Segurança da Informação

Advanced Encryption Standard (AES)
Modos de Operação

Advanced Encryption Standard (AES)

O Processo do AES

- Janeiro de 1997: NIST decide solicitar à comunidade criptológica internacional propostas para um algoritmo substituto do DES – o AES (*Advanced Encryption Standard*).

- Abril de 1997: parâmetros preliminares mínimos do AES: bloco e chave em três tamanhos (128, 192 e 256 bits).

- Junho de 1997: início oficial do concurso.

O Processo do AES

- Julho de 1997: Joan Daemen e Vincent Rijmen iniciam do projeto de uma variante do algoritmo SQUARE, com bloco e chave em 5 tamanhos (logo depois, o requisito de 3 tamanhos de bloco é retirado pelo NIST sob a alegação de que seria "irrealizável").

- Junho de 1998: fim do prazo de proposta de candidatos.

- RIJNDAEL (***RIJ***men a***N***d ***DAE***men).

O Processo do AES

- Agosto de 1998: 15 propostas apresentadas (de um total de 21) .
- Três algoritmos com elementos do SQUARE:
  - RIJNDAEL;
  - Crypton (Coréia);
  - Twofish (Counterpane - EUA).
- Agosto de 1999: Anúncio de 5 finalistas.
- Outubro de 2000: RIJNDAEL torna-se AES.
- Novembro de 2001: FIPS 197.

RIJNDAEL

- Estrutura simétrica e paralela:
  - mais flexibilidade de implementação.
  - impede ataques criptoanalíticos efetivos.
- Bem adaptado aos processadores modernos :
  - IA32 / IA64.
  - RISC e processadores paralelos.
- Adequado para *smart cards*.
- Eficiente em *hardware* dedicado.

Estratégia de Trilha Larga

- Método sistemático de implementar os princípios de *difusão e confusão* de Shannon.

- Cifra iterativa, com alternância de camadas de embaralhamento rápido através de códigos lineares MDS ("difusão") com camadas de substituição altamente não-lineares num corpo finito, para destruir relações algébricas e estatísticas entre a entrada e a saída ("confusão").

Aritmética no corpo GF(2^8)

- Polinômios com coeficientes binários e grau menor que 8.
- Representação cabe em 1 byte:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$
$$\leftrightarrow (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0), \qquad b_i \in \{0, 1\}.$$

- Adição e multiplicação conforme os axiomas de *corpo*:
  - elemento neutro,
  - elemento inverso,
  - associatividade,
  - comutatividade,
  - distributividade.

Aritmética no corpo GF(2^8)

- Adição em GF(2^8): ou-exclusivo bit a bit.
  - Elemento neutro: 0; cada elemento é seu próprio inverso aditivo.

- Multiplicação em GF(2^8): multiplicação comum de polinômios, seguida de redução módulo $m(x) = x^8 + x^4 + x^3 + x + 1$, i.e. para $a, b \in$ GF(2^8), o produto $a \cdot b \in$ GF(2^8) é o polinômio $c(x) = a(x)b(x) \bmod m(x)$.
  - Elemento neutro: 1; inverso multiplicativo a partir do algoritmo estendido de Euclides.

Aritmética no corpo GF(2^8)

- Exemplos:

$2 \cdot 3 = x \cdot (x + 1) = x^2 + x = 6.$

$3^2 = (x + 1)^2 = x^2 + 1 = 5.$

$16^2 = (x^4)^2 = x^8 =$
$x^4 + x^3 + x + 1 \pmod{x^8 + x^4 + x^3 + x + 1}.$

Aritmética no corpo GF(2^8)

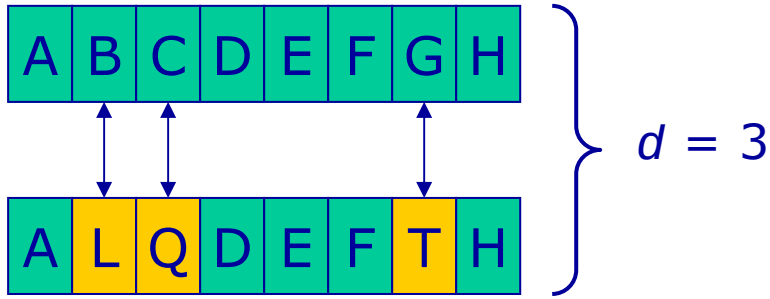
- Implementações efetivas não precisam usar diretamente a aritmética em GF(2^8).
- Técnicas diversas de implementação, conforme as características da plataforma.
- Exercício: implementar o AES em duas plataformas díspares:
  - *smart cards* (operações somente sobre bytes, pouco espaço de armazenamento);
  - IA32 ou IA64 (operações sobre words de 32, 64 ou mais bits, *cache* suficiente para tabelas maiores).

Códigos lineares

- Um *alfabeto* é um conjunto de símbolos.
- Uma *palavra* é uma seqüência de símbolos do alfabeto.
- Um *código* (de comprimento *n*) é um conjunto de palavras, todas de mesmo comprimento *n*.
- Um *código linear* é um código cujo alfabeto é um corpo (finito).

Códigos lineares

- A *distância* (de Hamming) entre duas palavras de um código é o número de posições onde elas diferem.



- A *distância de um código* é *d* se a distância entre duas palavras distintas quaisquer desse código é sempre ≥ *d*.

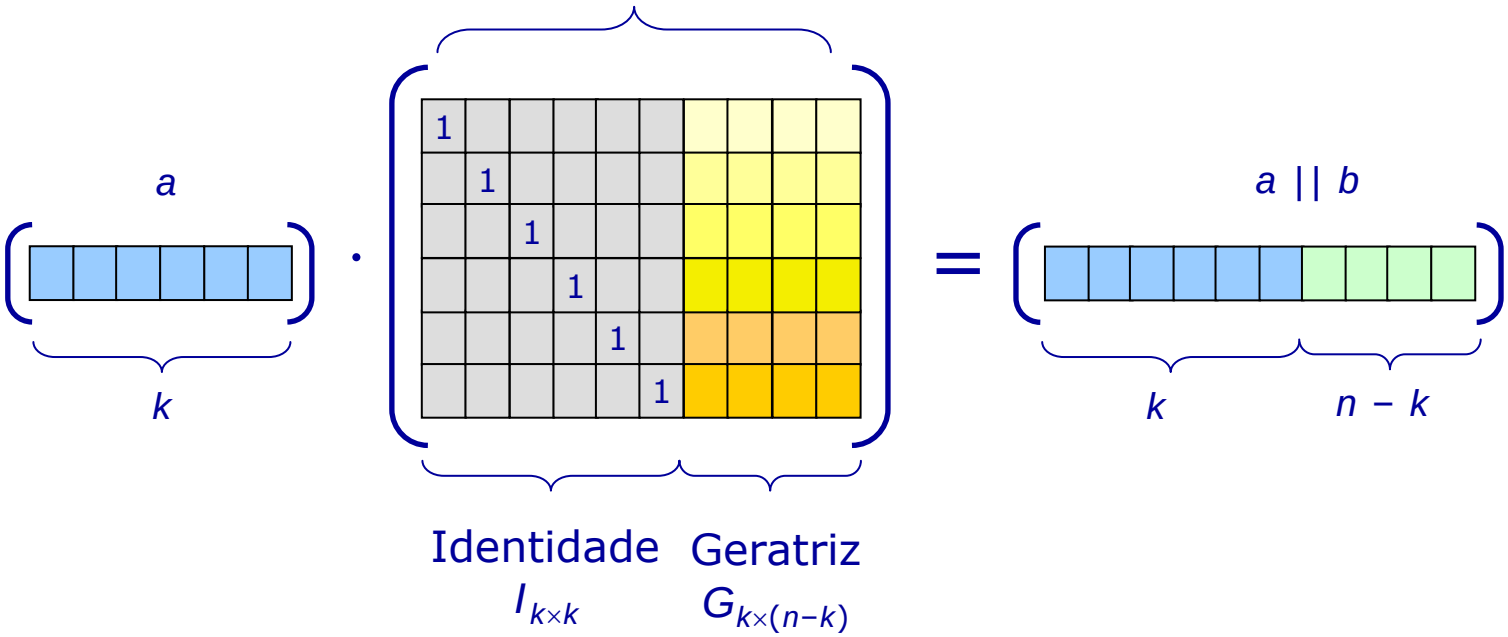
Códigos lineares

- Um código linear C sobre GF(q) é dito [n, k]-*separável* se qualquer palavra de C pode ser escrita como uma concatenação de duas cadeias $a \,||\, b$ tais que $|a| = k$ e $b = F(a)$ para alguma transformação linear F: $\mathrm{GF}(q)^k \rightarrow \mathrm{GF}(q)^{n-k}$.

- Em outras palavras, existe uma matriz (em forma escalonada) $H_{k \times n}$ tal que $a \cdot H = a \,||\, b$.

Códigos lineares

Matriz (escalonada) de Paridade $H_{k\times n} = [\ I_{k\times k} \mid G_{k\times(n-k)}\ ]$



Códigos lineares

- Um *código MDS* é um código linear [n, k]-separável com distância d máxima.
- Teorema: num código MDS, $d = n - k + 1$.
- Teorema: um código é MDS $\Leftrightarrow$ todas as submatrizes da geratriz $G_{k\times(n-k)}$ são não singulares (determinante não nulo).

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$$

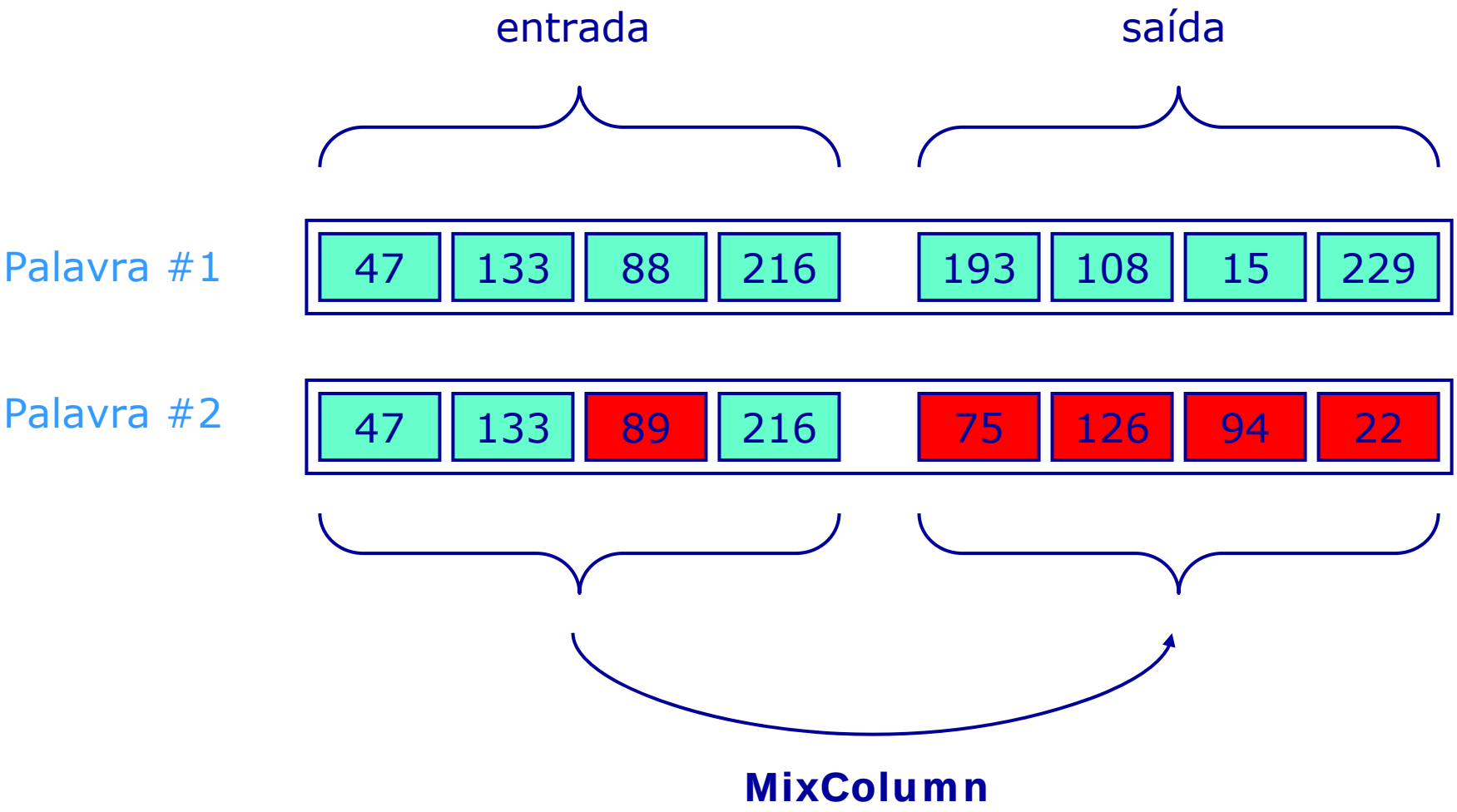
Códigos lineares

- RIJNDAEL utiliza, implicitamente, um código [8,4]-MDS sobre GF(2^8) (portanto com distância 5) em sua camada de difusão.

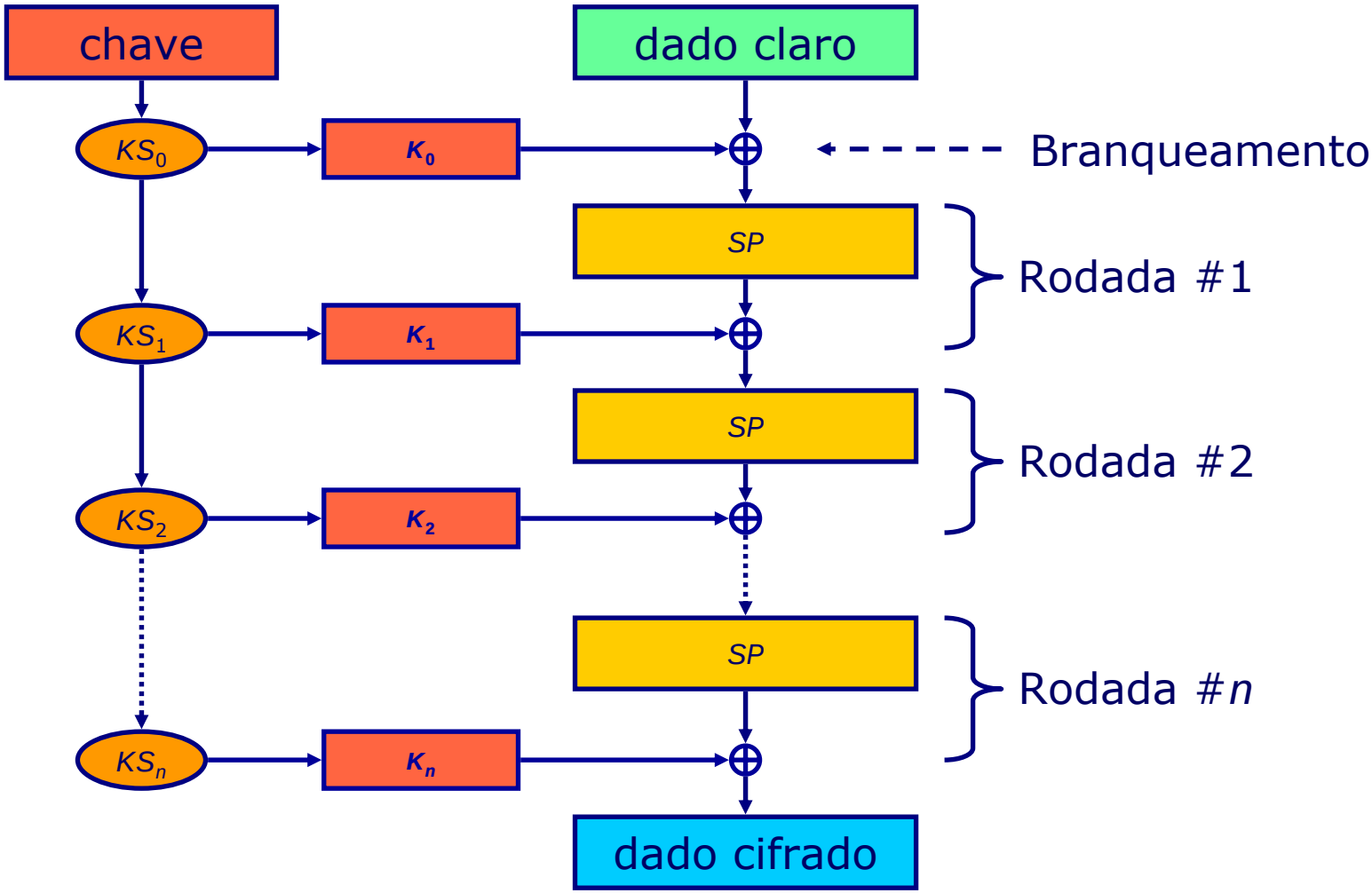
- Os 4 primeiros elementos de uma palavra do código (parte separável) representam a entrada de dados, e os 4 elementos seguintes a saída.

- Como a distância é 5, uma alteração de um único byte da entrada propaga-se para todos os 4 bytes de saída.

Códigos lineares



Rede de Substituição e Permutação



RIJNDAEL por dentro

- Os bytes da chave e do bloco de dados são organizados logicamente em forma de matrizes.

- Parâmetros:
  - Bloco de 128, 160, 192, 224 ou 256 bits.
  - Chave de 128, 160, 192, 224 ou 256 bits.

- Função de rodada aplicada de 10 a 14 vezes, segundo o tamanho da chave e do bloco.

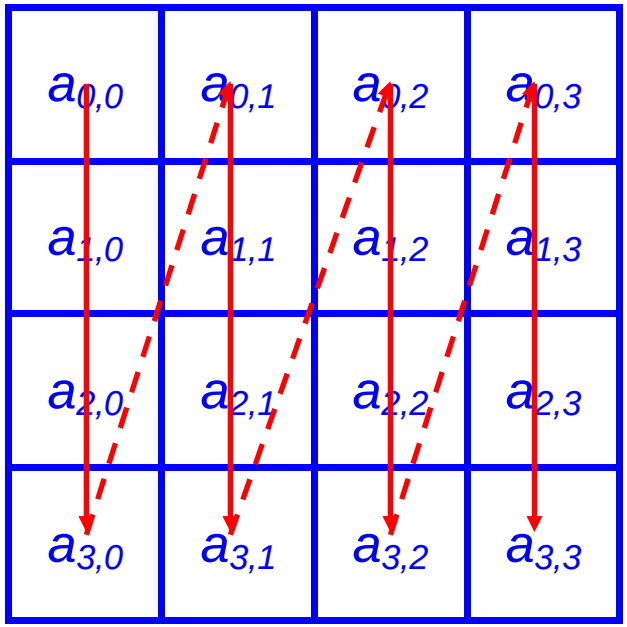
RIJNDAEL por dentro

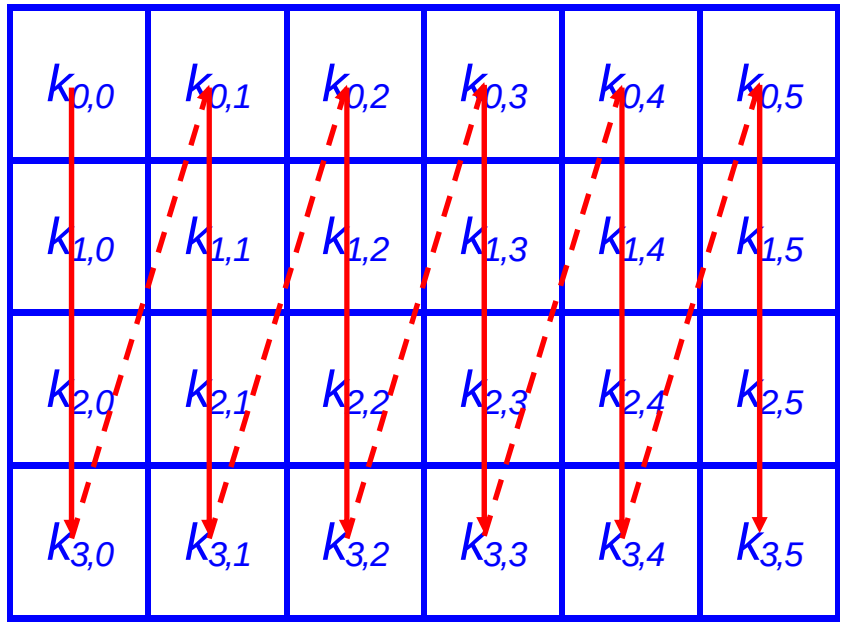
- Número de rodadas:

Bloco \ Chave	128	160	192	224	256
128	10	11	12	13	14
160	11	11	12	13	14
192	12	12	12	13	14
224	13	13	13	13	14
256	14	14	14	14	14

- AES é um subconjunto de RIJNDAEL:
  - Bloco de 128 bits.
  - Chave de 128, 192 ou 256 bits.

Organização dos dados e chaves





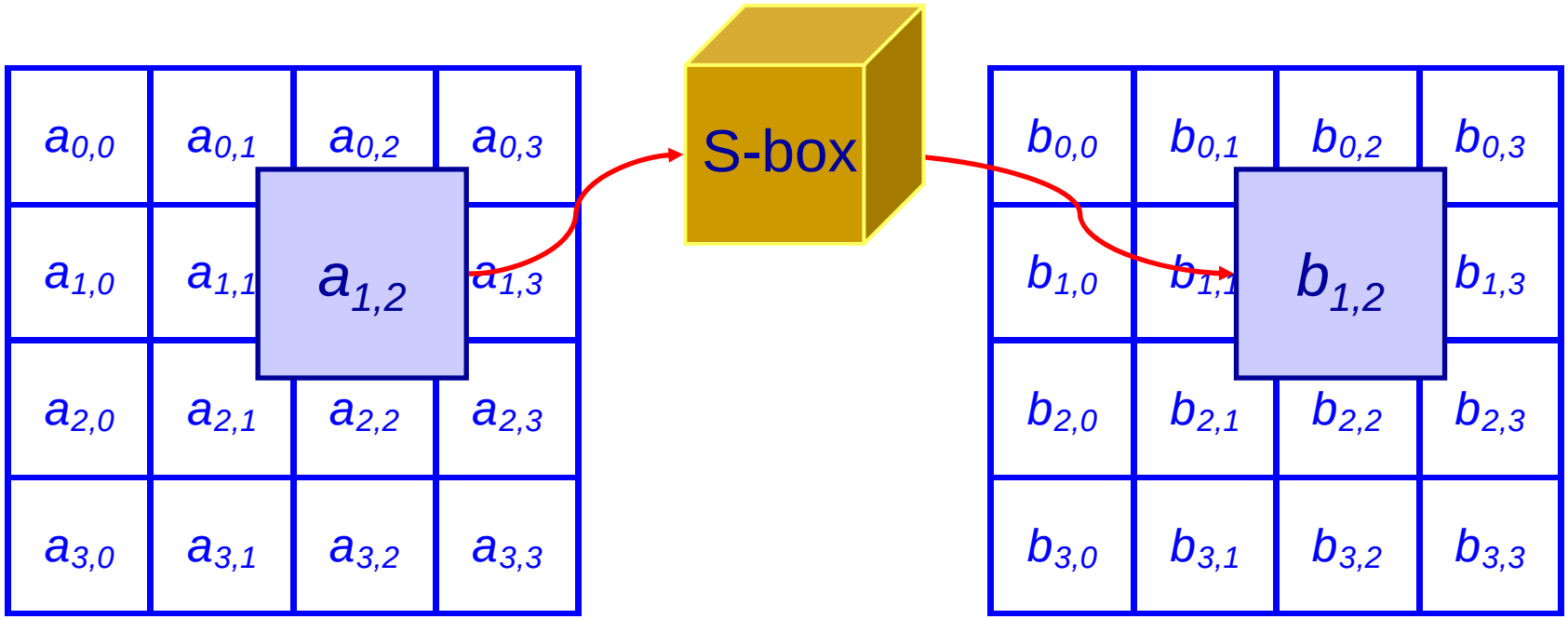
Estrutura das rodadas

- Quatro transformações, cada um com sua própria finalidade:
  - ***SubBytes***: não-linearidade.
  - ***ShiftRows***: difusão entre as colunas de dados.
  - ***MixColumns***: difusão dentro de cada coluna.
  - ***AddRoundKey***: aplicação da chave.

SubBytes

- Os bytes de dados são individualmente transformados pela aplicação de uma tabela de substituição (S-box) inversível.
- Uma única S-box para o algoritmo inteiro.
- Máxima não-linearidade: $u \rightarrow u^{-1}$ em GF(2^8) (com $0 \rightarrow 0$), seguido de uma transformação afim sobre GF$(2)^8$.

SubBytes


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
S-box
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

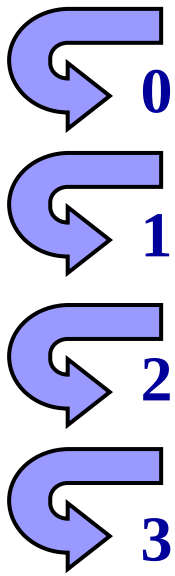
ShiftRows

- As linhas da matriz de dados são deslocadas à esquerda com 4 *offsets* diferentes.

- A interação com MixColumns resulta em difusão máxima sobre 4 rodadas (teorema do quadrado): dadas duas entradas diferentes, pelo menos 25 substituições de byte por aplicação da S-box produzem resultados diferentes.

ShiftRows

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

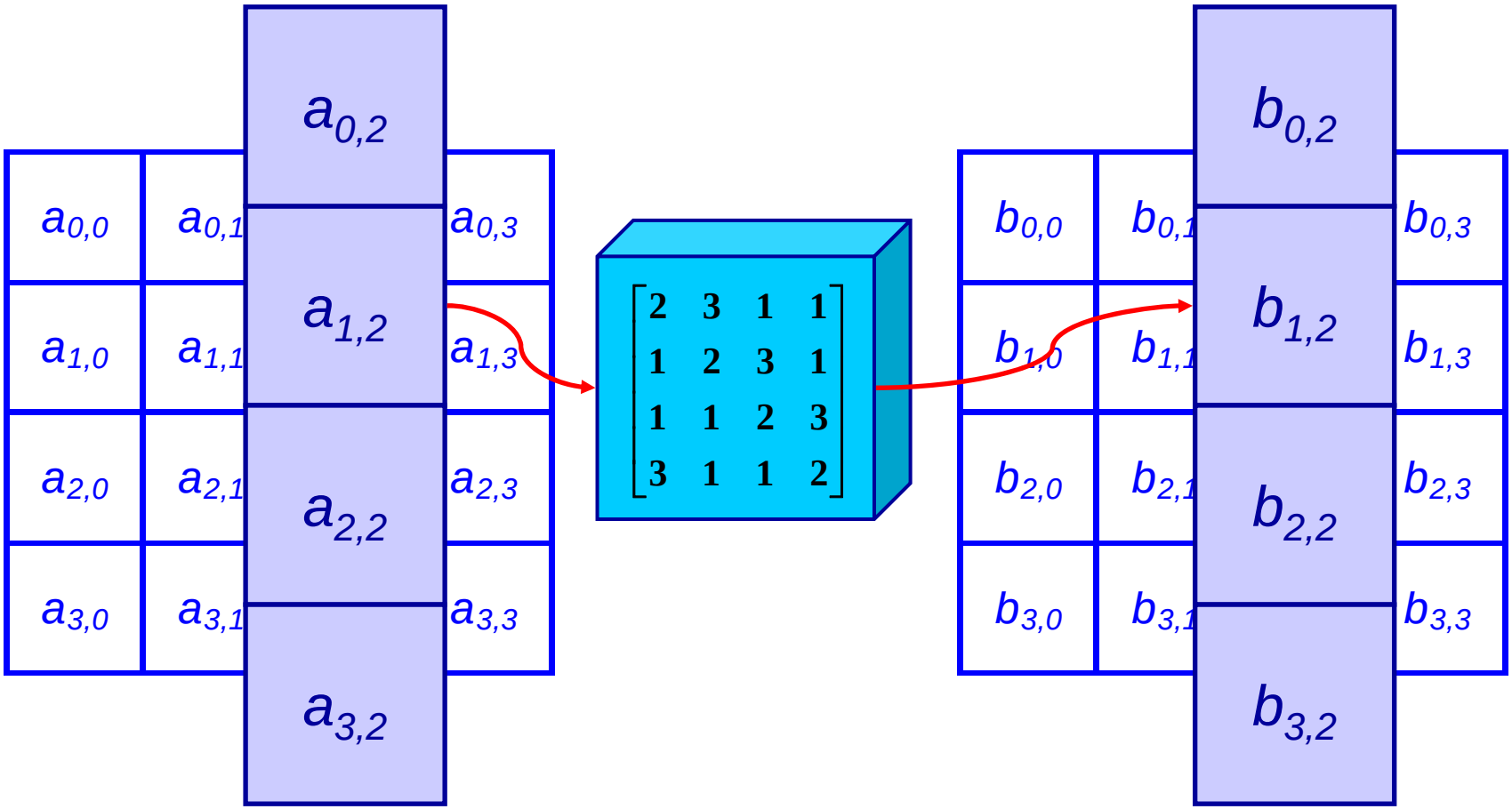


$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

MixColumns

- Os bytes em cada coluna são combinados linearmente.
- Difusão extrema dentro da cada coluna: transformação baseada em código MDS.
- Coeficientes adequados para implementação eficiente em hardware, smart cards e outras plataformas restritas (apenas 1, 2 e 3).

MixColumns


$a_{0,2}$
$a_{0,0}$ $a_{0,1}$ $a_{0,3}$
$a_{1,2}$
$a_{1,0}$ $a_{1,1}$ $a_{1,3}$
$a_{2,2}$
$a_{2,0}$ $a_{2,1}$ $a_{2,3}$
$a_{3,2}$
$a_{3,0}$ $a_{3,1}$ $a_{3,3}$
$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$
$b_{0,2}$
$b_{0,0}$ $b_{0,1}$ $b_{0,3}$
$b_{1,2}$
$b_{1,0}$ $b_{1,1}$ $b_{1,3}$
$b_{2,2}$
$b_{2,0}$ $b_{2,1}$ $b_{2,3}$
$b_{3,2}$
$b_{3,0}$ $b_{3,1}$ $b_{3,3}$

MixColumns

- Exercício: mostrar que a matriz

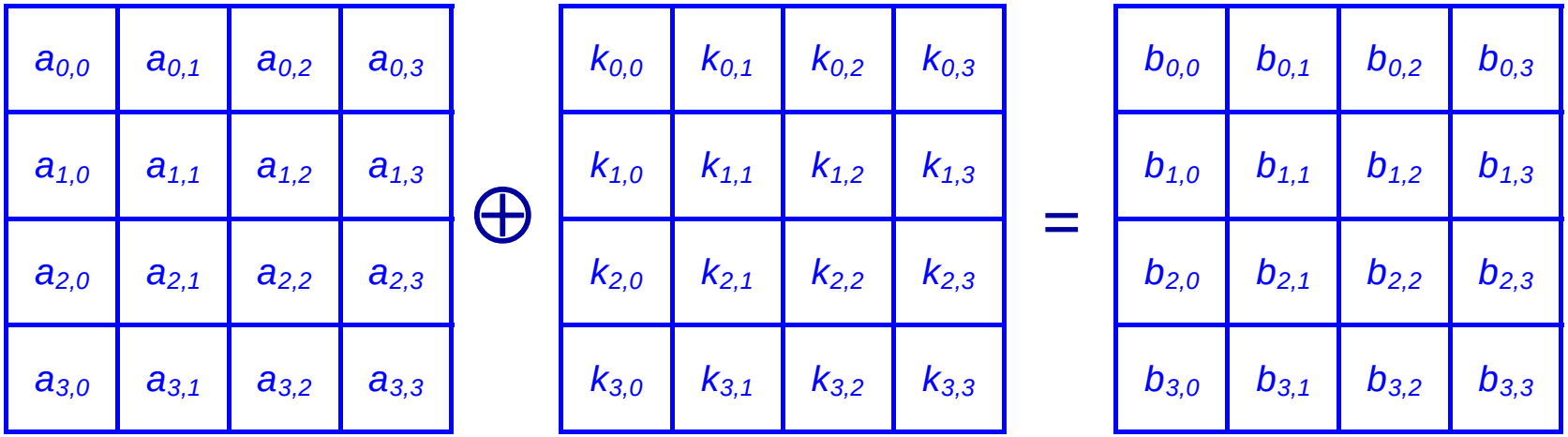
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

é a geratriz de um código MDS (calcular os determinantes de todas as submatrizes sobre GF(2^8) e verificar que são não nulos).

AddRoundKey

- Torna a função de passo dependente da chave.
- Pequeno número de operações.
- Pequena quantidade de memória.
- Eficiência em hardware (não há “carry”).

AddRoundKey



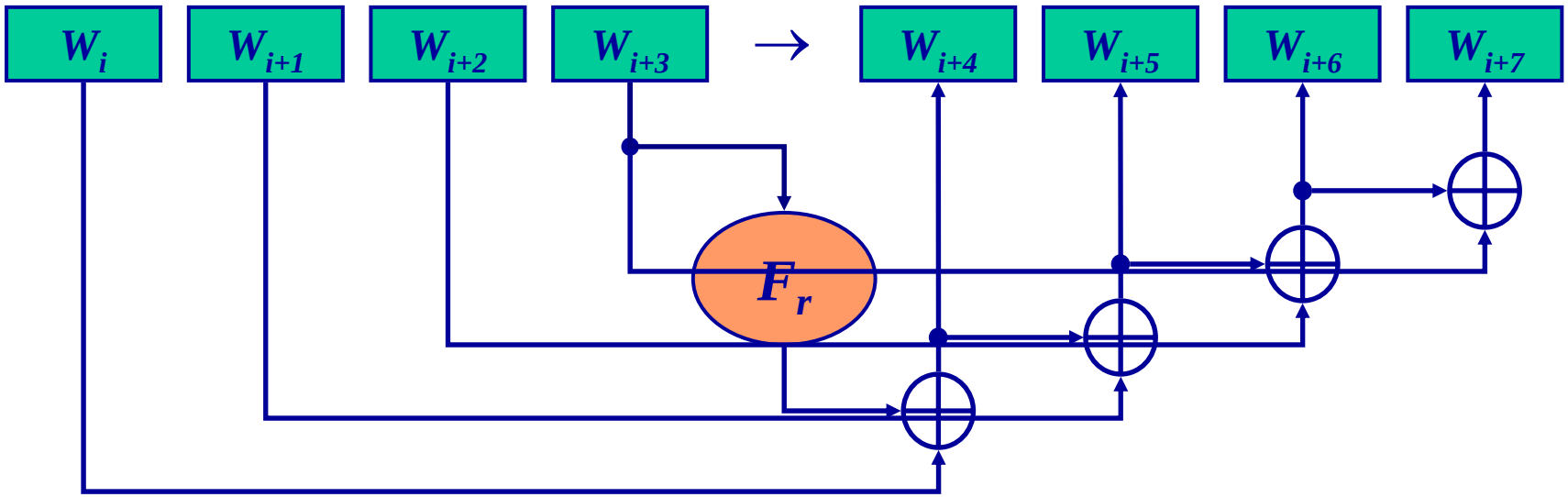
Escalonamento de chaves

- O método de cálculo das chaves de rodada (escalonamento de chave) é o mais simples possível mantendo imunidade a criptoanálise de chave relacionada.

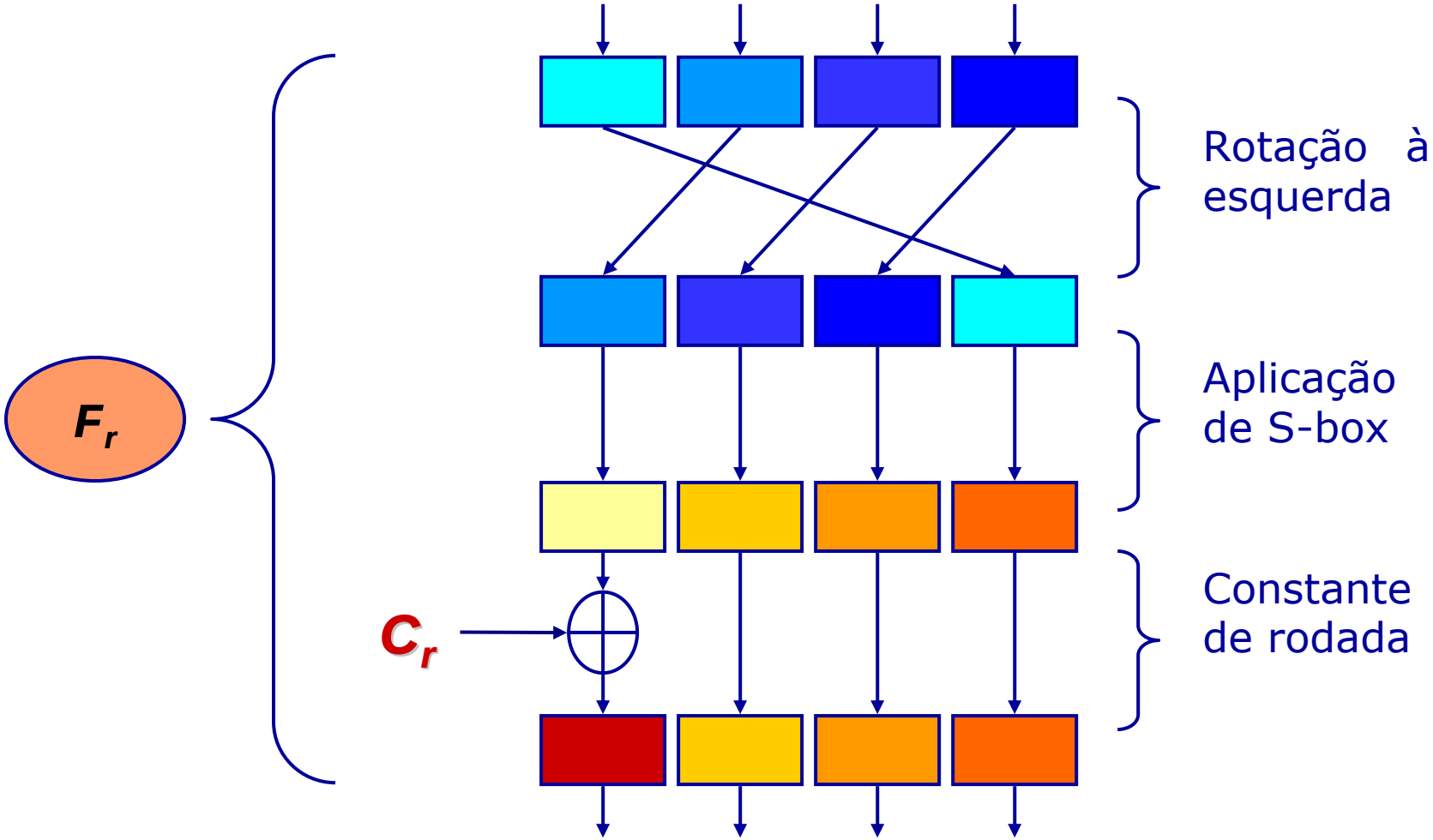
- A chave mestra é expandida numa seqüência de chaves de rodada com uma transformação linear, SubBytes e adição de constantes.

- Para obter as chaves de rodada para decifração, aplica-se a inversa de MixColumns.

Escalonamento de chaves



Escalonamento de chaves



Criptoanálise de Cifras de Bloco

- Área mais desenvolvida da criptoanálise moderna.

- Técnicas classificam-se segundo as diversas hipóteses sobre o nível de acesso do agressor ao sistema atacado:
  - Texto cifrado puro.
  - Texto conhecido.
  - Texto escolhido.
  - Chave relacionada.

Criptoanálise de Cifras de Bloco

- Criptoanálise diferencial (clássica, truncada, de ordem superior, bumerangue, quadrado, retângulo, cubo).
- Criptoanálise linear (clássica, de casco linear, diferencial-linear, não-linear).
- Criptoanálise de chave relacionada (ataque deslizante, ataque quadrado de chave).
- Ataques dedicados (interpolação, XSL, métodos algébricos).
- Ataques contra implementações (diferencial de consumo de potência, diferencial de falhas, diferencial de tempos de processamento).

Modos de Operação

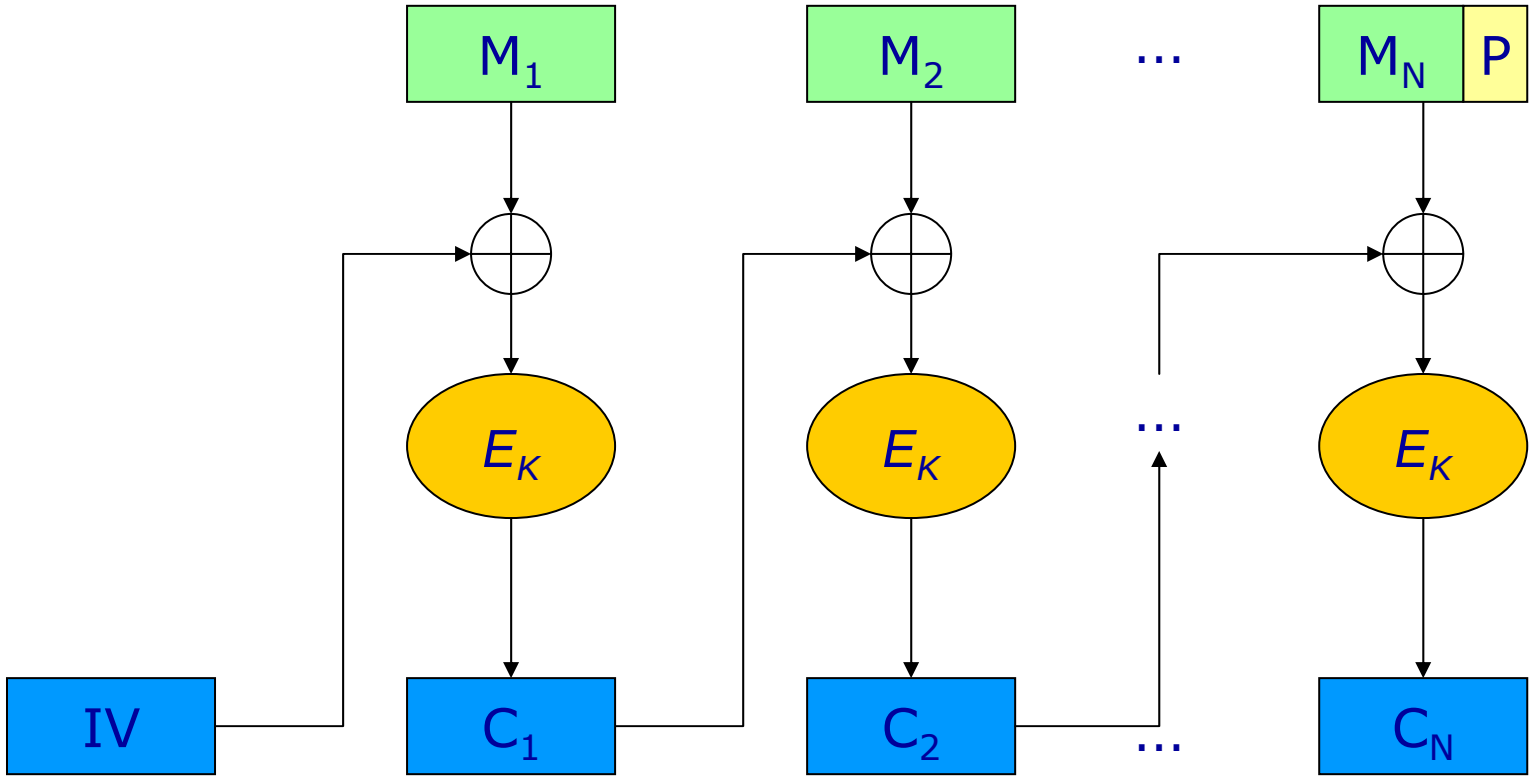
Modos de Operação

- Cifras de bloco apenas operam sobre mensagens de tamanhos fixo.
- Modo de operação: extensão para outros tamanhos de mensagem.
- Modos de confidencialidade:
  - ECB, CBC (*padding*).
  - XTS, LRW, XEX, EME (cifração setorial).
  - CFB, OFB, *CTR*.
- Modos de integridade: CBCMAC, CMAC.
- Modos híbridos: OCB, CCM, EAX, GCM.

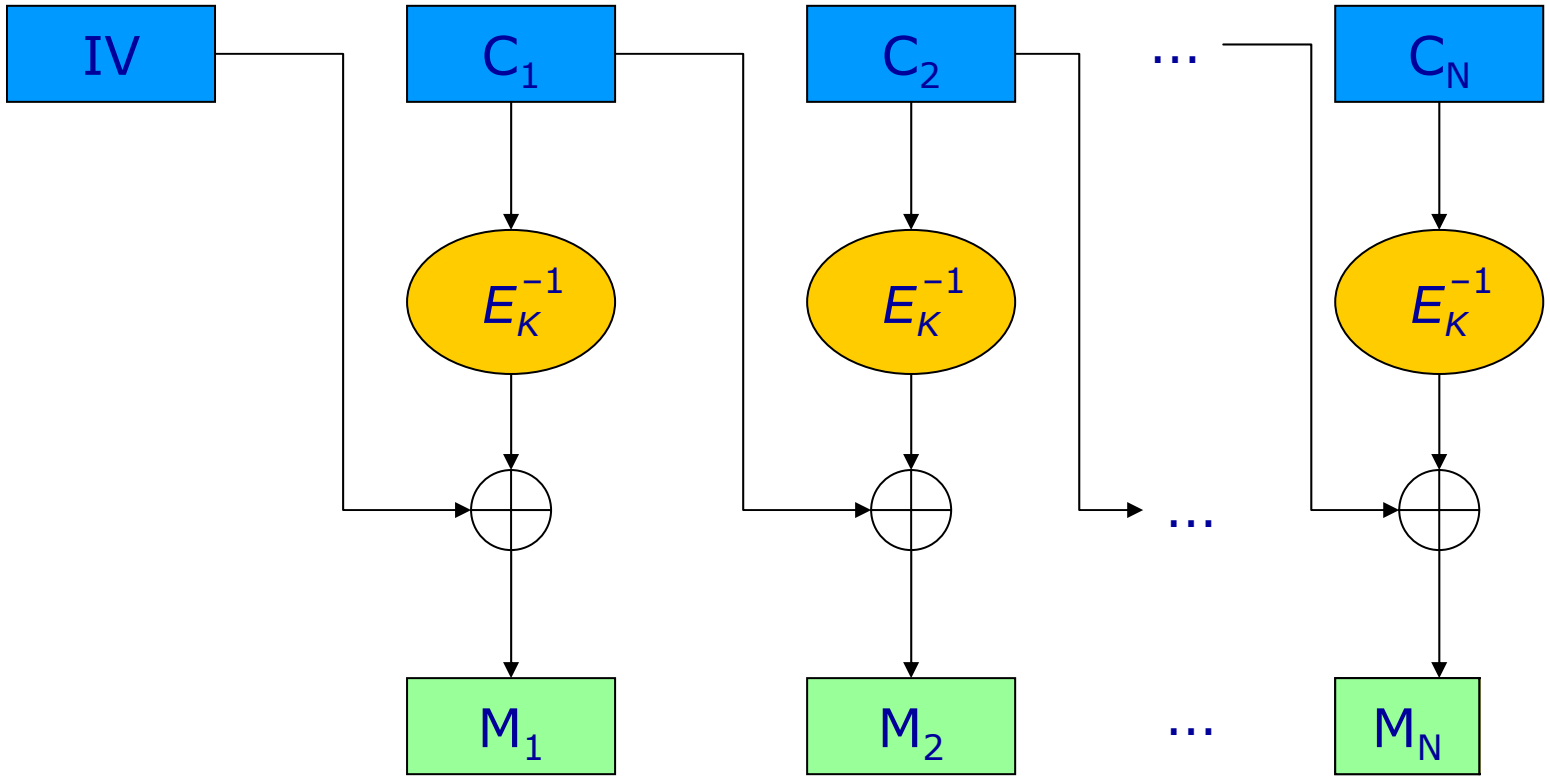
Padding

- Mensagem complementada para que o comprimento seja múltiplo do tamanho do bloco.
- Hipótese de não-ambigüidade: *padding* é sempre acrescentado.
- PKCS *padding*: byte a byte.
  - Acrescentar *m* bytes com o valor *m*.
- NIST *padding*: bit a bit.
  - Acrescentar um bit '1'.
  - Completar com bits '0'.

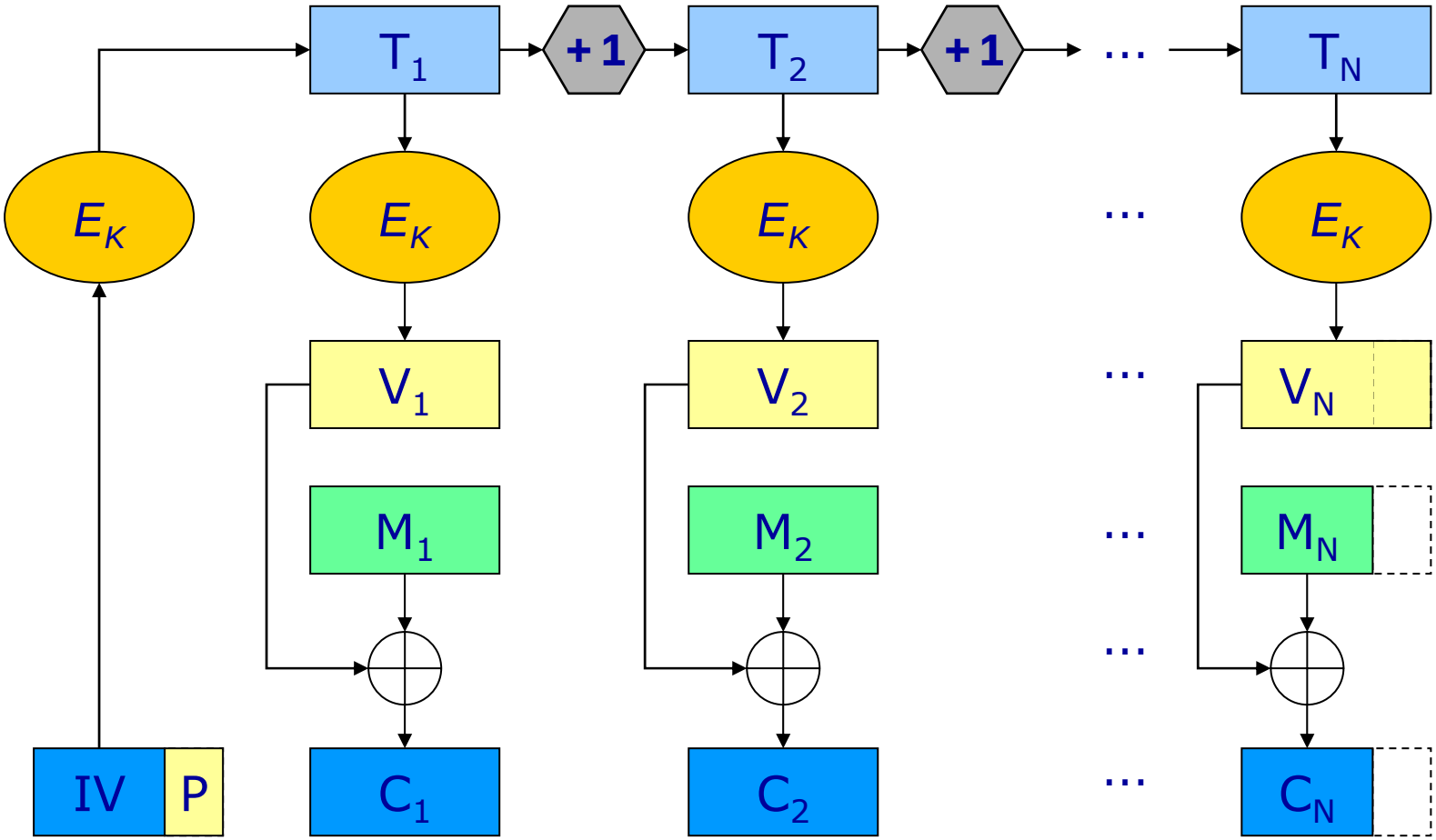
Modo CBC - Cifração



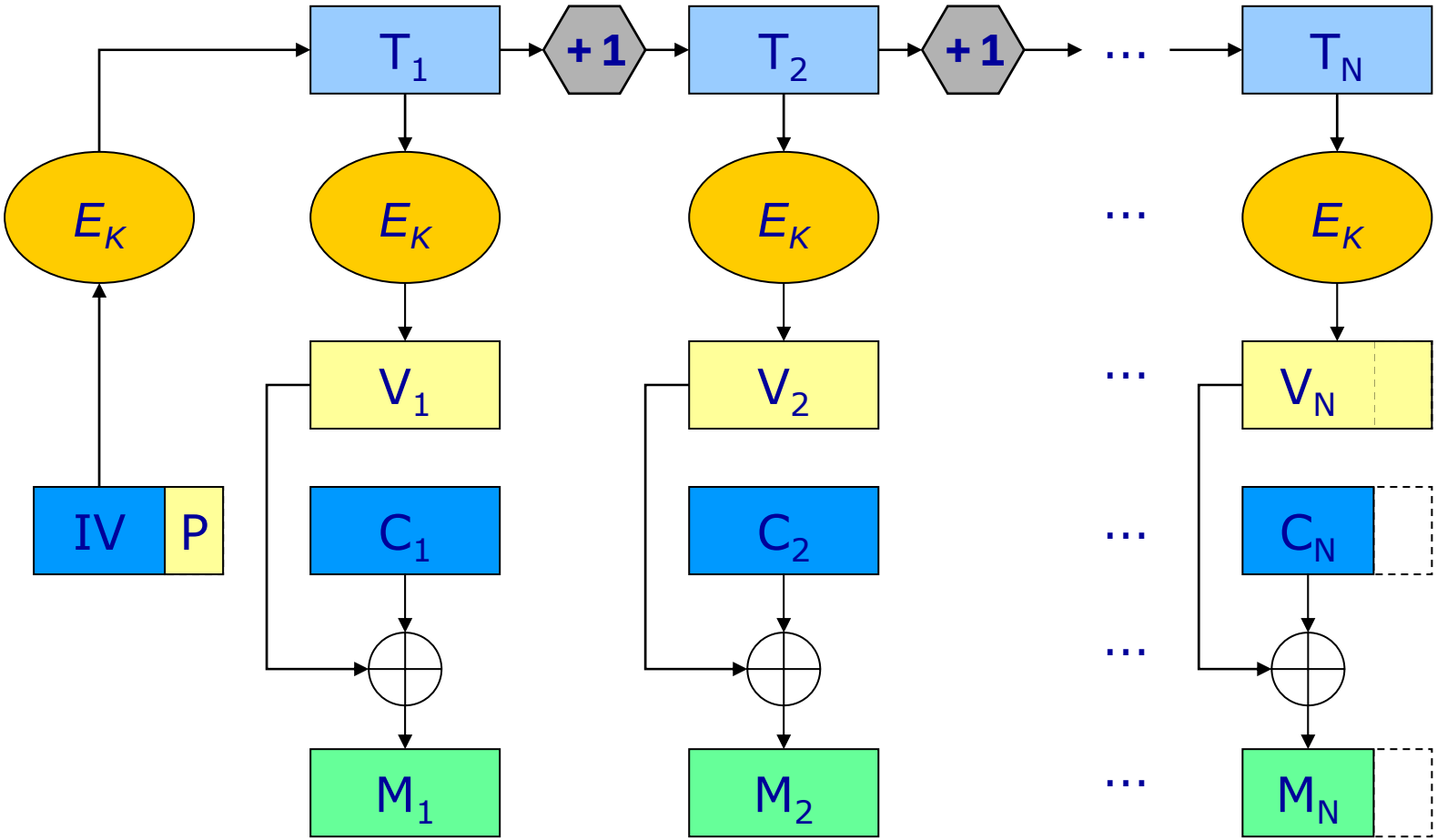
Modo CBC - Decifração



Modo CTR - Cifração



Modo CTR - Decifração



Modos LRW e XTS

- Projetado para a encriptação de setores de disco e outros dispositivos orientados a bloco (Liskov-Rivest-Wagner).
- Paralelizável: blocos independentes.
- Duas chaves: K e L (n bits: tamanho do bloco).
- Cifração do bloco M com índice lógico I:

$$Z \leftarrow \text{offset}(L, I)$$

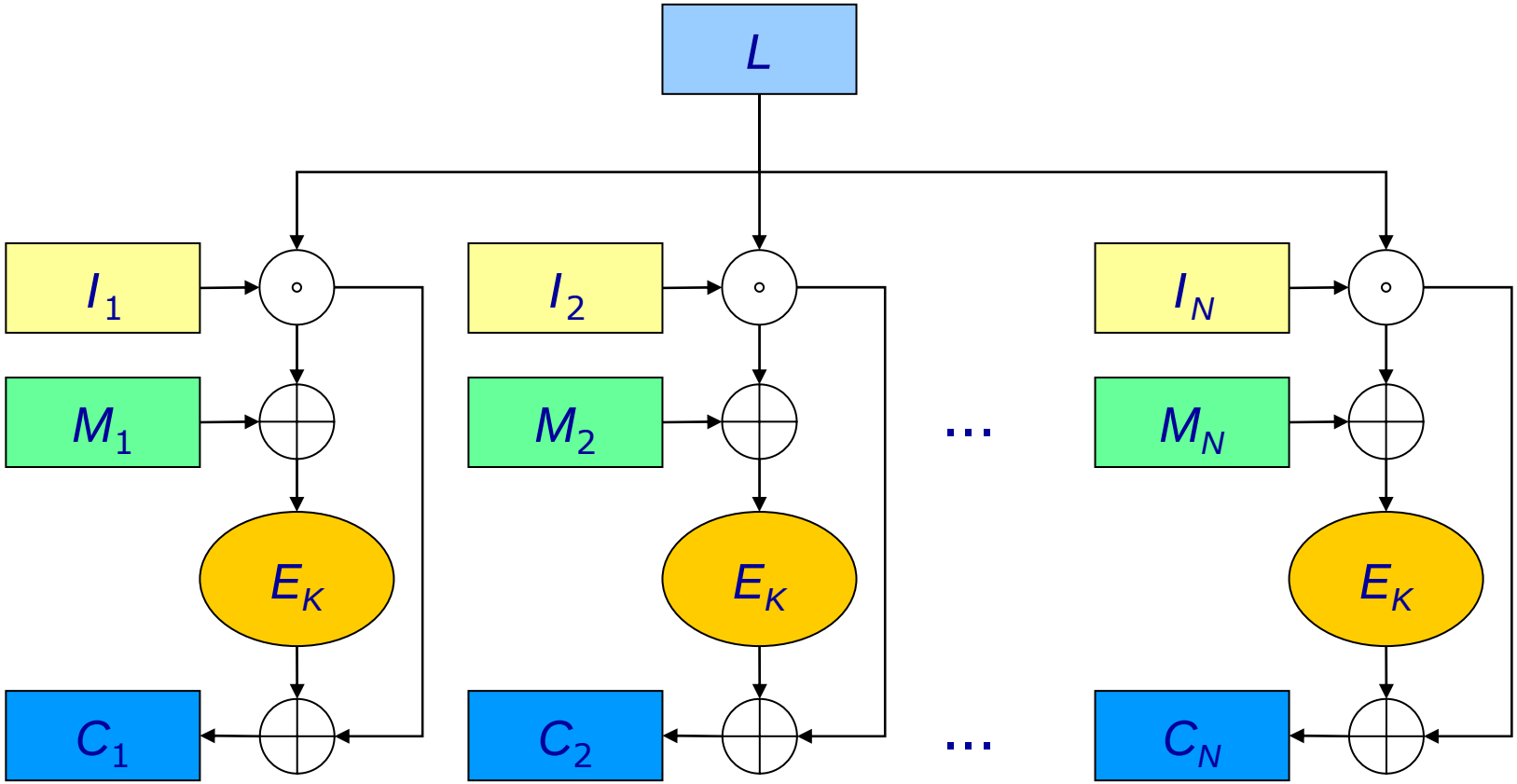
$$C \leftarrow E_K(M \oplus Z) \oplus Z$$

Modos LRW e XTS

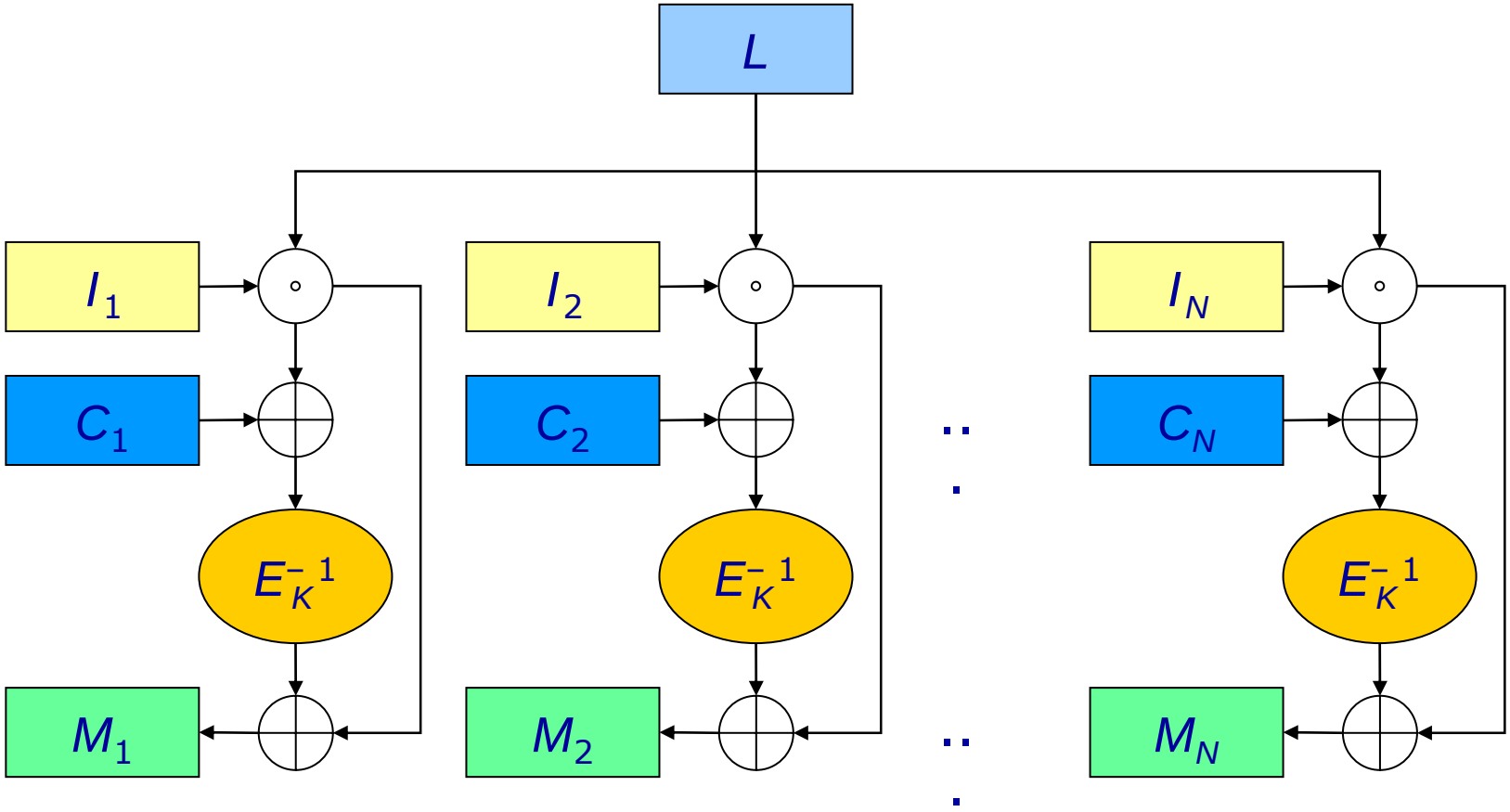
- Modo LRW original:
  - Operações em GF(2^n)
  - offset(L, I) $\leftarrow L \cdot I$
  - $L \cdot I = (L \cdot (I \text{ div } x^n)) \oplus (L \cdot (I \text{ mod } x^n))$
  - Overhead: Uma multiplicação por setor de m blocos e m multiplicações por chave

- Modo XTS (IEEE Std 1619:2007):
  - offset(L, I) $\leftarrow E_L(I \text{ div } m) \oplus x^{I \bmod m}$
  - Se $m = n$, $x^{I \bmod m}$ consiste de um único bit '1' na posição $0 \leq I < n$
  - Overhead: Uma cifração por setor de m blocos

Modo LRW - Encriptação



Modo LRW - Decriptação



Epílogo

Seleção de Chave

- Possuindo um algoritmo simétrico seguro, é possível manter a confidencialidade de uma mensagem?
- ... Sim, se uma chave adequada for escolhida.
- *O que é uma chave "adequada", e como escolher uma?*
- *O que acontece se a mensagem for alterada em trânsito?*

Apêndice

Multiplicação em GF(2^n)

- Objetivo: calcular $L \cdot I$ para L fixo, com $L, I \in \mathrm{GF}(2^n)$, sem recorrer a aritmética pesada no corpo finito.
- Importante para LRW, GCM e possivelmente outros algoritmos.
- 1ª observação: fácil calcular $P_k = L \cdot x^k$.
- Motivo: $P_0 = L$; $P_k = P_{k-1} \cdot x$ para $k > 0$.

Multiplicação em GF(2^n)

- 2ª observação: conhecendo $P_k = L \cdot x^k$ para $k = 0, \ldots, m-1$, é fácil calcular $L \cdot S$ para $\forall S \in \mathrm{GF}(2^n)$ com $\deg(S) < m$.

- Motivo: $S = S_{m-1}x^{m-1} + \ldots + S_1 x + S_0 \Rightarrow$
  $L \cdot S$
  $= L(x)S_{m-1}x^{m-1} + \ldots + L(x)S_1 x + L(x)S_0$
  $= S_{m-1}P_{m-1} + \ldots + S_1 P_1 + S_0 P_0$,
  onde $S_k \in \{0, 1\}$.

Multiplicação em GF(2^n)

- Estratégia básica: usando uma cifra de bloco de n bits, construir uma tabela contendo os polinômios $P_k = L \cdot x^k$ para $k = 0, \ldots, n-1$, e calcular $L \cdot I$ como soma ponderada desses valores:

$$L \cdot I = \bigoplus_{k=0}^{n-1} I_k P_k,$$

lembrando que $I_k \in \{0, 1\}$.

Multiplicação em GF(2^n)

- A estratégia básica pré-calcula n polinômios e executa n adições em GF(2^n) para calcular cada produto $L \cdot I$.

- *Trade-off* para $m \mid n$: construir n/m tabelas de 2^m elementos cada uma, e calcular:

$$L \cdot I = \bigoplus_{k=0}^{n/m-1} I_k \cdot x^{mk} = \bigoplus_{k=0}^{n/m-1} P_k[I_k],$$

onde $I_k \in \mathrm{GF}(2^m)$ e $P_k[u] = u(x) \cdot x^{mk}$.

Multiplicação em GF(2^n)

- Exemplo para AES-128:
  - Estratégia básica (128×16 = 2 KiB):

$$L \cdot I = \bigoplus_{k=0}^{127} (I_k = 1 \ ? \ P[k] \ : \ 0).$$

  - *Trade-off* para m = 8 (16×256×16 = 64 KiB):

$$L \cdot I = \bigoplus_{k=0}^{15} P[k][I_k].$$