Introdução à Segurança e Primitivas Criptográficas

March 9, 2018

Sumário

Sumário

Segurança: Definição

Segurança em Sistemas Computacionais: *"deals with the prevention and detection of unauthorised actions by users of a computer system"*, Dieter Gollmann in Computer Security, John Wiley & Sons, 1999

- Autorização requer ***autenticação*** e ***controlo de acesso***.
- Prevenir acções não autorizadas nem sempre é possível/economicamente viável, nesse caso teremos que nos contentar com a **detecção** dessas acções.
- Essencial para garantir a segurança dum sistema é definir a **política de segurança**, i.e. quais são as acções autorizadas e quais são as acções não autorizadas.

Segurança: Mais Definições

- Por vezes define-se segurança em termos de garantir:

  Integridade: impedir modificação não autorizada de informação;

  Confidencialidade: impedir o acesso não autorizado a informação;

  Disponibilidade: impedir que o acesso autorizado a informação seja negado.
- Esta definição é mais restrita e aplica-se apenas a informação, embora possa ser generalizada.
  - Em última análise, para que servem os computadores senão para aceder a informação (possivelmente processada)?
- Tal como na definição anterior, é notório que para garantir segurança é essencial definir o que é e o que não é autorizado.

Segurança: Processo

- Não há sistemas 100% seguros.
  - Mesmo que tecnicamente seja possível, tal poderá não se justificar em termos económicos.
- Implementar segurança requer uma *análise de risco*, formal ou não.
  - É essencial determinar as ameaças à segurança a que um sistema computacional pode estar sujeito.
- Desta análise resulta a especificação da política de segurança.
- Para implementar a política de segurança, recorre-se a mecanismos de segurança.
- Para verificar a conformidade da implementação com a política de segurança recorre-se à auditoria e à monitorização da operação através de *logs*.

Segurança: Ameaças

- Internas vs. externas;
- Passivas vs. activas;
- Ou ainda, quanto ao tipo de acção:

  Intercepção p.ex. ouvir a comunicação entre 2 entidades;
  Interrupção p.ex. impedir o acesso a um serviço Web, através dum ataque de *denial of service*;
  Modificação p.ex. alterar o conteúdo duma mensagem ou dum registo duma BD;
  Fabricação p.ex. acrescentar uma *password* a uma conta.
- Contrariar as ameaças de forma a satisfazer as políticas (requisitos) de segurança requer o recurso a **mecanismos de segurança**.

Segurança: Concepção

- A segurança **não deve** ser acrescentada no fim do projecto como mais uma camada:
  - Nessa altura, decisões previament tomadas podem restringir seriamente as opções.
- Alguns aspectos de projecto a considerar são:

  Camada em que camada dum sistema computacional (p.ex. sistema de comunicações, SO, aplicação) se deve implementar os mecanismos de segurança?

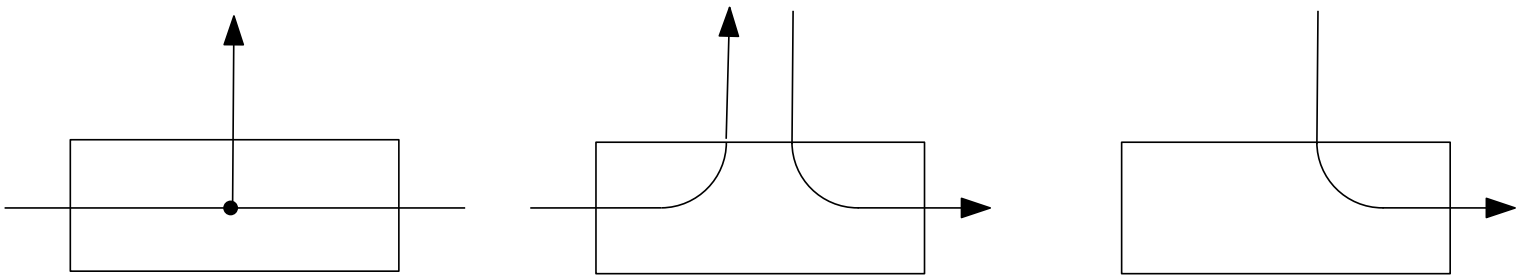
  Complexidade vs. Simplicidade o sistema deverá ter muita funcionalidade ou é importante garantir um grau de confiança elevado?

  Centralização vs. Descentralização De que componentes depende a segurança do sistema? Por outras palavras, qual a sua ***Trusted Computing Base (TCB)***?

Sumário

Criptografia

- É um dos mecanismos de segurança mais usados em sistemas distribuídos:
  - Permite proteger a comunicação entre entidades contra diferentes ameaças:



Primitivas Criptográficas

1. Algoritmos para Encriptação/Descodificação
2. Funções para Verificação de Integridade (funções de *hashing* criptográficas)
3. Algoritmos para Assinatura Digital

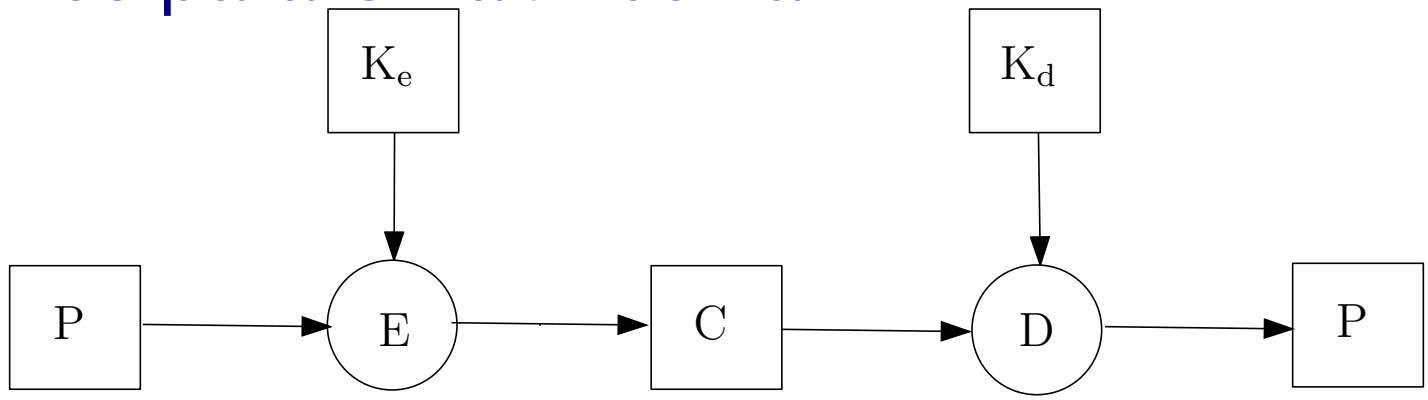
Princípio Fundamental Os algoritmos devem ser públicos. A segurança é obtida ***parametrizando*** os algoritmos com ***chaves***.

Tipos de Sistema Criptográfico Dois:

- **Simétricos (ou de chave partilhada)** usam uma única chave que é ***partilhada*** (K);
- **Assimétricos (ou de chave pública)** usam duas chaves uma das quais é ***pública*** (K^+) e a outra ***privada*** (K^-).

Sumário

Algoritmos para Cifrar/Decifrar



Simétricos, ou de chave partilhada: neste caso, as chaves para cifrar e decifrar são iguais:

$$K_e = K_d = K$$

- A chave deverá ser partilhada por todas as entidades autorizadas a aceder à informação.
- A chave deverá ser do conhecimento **apenas** dessas entidades.

Assimétricos, ou de chave pública: neste caso, as chaves para cifrar e para decifrar são diferentes:

$$K_e \neq K_d$$

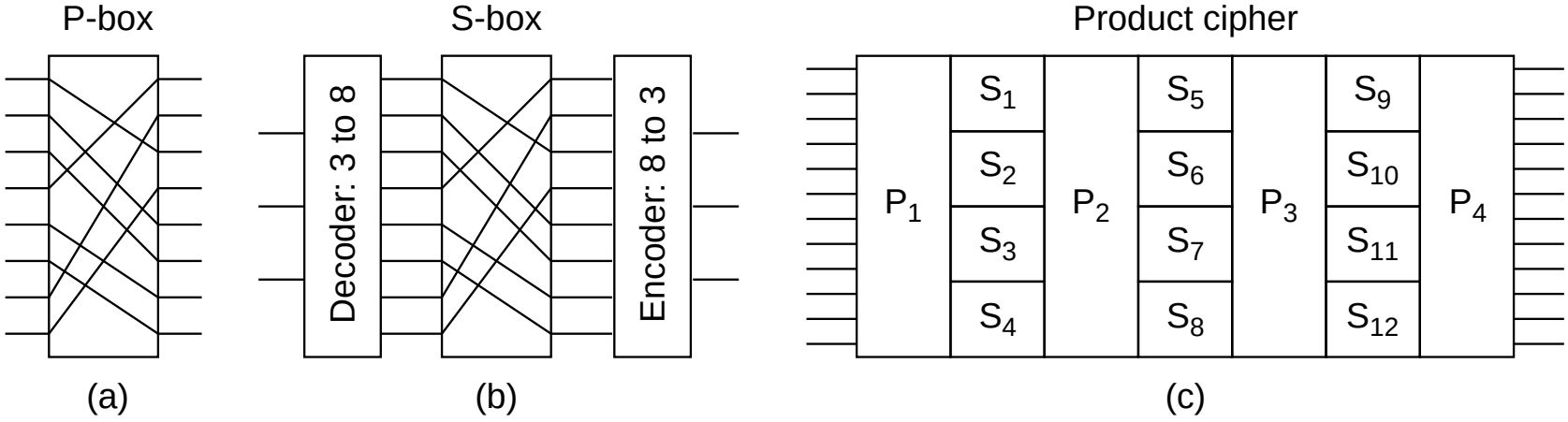
- Uma das chaves é pública e a outra privada. Qual é o quê?

Encriptação com Chave Partilhada: DES (1/3)

- *Data Encryption Standard (DES)* é uma norma dos EE.UU. para encriptação considerada vulnerável desde os meados dos anos 90:
  - Foi derrotada pela lei de Moore, como os seus conceptores previram.
- O algoritmo em si é relativamente simples e baseia-se na aplicação repetida de 2 operações básicas:

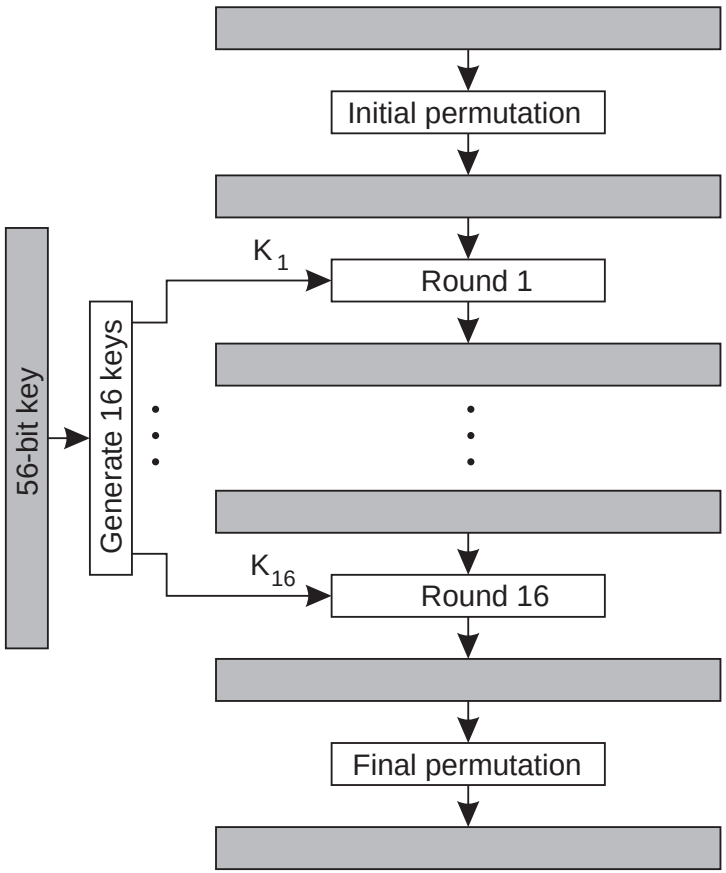
  Permutação de bits dum bloco;

  Substituição de subblocos de 6 bits, por outros sub-blocos de 4 bits.

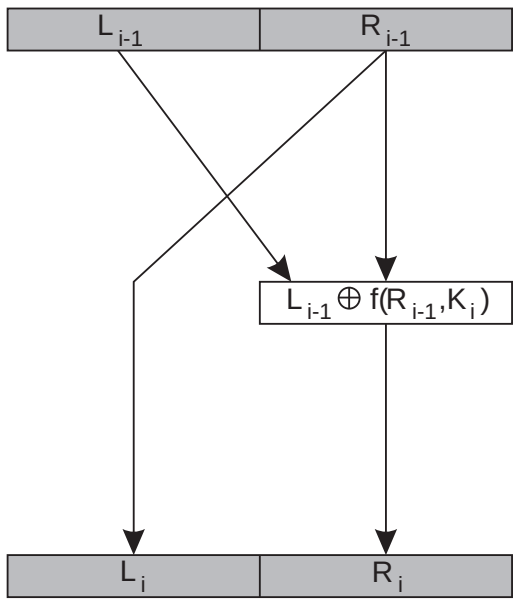


Encriptação com Chave Partilhada: DES (2/3)

- O algoritmo básico opera sobre blocos de 64 bits, que são transformados em blocos com o mesmo comprimento.
- O processo de encriptação dum bloco exige 16 passos (*rounds*).
  - Em cada passo usa-se uma chave diferente de 48 bits, gerada a partir da chave principal (*master*) de 56 bits.



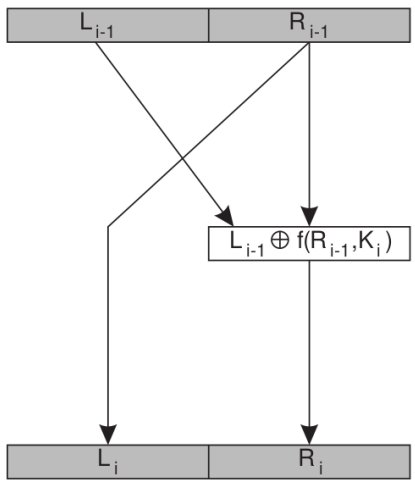
(a)



(b)

Encriptação com Chave Partilhada: DES (3/3)

- A permutação final é a inversa da permutação inicial.
- O verdadeiro trabalho é feito pela função não linear *(mangler function) (f)*



1. Expande R_{i-1} para um bloco de 48 bits;
2. Faz o XOR do resultado com a chave do passo, K_i;
3. Parte o resultado em 8 subblocos de 6 bits cada;
4. Cada subbloco é processado por uma função de substituição diferente que o converte num subbloco de 4 bits.
5. O conjunto de 8 subblocos de 4 bits é combinado num único de 32-bis, que é permutado.

- Esta mesma função pode ser usada para decifrar uma mensagem cifrada.
- DES foi substituída como norma por AES.

Encriptação com Chave Pública: RSA (1/3)

- RSA baseia-se na seguinte propriedade de aritmética módulo n:
  - Sejam p e q dois números primos
  - Sejam $n = p.q$ e $z = (p-1)(q-1)$
  - Sejam d e e dois números tais que: $d.e = 1 \ mod \ z$
  - Então, para qualquer x $(0 \leq x < n)$:
    $$x^{d.e} = x \ mod \ n$$

Encriptação com Chave Pública: RSA (2/3)

- O algoritmo para cifrar vem:
  1. Dividir a mensagem a enviar em blocos de comprimento fixo pré-estabelecido, tal que cada bloco m_i, interpretado como um número binário, seja menor do que n.
  2. Calcular para cada bloco:
     $$c_i = m_i^e \ mod \ n$$
- O algoritmo para decifrar a mensagem, vem:
  1. Decompor a mensagem recebida em blocos de comprimento fixo,
  2. Calcular: $m_i = c_i^d \ mod \ n$
- Assim, para garantir confidencialidade com RSA:
  - A chave para decifrar, $K_d = (d, n)$, deve ser secreta;
  - A chave para cifrar, $K_e = (e, n)$, deve ser pública.

Encriptação com Chave Pública: RSA (3/3)

- Como calcular as chaves?
  1. Escolher p e q, 2 números primos muito grandes, e.g. $> 10^{100}$;
  2. Calcular $n = pq$ e $z = (p-1)(q-1)$
  3. Escolher um valor d arbitrário.
  4. Usar o algoritmo de Euclides para calcular e:
     $$ed = 1 \ mod \ z$$
- A segurança de RSA está relacionada com a dificuldade da determinação dos factores dum número (n) muito grande.

Modos de Operação de *Block Ciphers* (1/3)

Observação A maioria dos algoritmos de encriptação operam sobre blocos de comprimento fixo (64 bits no caso de DES, p.ex.), sendo por isso designados por ***block ciphers***

- ***Stream ciphers*** são outro tipo de algoritmos que operam diretamente sobre sequências de bytes de comprimento arbitrário

Problema Como se pode cifrar dados/mensagens com comprimento superior ao de um bloco?

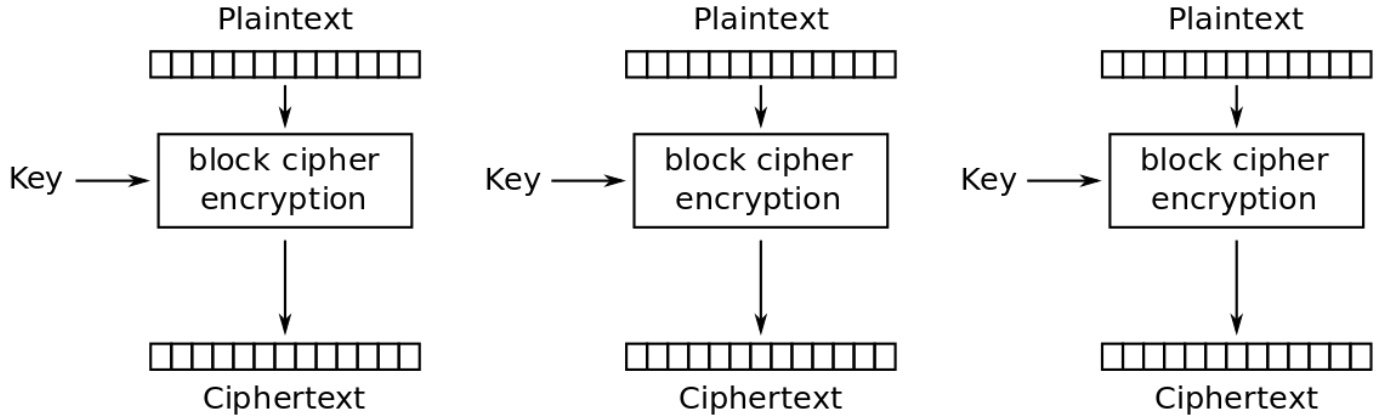
Solução Simplesmente:

1. Usar *padding* para que o comprimento dos dados a cifrar seja múltiplo do do bloco usado pela cifra
2. Decompor os dados em blocos que são posteriormente cifrados

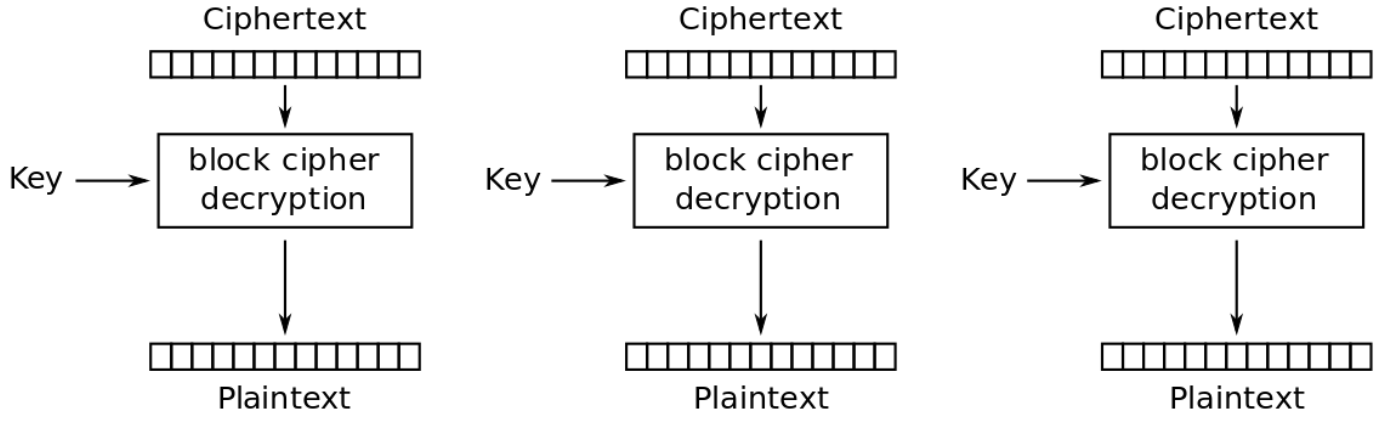
Este último passo pode ser realizado de diferentes formas designadas por ***modo de operação***

Modos de Operação de *Block Ciphers* (2/3)

Electronic Code Book (ECB)



Electronic Codebook (ECB) mode encryption



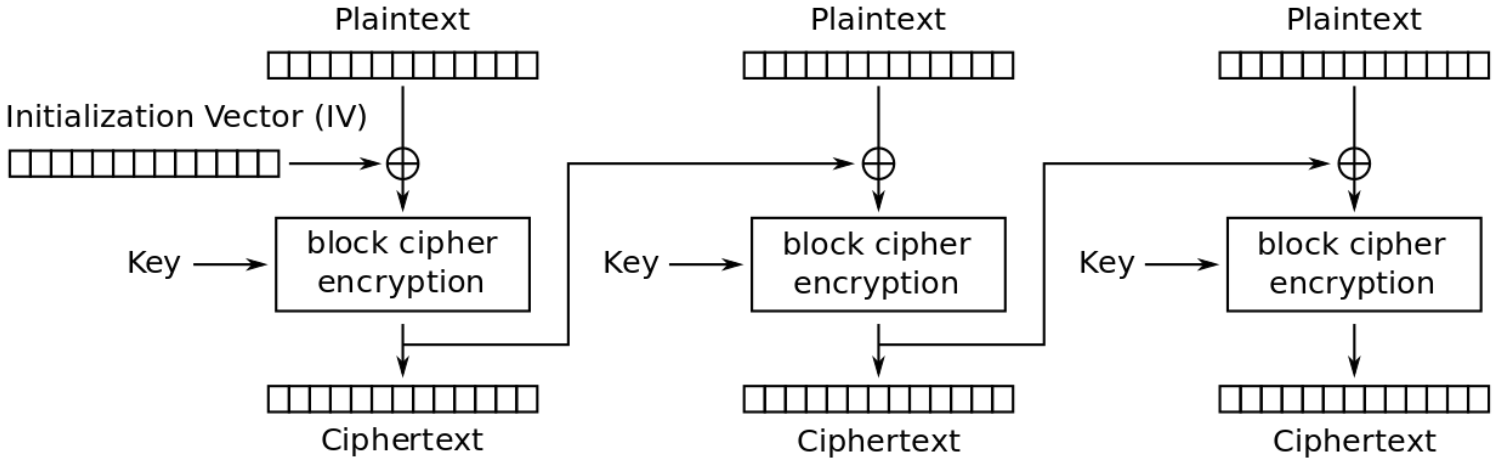
Electronic Codebook (ECB) mode decryption

Problema Facilita criptanálise

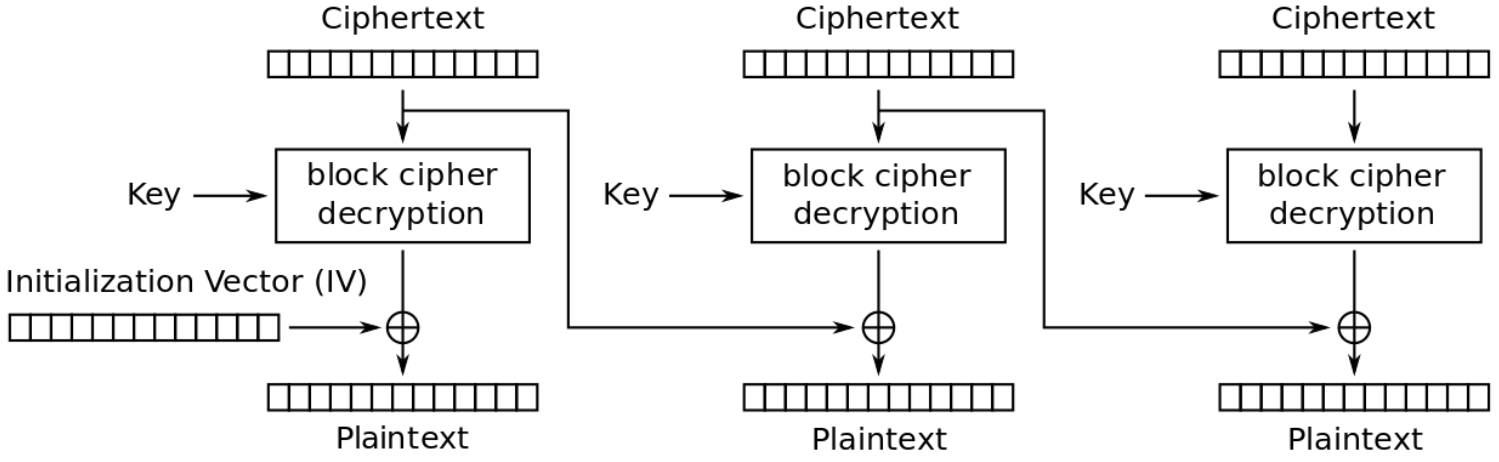
- Blocos de dados idênticos são sempre a blocos cifrados idênticos

Modos de Operação de *Block Ciphers* (3/3)

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

- O ***Initialization Vector*** é normalmente um valor (pseudo-)aleatório

Funções de *Hashing* Criptográficas

- São usadas para verificar a integridade de dados.
- Propriedades desejáveis duma função de *hashing* (h):

  Compressão mapeia a entrada de comprimento arbitrário num valor de *hashing* de comprimento fixo;

  Facilidade de Computação

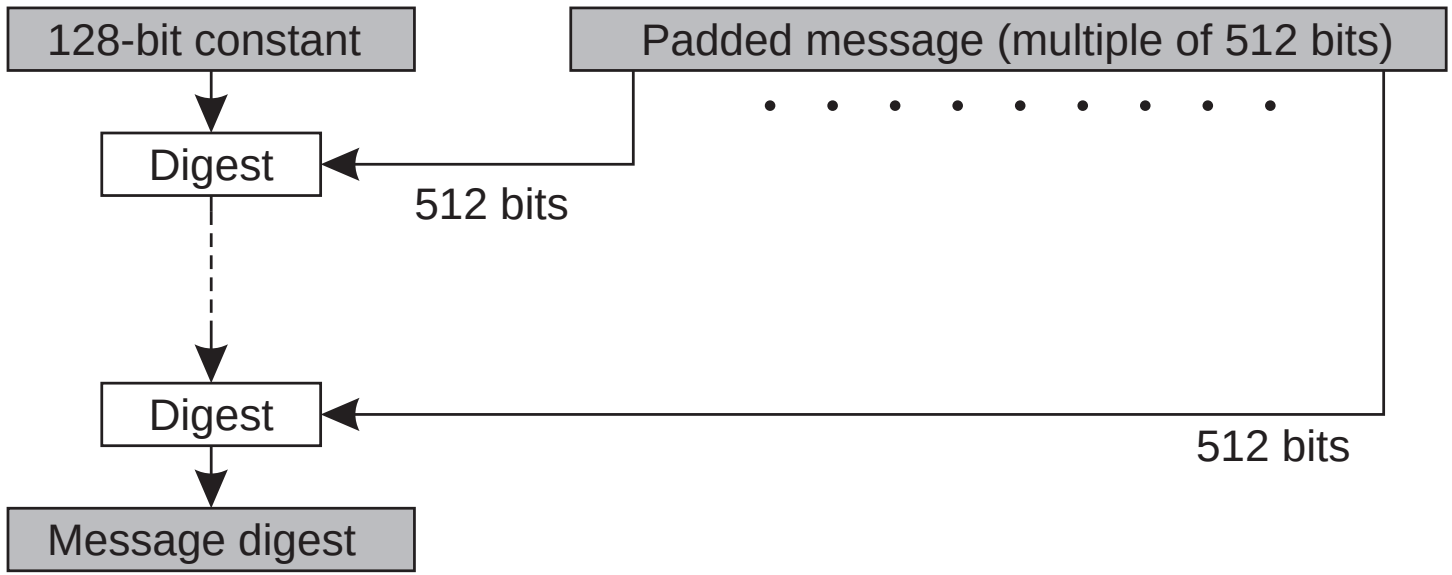
  Não Inversibilidade (*One-way*) dado um valor de *hashing*, y é computacionalmente inviável determinar um valor x tal que $y = h(x)$

  Fracamente Resistente a Colisões dado um valor x é computacionalmente inviável encontrar um valor x' diferente, tal que $h(x) = h(x')$

  Fortemente Resistente a Colisões é computacionalmente inviável encontrar dois valores x e x' tal que $x \neq x'$ e $h(x) = h(x')$

Funções de *Hashing* Criptográficas: MD5

- O algoritmo é executado em k fases, sendo k o número de blocos de 512 bits:
  - A entrada é pré-processada para garantir que o seu comprimento é múltiplo de 512 bits.
- Cada fase usa como entrada um número de 128 bits e um bloco de 512 bits, sendo a saída um número de 128 bits.



- Uma fase consiste em 4 passos (*rounds*) de computação;

MD5: Primeira Ronda duma Fase

- Cada sequência de 512 bits é decomposta em 16 blocos $(b_0, b_1, \ldots, b_{15})$ de 32 bits.
- As operações executadas na primeira ronda são:

Iterations 1-8	Iterations 9-16
$p \leftarrow (p + F(q,r,s) + b_0 + C_1) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_8 + C_9) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_1 + C_2) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_9 + C_{10}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_2 + C_3) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{10} + C_{11}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_3 + C_3) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{11} + C_{12}) \lll 22$
$p \leftarrow (p + F(q,r,s) + b_4 + C_5) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_{12} + C_{13}) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_5 + C_6) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_{13} + C_{14}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_6 + C_7) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{14} + C_{15}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_7 + C_8) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{15} + C_{16}) \lll 22$

  - p, q, r e s são variáveis de 32 bits – no total 128 bits– que são passadas duma fase para a seguinte.
  - Os C_i são constantes, no total 64 delas C_1 a C_{64}
  - F é *F(x,y,z) = (x AND y) XOR ((NOT x) AND z)*;
  - Em cada um dos outros 3 passos usam-se funções semelhantes G, H, I;

Autenticação com Funções de *Hashing*

- Se a função de *hashing* além da mensagem/dados tomar como entrada uma chave, pode ser usada também para autenticar a fonte e garantir a integridade da mensagem.
  - O valor de *hashing*, e por vezes a função, é conhecido por *message authentication code (MAC)*.
- Neste caso a função de *hashing* deve satisfazer uma propriedade adicional:

  Resistência Computacional para qualquer valor de k desconhecido, dados os valores $(x, h_k(x))$ é computacionalmente inviável calcular $h_k(y)$ para um valor y diferente.

  Porquê?
- HMAC (RFC) é uma MAC que garante o mesmo *nível de segurança* que a função de *hashing* usada.
  - MD5 é considerada pouco segura desde 2004
- A chave deve ser partilhada por ambos os lados
  - Um MAC não é equivalente a uma assinatura digital.

Assinaturas Digitais

- Uma assinatura digital deverá:
  1. Identificar o seu autor;
  2. Ser verificável por outros;
- MACs permitem identificar o autor duma mensagem face ao receptor, mas não permitem que um terceiro identifique o autor
  - O MAC pode ser gerado por qualquer entidade que conheça a mensagem e a chave – em princípio, as 2 entidades comunicantes.

  I.e., MACs não permitem **não-repudiação**.
- Primitivas para assinatura digital baseiam-se, tipicamente, em sistemas de encriptação assimétricos.

Assinaturas Digitais com RSA

- Algoritmos de encriptação de chave pública, e.g. RSA, podem ser usados para gerar assinaturas digitais.
- Na sua forma mais básica, a assinatura é a própria mensagem cifrada (C)
  - A obtenção de P usando a chave pública para decifrar, é prova suficiente.
- Na prática:
  1. Calcula-se um valor de *hash* da mensagem a assinar;
  2. Cifra-se esse valor – o resultado é a assinatura.
- Algoritmos para assinatura digital não precisam ser invertíveis, p.ex. DSA.

```
Signature sign(Message m, Key K⁻)
Boolean check(Message m, Signature s, Key K⁺)
```

Sumário

Força dos Mecanismos Criptográficos (1/2)

Empiricamente seguro , usa o teste do tempo. P.ex. DES.

- Não tem falhas óbvias;
- Embora não haja provas da sua segurança, é reconhecido seguro pela comunidade criptográfica.

Demonstravelmente seguro , usa a teoria da complexidade. Se quebrá-lo exigir a resolução dum problema para o qual não há uma solução computacionalmente eficiente. P.ex. RSA:

- A complexidade é medida em termos assimptóticos: quanto é *suficientemente grande*?
- Na realidade, não há prova de que a factorização não pode ser feita em tempo polinomial.
  - Se as constantes envolvidas fossem muito grandes, poderia não ser um problema.

Este tipo de algoritmos pode ser quebrado por um atacante com capacidade de processamento suficiente.

- É uma questão de tempo;
- ... e de comprimento das chaves.

Força dos Mecanismos Criptográficos (2/2)

Incondicionalmente seguro usa a teoria da informação. Um algoritmo é seguro se um atacante não conseguir extrair informação sobre a informação cifrada a partir da observação da cifra.

- A história mostra que algoritmos criptográficos **publicados** são quebrados normalmente por uma gestão inadequada das chaves e não tanto por vulnerabilidades intrínsecas aos algoritmos.
  - E estas últimas normalmente aparecem quando se consideram cenários de ataque que violam os pressupostos usados na sua concepção.
- Com algoritmos não publicados normalmente a história é outra.
  - DeCSS é talvez o exemplo mais recente e mais publicitado.

A Última Palavra aos Peritos

- *"If you think cryptography will solve your problem then you don't understand cryptography ... and you don't understand your problem."*, Bruce Schneier
- *"Cryptography is rarely ever the solution to a security problem. Cryptography is a translation mechanism, usually converting a communications security problem into a key management problem and ultimately into a computer security problem."*, Dieter Gollmann in Computer Security, John Wiley & Sons, 1999

Sumário

Leitura Adicional

- Capítulo 9 de Tanenbaum e van Steen, *Distributed Systems, 2nd Ed.*
  - Secção 9.1: *Introduction to Security*