UNIVERSIDADE DE ÉVORA

Sistemas Distribuídos

Segurança em SD: fundamentos de criptografia

Introdução

Conceitos:

- política de segurança: **regulamento** para alcançar os objetivos de segurança
- mecanismo de segurança: **técnica** usada para implementar uma política de segurança (ex: fechadura, algoritmo de criptografia)

- *Principal*: entidade (utilizador ou processo) envolvida numa operação

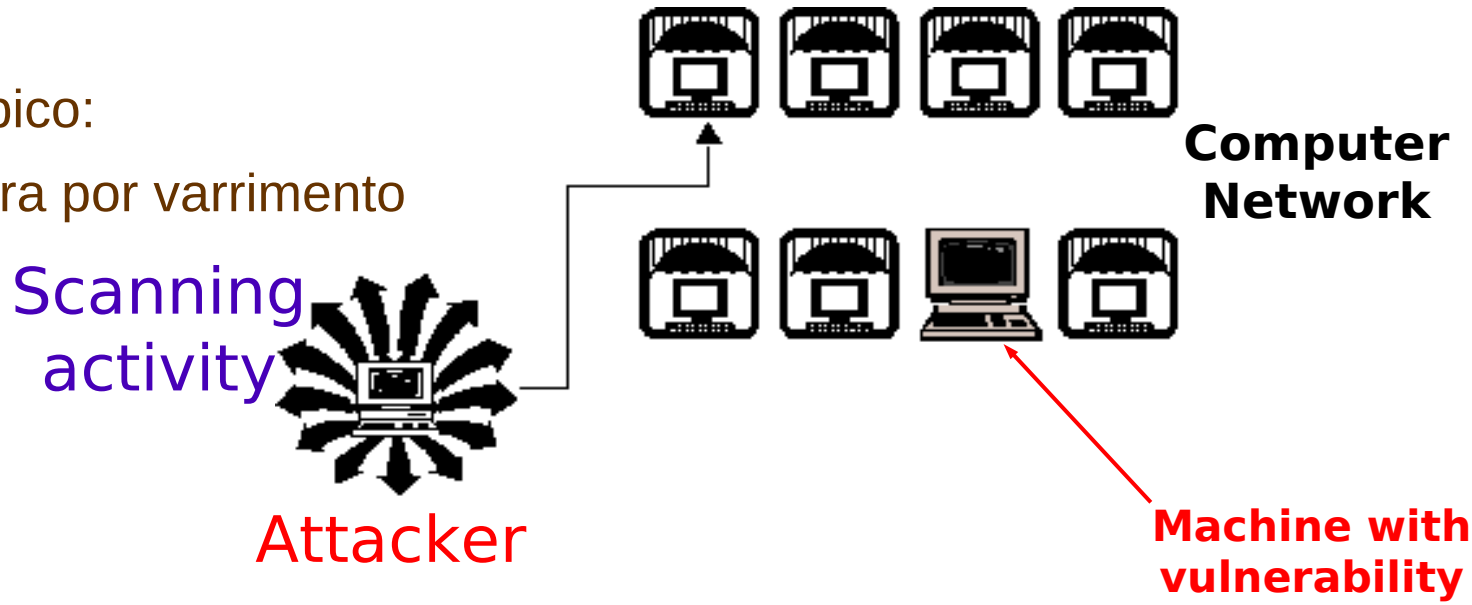
- Criptografia é uma ciência que utiliza funções matemáticas para cifrar e decifrar dados
- Criptoanálise é a ciência da análise e quebra de segurança de sistemas baseados em criptografia
- Criptologia envolve Criptografia e Criptoanálise
- Chave é um parâmetro do algoritmo criptográfico

Introdução

- **Segurança**: requisito que se pretende num sistema com vista a garantir objetivos como:
  - autenticação
    - garantia sobre a identidade de um interveniente ou a origem de uma mensagem
  - integridade
    - não houve alteração na informação
  - confidencialidade/sigilo
    - o acesso à informação é feito apenas por intervenientes autorizados
  - não-repúdio
    - Garantia de que o envolvimento numa operação não pode ser posteriormente negado
  - assinatura digital
    - Atestado de ligação de uma entidade a um documento, para provar a sua origem, ou que a entidade teve conhecimento do respetivo conteúdo

Intrusões (1)

- Intrusões são ações para contornar os mecanismos de segurança em sistemas informáticos.
- Podem ser causados por:
  - Atacantes remotos que acedem pela rede
  - *Insiders* (utilizadores locais) – utilizadores do sistema que tentam ganhar privilégios ou fazer uso indevido das suas capacidades
- Cenário típico:
  - Procura por varrimento



Vipin Kumar, "Parallel and Distributed Computing for Cybersecurity," *IEEE Distributed Systems Online*, vol. 6, no. 10, 2005

Ataques a um Sistema

ATAQUES onde se materializam uma ou mais ameaças

- *eavesdropping*
  - obter cópias de mensagens sem autorização
- *masquerading*
  - envio ou recepção de mensagens utilizando uma identidade de outro *principal* sem o seu consentimento
- *message tampering*
  - interceção e alteração de mensagens, que em seguida são enviadas para o destinatário original (Ex: *man-in-the-middle*)
- *replaying*
  - guardar uma mensagem interceptada para enviar mais tarde (pode funcionar mesmo com o uso de autenticação e cifra de mensagens)
- *denial of service*
  - congestionamento de um canal ou recurso para impedir o acesso por parte dos utilizadores comuns

Técnicas de Segurança

Firewalls

- filtros (origem, destino, porto, protocolo) aplicados ao tráfego da rede

Controlo de Acesso (de um processo a um recurso)

- tabela de permissões verificada no servidor

Certificados, Credenciais

Elementos que atestam algo sobre quem o detém

Identidade?

Autorização?

Criptografia, com o propósito de conseguir:

autenticação

integridade

confidencialidade

assinaturas digitais

não repúdio

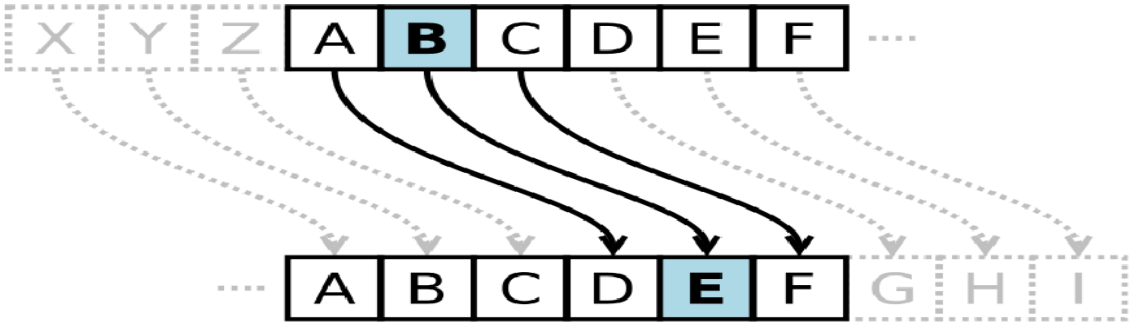
Criptografia: encriptação antiga

Scytale Transposition Cipher

- *O segredo é o diâmetro do cilindro*

Caesar Substitution Cipher

- *Rotação de N posições no alfabeto*



Criptografia: encriptação antiga

Vigenère Polyalphabetic Substitution

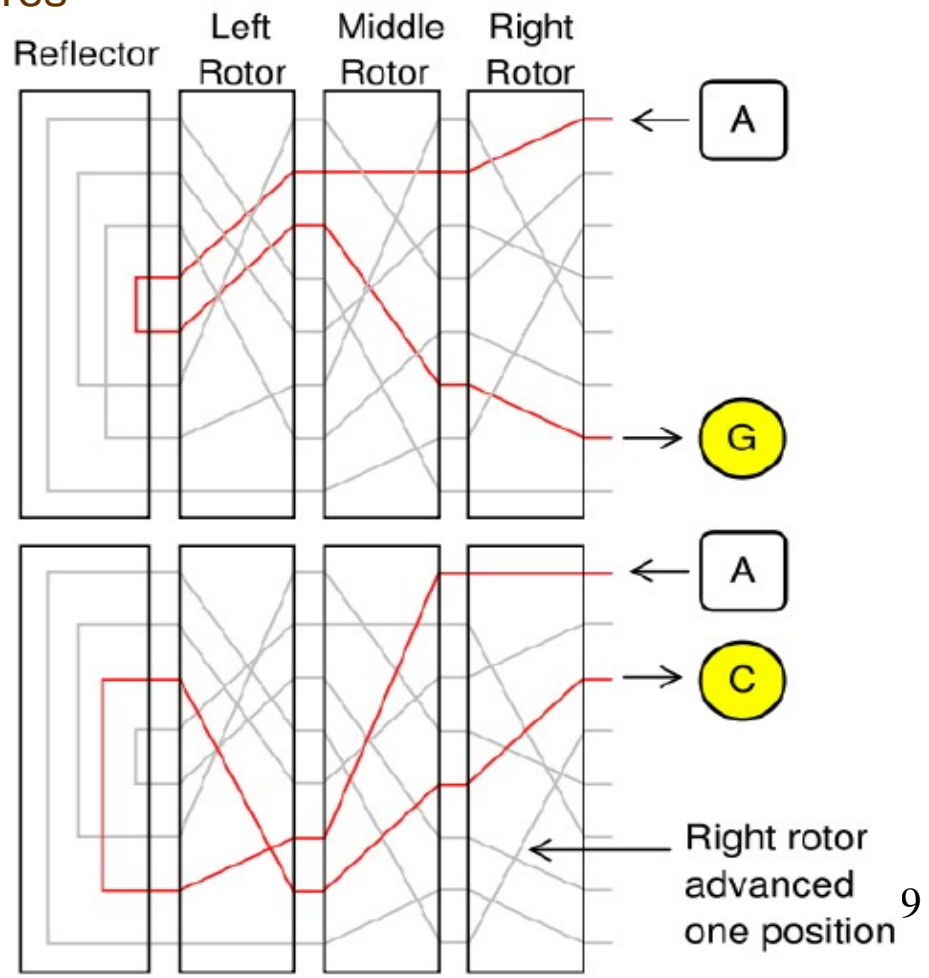
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Key:
GOOGLE
Plaintext:
BUYYOUTUBE
Ciphertext:
HIMEZYZIPK

Criptografia: encriptação antiga

- Cifra baseada em mecanismos com rotores
  - Maior variabilidade que os anteriores

Enigma – *2ª Guerra Mundial*



Notações Criptográficas

- Convenção simbólica

K_A	Alice's **secret key**
K_B	Bob's secret key
K_{AB}	Secret key shared between Alice and Bob
K_{Apriv}	Alice's **private key** (known only to Alice)
K_{Apub}	Alice's **public key** (published by Alice for all to read)
$\{M\}_K$	Message M encrypted with key K
$[M]_K$	Message M signed with key K

Algoritmos de Encriptação

- algoritmos usados para
  - transformar *plaintext* (mensagem ou dados no formato original) em *ciphertext* (dados codificados de modo ofuscado)
    - $E(K,M) = \{M\}_K$
  - o receptor do *ciphertext* aplica-lhe outra função do algoritmo para obter o *plaintext*
    - $D(K,E(K,M)) = M$

- **Tipo de Algoritmos de Encriptação:**
  - simétricos
  - assimétricos
  - outros: híbridos, block cipher, stream cipher

Algoritmos Simétricos

- chave secreta é partilhada (e escondida de todos os outros)
  - é argumento da função de encriptação E e da função de desencriptação D
- baseados em funções *one-way*
  - $F_k([M]) = E(K, M)$ fácil de calcular
  - função inversa $F^{-1}{}_k([M])$ tão difícil que é impraticável
- exemplo: DES

Algoritmos Simétricos: TEA

- Tiny Encryption Algorithm (TEA), Wheeler and Needham 1994
  - o plaintext é visto como sequência de blocos de 64 bits (2 inteiros 32 bits – *vector text[]*)
  - chave de 128 bits (4 inteiros de 32 bits)
  - usa
    - adição de inteiros (*linhas 4, 5, 6*)
    - bitwise logical shifts >> e << (*linhas 5, 6*), procura alcançar:
      - difusão
        - esconde repetição e redundância no *plaintext*
      - confusão
        - combina cada bloco do *plaintext* com a chave
        - ofusca a relação dos blocos em M com os de {M}K
        - evita a dedução da chave pela análise da frequência de caracteres no texto

Algoritmos Simétricos: TEA

- Função de encriptação

```
void encrypt(unsigned long k[], unsigned long text[]) {
        unsigned long y = text[0], z = text[1];                                  1
        unsigned long delta = 0x9e3779b9, sum = 0; int n;                        2
        for (n= 0; n < 32; n++) {                                                3
                sum += delta;                                                    4
                y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);            5
                z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);            6
        }
        text[0] = y;  text[1] = z;                                               7
}
```

Exclusive OR

logical shift

Algoritmos Simétricos: TEA

- Função de desencriptação

```
void decrypt(unsigned long k[], unsigned long text[]) {
        unsigned long y = text[0], z = text[1];
        unsigned long delta = 0x9e3779b9, sum = delta << 5;  int n;
        for (n= 0; n < 32; n++) {
                z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
                y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
                sum -= delta;
        }
        text[0] = y; text[1] = z;
}
```

Algoritmos Simétricos: TEA

- aplicação...

```
void tea(char mode, FILE *infile, FILE *outfile, unsigned long k[]) {
/* mode is 'e' for encrypt, 'd' for decrypt, k[] is the key.*/
        char ch, Text[8]; int i;
        while(!feof(infile)) {
                i = fread(Text, 1, 8, infile);          /* read 8 bytes from infile into Text */
                if (i <= 0) break;
                while (i < 8) { Text[i++] = ' ';}       /* pad last block with spaces */
                switch (mode) {
                case 'e':
                        encrypt(k, (unsigned long*) Text); break;
                case 'd':
                        decrypt(k, (unsigned long*) Text); break;
                }
                fwrite(Text, 1, 8, outfile);            /* write 8 bytes from Text to outfile */
        }
}
```

Algoritmos Simétricos: TEA

- Desempenho
  - cerca de 3 vezes mais rápido que DES
- 128 bits na Chave
  - resistente contra ataques de força bruta

Algoritmos Simétricos: DES

- Data Encryption Standard (DES)
  - desenvolvido pela IBM, ANSI standard (1977)
  - encripta blocos de 64 bits de *plaintext em ciphertext* com igual tamanho
  - chave de 56 bits
  - encriptação
    - 16 “rondas” de rotação de bits
    - number of bits to be rotated determined by key plus 3 key-independent transpositions
  - algoritmo lento quando implementado em software, para os computadores da década de 70 e 80, mas rápido quando executado em *VLSI hardware* (*very large scale integration*)

Algoritmos Simétricos: DES

- DES foi quebrado/cracked pela 1ª vez em Junho de 1997
  - ataque de força bruta para descobrir a chave dado um para plaintext/cyphertext, e usá-la para decifrar uma mensagem encriptada
  - envolveu a participação de 14000 utilizadores de Internet, que correram uma aplicação em *background* nas suas máquinas
  - capacidade média estimada das máquinas: 200 MHz Pentium Processor
  - a chave foi descoberta em 12 dias, depois de se analisarem 25% dos 2^{56} valores possíveis (houve alguma sorte também!)
  - um segundo ataque em 1998, com hardware dedicado, levou 3 dias
  - os ataques recentes precisam de menos de 24 horas

- DES com chave de 56 bits pode considerar-se obsoleto

Algoritmos Simétricos: triple-DES

- triple-DES
  - aplica o DES por 3 vezes, com duas chaves
  - $E_{3DES}(K_1, K_2, M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_1, M)))$
  - semelhante a um algoritmo simétrico comum com chave de 112 bits
  - mais resistente a ataques de força bruta que DES
  - desvantagem: má performance
    - são três operações DES

Algoritmos Simétricos: IDEA

- International Data Encryption Algorithm (IDEA)
  - desenvolvido em 1990, *Lai and Massey*
  - sucessor do DES
  - chave de 128 bits para encriptar blocos de 64 bits
  - baseado na álgebra de grupos
    - oito “rondas” de XOR, adição, multiplicação
  - como no DES, a mesma função serve para encriptar e desencriptar
    - uma vantagem quando se pretende implementar por hardware
  - mais seguro que o DES
  - 3 vezes mais rápido

Algoritmos Simétricos: AES

- Advanced Encryption Standard (AES)
  - resultou de “invitation for proposals” (US NIST 1997-2001)
  - Rijndael (Daemen and Rijmen*) algorithm
    - algoritmo baseado em iterações sobre blocos
    - comprimento variável para chaves e blocos
      - cada um pode ter, de forma independente: 128, 192 ou 256 bits de comprimento

- provavelmente o algoritmo simétrico mais utilizado

* - http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf

Algoritmos Assimétricos

- par de chaves pública e privada
  - a chave **pública** é divulgada e a **privada** mantida em segredo
- baseados em funções *trap-door*
  - *"função one-way com escapatória"*
  - inversa é muito difícil de calcular, excepto com o conhecimento de um segredo
- na encriptação usa-se a chave pública do destinatário (para confidencialidade)
- a desencriptação só é possível com a chave privada
- Vantagem: não há necessidade de
  - confiar uma chave secreta a outro interveniente (que a pode difundir)
  - mecanismo seguro de distribuição de chaves secretas
- Computacionalmente mais pesados que os simétricos, devido às operações com nºs primos elevados
- Exemplo: RSA

Algoritmos Assimétricos: RSA

- Rivest, Shamir and Adleman (RSA), 1978
- princípio
  - encriptação: baseada na multiplicação de nºs primos elevados
    - é computacionalmente inviável tentar fatorizar o resultado (para tentar descobrir os multiplicandos primos a partir dos quais se geram as chaves)
    - cada bloco de plaintext é tratado como um inteiro que vai ser alterado com operações potência e módulo
  - desencriptação: *trap door function*
    - é necessária outra chave do par*
    - operações de potência e módulo

* - a chave usada depende do propósito (confidencialidade/assinatura – ver adiante)

Comunicação com chaves públicas

Bob has a public/private key pair <K_{Bpub}, K_{Bpriv}>

1. Alice obtains Bob's public key K_{Bpub}
2. Alice creates a new shared key K_{AB} , encrypts it using K_{Bpub} using a public-key algorithm and sends the result to Bob.
3. Bob uses the corresponding private key K_{Bpriv} to decrypt it.

- Mallory pode intercetar o pedido inicial de Alice
  - Interfere enviando a sua chave pública no lugar da chave pública de Bob
    - Pode enganar ambos os interlocutores (Alice,Bob) ficando no meio das mensagens de ambos, dissimulado
    - *Man In The Middle Attack*

Algoritmos Híbridos

- resolvem o problema de exigência computacional dos algoritmos assimétricos
- robustos

- combinam técnicas de encriptação simétrica e assimétrica
  - criptografia de chave pública para autenticar os intervenientes e para transmissão de chaves secretas
  - algoritmos simétricos de chave secreta para restante encriptação

- ex: PGP, SSL

Algoritmos Híbridos: SSL

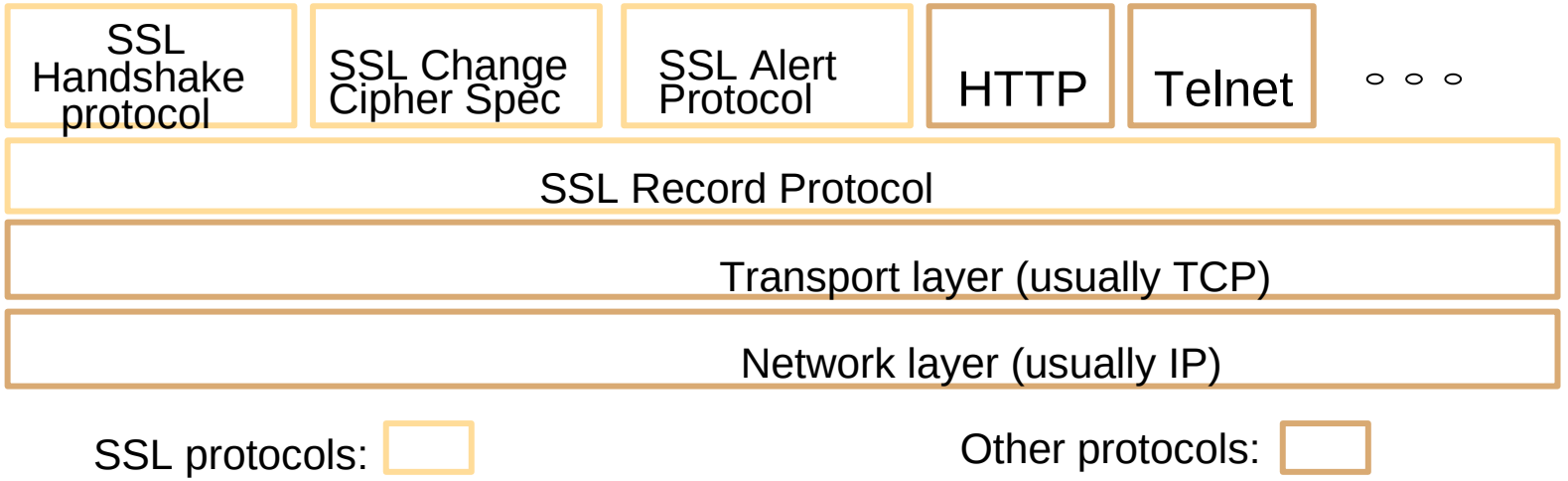
- Secure Sockets Layer (SSL) - Netscape Corporation, 1996
  - mecanismo híbrido: autenticação e troca de chaves secretas via criptografia de chave pública
  - TLS: uma norma que resulta da extensão do SSL
  - requer apenas certificados de chave pública atribuídos por uma CA reconhecida por ambas as partes
  - APIs do protocolo disponíveis em Java (e outras linguagens e ferramentas)

  - Permite
    - negociação dos algoritmos de autenticação e encriptação (facilita a comunicação entre plataformas distintas num sistema heterogéneo – *porque podem negociar algoritmos suportados por ambas as partes*)
    - estabelecimento de canal seguro sem contacto prévio ou ajuda de terceiros
      - os certificados usados devem ser emitidos por entidades reconhecidas por ambas as partes
      - o nível de segurança é acordado, em cada sentido. Pode haver apenas autenticação de uma das partes, por exemplo (ou então das duas).

Algoritmos Híbridos: SSL

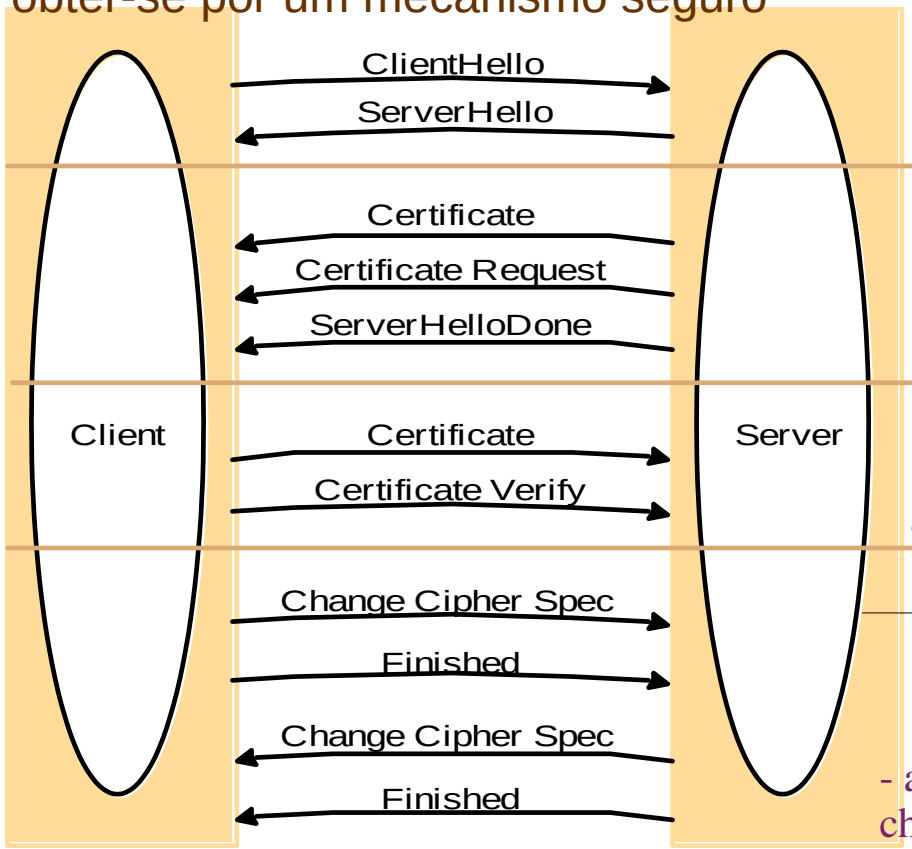
- Duas Camadas *(ao nível da camada de sessão do modelo OSI)*
  - Handshake Layer
    - negociação de algoritmos, geração e envio de chaves
  - SSL Record Protocol Layer
    - encriptação de mensagens e autenticação através de um protocolo com conexão (ex: TCP)
    - **pode garantir: integridade, confidencialidade e autenticação da fonte**

      mas depende da configuração usada



Algoritmos Híbridos: SSL

- Handshake
  - vulnerável a *man-in-the-middle*
    - *Para evitar isso:* a chave pública para validar o certificado do interlocutor deve obter-se por um mecanismo seguro



* pre-maste-secret
um elevado valor aleatório, usado por ambos para gerar as 2 chaves de sessão, para encriptar em cada sentido, e ainda para gerar os message authentication secrets

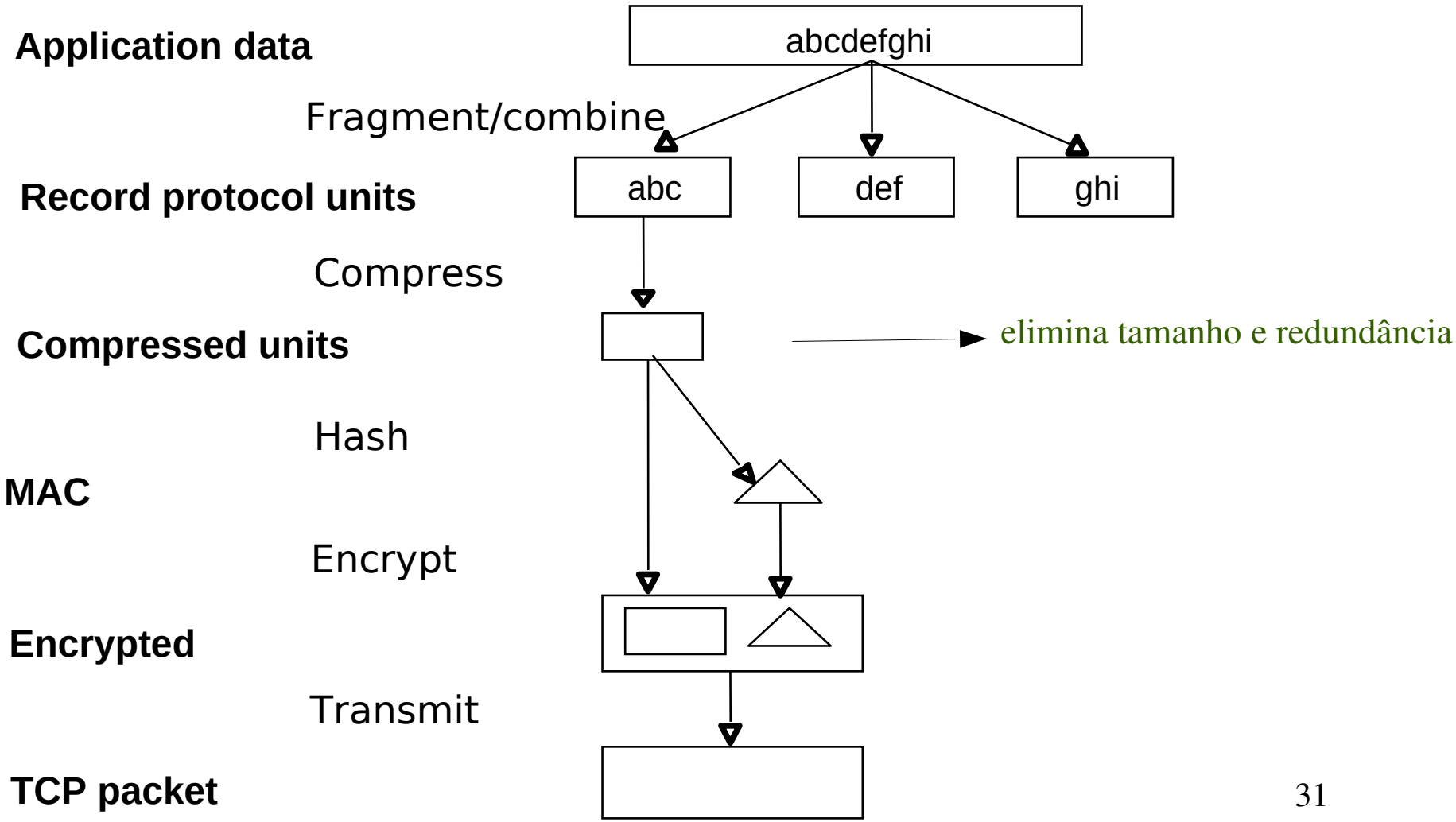
Algoritmos Híbridos: SSL

- opções configuradas no handshake

Component	*Description*	***Example***
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA
Message digest function	for creating message authentication codes (MACs)	SHA

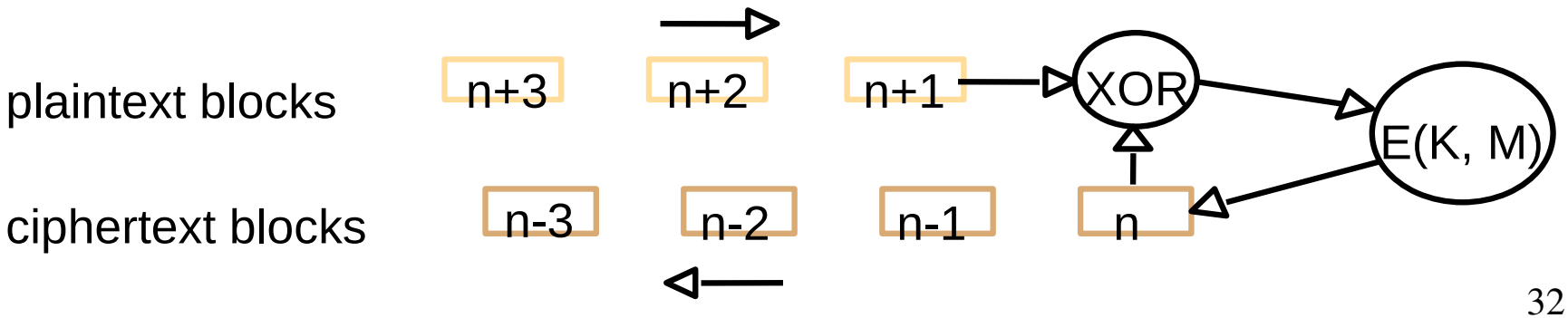
Algoritmos Híbridos: SSL

- camada SSL record: a transmissão dos dados



Encriptação de Blocos: ***Block Cipher***

- encriptar uma mensagem em blocos independentes
  - integridade da mensagem não é garantida sem um hash ou checksum
  - o atacante pode reconhecer padrões nos blocos de ciphertext e relacionar com o plaintext
- cipher block chaining (CBC)
  - cada bloco é combinado com o ciphertext precedente (xor) antes de ser encriptado.
  - Para decifrar, o bloco desencriptado é XOR-ed com o ciphertext do bloco anterior, resultando o formato inicial do bloco.

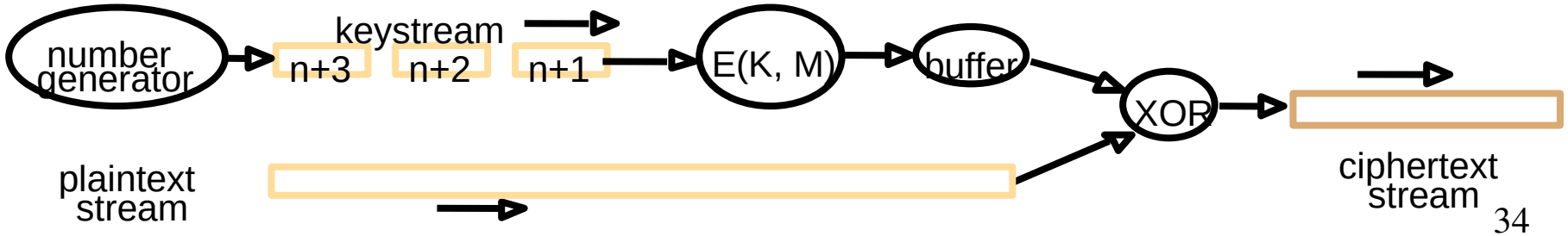


Encriptação de Blocos - CBC

- Desvantagem
  - pode ser usado apenas em canais fiáveis
    - Se um bloco se perder não será possível decifrar os restantes
- Vantagem
  - Dois blocos iguais de *plaintext* **não** terão o mesmo *ciphertext*
- Mesma mensagem, 2 destinos
  - A transmissão será a mesma (o que poderia dar pistas a um adversário)
  - Excepto se se adicionar um valor inicial antes dos dados, distinto para cada destino
    - *Initialization Vector*

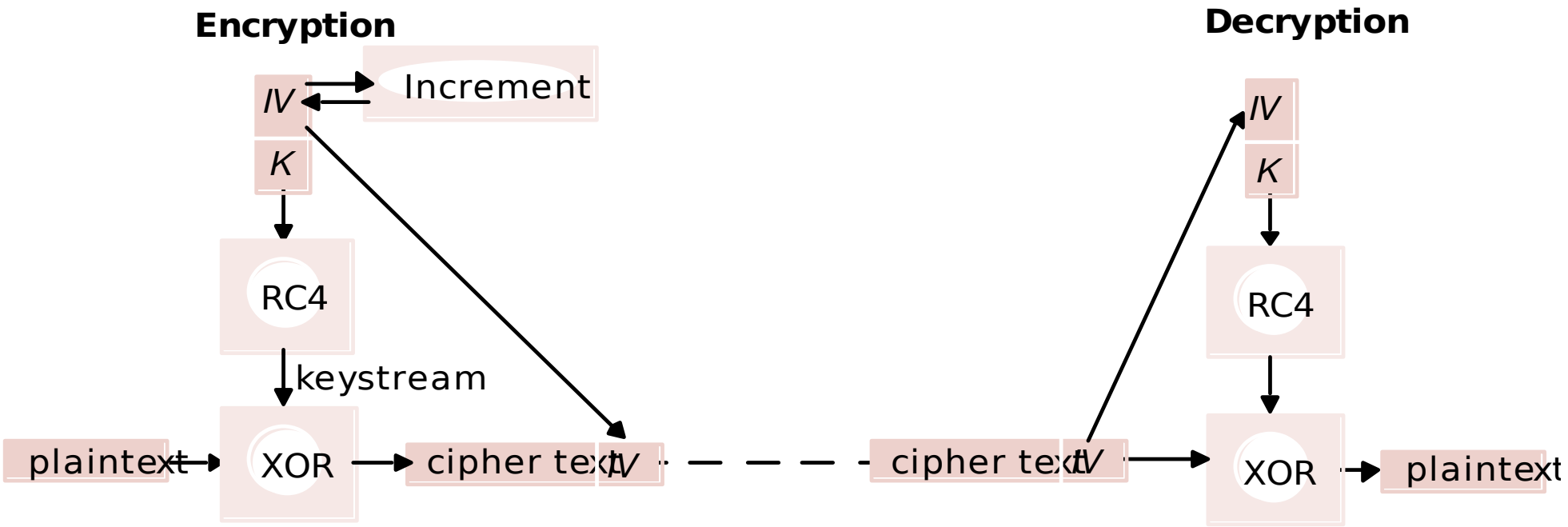
Encriptação de uma Stream: *Stream Cipher*

- usado para transmissões em tempo real, quando não se pode esperar para completar um bloco
- é gerada uma **sequência** de nºs, para gerar blocos que são encriptados com chave secreta e que depois são **XOR**-*ed* com o ***plaintext*** disponível
  - os blocos de *keystream* depois de encriptados podem ainda ser encadeados como CBC
- receptor: conhece a sequência e sabe como desencriptar a keystream. Usa o XOR para recuperar o plaintext
  - NOTA: ((a XOR b) XOR b) == a
- nº inicial combinado entre sender e receiver
- Suporta variação no volume de dados ao longo do tempo e faz um tratamento rápido dos dados (XOR ; a keystream pode ser preparada antes)



RC4 stream cipher in IEEE 802.11 WEP

- IEEE 802.11 – rede sem fios, WiFi
  - Dados em trânsito vulneráveis a qualquer dispositivo no alcance da transmissão
- WEP: wired equivalent privacy



IV: initial valu
K: shared key

802.11 WEP

- WEP: wired equivalent privacy *(versão base)*
  - Problemas
    - Partilha da chave é um ponto de risco
      - Solução: usar criptografia de chave pública para negociar e transmitir chaves individuais, como acontece em TLS/SSL
    - O ponto de acesso não era autenticado
      - Atacante com K podia fazer spoof & masquerading, controlando os dados em tráfego...
        - Solução: autenticação do pronto de acesso com C. de chave pública
    - Problemas com o stream cipher keystream reset
      - Se há perda de pacotes, o reset/sincronização pode dar pistas ao atacante
    - Chaves de 40 e de 64 bits vulneráveis a ataques de força bruta
      - Solução: chaves de 128 bits
    - RC4 stream cipher tem características que comprometem o secretismo da chave (mesmo que de 128 bits)
      - Solução: permitir a negociação das especificações da cifra, como em TLS

Hoje há vários protocolos seguros para redes wireless, como o usado na UE

Funções Seguras de Hash ou Digest

- uma função de digest h=H(M) é segura se:
  - dado M é fácil calcular h
  - dado h é inviável/difícil calcular M
  - dado M, é muito difícil encontrar M'!=M tal que H(M)=H(M')
- tais funções têm a propriedade *one-way*

- Como o *hash* tem um tamanho fixo, é possível encontrar mensagens naquelas condições... a probabilidade disso acontecer deve ser baixa)
- Se o *signer* conhecer duas mensagens M e M' com o mesmo digest poderá posteriormente alegar que enviou M' e não M, tendo ocorrido um erro de transmissão de M'

Funções Seguras de Hash ou Digest

- MD5 (1992)
  - em quatro ciclos. Cada aplica uma função não linear a um dos 16 segmentos de 32 bits um bloco de 512 bits da mensagem original
  - digest de 128 bits
  - um dos algoritmos mais eficientes em uso atualmente
- SHA (1995)
  - baseado no algoritmo MD5, introduzindo operações adicionais
  - digest de 160 bits
  - relativamente mais lento que o MD5 mas o tamanho do digest oferece mais garantias contra ataques de força bruta
- é possível usar um algoritmo simétrico de encriptação para gerar digests, mas nesse caso a chave usada será necessária para a validação
  - *ver CBC*

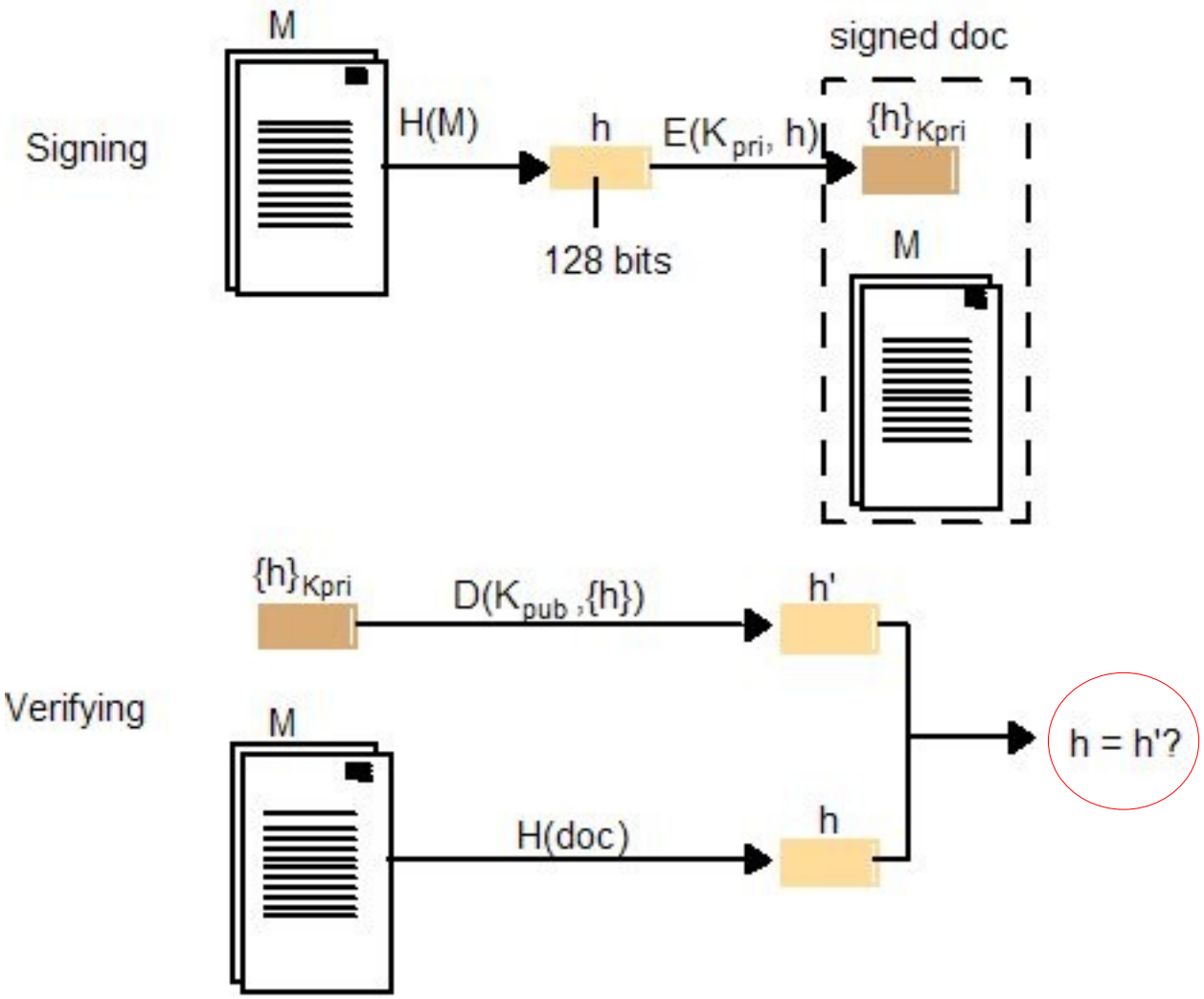
Assinaturas Digitais

- Dificuldades
  - documentos digitais
    - fáceis de copiar e modificar
  - o emissor pode deliberadamente divulgar a chave privada e alegar que não é o autor da mensagem (repúdio)

- Garantias desejáveis:
  - autenticidade de um documento (integridade)
  - impossibilidade de forjar uma assinatura (autenticação)
  - não repúdio

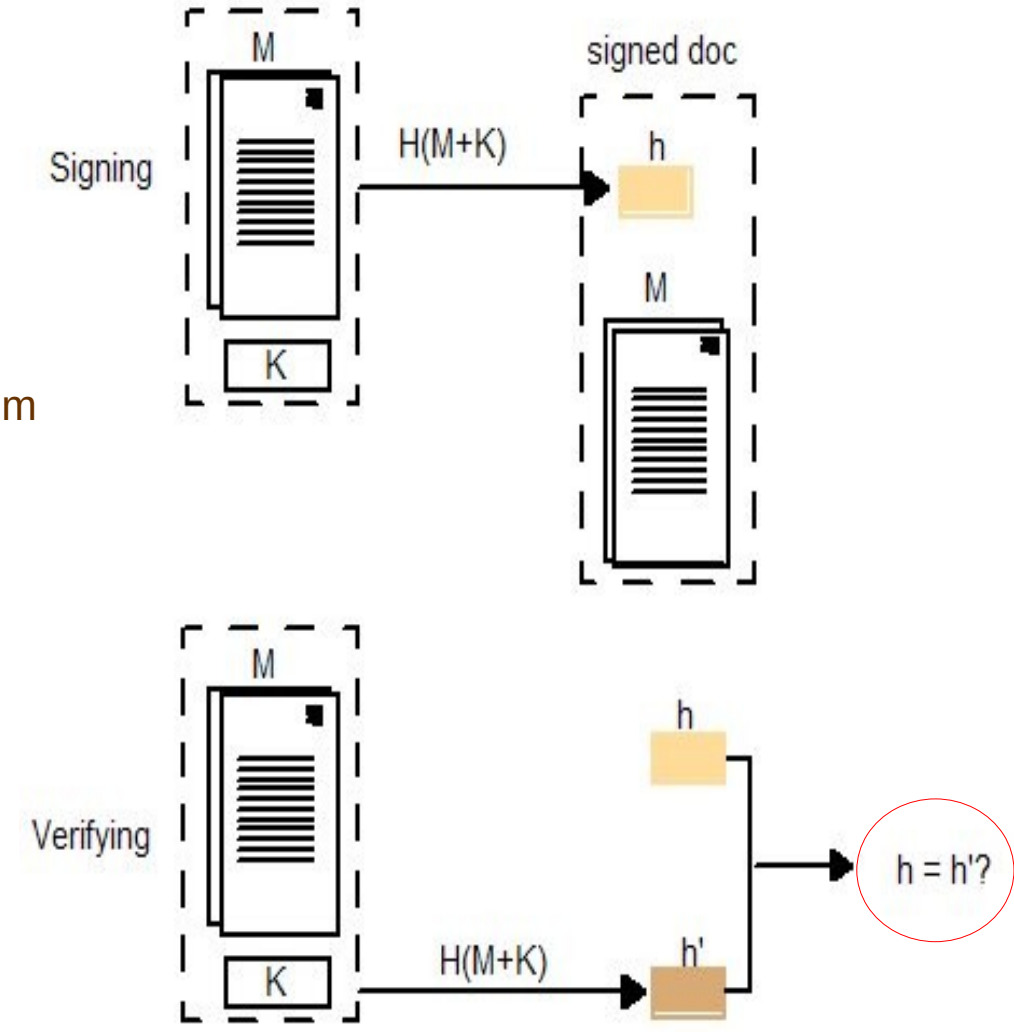
Assinatura Digital com Chave Pública

- Vantagens
  - simplicidade
  - dispensa comunicação prévia entre os intervenientes
- A encriptação é feita com a chave privada
  - o objetivo não é a confidencialidade da mensagem



Assinatura com Chave Secreta - **MAC**

- Algoritmo simétrico de encriptação
- Dificuldades
  - requer processo seguro para transmitir a chave secreta até ao *verifier*
  - pode ser necessário verificar a assinatura numa fase posterior à sua criação e por *verifiers* que o *signer* não conhece e a quem não dá a chave
  - a partilha da chave secreta traz fraqueza: um detentor da chave pode forjar a assinatura do signer original
- Vantagem: performance (<u>não há encriptação</u>)
  - funções de hash são 3 a 10 x mais rápido que alg. simétricos



Assinaturas digitais de chave pública

- Exemplo: as assinaturas que fazemos com:
  - Cartão do Cidadão
  - Chave Móvel Digital (um serviço inovador de desmaterialização)

- Na prática: assinaturas da mesma pessoa, em cada opção acima, usarão pares de chaves diferentes, mas o relevante é a validade das mesmas
  - No CC usam a chave privada inerente ao CC
  - Na CMD, usam outra chave privada associada ao cidadão, mas na posse do estado, alojada centralmente no serviço, e usada mediante autenticação

https://www.autenticacao.gov.pt/cc-assinatura
https://www.autenticacao.gov.pt/cmd-assinatura