

# Minicrypt *world*

Tecnologia Criptográfica (MIEI)

José Carlos Bacelar Almeida  
(jba@di.uminho.pt)



## MINICRYPT *world*

- › Cifras Simétricas:
  - › Cifras Sequenciais
  - › Cifras por Blocos
- › Funções de Sentido Único
  - › Funções de Hash Criptográficas
  - › Message Authentication Codes (MACs)
  - › (Cryptographically Secure) Pseudo-Number Generators (PRNGs)
- › ...

# Cifra Sequenciais

## Cifra One-Time-Pad

- › Cifra OneTimePad (Vernam) oferece garantias de confidencialidade!



MAS:

$$K_i = T_i \oplus C_i$$

$$C_1^i = T_1^i \oplus K_i \text{ e } C_2^i = T_2^i \oplus K_i \text{ determina que } T_1^i \oplus T_2^i = C_1^i \oplus C_2^i$$

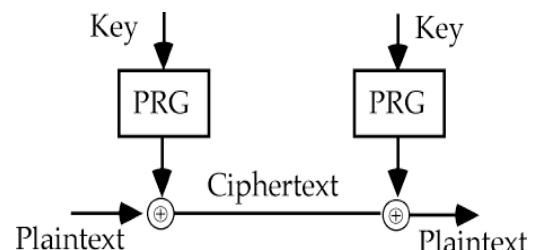
# Problemas da cifra OTP

- › Sequência de chave deve ser verdadeiramente aleatória e de comprimento igual à mensagem.
- › Chaves nunca devem ser reutilizadas. Um ataque com texto limpo conhecido é trivial. Mesmo dispondendo só dos criptogramas podemos retirar muita informação útil sobre o texto limpo.
- › Não promove a difusão da informação no criptograma. Se dispusermos de informação sobre a estrutura de uma mensagem podemos “trocar” os bits pretendidos.



## Cifras sequenciais

- › A ideia base consiste em “aproximar” a cifra OneTimePad por intermédio de um gerador de chaves (que produz uma sequência de chave a partir de uma chave de comprimento fixo).
- › Processam o texto limpo “símbolo a símbolo” (bit a bit, carácter a carácter, dígito a dígito...).
- › Tendem a ser muito eficientes e facilmente implementáveis em hardware.
- › O processo de geração da sequência de chave tem de ser reproduzível (determinístico) – pode por isso ser visto como uma máquina de estados finitos.
- › Logo, a sequência tem necessariamente de ser cíclica. Diz-se que o **período** é o comprimento da sequência antes de se começar a repetir.



# Critérios para o desenho de Cifras Sequenciais

- › Período deve ser tão grande quanto possível (sempre maior do que a mensagem a transmitir).
- › Sequência de chave deve ser:
  - **pseudo-aleatória:** propriedades estatísticas análogas a uma sequência “verdadeiramente” aleatória.
  - **imprevisível:** dado um segmento inicial não deve ser possível prever a sua continuação.
- › Outras características:
  - Sincronismo (síncronas ou auto-sincronizáveis).
  - Propagação de erros.
  - ...



## Cifras Síncronas

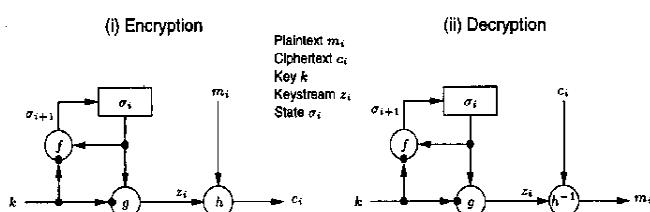


Figure 6.1: General model of a synchronous stream cipher.

- › A sequência de chave é independente do texto limpo/criptograma.
- › A perca/inserção de bits no criptograma determinam a “perca de sincronismo”. Ao decifrar, toda a mensagem a partir desse ponto é corrompida.
- › Erros (alterações de bits) só alteram a posição correspondente da mensagem original.
- › A chave (parâmetro de segurança) pode afectar:
  - › A função  $f$  que determina o próximo estado – Output Feedback Mode.
  - › A função  $g$  de saída – Counter Mode.
  - › Ambas...

# Cifras Auto-Sincronizáveis

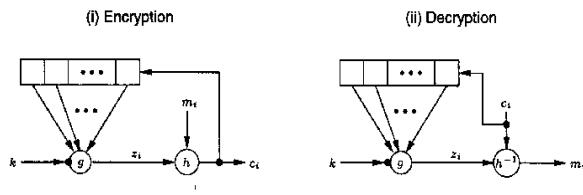


Figure 6.3: General model of a self-synchronizing stream cipher.

- › Cada bit da chave é calculado a partir dos últimos  $n$  bits do criptograma (e da chave, naturalmente).
- › Introduz-se um prefixo de  $n$  bits aleatórios no texto limpo para permitir sincronização da recepção.
- › Ao fim de  $n$  bits a decifragem sincroniza (após erro de transmissão; omissão/inserção de bits no criptograma).
- › Problema: vulnerável a ataques por repetição (o intruso pode reenviar uma porção do criptograma).

9



## Sequências Pseudo-Aleatórias

### Critérios de *Golomb*

1. A diferença no número de 1s e de 0s deve ser tão pequeno quanto possível.
2. Quando particionamos a sequência em sub-sequências de símbolos repetidos (*runs*), devemos encontrar um número de *runs* de comprimento  $l$  dado por  $r(l)=2^{-l}*r$  (se  $2^{-l}*r>l$ ), onde  $r$  é o número de *runs*.
3. A auto-correlação deve ser um valor constante para qualquer desvio diferente de 0 (mod p).

$$C_p(\tau) = \frac{1}{p} \sum_{i=1}^p x_i x_{i+\tau}$$

### Segurança Criptográfica

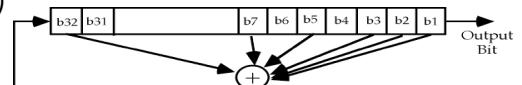
*Indistinguível de uma sequência aleatória para qualquer adversário polinomial.*

10



# Desenho baseado em LFSRs

- › Os Linear Feedback Shift Registers (LFSR) constituem uma das principais abordagens na geração de sequências pseudo-aleatórias (em aplicações não criptográficas).
- › Um LFSR é um dispositivo “síncrono” (controlado por um “relógio”) onde os bits são deslocados (produzindo um bit de saída) e o bit de entrada é determinado por uma função linear (e.g.  $f(x)=x^{32}+x^7+x^5+x^3+x^2+x+1$ )
- › LFSRs são ainda especialmente apropriados para implementações em hardware.
- › Um LFSR dispõe de período máximo ( $2^n-1$ ) se e só se o polinómio que lhe está associado for primitivo. Além disso, a sequência resultante satisfaz os critérios de Golomb.
- › ...mas infelizmente não cumprem o requisito de imprevisibilidade (só são necessários  $2^n$  bits da sequência para recuperar os parâmetros do LFSR)
- › **Em aplicações criptográficas, torna-se então necessário “esconder” a forte estrutura matemática que lhes está subjacente.**

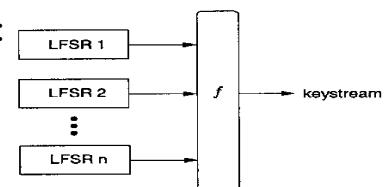


11



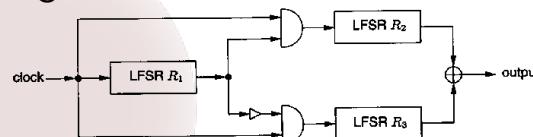
- › Tipicamente, para obscurecer estrutura linear subjacente à utilização de LFSRs, faz-se uso de uma combinação de LFSRs:

- utilização de função não linear de combinação:



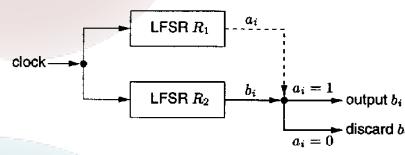
*Figure 6.8: A nonlinear combination generator.  $f$  is a nonlinear combining function.*

- Utilização de LFSRs para regular a cadência de outros LFSRs:



*Figure 6.12: The alternating step generator.*

- Utilização de LFSRs para filtrar a saída de outros LFSRs:



*Figure 6.13: The shrinking generator.*

12

- Multiplexers; Flip-Flops; etc.; etc...



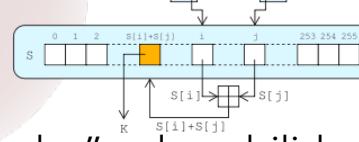
# Exemplos de Cifras baseadas em LFSRs

- › A5 (A5/2), utilizada no standard GSM
    - Utiliza três LFSRs (de 19, 22 e 23 bits).
    - Já quebrada!!! ( $2^{40}$ , 32Gb)
    - Vulgarmente reconhecida como um “bom desenho” mas *propositadamente fraco* em termos de segurança (registos pequenos e polinómios esparsos)
  - › E0, utilizado no protocolo bluetooth (mas já identificadas)
  - › CSS (Content Scramble System), utilizado na protecção de discos DVD — também quebrado...
- 
- › Podemos afirmar que **o desenho de cifras baseadas em LFSRs está, em grande medida, desacreditado.**



## RC4

- › o RC4 (ArcFour) é uma cifra sequencial síncrona vocacionada para ser executada em software.
  - Cifra desenvolvida por Ron Rivest (RSA Labs). Originalmente “trade secret”, mas algoritmo foi divulgado por um post anônimo na newsnet (descoberto por engenharia reversa).
  - Admite chaves de comprimento variável (até 2048 bit), mas não prevê utilização de Nonce
  - Opera em **Output Feedback Mode**:
    - Estado interno (256+2 bytes) inicial obtido a partir da chave;
    - iterada depois função que actualiza estado (e produz 1 bit para stream chave)



- › Já foram identificadas “demasiadas” vulnerabilidades! É **desaconselhado o seu uso.**

# Cifras baseadas em PRFs

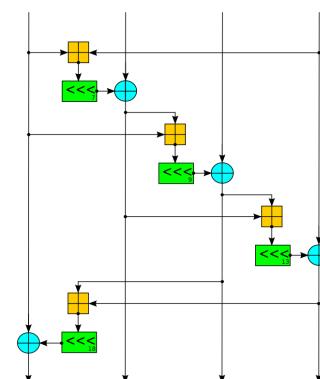
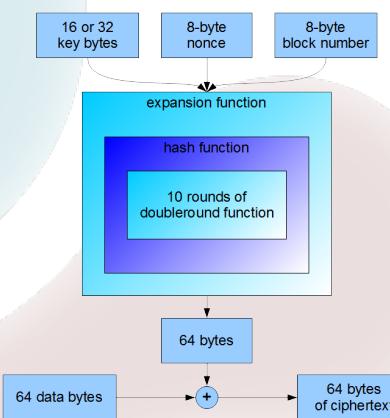
- › A generalidade das cifras recentes baseiam-se em Pseudo-Random Functions (PRFs)...
  - › PRFs são funções que dependem de uma chave e que são indistinguíveis de funções aleatórias
- › ...utilizadas de acordo com uma “construção criptográfica standard”, como “output-feedback mode” e “counter mode” (irão ser estudadas adiante)
- › Assim, admitem uma justificação mais rigorosa para o seu desenho (redução à segurança da primitiva PRF).

15



## E.g. Salsa20

- › Cifra proposta no âmbito da iniciativa eSTREAM (<http://www.ecrypt.eu.org/stream/index.html>) por Daniel Bernstein.
- › Opera sobre estado interno de 512 bit (4\*4\*32 bit). Estado inicial dependente de chave (256 bit), Nonce (64 bit) e contador (64 bit)



- › Opera em 10 rounds, em que cada um destes é composto por 4 execuções paralelas da quarter-round function.

16



# Outras Cifras Sequenciais

## › Chacha20

- evolução do algoritmo Salsa20 (mais seguro, mais eficiente)
- chave: 256bit, Nonce: 64bit, PositionCounter: 64bit
- por vezes (IETF RFC7539), utilizado com Nonce: 96bit; PositionCounter: 32bit (segurança reforçada; mensagem limitada a  $2^{32}$  blocos de 64 bytes = 256GB)

## › HC-256

- cifra vocacionada para “bulk encryption”
- muito pesada numa fase de pré-processamento, mas rápida na cifra.
- chave e IV de 256 bit

17



Universidade do Minho

# Cifra por Blocos

# Cifras por Blocos

› Processam blocos de comprimento fixo:

- Tamanhos típicos para os blocos: 64, 128, 256 bit.
- Mensagem é partida em blocos do comprimento requerido.
- Muitas vezes, torna-se necessário fazer como que a mensagem tenha um comprimento múltiplo do tamanho do bloco.
- ...que é conseguido convencionando-se que o último bloco é preenchido de acordo com regra pré-estabelecida (*padding*).

› Conceptualmente, corresponde a uma substituição a operar num alfabeto enorme (e.g. em blocos de 128 bit existirão  $2^{128}!$  possíveis substituições).



## Confusão e Difusão

› Uma cifra, a operar num bloco, espera-se que promova (Shannon 1949):

**Difusão** – *cada bit do texto limpo deve afectar o maior número de bits do criptograma.* Desta forma escondemos propriedades estatísticas da mensagem.

**Confusão** – *cada bit do criptograma deve ser uma função complexa dos bits do texto limpo.* Desta forma torna-se “complicada” a relação entre propriedades estatísticas do criptograma face às propriedades do texto limpo.

# Cifras Blocos vs. Sequenciais

- Unidade de processamento distinta...
- Cifras por blocos mais complexas...
- As cifras por blocos são consideravelmente mais lentas (e.g. DES é cerca de 10x mais lento do que RC4).
- Cifras sequenciais não promovem a difusão da influência dos bits do texto limpo.
- Cifras por blocos *protegem* a chave, no sentido em que uma mesma chave pode ser utilizada para cifrar múltiplas mensagens. Nas cifras sequenciais, o mesmo efeito só é conseguido com recurso a um mecanismo de certa maneira externo (o *Nonce*). – Obs: só por isso é que nas cifras por blocos faz sentido considerarmos ataques de “texto limpo conhecido” e “texto limpo escolhido”.

21



## Padding

- › Estratégia para completar o último bloco de texto-limpo...
- › ...sem perder informação sobre comprimento efectivo da mensagem.
- › Várias métodos:
  - bit 1 seguido de 0s;
  - $n$  bytes  $n$ ;
  - ...
- › Por vezes (cifras assimétricas) é explorado para introduzir aleatoriedade na mensagem.

22



# Modos de Operação

- › Dependendo da aplicação, podemos cifrar uma mensagem com uma cifra por blocos de diversos **modos**:
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher FeedBack (CFB)
  - Output FeedBack (OFB)
  - Counter Mode (CTR)

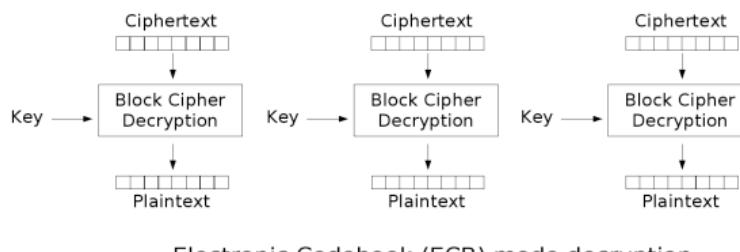
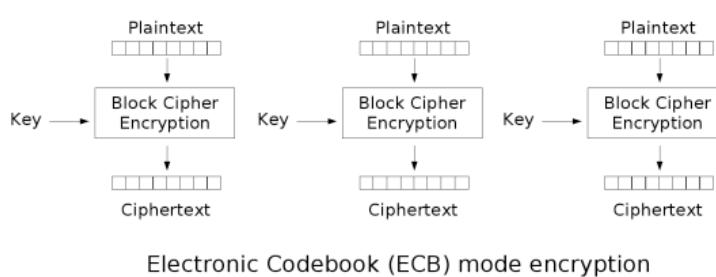
...

- › Certos modos garantem simultaneamente integridade (Authenticated Encryption)

23



## Electronic Code Book (ECB)

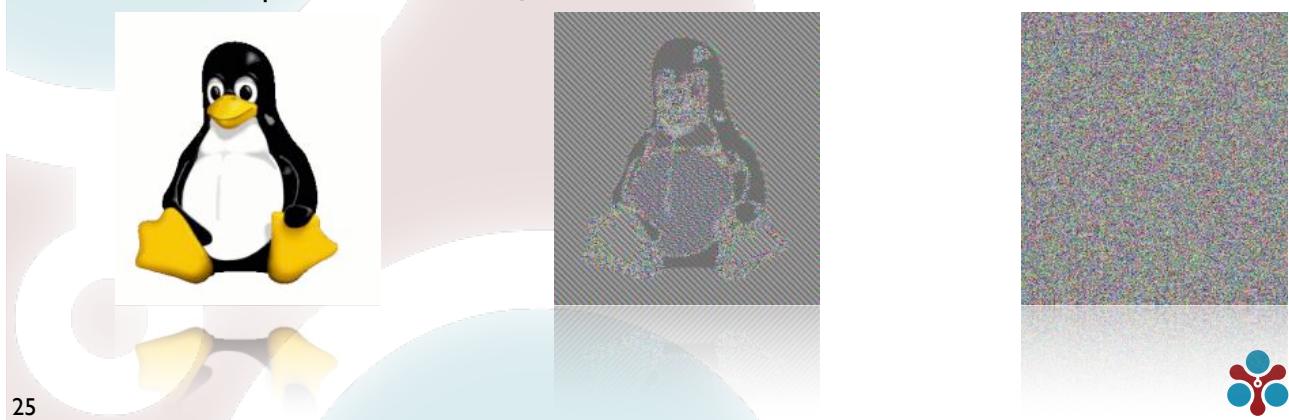


24

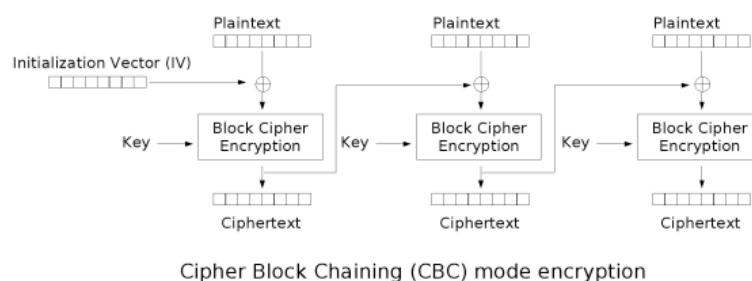


# ECB (cont.)

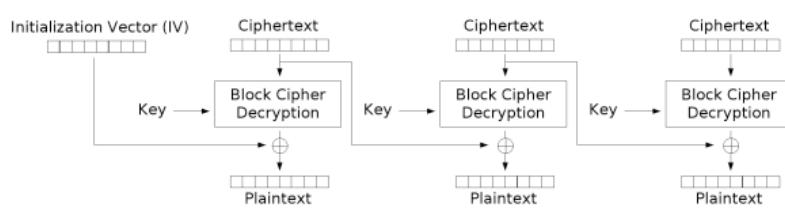
- Eventual repetição de blocos é detectável - *code book attack*.
- Vulnerável a ataques por repetição/substituição.
- Só deve ser utilizado para cifrar mensagem de um só bloco (ou poucos...).
- Um erro de um bit num bloco do criptograma afecta um só bloco após a decifragem (mas todo o bloco).



## Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

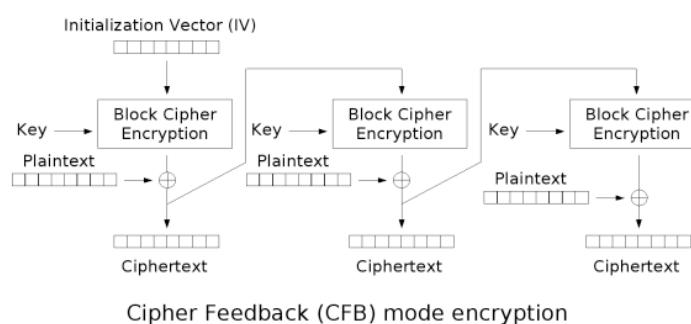
# CBC (cont.)

- › É utilizado um “vector de inicialização” (IV) previamente conhecido para iniciar o processo (enviado em claro ou, preferencialmente, enviado cifrado em ECB). Obs.: se o IV for enviado em claro, o intruso pode alterar bits do primeiro bloco alterando os respectivos bits do IV. Obs2.: Se IV for fixo, os criptogramas de duas mensagens com um prefixo comum vão preservar essa propriedade.
- › Um erro num bloco do criptograma corrompe dois blocos após a decifragem (mais precisamente, um bloco e um bit).
- › Encadeamento do processo faz depender a operação de cifra de um bloco de todos os que o antecedem.
  - Pode assim ser utilizado como MAC (utilizando somente o último bloco do criptograma). O problema é que pode ser um código muito pequeno... (e.g. 64 bit no DES).

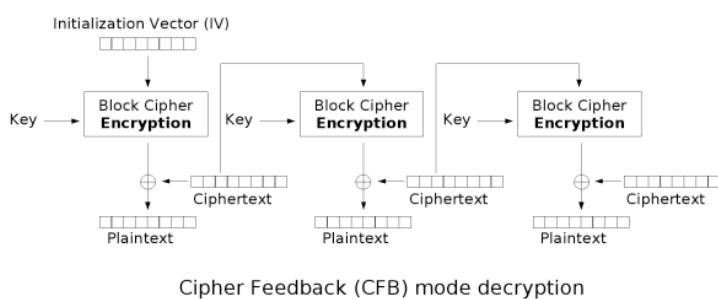
27



# Cipher FeedBack (CFB)



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

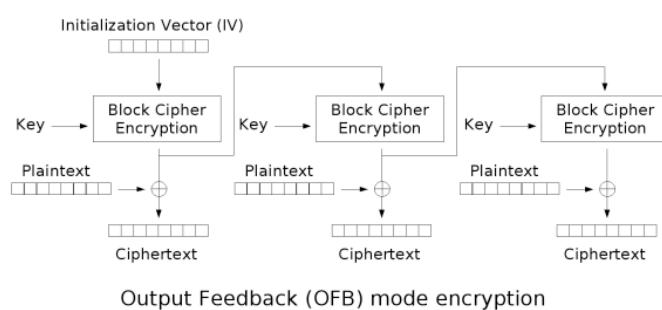
# CFB (cont.)

- Modo que implementa uma cifra sequencial auto-sincronizável com uma cifra por blocos.
- Note-se que se utiliza sempre a operação de "cifrar".
- Número de bits no FeedBack é variável ( $CFBn$  - standard prevê  $n=1, 8$  ou  $64$ ). A realimentação transfere os  $n$  bits mais significativos para os menos significativos (com *shift* dos restantes).
- IV deve ser único por cada utilização (c.f. reutilização de chaves em cifras sequenciais). E.g. pode ser enviado em claro.
- Um erro num bit do criptograma afecta o bit respectivo no bloco e todos do bloco seguinte.
- Sequência de chave depende do IV, chave da cifra e de todo o texto limpo já cifrado.

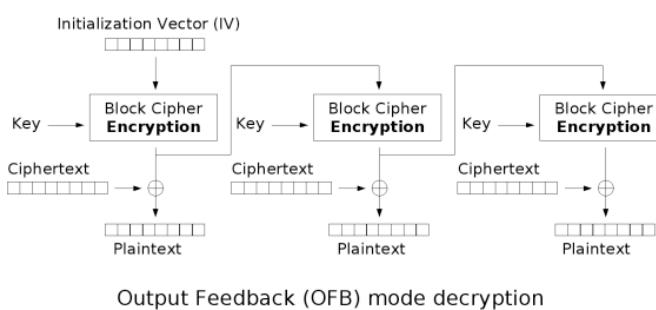
29



## Output FeedBack (OFB)



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

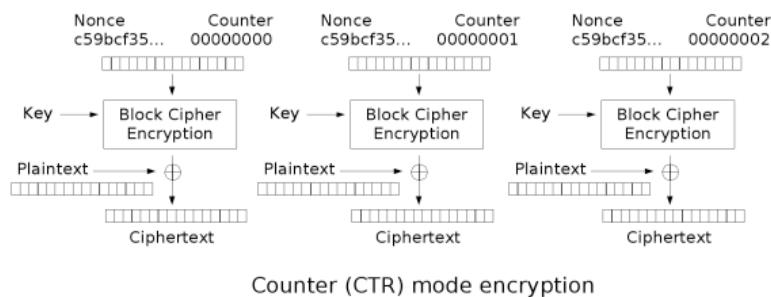
# OFB (cont.)

- Modo que implementa uma cifra sequencial síncrona com uma cifra por blocos.
- Sequência de chave é obtida iterando a cifra sobre um bloco inicial (IV).
- Sequência de chave é independente da mensagem (pode assim ser processada independentemente de se ter já disponível a mensagem).
- Erros de bits no criptograma só afectam os respectivos bits na mensagem original.

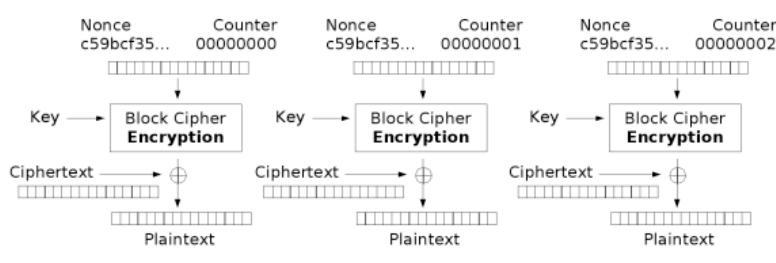
31



# Counter Mode (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

32



# CTR (cont.)

- › Tal como OFB, simula uma cifra sequencial síncrona (mas agora em counter mode).
- › Nonce (IV) e Counter podem ser conjugados de diferentes formas (concatenados, xored, ...).
- › Único requisito para o Counter é produzir valores distintos para todos os blocos (o mais simples é ser mesmo implementado como um contador).
- › Não impõe dependência entre processamento dos vários blocos (podem ser processados em paralelo; acesso aleatório; ...)
- › Análise teórica da respectiva segurança mais simples...
  - Segurança análoga ao modo CBC.

33



# Authenticated Encryption

- › Tendência recente de chamada a propostas para modos de operação que ofereçam garantias de integridade
- › Normalmente suportam “Associated Data”, não cifrada, mas que é acoplada para efeitos de garantia de integridade (e.g. metadados)
- › Alguns exemplos:
  - EAX (Encrypt-then-Mac-then-Translate)
  - GCM (Galois/Counter Mode)
  - OCB (Offset CodeBook)

34



# Construção de uma cifra por blocos...

- › Procurar implementar uma cifra por blocos directamente como uma substituição arbitrária não é nem exequível nem prático (pense-se no espaço de chaves...). Essa substituição “virtual” terá então de ser construída a partir de blocos mais simples. E.g.:
  - Substituições (de tamanho razoável)
  - Transposições

35



## Substituição

- › Uma substituição promove uma permutação entre os símbolos de um alfabeto.
- › Quando vista sobre palavras em binário, pode ser entendida como uma permutação (troca de fios) entre um par descodificador/codificador. Obs.: note a correspondência exponencial entre o número de bits de entrada e o número de ligações envolvidas.

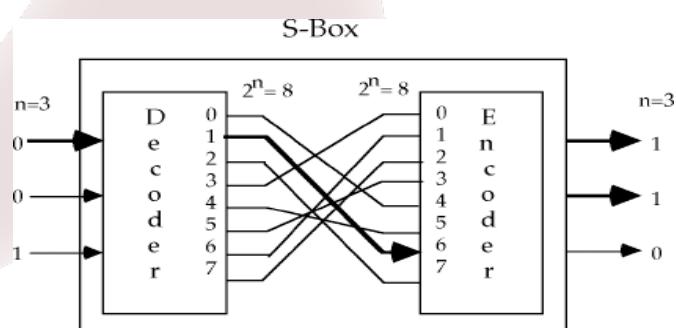


Fig 2.1 Substitution Operation

36



# Permutação

- › As transposições são simplesmente uma “troca de fios” entre os bits de entrada e os de saída.
- › A sua implementação é muito simples (em *hardware*... já em *software* é uma operação particularmente frustrante) e é comportável operar sobre todo o bloco.

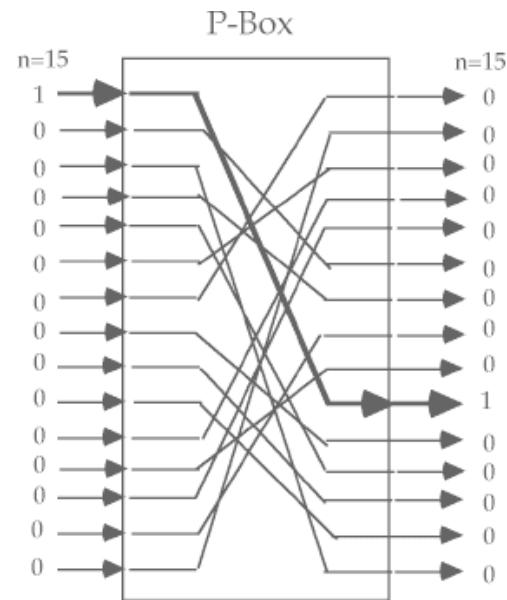


Fig 2.2 - Permutation or Transposition Function



## S-P networks

- Se utilizássemos unicamente uma das construções (substituições de tamanho comportável ou permutações) não teríamos sucesso na construção de uma cifra segura (ambas as técnicas constituem cífras *idempotentes*).
- ...mas, combinando ambas as técnicas, podemos construir uma “cifra produto” não idempotente e que designamos por *round*.
- Um *round*, por si só, não é suficientemente seguro. Mas quando iterado permite obter os níveis de segurança pretendidos.
- É esta a essência das designadas “S-P networks”.

...mais um conceito introduzido no artigo basilar de C. Shannon (1949).

# S-P networks (cont.)

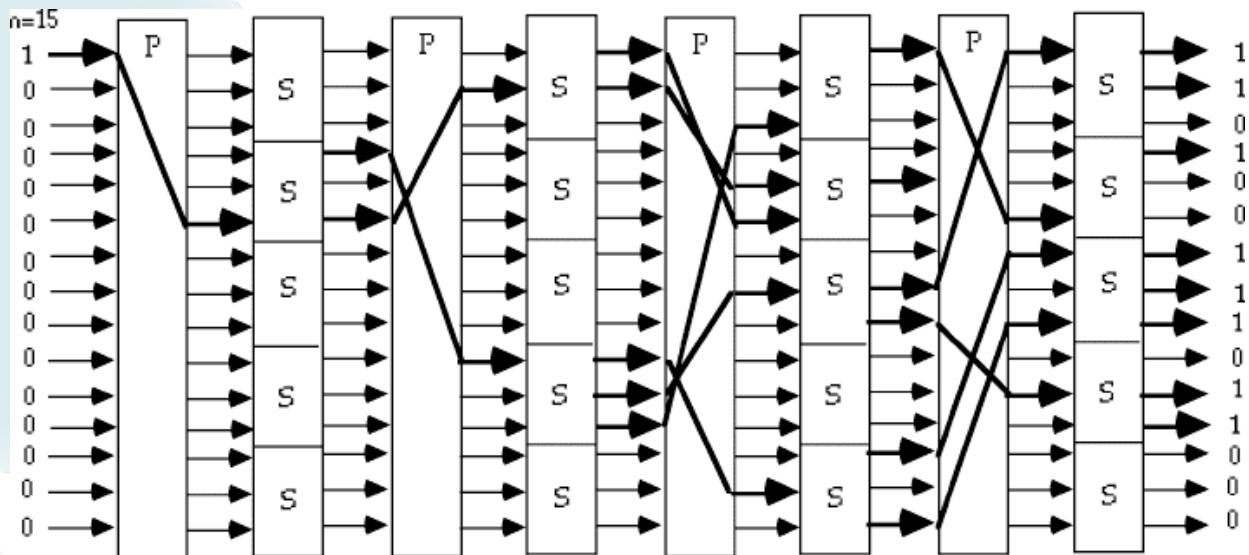


Fig 2.3 - Substitution-Permutation Network, with the Avalanche Characteristic

39



## Circuitos de Feistel

- A utilização de uma S-P *network* numa cifra não é prática porque requer a construção de uma rede inversa para decifrar o criptograma (i.e. obriga à duplicação dos dispositivos/programas)
- Para evitar esse problema devemos encontrar formas expeditas de desenhar um *round* por forma ao mesmo procedimento seja usado para calcular a sua "inversa"
- Um **circuito de Feistel** é um exemplo de uma tal forma expedita de desenho dos *rounds*: cada bloco é partido em dois sub-blocos - num é aplicada a transformação e o outro é preservado. Os sub-blocos são ainda trocados para que o *round* seguinte afecte agora o sub-bloco que ficou inalterado.

40



# Circuitos de Feistel (cont.)

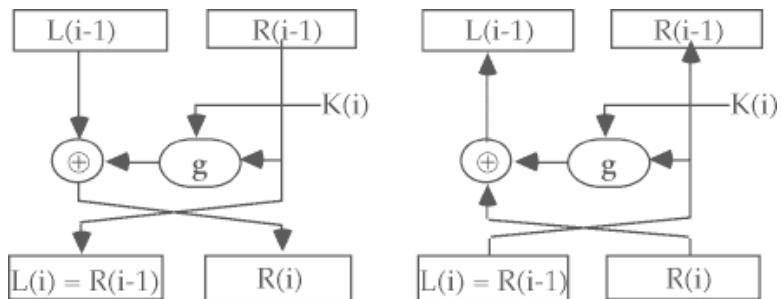


Fig 2.4 - A Round of a Feistel Cipher

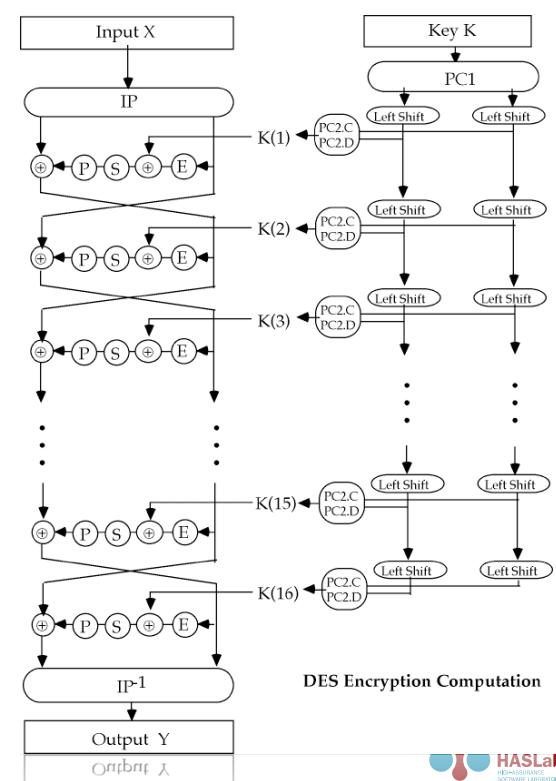
$$L(i) = R(i-1); \quad R(i) = L(i-1) \oplus g(K(i), R(i-1))$$

- Um *round* pode ser considerado a sua própria operação inversa se realizarmos algumas “trocas” nos sub-blocos...

## e.g. DES

- O algoritmo (Lucifer, IBM) escolhido como Standard DES (1975).

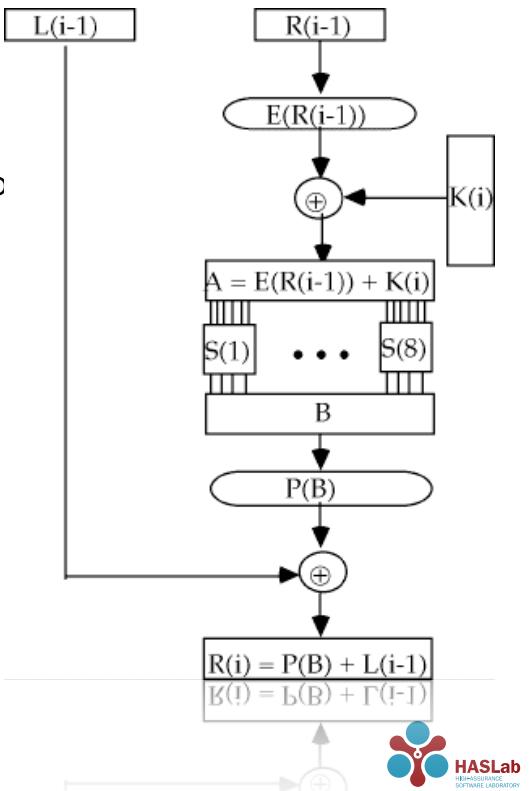
- Ao bloco (64 bit) é aplicada uma permutação inicial e partido em dois sub-blocos (32 bit).
- Seguem-se 16 rounds segundo a estrutura típica determinada por um circuito de Feistel.
- A chave (56 bit) é processada para produzir 16 chaves de round (48 bit).
- O algoritmo conclui-se com a inversão da permutação inicial.



# Round do DES

- › Ao sub-bloco a transformar (32 bit) é aplicada uma **permutação de expansão** para 48 bit (alguns bits são duplicados).
- › 8 **S-boxes** distintas realizam a operação não linear do *round*.
- › Cada **S-box** processa 6 bit de entrada, produzindo 4 de saída. O bloco (*B*) volta a dispor de 32 bit.
- › O *round* finaliza com uma permutação.

Obs.: A expansão no *round* DES permite acelerar o *efeito de avalanche*.



## Propriedades e características do DES

- Complementaridade:  $\overline{E_K(X)} = E_{\bar{K}}(\bar{X})$
- DES não é um grupo: dadas duas chaves  $K_1$  e  $K_2$ , não é em geral possível determinar  $K_3$  tal que  $E_{K_3}(X) = E_{K_2}(E_{K_1}(X))$
- O DES foi desenhado para permitir implementações eficientes em *hardware*. Em contrapartida, exibe algumas operações particularmente ingratas para implementações em *software* (em particular, permutações e manipulações várias ao nível do bit).

# Chaves fracas

- › Dado o mecanismo de escalonamento de chaves do DES (produção das chaves de *round* a partir da chave inicial), existem algumas chaves com problemas de segurança associados:
  - dão origem a chaves  $K_1..K_{16}$  todas iguais - **chaves fracas**.
  - dão origem a apenas duas sub-chaves diferentes - **chaves semi-fracas**.
  - dão origem a apenas 4 sub-chaves diferentes - **chaves possivelmente fracas**.
- › Em chaves fracas (destes três tipos), temos que  $K_i = K_{16-i}$ . Assim a operação de cifra é uma involução (cifrar duas vezes permite recuperar o texto original). Estas chaves dão origem a pontos fixos ( $f(x)=x$ ) e a anti pontos fixos ( $f(x)=\text{neg}(x)$ ).

Obs.: Todos estes conjuntos de chaves estão tabelados, pelo que basta a implementação assegurar que não é escolhida nenhuma destas chaves para não incorrer nos problemas de segurança associados.

45



# Cifra múltipla

- › Dado que DES não é um grupo, é razoável aumentar a segurança iterando a operação de cifra.
- › Cifra dupla ( $E_{K2}(E_{K1}(X))$ ) oferece segurança análoga à cifra simples - ataque *meet-in-the-middle*.
  - Sabendo que  $C = E_{K2}(E_{K1}(P))$ ,
    - Construímos tabela com  $E_K(P)$ , para toda a chave K
    - Basta-nos agora encontrar  $K_2$ , verificando se  $D_{K2}(C)$  se encontra na tabela construída.
    - Par determinado é verificado com outro par...
    - ...mas obriga a uma utilização enorme de memória...
- › ...por isso, é normal utilizar-se cifra tripla.

46



# Triple DES

- › Encadeiam-se operações de cifra com decifragem:
$$E_{(K_1, K_2, K_3)}(X) = E_{K_3}(D_{K_2}(E_{K_1}(X)))$$
- › Chave dispõe agora de 168 bit.
- › Quando  $K_1 = K_2 = K_3 = K$ , é simplesmente uma cifra DES simples.
- › Nível de segurança é análogo a duas chaves (modificação de ataque *meet-in-the-middle*).
- › Também disponível numa versão com apenas duas chaves:

$$E_{(K_1, K_2)}(X) = E_{K_1}(D_{K_2}(E_{K_1}(X)))$$

47



## Outras cifras por blocos

- › Baseadas em circuitos de Feistel:
  - LOKI (64 bit/bloco; 64 bit/chave)
  - Blowfish (64 bit/bloco; tam. chave variável)
  - ...
- › Outros desenhos:
  - IDEA (...)
  - RC5 (tam. bloco, chave e n° rounds variável - muito eficiente)
  - AES (...)
  - ...

48



# Advanced Encryption Standard

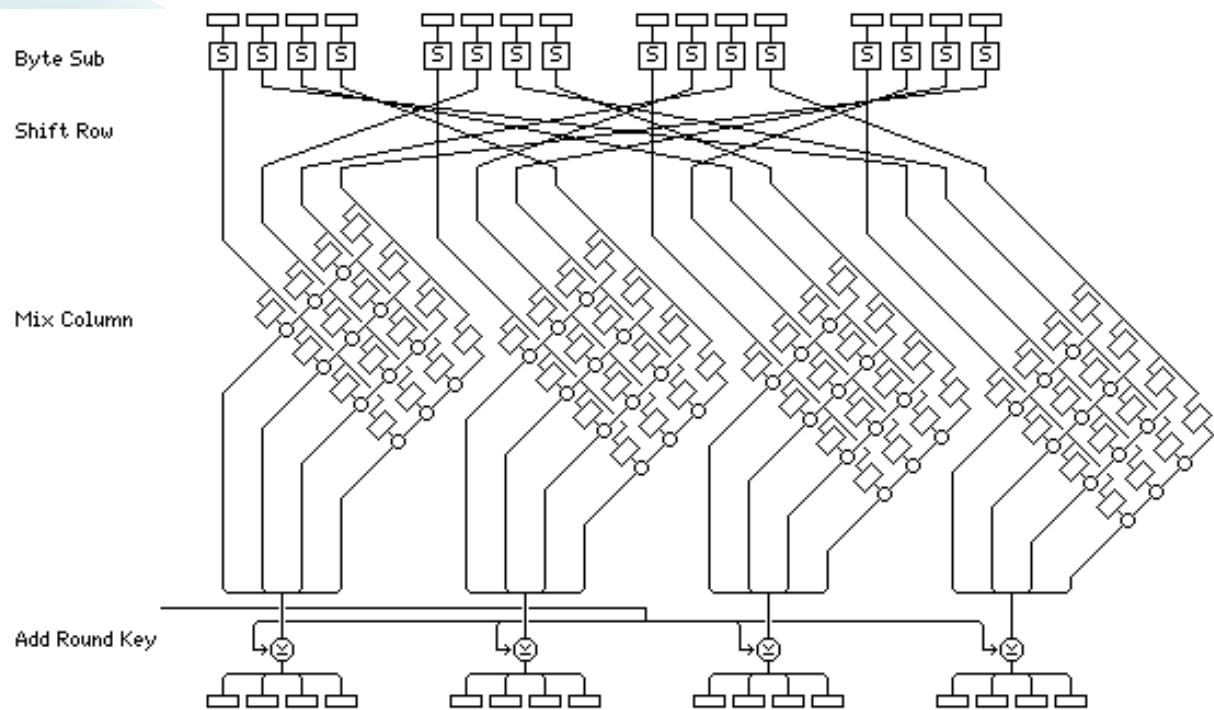
- › Chamada para algoritmos criptográficos pela NIST para elaboração de um standard para substituir o DES - anúncio:1997, submissões: 1998, decisão 2000(1).
- › Requisitos:
  - 128+ bit/bloco; 128+ bit chave
  - Mais seguro e mais rápido do que TripleDES
  - Especificação e detalhes de desenho completos.
  - Realizações em C e Java.
- › Finalistas:
  - **MARS** - complexo, rápido, alta margem de segurança
  - **RC6** - muito simples, muito rápido, pequena margem de segurança
  - **Rijndael** - limpo, rápido, boa margem de segurança (**VENCEDOR**)
  - **Serpent** - lento, limpo, muito alta margem de segurança
  - **Twofish** - complexo, muito rápido, alta margem de segurança



## AES (cont.)

- Desenhado por **Rijmen-Daemen** (Bélgica).
- *Rounds* compostos por 4 camadas:
  - *Byte Substitution*
  - *Shift Rows*
  - *Mix Columns*
  - *Key Addition*
- Todas as operações podem ser realizadas por *Xor* e *lookup* a tabelas (que permite realizações muito eficientes em SW).
- Desenho dispõe de uma forte fundamentação matemática baseada em *corpos finitos* ( $GF(2^8)$ ,  $GF(2^4)$ ,...).
- Particularmente adaptado para processadores modernos (e *smart-cards*).

# AES (cont.)



51



## Rijndael - descrição

- Tamanho de bloco variável (16, 24, 32 bytes) - i.e. 128, 192, 256 bit.
- Tamanho da chave variável (e independente do tamanho do bloco)
- Número de *rounds* dependente do tamanho da chave e do bloco...

a <sub>0,0</sub>	a <sub>0,1</sub>	a <sub>0,2</sub>	a <sub>0,3</sub>	a <sub>0,4</sub>	a <sub>0,5</sub>	a <sub>0,6</sub>	a <sub>0,7</sub>
a <sub>1,0</sub>	a <sub>1,1</sub>	a <sub>1,2</sub>	a <sub>1,3</sub>	a <sub>1,4</sub>	a <sub>1,5</sub>	a <sub>1,6</sub>	a <sub>1,7</sub>
a <sub>2,0</sub>	a <sub>2,1</sub>	a <sub>2,2</sub>	a <sub>2,3</sub>	a <sub>2,4</sub>	a <sub>2,5</sub>	a <sub>2,6</sub>	a <sub>2,7</sub>
a <sub>3,0</sub>	a <sub>3,1</sub>	a <sub>3,2</sub>	a <sub>3,3</sub>	a <sub>3,4</sub>	a <sub>3,5</sub>	a <sub>3,6</sub>	a <sub>3,7</sub>

k <sub>0,0</sub>	k <sub>0,1</sub>	k <sub>0,2</sub>	k <sub>0,3</sub>	k <sub>0,4</sub>	k <sub>0,5</sub>	k <sub>0,6</sub>	k <sub>0,7</sub>
k <sub>1,0</sub>	k <sub>1,1</sub>	k <sub>1,2</sub>	k <sub>1,3</sub>	k <sub>1,4</sub>	k <sub>1,5</sub>	k <sub>1,6</sub>	k <sub>1,7</sub>
k <sub>2,0</sub>	k <sub>2,1</sub>	k <sub>2,2</sub>	k <sub>2,3</sub>	k <sub>2,4</sub>	k <sub>2,5</sub>	k <sub>2,6</sub>	k <sub>2,7</sub>
k <sub>3,0</sub>	k <sub>3,1</sub>	k <sub>3,2</sub>	k <sub>3,3</sub>	k <sub>3,4</sub>	k <sub>3,5</sub>	k <sub>3,6</sub>	k <sub>3,7</sub>

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Number of rounds (Nr) as a function of the block and key length.

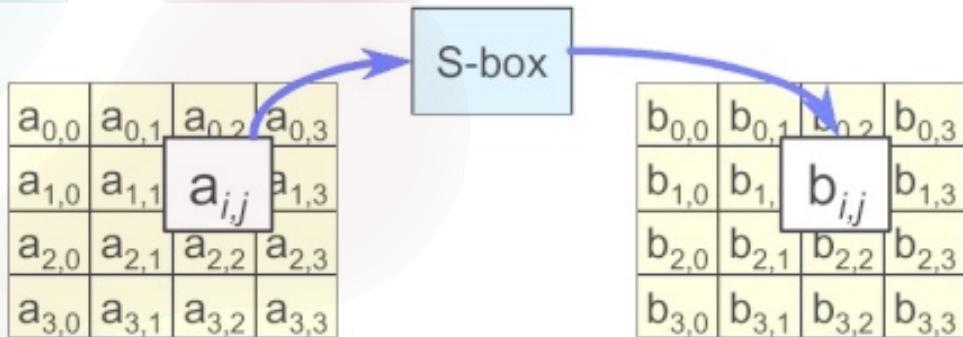
52



# AES - descrição (cont.)

## › ByteSub:

- Fortemente não linear
- Uma única S-box

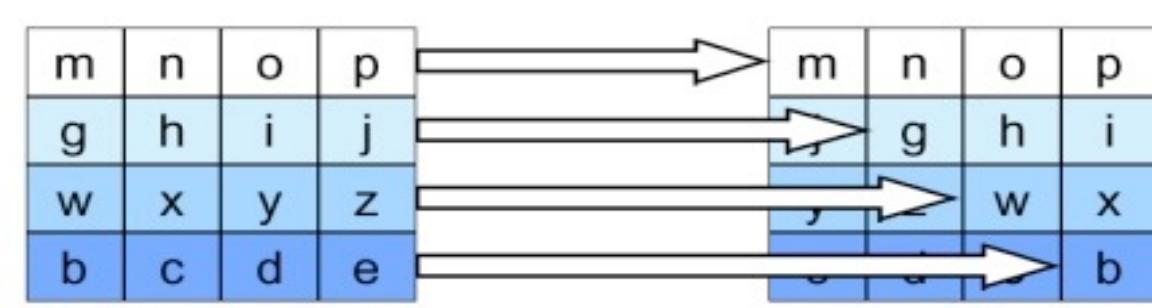


53



# AES - descrição (cont.)

## › ShiftRow:



Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Table 2: Shift offsets for different block lengths.

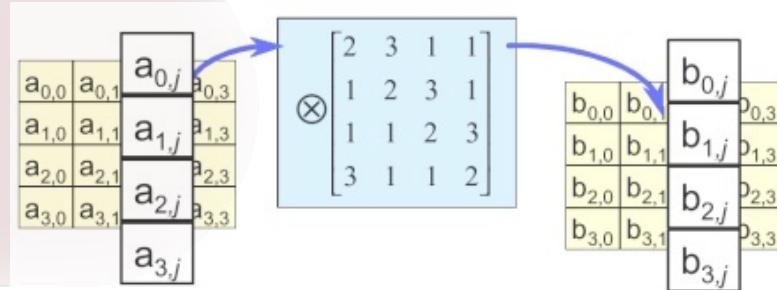
54



# AES - descrição (cont.)

## › Mix column:

- Alta difusão intra-coluna
- Interacção com *ShiftRow* (alta difusão em múltiplos *rounds*)
- $c(x) = '03'x^3 + '01'x^2 + '01'x + '02'$



55

# AES - descrição (cont.)

## › Key addition:

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} + \begin{array}{|c|c|c|c|} \hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

56

# AES - descrição (cont.)

## › Round transformation...

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} \text{ and } \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}.$$

For the ShiftRow and the ByteSub transformations, we have:

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-C1} \\ b_{2,j-C2} \\ b_{3,j-C3} \end{bmatrix} \text{ and } b_{i,j} = S[a_{i,j}].$$

In this expression the column indices must be taken modulo 4. By substitution, the above expressions can be combined into:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-C1}] \\ S[a_{2,j-C2}] \\ S[a_{3,j-C3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}.$$

# AES - descrição (cont.)

The matrix multiplication can be expressed as a linear combination of vectors:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = S[a_{0,j}] \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \oplus S[a_{1,j-C1}] \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \oplus S[a_{2,j-C2}] \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \oplus S[a_{3,j-C3}] \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

The multiplication factors  $S[a_{i,j}]$  of the four vectors are obtained by performing a table lookup on input bytes  $a_{ij}$  in the S-box table S[256].

We define tables  $T_0$  to  $T_3$ :

$$T_0[a] = \begin{bmatrix} S[a] \bullet 02 \\ S[a] \\ S[a] \\ S[a] \bullet 03 \end{bmatrix} \quad T_1[a] = \begin{bmatrix} S[a] \bullet 03 \\ S[a] \bullet 02 \\ S[a] \\ S[a] \end{bmatrix} \quad T_2[a] = \begin{bmatrix} S[a] \\ S[a] \bullet 03 \\ S[a] \bullet 02 \\ S[a] \end{bmatrix} \quad T_3[a] = \begin{bmatrix} S[a] \\ S[a] \\ S[a] \bullet 03 \\ S[a] \bullet 02 \end{bmatrix}.$$

These are 4 tables with 256 4-byte word entries and make up for 4KByte of total space. Using these tables, the round transformation can be expressed as:

$$e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j-C1}] \oplus T_2[a_{2,j-C2}] \oplus T_3[a_{3,j-C3}] \oplus k_j.$$

# Técnicas de Cripto-Análise

- › Força Bruta – na prática inviável... (e.g. AES dispõe de  $2^{128}$  hipóteses de chaves)
- › Ataques à estrutura do algoritmo:
  - Cripto-análise Diferencial - ataque de texto-limpo escolhido. Compara o comportamento da cifra para blocos com diferenças determinadas.
  - Cripto-análise Linear - ataque de texto-limpo conhecido. Aproxima as S-boxes por funções lineares.
  - Cripto-análise Algébrica - ataque de texto-limpo conhecido. Explora a estrutura algébrica da cifra.
- › Ataques à implementação:
  - Side-Channel attacks - timing, power consumption, electromagnetic leaks, etc.

59



## Cripto-análise Diferencial

- › Desenvolvida em 1990 por Eli Biham e Adi Shamir para atacar o DES.
- › Ataque de texto limpo escolhido.
- › Analisa as diferenças no processamento das S-boxes do último round por forma a determinar a chave utilizada nesse round.
- › Restantes bits da chave (8) podem ser obtidos por força bruta...

60



- › No desenho das S-boxes existiu a preocupação de garantir que é produzida uma saída aleatória face a uma entrada aleatória.
- › Mas quando comparamos a diferença dos resultados de uma S-box para pares de entradas com uma diferença determinada o resultado não é mais aleatório.
- › Exemplo: para S-1 - quando analisamos a frequência da diferença dos resultados nos pares de entrada com diferença 0x01 (64 possíveis pares) obtemos:

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Output
	01	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4

- › ...este comportamento está longe de ser aleatório: muitas das possibilidades de resultado estão excluídas (0,1,2,4,8) e podemos identificar uma maior probabilidade de ocorrer A como diferença dos resultados.
- › Dado que as S-boxes são fixas, podemos construir uma tabela que agregue a informação referida...

61



- › A cripto-análise diferencial baseia-se no conhecimento das diferenças na entrada/saída das S-boxes...
- › As diferenças nas saídas da S-box do último round são directamente calculadas dos criptogramas disponíveis. Já as diferenças na saída terão de ser calculadas com base em estimativas (probabilísticas)
- › Este é o motivo pelo qual a viabilidade da cripto-análise diferencial é fortemente influenciada pelo número de rounds: para 3 rounds é possível conduzir uma análise determinística. Para mais do que isso é necessário uma abordagem probabilística que aumenta muito o número de pares texto limpo/criptograma a analisar e complica substancialmente o ataque...

62



# Resumo de resultados

## Cripto-análise diferencial do DES

Nº Rounds	Pares escolhidos
3	2/3
6	120
8	$\approx 2^{14}$
10	$\approx 2^{24}$
13	$\approx 2^{39}$
16	$\approx 2^{47}$

63



## Cripto-análise Linear

- › Desenvolvida em 1993 por Mitsuru Matsui.
- › Utiliza aproximações lineares para descrever as acções da cifra.
- › Explora “desvios” (bias) das probabilidades com que essas aproximações se verificam ( $p \neq 1/2$ ).
  - › Exemplo: A acção da S-box 5 permite-nos derivar a aproximação (que se verifica com probabilidade  $1/2 - 5/16$ ): “ $K_{26} = X_{17} \oplus Y_3 \oplus Y_8 \oplus Y_{14} \oplus Y_{25}$ ”
- › As aproximações lineares de diferentes rounds podem ser associadas...
- › Análise completa permite recuperar uma chave DES com  $2^{43}$  pares conhecidos...

64



# Cripto-análise Algébrica

- › Introduzido por Nicolas Courtois and Josef Pieprzyk em 2002.
- › Aproxima a cifra por um enorme sistema equações quadráticas (e.g. AES é aproximado por 8000 equações com 1600 variáveis).
- › Os autores propõe um método heurístico para resolverem tais sistemas (aplicável em toy-ciphers, mas desapropriado para cifras como o AES).
- › ...mas existe a esperança/risco de vir a estender esses métodos por forma a permitirem comprometer cifras actuais...
- › Em geral, a comunidade científica mantém-se muito céptica relativamente à viabilidade deste ataque em cifras como o AES.

65



## Side-Channel attacks

- › Muitas técnicas de cripto-análise não atacam directamente o algoritmo criptográfico mas procuram retirar informação do “ambiente” onde este é executado...
  - Power analysis - analisa padrões no consumo de energia;
  - Timming analysis - analisa tempos de execução das operações;
  - Differential fault analysis - analisa as alterações observadas após injecção de erros;
  - ...
- › Estas técnicas não são, de forma alguma, específicas da criptografia simétrica. Algumas são até particularmente adequadas para atacar cifras assimétricas...

66



# Timing attacks...

- › Analisa tempos de execução das operações criptográficas por forma a retirar informação sobre os segredos criptográficos.
- › Particularmente efectiva para algoritmos assimétricos, onde as operações são fortemente dependentes dos dados (e.g. exponenciação modular)
- › ...ou implementações baseadas em “loopup tables” (e.g. cache attacks)
- › Cuidados especiais nas implementações impedem (ou dificultam) estes ataques (e.g. dummy code; randomized delays; etc.)

67



# Funções de Sentido Único

# Tópicos de Teoria da Complexidade

Podem-se classificar os problemas computacionais como:

- **Tratáveis** - existe um algoritmo eficiente para a sua resolução;
- **Intratáveis** - o melhor algoritmo para resolver o problema requer recursos computacionais inviáveis;
- **Insolúveis** - não é possível estabelecer um algoritmo para a resolução do problema.

69



## Complexidade

- › Procura avaliar a eficiência (temporal e/ou espacial) dos algoritmos em função do “tamanho” dos dados de entrada.
- › Modelo computacional adoptado é normalmente a máquina de Turing (mas...)
- › É normal conduzirmos a análise para o *pior caso*, mas também é possível considerar o *caso médio* (análise estatística).
- › Análises assimptótica (e.g.  $O(2^n)$ ;  $O(n^5)$ )

70



# Classes de Complexidade

- › **P – Polinomial** – tempo de execução é limitado por um polinómio (sobre o tamanho dos dados de entrada).
- › **NP – Não determinístico Polinomial** – tempo de execução é limitado por um polinómio numa máquina de Turing não determinística (modelo computacional pode criar execuções concorrentes para prosseguir diferentes alternativas...).
- › **EXP – Exponencial** – tempo de execução é exponencial...

71



## P versus NP

- › É normal identificar-se a classe **P** com a classe dos problemas *tratáveis*.
- › Um problema diz-se P(NP)-completo quando qualquer problema em P(NP) dispõe de uma redução polinomial nele.
- › Exemplo de problema NP-Completo: *validade em lógica proposicional*.
- › Conjectura **P ≠ NP** ...

72



# Funções de sentido único

- › Em criptografia gostaríamos de dispor de funções que:
  - possuam um algoritmo eficiente para o seu cálculo,
  - não disponham de um algoritmo eficiente que calcule uma sua (pseudo-)inversa.

essas funções são designadas por

## Funções de Sentido Único

73



## FSU (cont.)

- › É óbvio que funções de sentido único com domínio finito pertencem à classe NP.
- › Mas:
  - **P ≠ NP** não implica a existência de funções de sentido único.
  - Complexidade do “pior caso” não é adequada para garantir segurança.
  - Mas acredita-se que certas funções cumprem os requisitos (a **teoria dos números computacional** tem-se revelado a principal fonte para esses problemas).

74



# Funções de Hash criptográficas

- › Um exemplo de aplicação de funções de sentido único são a *funções de hash criptográficas*.
- › A sua segurança baseia-se, portanto, em argumentos de natureza de *complexidade computacional*.
- › O objectivo é que mensagens de comprimento arbitrário sejam mapeadas num contra-domínio de tamanho fixo.
- › ... mas devem ser “de sentido único” no sentido em que não deve ser possível inverter essa função.
- › Exemplos: MD5, SHA-1, SHA-256, RIPEMD-160, SHA-3

75



## Propriedades

Os requisitos das funções de hash são normalmente expressos pela seguinte hierarquia de propriedades:

- **(First) pre-image resistant:** dado um valor de hash  $h$ , deverá ser inviável conseguir obter uma mensagem  $m$  tal que  $\text{hash}(m)=h$ .
- **Second pre-image resistant:** dada uma mensagem  $m_1$ , deverá ser inviável obter uma mensagem  $m_2$  distinta de  $m_1$  tal que  $\text{hash}(m_2)=\text{hash}(m_1)$ .
- **Collision resistant:** não é viável encontrar mensagens distintas  $m_1$  e  $m_2$  tais que  $\text{hash}(m_1)=\text{hash}(m_2)$ .

A resistência a colisões é a propriedade que se deseja nas funções de hash criptográficas. No entanto, em certas aplicações é suficiente uma das propriedades mais fracas.

76



# Birthday attack

- › Um resultado famoso da teoria das probabilidades indica-nos que necessitamos de um contra-domínio de “tamanho razoável” para se conseguir resistência a colisões.

*Quantas pessoas se tem (em média) que perguntar a idade numa festa de anos para encontrar duas com o mesmo dia de aniversário?*

...testando cerca de  $\sqrt{N}$  valores aleatórios do domínio dispõe-se de probabilidade superior a  $1/2$  de encontrar uma colisão!

- › Valores típicos para contra-domínios de funções de Hash criptográficas: 128..512 bit.
- › ...assim, um ataque por força bruta para encontrar colisões deve testar entre  $2^{64}$  e  $2^{256}$  mensagens.

77



## Aplicações das funções de hash criptográficas

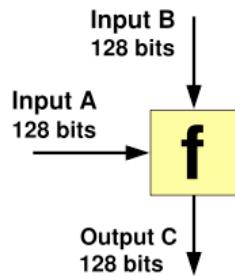
- › Armazenamento de passwords.
- › *Commitment schemes* (provas de “posse” de informação).
- › Amplificação de *Entropia* (e.g. Password-based Key Derivation Functions)
- › como componentes de outras técnicas:
  - MACs
  - Geradores de sequências aleatórias seguras (PRNG)
  - Cifras
  - ...

78

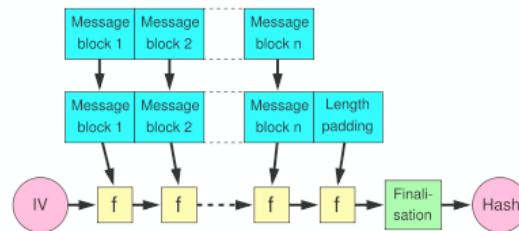


# Desenho de Funções de Hash Criptográficas

- › Um ingrediente fundamental no desenho de funções de hash criptográficas são as **funções de compressão**:
  - Estas são funções de sentido único nos seguintes sentidos:
    - conhecendo ambas as entradas, é fácil calcular a saída;
    - conhecendo a saída, é difícil calcular qualquer uma das entradas;
    - conhecendo a saída e uma das entradas, é difícil calcular a outra.
  - Devem também ser *resistentes a colisões*.
- › Podem ser construídas a partir de cifras por blocos... (cumprem metade dos requisitos directamente - existem construções standard que permitem obter funções de compressão a partir dessas cifras)



- › A generalidade das funções de hash baseia-se na construção de Merkle-Damgård:



- › A função de compressão é responsável por fazer evoluir o estado interno (do estado anterior e de um bloco da mensagem).
- › O IV é normalmente específico do algoritmo (constante).
- › Demonstra-se que, se  $f$  é uma função de compressão livre de colisões, a função de hash resultante também o é.
- › É importante o *padding* conter informação relativa ao comprimento da mensagem.

# MD5

- › Proposto pela *RSA Labs* (Donald Rivest).
- › Melhoramento da função MD4.
- › Tamanho do contra-domínio: 128 bit.
- › Processa a mensagem em blocos de 512 bit.
- › Opera por *rounds* (4).
- › Nos últimos anos tem surgido avanços importantes na sua cripto-análise. Em particular, já foram encontradas colisões.
- › É por isso desaconselhada para aplicações com requisitos de segurança elevada.

81



# SHA-1

- › Função de hash incluída no DSS (Digital Signature Standard)
- › Desenvolvida pela NSA (para a NIST) – 1993/1995.
- › ...também um melhoramento (correcção) de uma versão anterior (SHA-0).
- › Tamanho do contra-domínio: 160 bit.
- › Optimizada para arquitecturas *big-endian*.
- › Sua cripto-análise também tem sido alvo de avanços recentes significativos.
- › Já foram propostas variantes mais seguras (SHA-2: sha-224, sha-256, sha-384, sha-512)

82



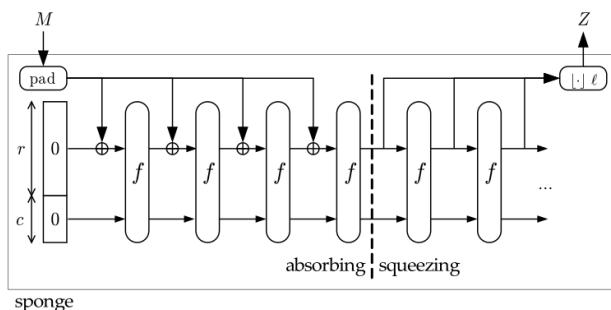
# SHA-3

- › Novo standard para função de hash da NIST (FIPS 202 - 08/2015)
- › Seleccionado após concurso de selecção lançado em 2006 (51 candidatos; 14 seleccionados para 2ª fase).
- › Algoritmo vencedor: Keccak (instâncias específicas)
- › Utiliza a *sponge-construction*, dispondo de variadíssimos parâmetros de configuração que permitem ajustar nível de segurança/eficiência.
- › Tamanho do contra-domínio: 224/256/384/512 bit.

83



## sponge construction



- › permite processar *input* e gerar *output* de tamanho arbitrário
- › apenas uma fração do estado (bitrate  $r$ ) intervém nas fases em que *input* é consumido (*absorbed*), e *output* gerado (*squeezed out*)
- › Capacidade  $c$  é determinante na resistencia a colisões

84

# Message Authentication Codes (MAC)

- › As funções de hash, por si só, não garantem nem a integridade nem autenticidade! (... mas quando utilizadas com uma cifra já permitem estabelecer essas propriedades)
- › Um **código de autenticação** (MAC), pode ser entendido como “uma função de hash com segredo” e visa garantir essas propriedades.

85



## HMac

- › A forma mais simples de construir um MAC é combinar uma função de hash com um segredo (de forma apropriada).
- › Uma dessas construções o designada por **HMAC**.
- › Dada uma função de hash  $h$ , define-se HMAC- $h$  como:
  - $\text{HMAC-}h(K,M) = h((K \oplus \text{opad}) \parallel h((K \oplus \text{ipad}) \parallel M))$
  - B = tamanho dos blocos em que opera a função de hash (em bytes)
  - L = tamanho do resultado da função de hash (em bytes)
  - K = chave (tamanho variável entre L e B)
  - ipad = byte 0x36 repetido B vezes
  - opad = byte 0x5C repetido B vezes

86



# MACs derivados de Cifras por Blocos

- › Já referimos que o último bloco de criptograma do modo CBC pode ser utilizado como um MAC (**CBC-MAC**).
- › No entanto, esse método só é seguro para mensagens de comprimento fixo. Existem no entanto construções que ultrapassam essa limitação (CMAC ou PMAC).
- › Existem também modos que combinam as garantias de confidencialidade com integridade/autenticação (e.g. EAX, GCM, OCB, etc.).

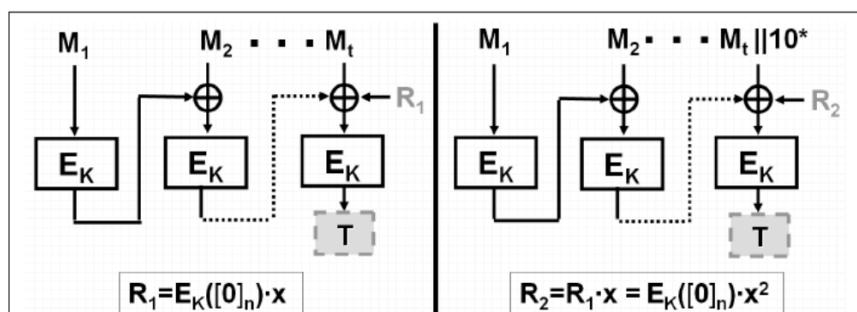
87



## CMAC e PMAC (MACs)

### › CMAC

- surge no seguimento de propostas de refinamentos do CBC-MAC (XCBC de Black & Rogaway e OMAC de Iwata & Kurosawa)



### › PMAC

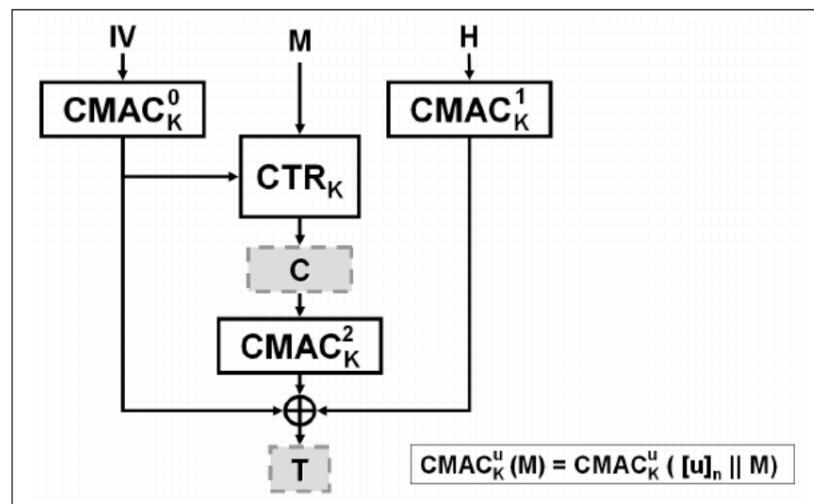
- proposta com eficiência e segurança similar ao CMAC, mas com a característica de permitir implementação paralelizável.

88



# Modo EAX (authenticated encryption)

- › Modo que combina cifra em modo CTR com mac CMAC (2 passagens)

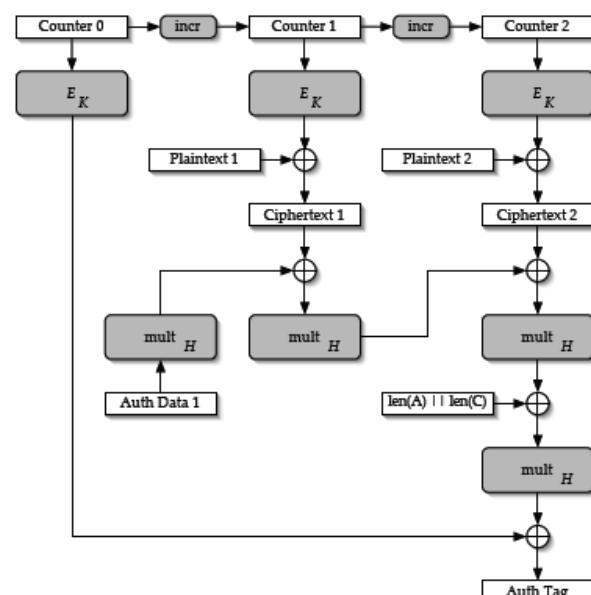


89



# Modo GCM (authenticated encryption)

- › Combina um MAC de avaliação polinomial (GHash) com o modo de cifra CTR.



90



# MACs baseados em Universal Hashing

- › *Universal Polynomial Hash Function* ( $m=m_1, \dots, m_n$ ):  
$$h(m) = m_1 * r^n + \dots + m_n * r$$
one:  $m_i$  são considerados coeficientes de um polinómio (num determinado corpo finito), e  $r$  é um ponto aleatório.
- › Quando combinado com *One-Time-Pad* permite obter um MAC com segurança absoluta (*Information Theoretic*)  
$$\text{Tag}(m) = h_r(m) \oplus s$$
- › Na prática, é usado em combinação com uma cifra (e.g. AES), como no modo GCM.
- › Para a eficiência da construção, é crítica a escolha do corpo finito onde a avaliação do polinómio é realizada...

91



## Poly-1305

- › MAC proposto por D. Bernstein, baseado numa *universal polynomial hash*.
- › Originalmente desenhado para ser combinado com o AES (Poly1305-AES), mas que pode ser adaptado para qualquer cifra (e.g. Poly1305-Chacha20)
- › Escolha criteriosa do corpo finito (primo  $2^{130}-5$ ) permite implementação muito eficiente das operações modulares.
- › Utilizado na cifra autenticada Chacha20-Poly1305 (rfc8439, e TLS-1.3)

92



# PBKDF (Password-Based Key Derivation Functions)

- › Por vezes há necessidade de construir uma chave apropriada para uma dada técnica a partir de **chaves fracas** (e.g. passwords ou passphrases).
- › O principal problema é ficar-se vulnerável a ataques de dicionário
  - o adversário pode “catalogar” todo o espaço de chaves.
- › Estratégias para dificultar esses ataques:
  - Considerar factores aleatórios (designados por **salt**, ou IV). Assim procura-se impedir a pré-computação do dicionário. Na sua forma mais simples, o salt é concatenado com o segredo.
  - Aumentar o “peso computacional” da função de derivação da chave. Assim dificulta-se a realização de ataques em tempo real.

93



## PBKDF1

- › Função de geração de geração de chaves proposta no standard PKCS5 (Password-based encryption).
- › Considera um valor aleatório  $S$  (salt) e um número de iterações  $C$  (iteration count).
- › Itera uma função de hash  $C$  vezes aplicada sobre  $P||S$ .
- › Limita o segredo obtido ao tamanho do resultado da função de hash.

94



# PBKDF2

- › Substitui PBKDF1 no standard PKCS5.
- › Não limita o segredo ao tamanho da função de Hash.
- › Parametrizada por uma *pseudorandom function PRF* (e.g. HMAC-sha1).
- › Para produzir um segredo  $(T_1 \parallel \dots \parallel T_k)$  a partir de uma password P (salt=S, iCount=c):
  - $T_i = F(P, S, c, \text{INT4}(i))$
  - $F(P, S, c, i) = U_1 \oplus U_2 \oplus \dots \oplus U_c$ 
    - $U_1 = \text{PRF}(P, S \parallel \text{INT4}(i))$  ,
    - $U_2 = \text{PRF}(P, U_1)$  ,
    - ...
    - $U_c = \text{PRF}(P, U_{c-1})$  .

95



## SCrypt

- › Uma KDF especificamente desenhada para resistir a “ataques por utilização de hardware dedicado”, como os que passam pela utilização de Application Specific Integrated Circuits (ASICs) ou Field Programmable Gate Arrays (FPGAs).
- › Estratégia passa por forçar a utilização de uma quantidade de memória intermédia considerável (que se traduz numa área significativa do respectivo circuito quando implementado em hardware).
  - › Internamente, usa repetidamente PBKDF2 para a construção de um estado interno...
  - › ...juntamente com uma construção (ROMix) que impede a paralelização efectiva do processo.

96



# ...mas no limite, tudo depende da entropia da pass-phrase...

Estimated cost of hardware to crack a password in 1 year.

KDF	6 letters	8 letters	8 chars	10 chars	40-char text
DES CRYPT	< \$1	< \$1	< \$1	< \$1	< \$1
MD5	< \$1	< \$1	< \$1	\$1.1k	\$1
MD5 CRYPT	< \$1	< \$1	\$130	\$1.1M	\$1.4k
PBKDF2 (100 ms)	< \$1	< \$1	\$18k	\$160M	\$200k
bcrypt (95 ms)	< \$1	\$4	\$130k	\$1.2B	\$1.5M
scrypt (64 ms)	< \$1	\$150	\$4.8M	\$43B	\$52M
PBKDF2 (5.0 s)	< \$1	\$29	\$920k	\$8.3B	\$10M
bcrypt (3.0 s)	< \$1	\$130	\$4.3M	\$39B	\$47M
scrypt (3.8 s)	\$900	\$610k	\$19B	\$175T	\$210B

