa, junto com a mensagem que está sendo assinada, resolveria estes problemas. Assim, uma assinatura digital seria o análogo eletrônico de uma assinatura tradicional.

Do exposto acima conclui-se que uma assinatura digital deve

ter as seguintes propriedades.

1) Deve identificar unicamente seu autor.

2) Deve ser impraticável a qualquer outra pessoa forjá-la, inclusive o destinatário da mensagem.

Contrariamente a uma assinatura normal, a assinatura digital uma vez "escutada" numa linha de transmissão, pode ser duplicada pois não passa de uma seqüência de bits. Obriga-se a propriedade 3:

3) A assinatura digital deve ser dependente de cada mensagem transmitida; a forma dessa dependência não deve ser óbvia, caso contrário,qualquer pessoa poderia forjar uma assinatura.

Se a assinatura varia a cada mensagem, como reconhecê-la ? Propriedade 4:

4) Deve ser possível a uma terceira parte credenciada decidir um conflito entre dois usuários, reconhecendo ou não uma assinatura digital.

Veremos a seguir como produzir assinaturas digitais em contextos de criptossistemas simétricos ou assimétricos.

• ASSINATURAS DIGITAIS NO CONTEXTO DE CRIPTOSSISTEMAS SIMÉTRICOS

Dois usuários A e B que trocam mensagens cifradas usando um criptossistema simétrico, após estabelecerem uma chave k, não podem provar a um juiz que uma mensagem cifrada com k é de autoria de um ou de outro. Isto porque ambos poderiam ter gerado essa mensagem. As propostas de assinaturas em ambientes que usam criptossistemas simétricos implementam algum tipo de protocolo que impossibilita tanto ao remetente quanto ao destinatário, negar o envio ou recebimento de mensagens, ou forjar assinaturas falsas.

O primeiro desses protocolos é de autoria de Rabin [RABI1-78], e supõe a existência de uma função unidirecional E, com as seguintes propriedades:

i) Para qualquer k e qualquer m, E(k,m) é rapidamente computável.

ii) Para qualquer k e qualquer 2n-upla:

$$(m_1, E(k,m_1),\ m_2, E(k,m_2), \ldots, m_n, E(k,m_n)),$$

é *difícil* computar (m,u) tal que $u = E(k,m)$, com $m \neq m_i$, $1 \leq i \leq n$; donde se conclui que é difícil deduzir a chave k a partir de qualquer 2n-upla.

iii) Para qualquer m, é *difícil* produzir (k_1,k_2), $k_1 \neq k_2$, tal que $E(k_1,m) = E(k_2,m)$. Isto é, mesmo que se escolha m e se conheça o resultado $E(k_1,m)$, deve ser difícil deduzir $k_2 \neq k_1$, mesmo que k_2 exista, tal que $E(k_2,m) = E(k_1,m)$.

As propriedades acima impõem que a função $E(k,m)$, seja portanto unidirecional nos dois parâmetros: a chave k e a mensagem m. As funções de ciframento em bloco, descritas até aqui, acredita-se, têm essas propriedades.

O protocolo segue os seguintes passos:

1) Antes de qualquer troca de mensagens, cada usuário A escolhe, mas não revela aos outros, uma lista de 40 chaves:

$$x_1, x_2, \ldots, x_{40}$$

2) Usando uma mensagem padrão, mp, A calcula a seguinte seqüência de *autenticadores*:

$$a_i := E(x_i, mp), \quad 1 \leq i \leq 40 \qquad (1)$$

3) O usuário A se encaminha a um cartório público e registra um contrato assinado contendo os 40 valores $E(x_i, mp)$, $1 \leq i \leq 40$

A
$a_1 := E(x_1,mp)$
$a_2 := E(x_2,mp)$
.
.
.
$a_{40} := E(x_{40},mp)$
assinado: A

4) Desejando enviar uma mensagem m, para um outro usuário B, A inicialmente comprime m, usando por exemplo a função E da seguinte maneira:

$$r(m) := E(m_1,E(m_2 , \ldots , E(m_r,mp) \ldots) \qquad (2)$$

onde $m_1,m_2 , \ldots , m_r$ são os blocos em que m foi segmentada para poder-se usar a função E. O resultado, r(m), é a *compressão* de m. Outra maneira de comprimir m seria cifrá-la usando algum outro método de encadeamento com propagação de erro. Por simplicidade, denotaremos (2) por

$$r(m) := E(m,mp).$$

Convém tomar o cuidado de verificar se r(m) := mp; se isso

acontecer, deve-se modificar m ligeiramente, de alguma maneira padrão previamente estabelecida, antes de assiná-la. A razão disso ficará clara adiante.

5) A *assina* m calculando a assinatura s(m):

$$s_i(m) := E(x_i, r(m)) \qquad (1 \leq i \leq 40)$$
$$s(m) := \{s_i(m) \mid \quad 1 \leq i \leq 40\} \tag{3}$$

6) Finalmente, A envia m e a assinatura s(m) para B:

$$A \rightarrow B: \quad m, s(m)$$

B verifica a assinatura de A como segue:

7) B gera ao acaso 20 índices $i_1, i_2, \ldots, i_{20}$, no intervalo $1 \leq i_j \leq 40$, $(1 \leq j \leq 20)$, e solicita a A que lhe envie as 20 chaves correspondentes $x_{i_1}, x_{i_2}, \ldots, x_{i_{20}}$. Chamaremos de $x'_{i_1}, x'_{i_2}, \ldots, x'_{i_{20}}$ às 20 chaves remetidas por A. Em seguida, B verifica a autenticidade dessas 20 chaves, conferindo, de acordo com (1), se

$$E(x'_{i_j}, mp) = a_{i_j}, \tag{4}$$

para cada j, $1 \leq j \leq 20$. Se a igualdade (4) não valer para algum

dos índices i_j selecionados, B rejeita a assinatura. Caso contrário, segue o processo de reconhecimento de s(m).

8) B é capaz de calcular r(m) pois conhece E, m e mp. Usando as 20 chaves que A lhe revelou e s(m), B verifica se, de acordo com (3),

$$E(x_{i_j}, r(m)) = s_{i_j}(m), \tag{5}$$

para cada j, $1 \leq j \leq 20$. Se as 20 igualdades valerem, B reconhece s(m); caso contrário a rejeita.

No caso de um juiz ser chamado a resolver um conflito entre A e B, onde por exemplo A nega o envio de uma mensagem assinada a B, que alega tê-la recebido, resolve-se o impasse como segue:

9) O juiz pede a A que revele as 40 chaves usadas para gerar s(m), a saber $x_1, x_2, \ldots x_{40}$, e verifica sua autenticidade conferindo se, de acordo com (1),

$$E(x_i, mp) = a_i, \tag{6}$$

para cada i, $1 \leq i \leq 40$. Se ao menos uma das 40 chaves não for autêntica, o juiz dá razão a B, que alega o recebimento da mensagem.

10) Autenticadas as 40 chaves, o juiz questiona a autenticidade de $s(m)$, verificando se, de acordo com (3),

$$E(x_i , r(m)) = s_i(m), \qquad (7)$$

para todo i, $1 \leq i \leq 40$. Se 20 ou menos igualdades de (7) se verificarem, o juiz dá razão a A, que nega o envio da mensagem; se 21 ou mais conferem, a razão é de B, que diz tê-la recebido.

Para analisarmos porque este protocolo resolve realmente o impasse, suporemos as seguintes situações:

1º CASO: Tendo aceito anteriormente a assinatura de uma mensagem m, B conseguiu forjar a assinatura de A numa outra mensagem m', diferente de m.

2º CASO: A consegue recusar a autoria de uma mensagem m, realmente enviada para B.

· 1ª SITUAÇÃO: (B desonesto)

Ao aceitar m anteriormente, B tomara conhecimento de 20 das 40 chaves usadas para assiná-la, digamos, sem perda de generalidade, $x_1, x_2, \ldots, x_{20}$. Com elas, B pôde produzir:

$$s_i(m') = E(x_i , r(m')), \qquad 1 \leq i \leq 20.$$

Para que o juiz lhe desse razão no momento de julgar a validade da assinatura de m', B precisou calcular ao menos mais um valor, digamos,

$$s_{21}(m') = E(x_{21}, r(m')).$$

Mas a_{21}, isto é $E(x_{21}, mp)$, consta do contrato assinado. Assim, B conseguiu obter o par

$\{r(m'), E(x_{21}, r(m'))\}$ válido, a partir de

$\{mp, E(x_{21}, mp)\}$ e

$\{r(m), E(x_{21}, r(m))\}$

onde $r(m') \neq r(m) \neq mp$; isto contraria a propriedade (ii) de E. Esta a rezão pela qual se tomou o cuidado inicial de não permitir que $r(m) = mp$.

Se por outro lado, $r(m) = r(m')$, isto significa que B foi capaz de gerar $m' \neq m$ tal que

$$r(m) = E(m, mp) = E(m', mp) = r(m'),$$

o que contraria a propriedade (iii) de E.

• 2ª SITUAÇÃO: (A desonesto)

Na tentativa de recusar a autoria de uma mensagem m assinada, realmente enviada para B, A precisa, ao mesmo tempo, fazer com que B aceite a mensagem assinada e o juiz acredite que ele não enviou a mensagem m a B.

Isso só é possível se as seguintes afirmações valerem:

a) As 20 chaves, $x'_{i_1}, x'_{i_2}, \ldots, x'_{i_{20}}$ selecionadas por B "passam" no teste da assinatura (passos 7 e 8 do protocolo):

$$E(x'_{i_j}, mp) = a_{i_j} \qquad (1 \leq j \leq 20) \qquad \text{e}$$

$$E(x'_{i_j}, r(m)) = s_{i_j}(m) \qquad (1 \leq j \leq 20).$$

b) As 40 chaves, $x_1, x_2, \ldots, x_{40}$ usadas para produzir s(m) "passam" pelo "teste do cartório" feito pelo juiz (passo 9):

$$E(x_i, mp) = a_i \qquad (1 \leq i \leq 40).$$

c) No máximo 20 dessas 40 chaves passam pelo teste da assinatura feito pelo juiz (passo 10):

$$E(x_i, r(m)) = s_i(m) \qquad (1 \leq i \leq 40).$$

Ou seja, as 40 chaves têm que passar pelo "teste do cartório"

e precisamente 20 delas pelo teste da assinatura: exatamente as 20 escolhidas pela vítima B. A probabilidade de que B escolha exatamente essas é muito pequena:

$$1/\binom{40}{20} \leq 10^{-11}.$$

Explica-se aqui porque B solicita a A que revele exatamente 20 das 40 chaves e não menos ou mais que 20. Fixado m, o coeficiente binomial $\binom{m}{n}$ é maximizado quando $n = [\frac{m}{2}]$, conseguindo-se assim minimizar as chances de A enganar B.

Na realidade, o contrato assinado não contém somente 40 chaves, mas t blocos de 40 chaves, um para cada mensagem. Ou se esgotam os t blocos após t mensagens (e um novo contrato deve ser assinado no cartório), ou se reserva o último bloco de 40 chaves de cada contrato para ser usado no estabelecimento de um novo contrato de t blocos, que estaria assim automaticamente assinado.

Este protocolo foi apresentado em 1976 e pode ser utilizado em qualquer contexto baseado em criptossistemas simétricos. Embora muito elegante, a quantidade de chaves que deve ser gerada e armazenada pode ser proibitiva se o tráfego de mensagens for intenso; além disso, a renovação constante do contrato assinado pode ser inconveniente. Ao invés de usar o último bloco de cada contrato para assinar o próximo, Rabin aconselha que se use as

chaves do primeiro contrato, realmente assinado em cartório, apenas para assinatura de novos contratos, evitando que um juiz precise fazer autenticações de chaves de inúmeros contratos anteriores para finalmente resolver um impasse.

Uma restrição desse protocolo é que ele não pode ser usado em correio eletrônico:

Se A quiser enviar uma mensagem assinada e deixá-la armazenada em algum arquivo de B, este só poderá autenticar a assinatura numa comunicação posterior com A.

Yuval [YUVA1-79] apresenta a seguinte análise do protocolo de Rabin:

A propriedade (iii) da função E:

"Para qualquer m, é *difícil* produzir (k_1,k_2), $k_1 \neq k_2$, tal que $E(k_1,m) = E(k_2,m)$", pode ser "fraca". Pelo paradoxo do aniversário, a probabilidade de se encontrar (k_1,k_2) cresce quadraticamente com o número de chaves considerado; no caso do DES por exemplo, se tomarmos 2^{32} chaves a probabilidade de um empate é de 50%. Como vimos, Rabin usa essa propriedade para mostrar que a 1ª situação, onde B tenta forjar uma assinatura, é improvável.

Se m tiver mais que 64 bits como em geral tem, a existência de m', tal que $E(m,mp) = r(m) = r(m') = E(m',p)$, $m \neq m'$, é certa. Rabin alega que é *difícil* achar m'. Yuval mostra que pode

ser *fácil*, à custa de alguns bilhões de operações e algumas centenas de fitas magnéticas.

Convém notar que no protocolo proposto, a mensagem m e a assinatura s(m) são transmitadas às claras; deve-se cifrá-las como se faz com mensagens comuns, se se quiser sigilo de m e s(m), com os mesmos cuidados de autenticação de atualidade e conteúdo.

Outros esquemas foram propostos para assinaturas digitais em ambientes de criptossistemas simétricos, por exemplo os devidos a Diffie & Lamport [DIFF1-76] e Meyer & Matyas [MEYE2-82]. Ambos estão descritos também, com maior detalhe, em [MEYE1-82]. No entanto, todos padecem do mesmo mal: a quantidade relativamente grande de informações que têm que ser geradas para a produção de uma assinatura; além disso, essa informação não pode ser reaproveitada em outras mensagens.

• ASSINATURAS USANDO CRIPTOSSISTEMAS ASSIMÉTRICOS COMUTATIVOS

Um criptossistema assimétrico, como vimos, tem a seguinte propriedade:

$$D(\ell,E(k,m)) = m.$$

Ou seja, para todo par de chaves k, ℓ, respectivamente pública (de ciframento) e secreta (de deciframento), o resultado do deciframento, sob a chave ℓ, de uma mensagem m cifrada sob a chave k, é a própria mensagem m.

Dizemos que um criptossistema assimétrico é *comutativo*, se, além da propriedade acima, ele tiver a propriedade adicional:

$$E(k,D(\ell,m)) = m.$$

Isto é, toda mensagem m pode ser inicialmente "decifrada" sob a chave secreta ℓ, e o resultado posteriormente cifrado sob a chave pública k restaurando m. Diz-se então que as funções E e D, de ciframento e deciframento respectivamente, comutam. Dos criptossistemas assimétricos apresentados neste trabalho, apenas o RSA tem também a segunda propriedade; o baseado no problema da mochila tem somente a primeira.

Usando criptossistemas assimétricos comutativos temos um meio imediato de produzir assinaturas digitais:

Sejam kA, ℓA as chaves públicas (de ciframento) e secreta (de deciframento), respectivamente do usuário A.

• A pode enviar uma mensagem assinada m para B como segue:

$$s(m) := D(\ell A, m), \tag{1}$$

isto é, A "decifra" m sob ℓA, gerando a assinatura s(m) que é enviada para B,

• B usa kA, pública, e executa:

$$E(kA, s(m)) \tag{2}$$

isto é, B "cifra" s(m) obtendo m, pois

$$E(kA, s(m)) = E(kA, D(\ell A, m)) = m.$$

O usuário B certifica-se assim que m só poderia ter vindo de A, pois somente ele conhece ℓA. Além disso, é inviável gerar s(m) sem conhecer ℓA, pois E é unidirecional-alçapão, pela definição de criptossistemas assimétricos.

É interessante observar que não há sigilo neste esquema; qualquer outro usuário além de B pode executar (2), pois kA é pública. Se quiser sigilo, além da autenticação de m, o usuário A deve cifrar s(m) com a chave pública de B, kB, antes de

enviá-la:

$$E(kB,s(m)) = E(kB,D(\ell A,m)) \qquad (1')$$

B por sua vez decifra $E(kB,s(m))$ usando sua chave secreta ℓB:

$$D(\ell B,E(kB,s(m)) = s(m), \qquad (2')$$

obtendo $s(m)$; a seguir autentica $s(m)$ como antes, reavendo m:

$$E(kA,s(m)) = m. \qquad (3')$$

Este método é prontamente adaptado se o criptossistema em uso for o RSA:

Sejam $\ell A = (dA,nA)$ e $kA = (eA,nA)$, as chaves secreta e pública respectivamente, de A. Analogamente para B, $\ell B = (dB,nB)$ e $kB = (eB,nB)$.

- A envia m, assinada e cifrada, para B executando:

  - inicialmente

    $$s(m) := m^{dA} \bmod nA$$

  - a seguir

    $(s(m))^{eB} \bmod nB$, é enviada

• B obtém m igualmente em dois passos:

• primeiramente

$$((s(m))^{eB})^{dB} \bmod nB = s(m)$$

• obtida s(m), B se certifica da assinatura calculando

$$(s(m))^{eA} \bmod nA = (m^{dA})^{eA} \bmod nA = m.$$

Um juiz pode facilmente resolver conflitos neste esquema; basta que a parte interessada, digamos B, lhe apresente s(m), alegando que esta lhe foi inviada por A. O juiz tem apenas que aplicar o algoritmo de ciframento

$$E(kA,s(m)),$$

e validar a mensagem m, dando ou não razão a B. Note-se que não é necessário que o remetente revele sua chave secreta ℓA.

Surge aqui uma questão: como o juiz, ou mesmo o destinatário B podem distinguir m, uma mensagem válida, de uma seqüência qualquer de bits eventualmente inserida na linha por um oponente? É necessário que ao execuatar E(kA,s(m)), B obtenha informações que lhe permitam fazer essa distinção. Resolve-se esse problema concatenando a m, antes de assiná-la, algum prefixo ou sufixo público que permita o reconhecimento posterior de m, tais como

as identidades do remetente e destinatário, etc. Pode-se também usar as técnicas de autenticação de conteúdo e atualidade já descritas.

• ASSINATURAS USANDO CRIPTOSSISTEMAS BASEADOS NO PROBLEMA DA MOCHILA

Na seção anterior, descrevemos assinaturas digitais usando criptossistemas assimétricos comutativos, como o RSA. Aqui veremos como usar os criptossistemas baseado no problema da mochila que não oferecem um meio imediato de gerar assinaturas, pois as funções de ciframento / deciframento não comutam; como vimos, o resultado do ciframento de uma mensegem, vista como um inteiro entre 0 e $2^n - 1$, é um número inteiro entre 0 e Σa_i, com $\Sigma a_i >> 2^n$; apenas uma fração $(2^n / \Sigma a_i)$ do contradomínio da função de ciframento é resultado do ciframento de alguma mensagem do domínio da função; assim, obviamente, o criptossistema não é comutativo. Apesar das propostas de Merkele & Hellman [MERK 3-78], para tornar seu criptossistema útil para gerar assinaturas, os resultados de Lagaryas & Odlyzko [LAGA1-83] possibilitam que um oponente forje assinaturas válidas. Descreveremos a seguir uma versão de um criptossistema baseado no problema da mochila, para gerar assinaturas, devido a Shamir.

• O MÉTODO DE SHAMIR

O método, [SHAM 5-78] não pode ser usado para ciframento pois não é um-a-um; se baseia numa variante do problema da mochila,

descrito a seguir:

• Dados m, n e A, encontrar C, se existir, que satisfaça

$$\sum_{j=1}^{k} c_j a_j \equiv m \pmod{n} \qquad (1)$$

onde $A = (a_1, a_2, \ldots, a_k)$, a_j inteiros, $1 \leq j \leq k$

$C = (c_1, c_2, \ldots, a_k)$, $0 \leq c_j \leq \log n$, inteiros e n é primo.

É possível mostrar que este problema é NP-completo.

Neste sistema, são públicos o vetor A e o inteiro n. Querendo assinar m, o usuário X resolve, de alguma maneira que somente ele conhece, o problema da mochila enunciado, encontrando C, a assinatura de m, que é enviada para o usuário y, junto com m. Este calcula $\sum_{j=1}^{k} c_j a_j \bmod n$, restaurando m, se a assinatura for correta. É importante que toda mensagem m, vista como um inteiro entre 0 e n - 1, seja assinável. O alçapão, ou informação secreta, que serve para assinar mensagens é obtida como segue:

Seguindo a sugestão do autor vamos adotar $k = 200$ e $[\log_2 n] = 100$.

Inicialmente escolhe-se uma matriz $E_{100 \times 200}$, cujos elementos são zeros e uns escolhidos aleatoriamente. O vetor A de

200 elementos é uma das soluções do seguinte sistema linear.

$$\underbrace{\begin{pmatrix} e_{0,1} & \cdots & e_{0,200} \\ \vdots & & \vdots \\ e_{99,1} & \cdots & e_{99,200} \end{pmatrix}}_{E} \underbrace{\begin{pmatrix} a_1 \\ \vdots \\ a_{200} \end{pmatrix}}_{A} = \begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{99} \end{pmatrix} \pmod{n} \qquad (2)$$

Como há 200 equações a 100 incógnitas, pode-se escolher aleatoriamente valores para $a_{101},\ldots,a_{200}$, e resolver o sistema, agora de 100 equações a 100 incógnitas $(a_1,\ldots,a_{100})$. A probabilidade de que não consigamos resolver o sistema é muito pequena, já que a matriz reduzida de 100×100, que é a metade esquerda de E, será singular apenas se o seu determinante for um múltiplo do primo n, que é muito grande. Mesmo que isto aconteça, basta escolher outra matriz E. Assim, cada uma das potências de 2, de 2^0 a 2^{99}, é expressa como a soma, modulo n, de alguns elementos de A. Dados A e n porém, recuperar a matriz E significa resolver 100 casos do problema da mochila.

Veremos agora como assinar um bloco m de 100 bits.

Seja $m = (m_{99}, m_{98}, \ldots, m_1, m_0)$, a representação binária de m. Isto é,

$$m = \sum_{i=0}^{99} m_i \, 2^i .$$

De (2) segue que:

$$m = \sum_{i=0}^{99} m_i \left[\sum_{j=1}^{200} e_{i_j} a_j \right] \pmod{n} = \sum_{j=1}^{200} a_j \left[\sum_{i=0}^{99} e_{i_j} m_i \right] \pmod{n} \qquad (3)$$

Assim, fazendo

$$c_j := \sum_{i=0}^{99} e_{i_j} m_i \qquad (4)$$

geramos o vetor $C = (c_1, c_2, \ldots, c_{200})$ que é a assinatura de m e satisfaz (1).

Nota-se que (4) é rapidamente computável a partir da soma das linhas de E, coluna a coluna, que correspondem a bits iguais a 1 na representação binária de m, ou seja

$$[m_1 \cdot \cdot \cdot \cdot \cdot m_{99}] \begin{pmatrix} e_{0,1} \cdot \cdot \cdot \cdot \cdot e_{0,200} \\ \\ \\ e_{99,1} \cdot \cdot \cdot \cdot e_{99,200} \end{pmatrix} = [c_1 \cdot \cdot \cdot c_{200}]$$

Um intruso, para forjar a assinatura C' de uma mensagem m', deve encontar o vetor C' sem conhecer a matriz E.

É fácil observar que toda mensagem, vista como um inteiro entre 0 e $2^{99} - 1$, pode ser assinada, pois as equações em (3) valem para qualquer m nesse intervalo.

Há várias fraquezas deste método e que foram descritas e analisada por Shamir [SHAM5-79]; todas porém são pequenas em face do resultado de Lagaryas & Odlyzko [LAGA1-83], que inviabiliza aplicações deste esquema, pois a matriz E pode ser determinada eficientemente a partir de A e n. Assim, assinaturas podem ser forjadas.

• ARBITRAMENTO DE ASSINATURAS

É importante introduzir a seguinte observação sobre a validade de assinaturas: Num criptossistema simétrico um usuário pode renegar a autoria de uma mensagem assinada, alegando, posteriormente, roubo ou perda da chave anterior à emissão daquela assinatura. No caso de criptossistemas assimétricos, a situação é ainda pior: um usuário pode publicar sua chave secreta (de deciframento) e tornar suspeitas todas as assinaturas produzidas com ela. A simples inclusão de data e hora na mensagem não resolve o problema; além disso, se a chave foi roubada, o ladrão pode assinar mensagens com data anterior àquela alegada pelo usuário legítimo, para prejudicá-lo. É necessário também manter um registro de chaves usadas em assinaturas antigas mas que devem ser autenticadas eventualmente.

Por tudo isso criou-se um terceiro elemento no processo de autenticação de assinaturas (o juiz servia apenas para resolver impasses; ele não participa do processo de autenticação): o *mediador*, por quem passará toda a informação necessária à autenticação de uma assinatura. Convencionou-se chamar de assinaturas *arbitradas* àquelas que usam o mediador, e assinaturas *universais* ou *verdadeiras* as que não usam, como é o caso do protocolo de Rabin. Esta distinção vale tanto para assinaturas num contexto de criptossistemas simétricos, quanto para contextos de criptossistemas assimétricos.

Needham & Schroeder [NEED1-78] propuseram um protocolo de autenticação de assinaturas onde o distribuidor de chaves, citado no protocolo de autenticação de usuários, funciona como mediador, num ambiente que usa criptossistemas simétricos:

Vamos supor que A queira enviar a mensagem m assinada para B

1) A envia para Dc

A,E(kA,r(m))

onde: r é uma função de compressão de m

kA é a chave secreta para conversação entre A e Dc

2) Dc responde enviando

w := E(kDc,[A, r(m)])

onde kDc é uma chave de conhecimento exclusivo de Dc.

3) A envia para B:

E(kAB,m), w

onde kAB é a chave secreta para conversação entre A e e B previamente obtida usando o protocolo de distribuição de chaves.

4) Após decifrar E(kAB,m), B calcula e armazena x:=r(m)

Envia então para DC

B,w

5) DC decifra w e envia para B

E(kB,[A,r(m)])

6) B pode então comparar se r(m) recebido em 5 é igual a x, validando a assinatura ou rejeitando-a caso contrário.

A figura 410 ilustra este protocolo.

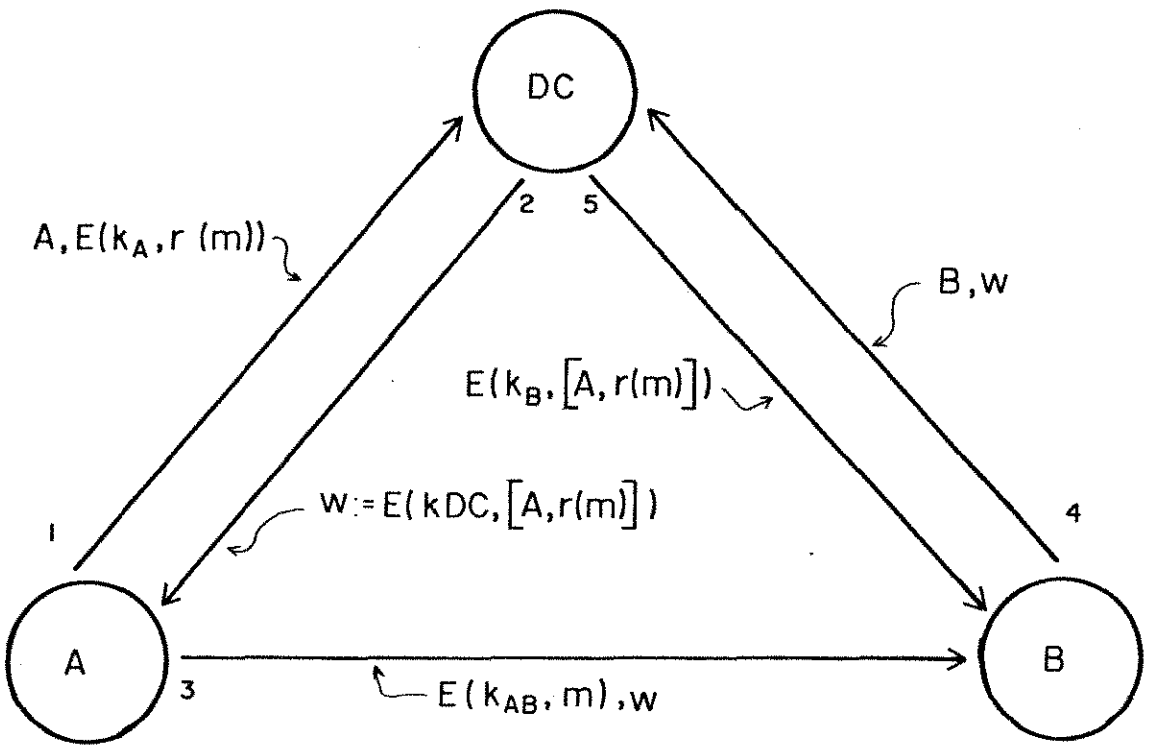


Figura 410: PROTOCOLO DE NEEDHAM & SCHROEDER PARA ASSINATURAS ARBITRADAS - CASO SIMÉTRICO.

A quantia w é chamada de *bloco de assinatura*, cujo conteúdo decifrado é inacessível a A e B, pois cifrado com kDC. Qualquer tentativa de modificação de w acarretará em não aceitação da assinatura. Conflitos são resolvidos tomando m e w e apresentando-os a um juiz. Este recalcula r(m) em colaboração com DC e consegue descobrir quem tem razão; supõe-se, é óbvio, que DC seja confiável. O papel do distribuidor de chaves, como vimos, é múltiplo; além de distribuir corretamente as chaves, ele autentica assinaturas e deve manter registros de chaves antigas. Por isso um nome mais adequado para ele seria *servidor de autenticação*, como está em [NEED1-78].

Um esquema ligeiramente diferente, mas baseados nos mesmos princípios foi proposto por Merkle [MERK 4-80].

Também para o caso de criptossistemas assimétricos comutativos, há o problema da autenticação de assinaturas antigas, isto é, mensagens assinadas com chaves não mais em uso. Há, além disso um novo problema: um usuário pode alegar perda ou roubo da sua chave secreta, exatamente aquela usada para confecção de assinaturas; ele poderá assim alegar que uma certa mensagem, que ele realmente enviou, foi assinada após a alegada perda da chave; mesmo que a mensagem seja assinada junto com "timestamps", isto não evita a ameaça descrita, pois o usuário pode declarar qualquer data como referente à da perda. Além disso se ele realmente perdeu sua chave ou foi roubado, o ladrão pode assinar mensagens

para si próprio com datas arbitrárias.

Novamente, torna-se necessário o emprego de assinaturas arbitradas, como as descritas para criptossistemas simétricos. Também aqui um servidor de autenticação DC, irá mediar a conversação entre dois usuários A e B, e além disso manterá um registro de chaves usadas em assinaturas antigas.

O protocolo que segue é uma modificação do descrito em [DENN 2-83], e usa o que se convencionou chamar *certificados de assinaturas* e *certificados de chaves públicas;* um e outro são informações expedidas pelo servidor DC a qualquer usuário, certificando a atualidade, ou melhor, validade, de uma chave pública. São usados como segue:

1) O usuário A, querendo enviar uma mensagem m assinada para B, computa inicialmente:

- r(m), a compressão de m, e
- $x := D(\ell A, [r(m), t_1])$, onde t_1 é um "timestamp";

isto é, A assina $[r(m), t_1]$ usando sua chave secreta ℓA.

Envia então para DC:

A, B, x.

2) O servidor DC, após se certificar da atualidade de x, computa:

$$p := D(\ell DC, [t_2, B, kB]) \quad e$$

$$g := D(\ell DC, [t_2, A, kA, x]);$$

isto é, DC assina as duas mensagens, usando sua chave secreta ℓDC; p é chamado *certificado da chave pública* kB de B; g é chamado *certificado da assinatura* x; t_2 é outro "timestamp".

3) A computa

E(kDC,p) obtendo t_2, B e kB e

E(kDC,g) obtendo t_2, A, kA, x;

O usuário A confere a atualidade das duas mensagens verificando t_2. Usando kB, A calcula

c := E(kB,m), o ciframento da mensagem m, e envia para B:

c, g

4) B finalmente calcula

• D(ℓB,c) = m, o deciframento de m.

• E(kDC,g) obtendo t_2, A, kA, x;

após conferir a atualidade de g, verificando t_2, B computa

E(kA,x), obtendo [r(m), t_2];

A seguir B recalcula r(m), usando o m obtido no decifra-mento de c, e confere com o valor recebido em x, aceitando ou rejeitando a assinatura.

Este protocolo garante que:

• Ambos os usuários A e B podem se certificar da atualidade das chaves de assinatura (secretas) e ciframento (público) de um e de outro, pois os certificados p e g, contendo o autenticador de atualidade t_2, são assinados por DC que, supõe-se, é confiável. Pode-se argumentar ainda que g deve ser calculado usando $t_3 > t_2$, para prever o atraso na retransmissão de A para B.

• Um espião não consegue se fazer passar por A perante DC, pois para isso teria que assinar uma mensagem com t_1 válido, o que é inviável.

• O sigilo de m continua garantido pois o acesso a r(m), dado x, não permite a um espião obter m. Poder-se-ia impor,

como medida adicional de segurança, que A cifrasse x, usando kDC, antes de enviá-la a DC.

O servidor DC mantém um registro de chaves públicas válidas e antigas, e assim pode autenticar assinaturas de qualquer época; basta que o interessado apresente o certificado correspondente, de onde o servidor obterá o "timestamp", resolvendo um possível impasse.

BIBLIOGRAFIA

[ADLE1 - 79] - L.ADLEMAN, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography" - *Proc. IEEE 20 th.Annual Symp. on Foundations of Computer Science, 55-60, Out 1979.*

[ADLE2 - 83] - L.ADLEMAN, "On breaking generalized knapsack public-key cryptosystems" - *Proc. 15 th. ACM Symposium on Theory of Computation,402-412, 1983.*

[BERL1 - 70] - E.R.BERLEKAMP, "Factoring polynomials over large finite fields" - *Mathematics of Computation,Vol. 24, 713-735, Julho 1970.*

[BLAK1 - 78] - B.BLAKLEY, G.R. BLAKLEY, "Security of number theoretic public key cryptosystems against random attack" - *Revista Cryptologia. Em três partes.Parte I: nº4, 305-321, 1978; Parte II: nº1, 29-42 , 1979; Parte III: nº2, 105-118,1979.*

[BLAK2 - 79] - G.R.BLAKLEY, I.BOROSH, "Rivest-Shamir - Adleman Public-key Cryptosystems do not always conceal messages" - *Comp. & Math. with Applic., Vol. 5 , 169-178, 1979.*

[BRAN1 - 79] - D.BRANSTAD,"Hellman's data does not support his conclusion". *IEEE Spectrum, Vol.16, 39,Julho 1979.*

[BREN1 - 80] - R.P.BRENT, "An Improved Monte Carlo Factorization Algorithm". *BIT, 20, 176-184, 1980.*

[DAVI1 - 79] - G.I.DAVIDA, "Hellman's Scheme Breaks DES in its

Basic Form", *IEEE Spectrum, Vol. 16, 39, Julho 1979.*

[DAVS1 - 79] - D.L.A.BARBER, D.W.DAVIES, W.L.PRICE, C.M.SOLOMONides, "Computer Networks and their Protocols" - *Wiley, New York, 1979.*

[DENN1 - 82] - D.E.DENNING, "Criptography and Data Security" - *Addison-Wesley, 1982.*

[DENN2 - 83] - D.E.DENNING, "Protecting Public keys and signature keys" - *Computer, 27 - 35, Fev, 1983.*

[DES1 - 77] - "Data Encryption Standard", *FIPS PUB 46, National Bureau of Standards, Washington, D.C., Jan 1977.*

[DIFF1 - 76] - W.DIFFIE, M.HELLMAN, "New Directions in Cryptography", *IEEE Trans. on Info. Theory, Vol. 22 , 397-427, Junho 1977.*

[DIFF2 - 76] - W.DIFFIE, M.HELLMAN, "A Critique of the Proposed Data Encryption Standard", *Comm. ACM, Vol. 19 , 164-165, 1976.*

[DIFF3 - 77] - W.DIFFIE, M.HELLMAN, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer, Vol. 10, 74-84, Junho, 1977.*

[DIFF4 - 81] - W.DIFFIE, "Cryptographic Technology:Fifteen year Forecast", *Secure Communications and Asymmetric Cryptosystems. AAAS Selected Symposia. Vol. 69, Westview Press, Boulder, Colorado, 1982.*

[EVAN1 - 74] - A.EVANS Jr, W.Kantrowitz, E.Weiss, "A User Authen

tication Scheme not requiring Secrecy in the Computer", *Comm. ACM, Vol. 17, 437-442, Agosto 1974.*

[FEIS1 - 73] - H.FEISTEL, "Cryptography and Computer Privacy" , *Scientific American, Vol. 228, 15-23, Maio 1973.*

[FEIS2 - 75] - H.FEISTEL, W.A. NOTZ, J.L.SMITH, "Some Cryptographic Techniques for Machine to Machine Data Communications", *Proc. IEEE Vol. 63, 1545-1554, Nov. 1975.*

[GORD1 - 82] - J.A.GORDON, H.RETKIN, "Are big S-boxes best?", *Lecture Notes in Computer Science, 149, 257-262, Springer-Verlag, 1983.*

[HARD1 - 65] - G.H. HARDY, E.M. WRIGHT, "An Introduction to the theory of numbers", *4º edição Oxford University Press, Londres, 1965.*

[HELL1 - 79] - M.HELLMAN, "Des will be totally insecure within ten years", *IEEE Spectrum, Vol. 16, 32-39, Julho 1979.*

[HELL2 - 80] - M.HELLMAN, "A Cryptanalytic time-memory tradeaff", *IEEE Trans. on Info. Theory, Vol. 26, 401-406 , Julho 1980.*

[HELL3 - 76] - M.HELLMAN, R. MERKLE, R.SCHROEPPEL,L.WASHINGTON, W.DIFFIE, S.POHLIG, P.SCHWEITZER,"Results of an initial attempt to Cryptanalyze the NBS Data Encryption Standard", *Information Systems Lab., Dept. of Electrical Eng., Stanford Univ, 1976.*

[HERL1 - 78] - T.HERLESTAM, "Critical remarks on some public — key cryptosystems", *BIT, nº18, 493-496, 1978.*

[KAHN1 - 67] - D.KAHN, "The Codebreakers", *Macmillan Co., Ney York, 1967.*

[KNUT1 - 69] - D.E.KNUTH, "The Art of Computer Programming" , *Vol. 2, Addison-Wesley, Reading, Mass, 1969.*

[KOLA1 - 77] - G.B.KOLATA, "Computer Encryption and the National Security Connection", *Science, Vol. 197,438-440, Julho 1977.*

[KOLA2 - 77] - G.B.KOLATA, "Cryptology: Scientist puzzle over science", *SCIENCE, Vol. 197, 1345-1346, setembro 1977.*

[KOLA3 - 78] - G.B.KOLATA, "Cryptology: a secret meeting at IDA?", *SCIENCE, Vol. 200, 184, Abril 1978.*

[KONH1 - 81] - A.G.KONHEIM, "Cryptography a primer", *Wiley-Interscience, 1981.*

[LAGA1 - 83] - J.C.LAGARIAS, A.M.ODLYZKO, "Solving Low-Density Subset Problems, *Proc. IEEE 24 th. Annual Symp. on Foundations of Comp. Science, 1-10, 1983.*

[LAMP1 - 81] - L.LAMPORT, "Password Authentication with Insecure Communication", *Comm. of ACM, Vol. 24, 770-772 , Novembro 1981.*

[LEMP1 - 79] - A.LEMPEL, "Cryptology in Transition", *Computing Surveys, Vol. 11, 285-303, Dezembro 1979.*

[LENN1 - 81] - R.E.LENNON et al., "Cryptographic Authentication of Time-Invariant Quantities", *IEEE Trans. on Comm., Vol. 29, 773-777, Junho 1981.*

[LUCC1 - 84] - C.L.LUCCHESI, "Introdução à Criptografia", *IV Escola de Computação, IME-USP, São Paulo, Julho 1984.*

[MEIJ1 - 82] - H.MEIJER, S.G.AKL, "Digital Signature Schemes", *Cryptologia, Vol,6, 329-338, Outubro, 1982.*

[MEIS1 - 76] - P.MEISSNER, "Report of 1976 Workshop on Estimation of Significant Advances in Computer Technology", *NBSIR 76-1199, National Bureau of Standards, Washington DC, Agosto 1976.*

[MERK1 - 78] - R.C.MERKLE, "Secure Communications over insecure channels", *Comm. ACM, Vol. 21, 294-299, Abril 1978.*

[MERK2 - 81] - R.C.MERKLE, M.E.HELLMAN, "On the Security of Multiple Encryption", *Comm. ACM, Vol. 27, 465-467 , Julho 1981.*

[MERK3 - 78] - R.C.MERKLE, M.E.HELLMAN, "Hiding Information and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Info. Theory, Vol. 24, 525-530, Setembro 1978.*

[MERK4 - 80] - R.C.MERKLE - "Protocolo for Public-key Cryptosystems", *Proc. 1980 IEEE Symp. Security and Privacy, 122-133, Oakland, Abril 1980.*

[MEYE1 - 82] - C.H.MEYER, S.M.MATYAS, "Cryptography: a new dimension in computer data security", *Wiley, 1982.*

[MEYE2 - 81] - C.H.MEYER, S.M.MATYAS, "Eletronic Signature for use with the Data Encryption Standard", *IBM Technical Disclosure Bulletin, Vol. 24, 2335-2336, Outubro 1981.*

[MILL1 - 75] - G.L.MILLER, "Riemman's Hypothesis and Tests for

Primality", *Proc. 7th. ACM Symp. on Theory of Computing, 234-239, Maio, 1975.*

[MORR1 - 79] - R.MORRIS, K.THOMPSON, "Password Security: A Case History", *Comm. ACM, Vol. 22, 594-597, Novembro 1979.*

[NEED1 - 78] - R.M.NEEDHAM, M.SCHROEDER, "Using Encryption for Authentication in Large Networks of Computers", *Comm. ACM, Vol. 21, 993-999, Dezembro 1978.*

[NEWS1 - 81] - "NSA'S Cryptic Alliance", Newsweek, 45, Agosto/24, 1981.

[NIVE1 - 72] - I.NIVEN, H.A.ZUCKERMAN, "An Introduction to the Theory of Numbers", *Wiley, New York, 1972.*

[POLL1 - 74] - J.M.POLLARD, "Theorems on factorization and primiality testing", *Proc. Camb. Phil. Soc. (1974), 76, 521-528.*

[PURD1 - 74] - G.P.PURDY, "A High Security Log-in Procedure", *Comm. ACM, Vol. 17, 442-445, Agosto 1974.*

[RABI1 - 78] - M.O.RABIN, "Digitalized Signatures", *Foundations of Secure Computation, ed. R.A. de Millo et. al., 155-166, Academic Press, New York, 1978.*

[RABI2 - 79] - M.O.RABIN, "Digitalized Signatures and Public-key functions as intractable as factorization", *MIT/LCS/TR-212, MIT Lab for Computer Science, Janeiro, 1979*

[RABI3 - 76] - M.O.RABIN, "Probabilistic Algorithms", *Algorithms*

and Complexity - New Directions and recent results, ed. J.F. Traub, 21-39, Academic Press, 1976.

[REPT1 - 81] - "Report of the Public Cryptography Study Group" - *Comm. ACM, Vol. 24, 434-450, Julho 1981.*

[RIVE1 - 78] - R.L.RIVEST, A.SCHAMIR, L.ADLEMAN, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *Comm. ACM, Vol. 21, 120-126, Fevereiro 1978.*

[RIVE2 - 77] - R.L.RIVEST, "Remarks on a Proposed Cryptanalytic Attack of the MIT Public-key Cryptosystem", *Cryptologia, Vol. 2, 62-65, Janeiro 1978.*

[RIVE3 - 78] - R.L.RIVEST, "Critical remarks on "Critical Remarks on Some Public-key cryptosystems" by T.Herlestam", *BIT, nº 19, 274-275, 1979.*

[RIVE4 - 84] - R.L.RIVEST, A.SHAMIR, "How to Expose an Eavesdropper", *Comm. ACM, Vol. 27, 393-395, Abril 1984.*

[SCHR1 - 79] - R.SCHROEPEL, A.SHAMIR, "A $TS^2 = O(2^n)$ Time / Space Tradeoff for Certain NP-complete Problems", *Proc. IEEE 20 th. Symp. on Found. of Comp. Sci, Outubro 1979.*

[SEN - 78] - UNITED STATES SENATE SELECT COMMITTEE ON INTELLIGENCE, "Unclassified Summary: Involvement of NSA in the Development of the Data Encryption Standard", *IEEE Communications Society Magazine, 53-55, Novembro 1978.*

[SHAM1 - 82] - A.SHAMIR, "A Polynomial Time Algorithm for breaking the basic Merkle-Hellman Cryptosystem", *Proc. 23*

rd. IEEE Symp. on Found. of Comp. Science, 145-152, 1982.

[SHAM2 - 80] - A.SHAMIR,R.E. ZIPPEL, "On the Security of the Merkle-Hellman Cryptographic Scheme", *IEEE Trans. on Info. Theory, Vol. 26, 339-340, Maio 1980.*

[SHAM3 - 79] - A.SHAMIR, "On the Cryptocomplexity of Knapsack Systems", *Proc. 11th. Symp. on the Theory of Computing, 118-129,1979.*

[SHAM4 - 78] - A.SHAMIR, "A Fast Signature Scheme", *MIT/LCS/TM-107, MIT Lab. for Computer Science, Cambridge , Mass, Julho 1978.*

[SHAM5 - 83] - A.SHAMIR, M.BEN-OR,B.CHOR, "On the Cryptographic Security of Single RSA bits", *Proceedings of the 15 th. Symp on the theory of Computing,402-412 , 1983.*

[SHAN1 - 49] - C.E.SHANNON, "Communication theory of Secrecy Systems", *Bell Syst. Tech. Journal, Vol. 28, 656-715, Outubro 1949.*

[SHAP1 - 78] - D.SHAPLEY, "Security Agency's Role in DES Confirmed", *SCIENCE, Vol. 200, 412-413, Abril 1978.*

[SHAP2 - 77] - D.SHAPLEY, "Telecommunications Eavesdropping by NSA on Private Messages Alleged", *SCIENCE, Vol . 197, 1061-1064, Setembro 1977.*

[SIMM1 - 79] - G.J.SIMMONS, "Symmetric and Asymmetric Encryption", *Computing Surveys, Vol. 11, 305-330,Dezembro 1979.*

[SIMM2 - 77] - G.J.SIMMONS, "Preliminary Comments on the MIT

Public-key Cryptosystem", Cryptologia, Vol.1,406-414, Outubro 1977.

[SOLO1 - 77] - R.SOLOVAY, V.STRASSEN, "A Fast Monte-Carlo Test for Primality", *SIAM Journal Computing, Vol.6,84-85, Março 1977.*

[TUCH1 - 79] - W.TUCHMAN, "Hellman Presents no Shortcut Solutions to the DES", *IEEE Spectrum, Vol. 16, 40-41, Julho 1979.*

[TUCH2 - 78] - W.TUCHMAN, *Palestra apresentada na National Computer Conference, Anaheim, California, Junho 1978.*

[WILL1 - 79] - H.C.WILLIAMS, B.SCHMID, "Some Remarks concerning the MIT Public-key cryptosystem", *BIT, nº 19,525-538, 1979.*

[WILL2 - 80] - H.C.WILLIAMS, "A Modification of the RSA Public-key Encryption Algorithm", *IEEE Trans. on Info. Theory, Vol. 26, 726-729, Novembro 1980.*

[YASA1 - 76] - E.K.YASAKI, "Encryption Algorithm: Key Size is the thing", *Datamation, Vol. 22, 164-166, Março 1976.*

[YUVA1 - 79] - G.YUVAL, "How to swindle Rabin", *Cryptologia, Vol. 3, 187-188, Julho 1979.*