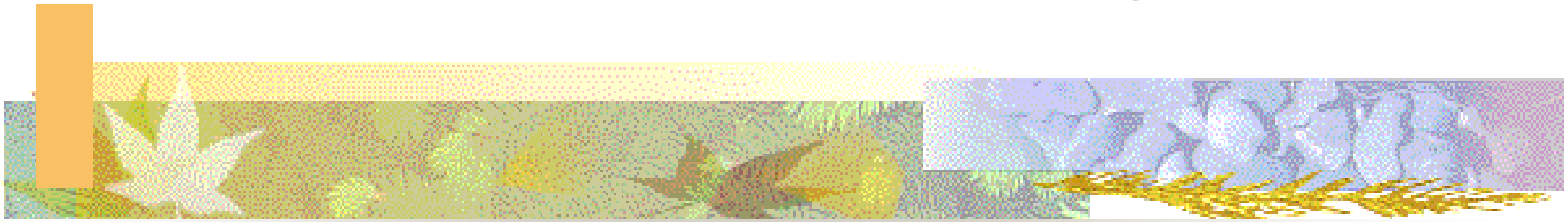
Introdução à Criptografia



Profa. Yeda

Aula 04 – Criptografia Simétrica: Advanced Encryption Standard (AES)

Cap. 4 Christof Paar & Jan Pelzl

Cap. 5 Stallings

Origem

- 1999 – NIST (FIPS PUB 46-3)
  - DES só deveria ser utilizado em sistemas legados
    - Há ataque teórico que pode quebrá-lo
    - Há ataque de busca exaustiva da chave demonstrado
  - 3DES deveria ser utilizado em seu lugar
    - Mas é lento e trabalha com blocos pequenos
- 1997 – NIST pediu propostas de cifradores
  - 15 candidatos foram aceitos em Jun/1998
  - 5 permaneceram na lista em Ago/1999
  - Rijndael foi selecionado como o AES em Out/2000
- 2001/Nov – NIST emitiu o FIPS PUB 197 (AES)

Requisitos para o AES

- Cifrador de blocos simétrico
- Dados de 128-bit, Chaves de 128/192/256-bits
- Mais forte e rápido que o Triple-DES
- Tempo de vida ativa de 20-30 anos
- Especificação completa e detalhes de projeto
- Implementações em C e Java
- NIST liberou todas as submissões e análise dos desclassificados.

Critério de Avaliação do AES

- Critério Inicial:
  - Segurança – esforço para criptoanálise prática
  - Custo – eficiência computacional
  - Características do algoritmo e implementação
- Critério Final
  - Segurança geral
  - Fácil implementação em software e hardware
  - Ataques de implementação
  - Flexibilidade (de/cifrar, usar chave, outros fatores)

Lista Reduzida do AES

- Após teste e avaliação – Ago/99:
  - MARS (IBM) - complexo, rápido, alta margem seg.
  - RC6 (USA) - muito simples e rápido, baixa margem seg.
  - Rijndael (Belgium) - claro, rápido, boa margem seg.
  - Serpent (Euro) – lento, claro, muito alta margem seg.
  - Twofish (USA) - complexo, muito rápido, alta margem seg.
- Visto o contraste entre algoritmos com
  - Rodadas pouco complexas **x** muito simples
  - Quais refinaram cifradores existentes **x** novas propostas

The AES Cipher - Rijndael

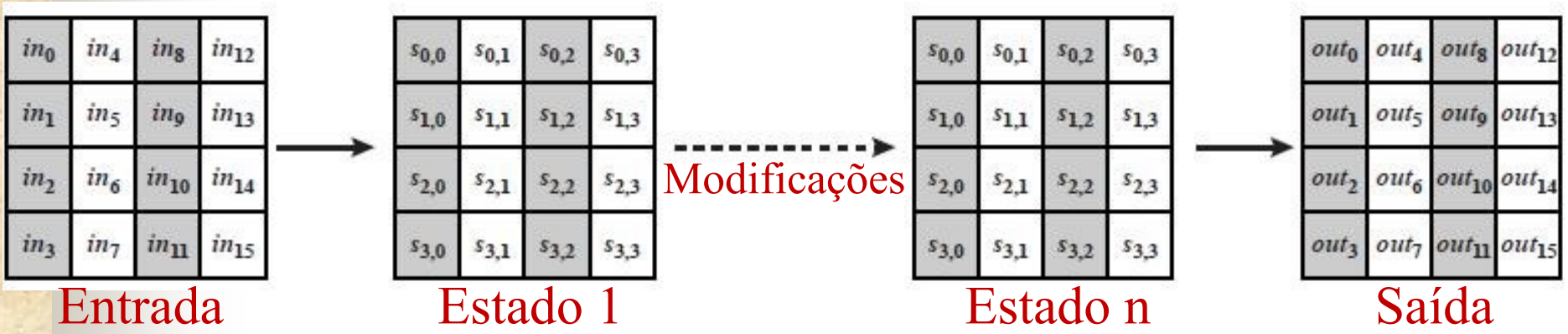
- Projetado pelos Belgas Rijmen-Daemen
- Utiliza dados de 128bits e chaves de 128/192/256 bits
- Cifrador **iterativo** ao invés de estrutura de **feistel**
  - processa dados como blocos de 4 colunas de 4 bytes
  - opera sobre todo o bloco de dados toda rodada
- Projetado para ser:
  - resistente contra ataques conhecidos
  - Velocidade e código sólido sobre muitas CPUs
  - Simplicidade de projeto

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

Estrutura Geral do Rijndael

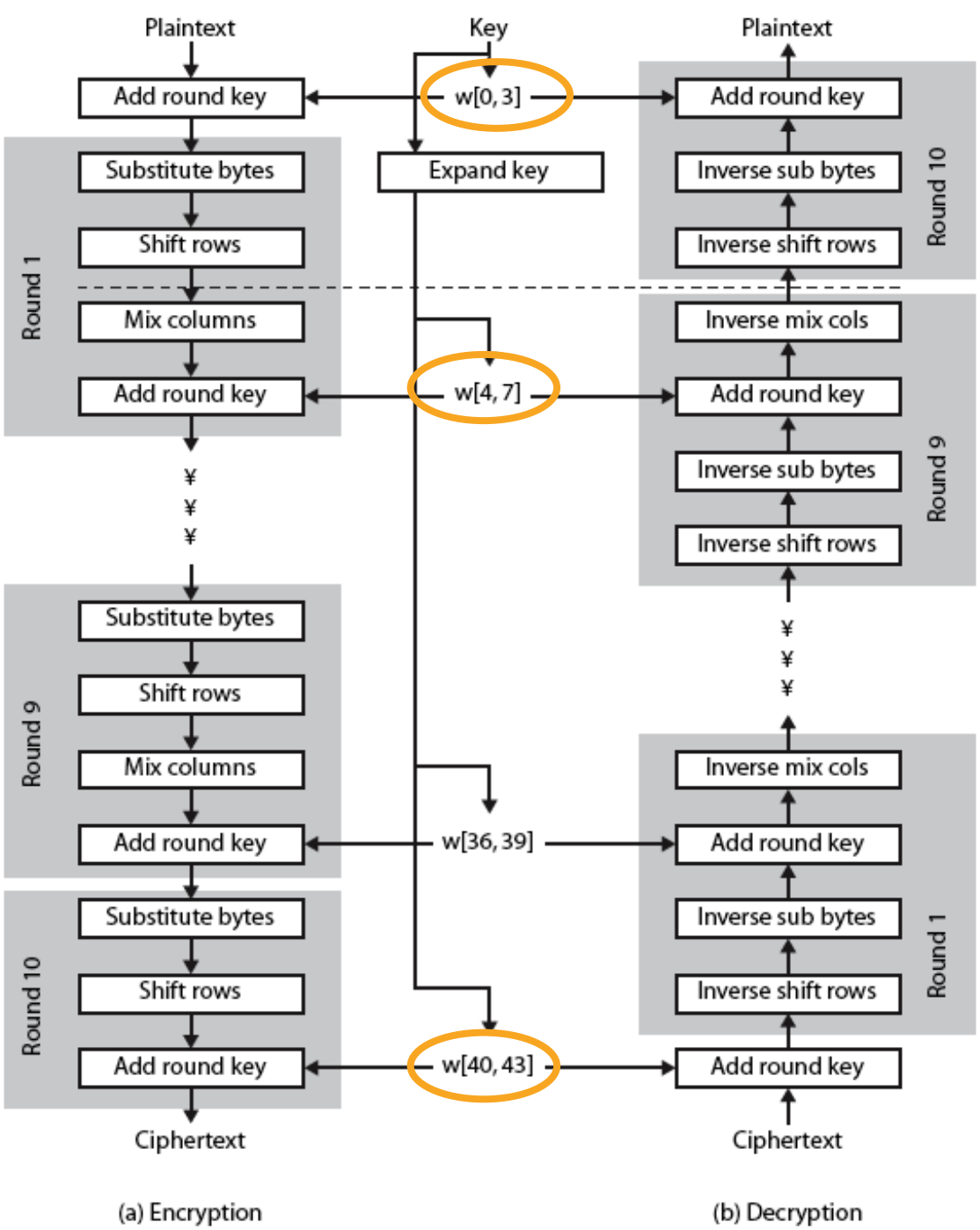
- Blocos de dados em 4 colunas de 4 bytes (128 bits)
- Chave expandida para matriz de palavras
- Tem 10/12/14 rounds p/ chaves de 128/192/256 bits:
  - substituição de byte (1 S-box usada sobre todo byte)
  - deslocamento de linhas (permuta bytes entre grupos/colunas)
  - mistura colunas (subs usando matriz de grupos)
  - Adiciona chave de rodada (XOR)
- XOR inicial com a chave & último round incompleto

AES – Entrada e Vetor de Estado

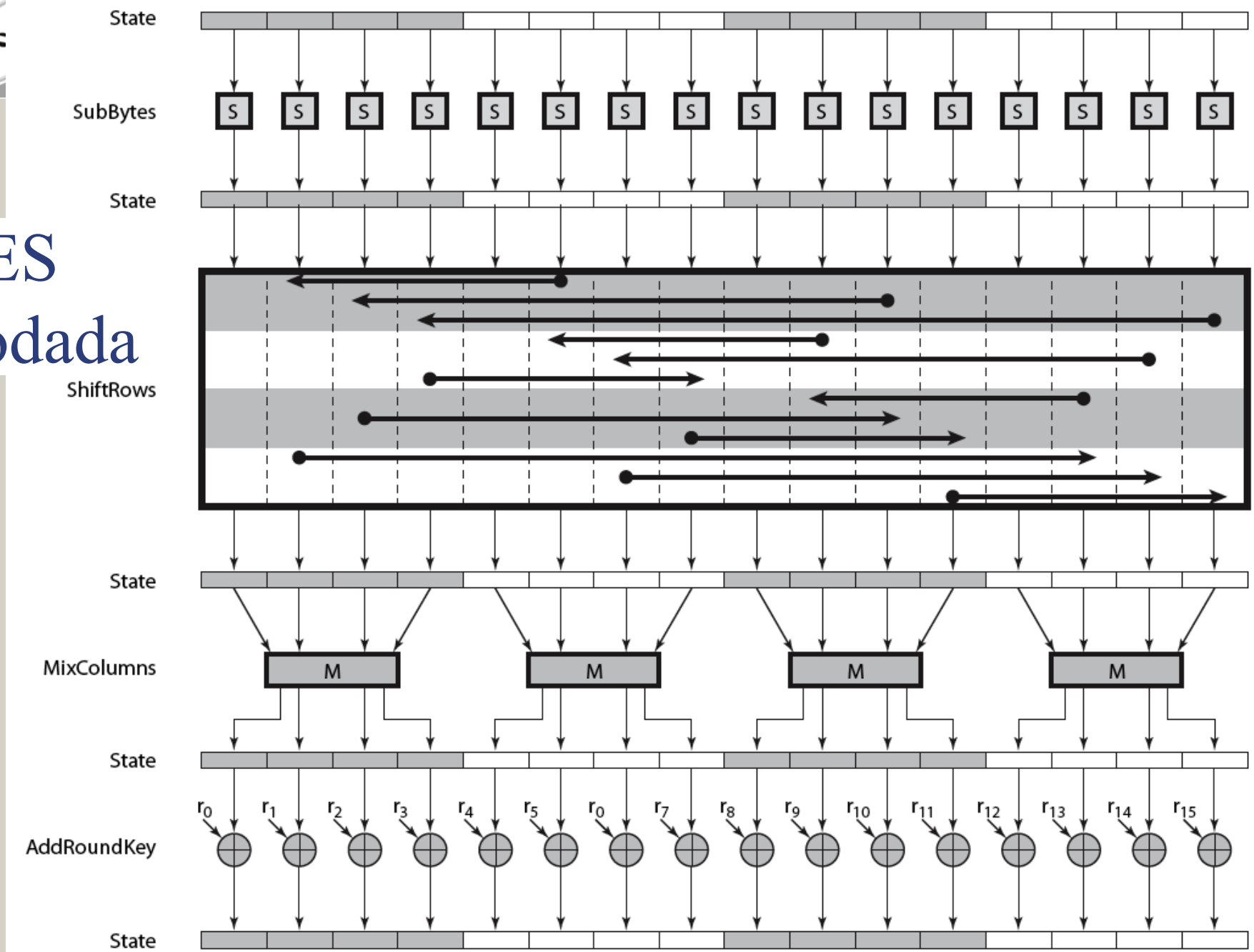


AES – Chave e Chave Expandida

Rijndael


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
¥
¥
¥
Substitute bytes
Shift rows
Inverse mix cols
Round 9
Mix columns
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Shift rows
Inverse shift rows
Round 10
Round 1
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

AES Rodada


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

Substituição de Bytes

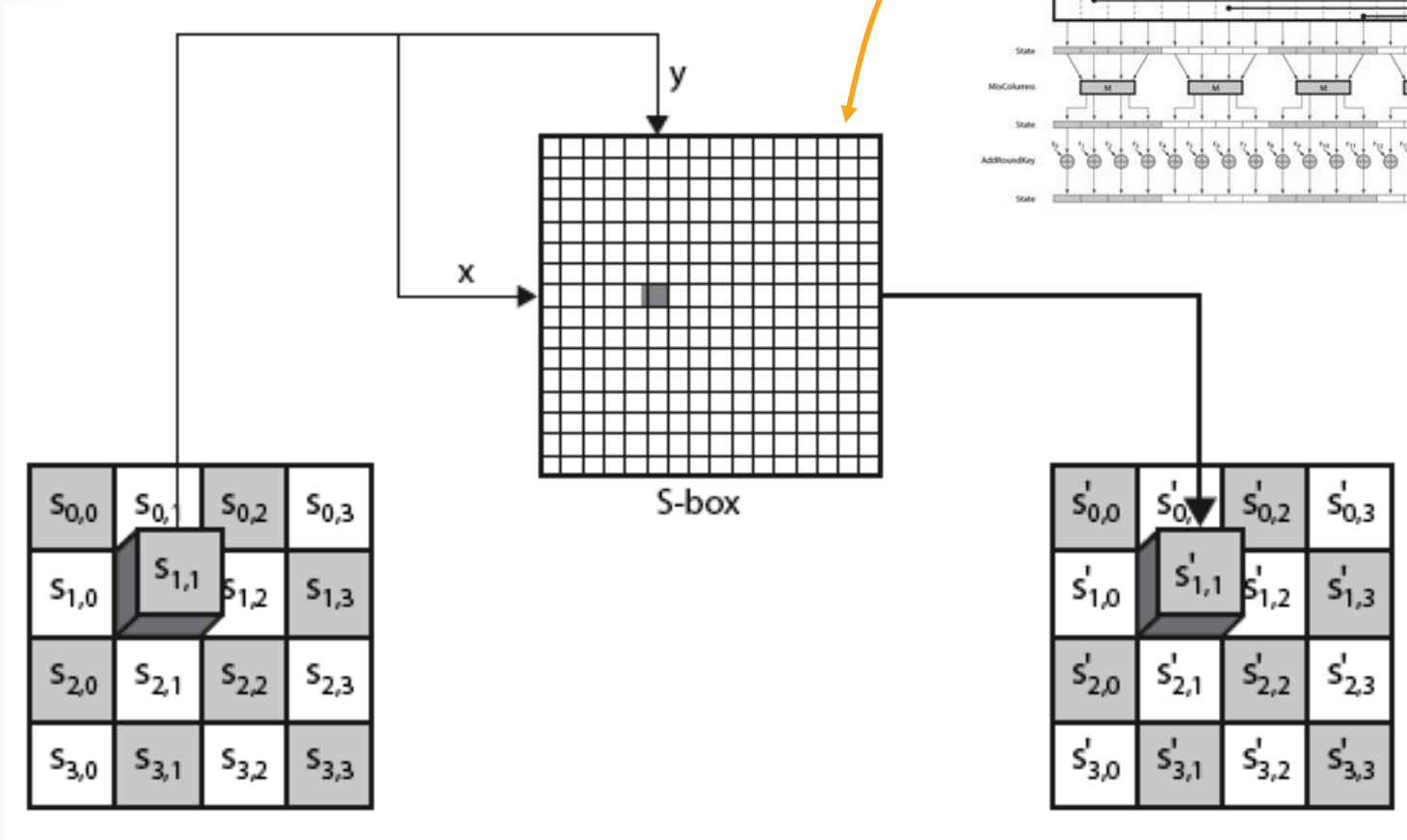


Tabela S-Box no arquivo CH05_V.pdf

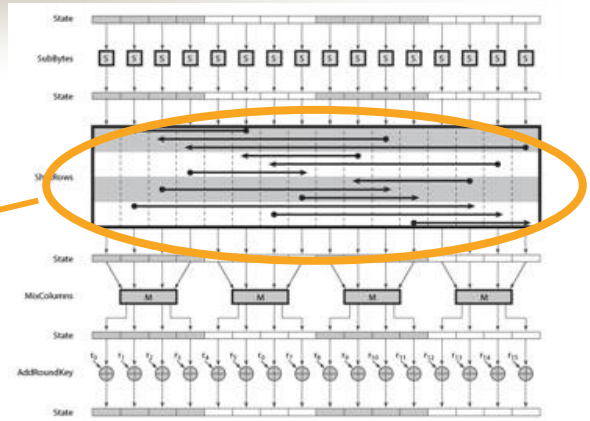
Substituição de Byte

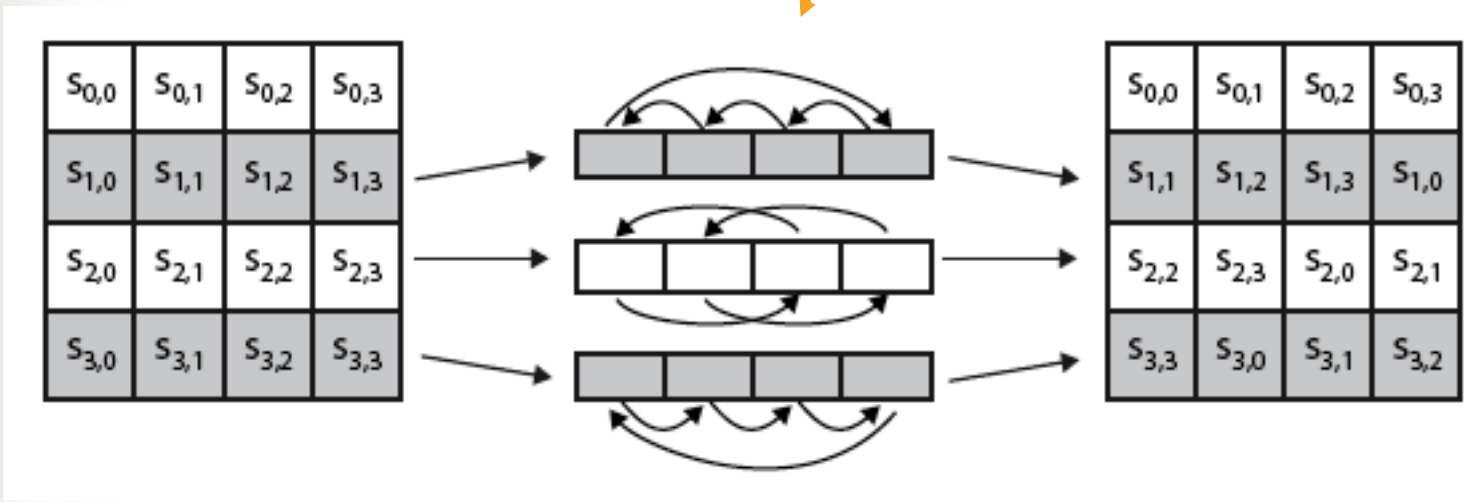
- Substituição simples de cada byte
- Usa uma tabela de 16x16 bytes contendo uma permutação de todos os 256 valores 8-bit
- Cada byte de estado é trocado pelo byte indexado pela linha (4-bits esquerda) e coluna (4-bits direita)
  - Ex.: byte {95} é trocado pelo byte na linha 9 coluna 5
  - Que tem valor {2A}
- S-box construído usando transformação definida de valores em GF(2^8) (Corpo Finito)
- Projetado para ser resistente a todos os ataques conhecidos.

Substituição de Byte

- Projetado para ser resistente a todos os ataques conhecidos.
  - Baixa correlação entre entrada e saída.
  - Saída não pode ser descrita como uma função matemática simples da entrada.
  - Não pontos fixos, cuja substituição leva ao um mesmo valor ou seu oposto (negação bit a bit).
  - Função reversível, mas não auto inversa (S(a) = InvS(a))
    - S(95)={2A}, mas InvS(95)={AD}

Deslocamento de Linhas

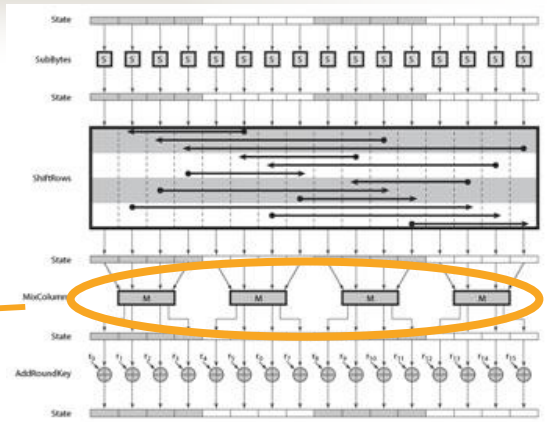


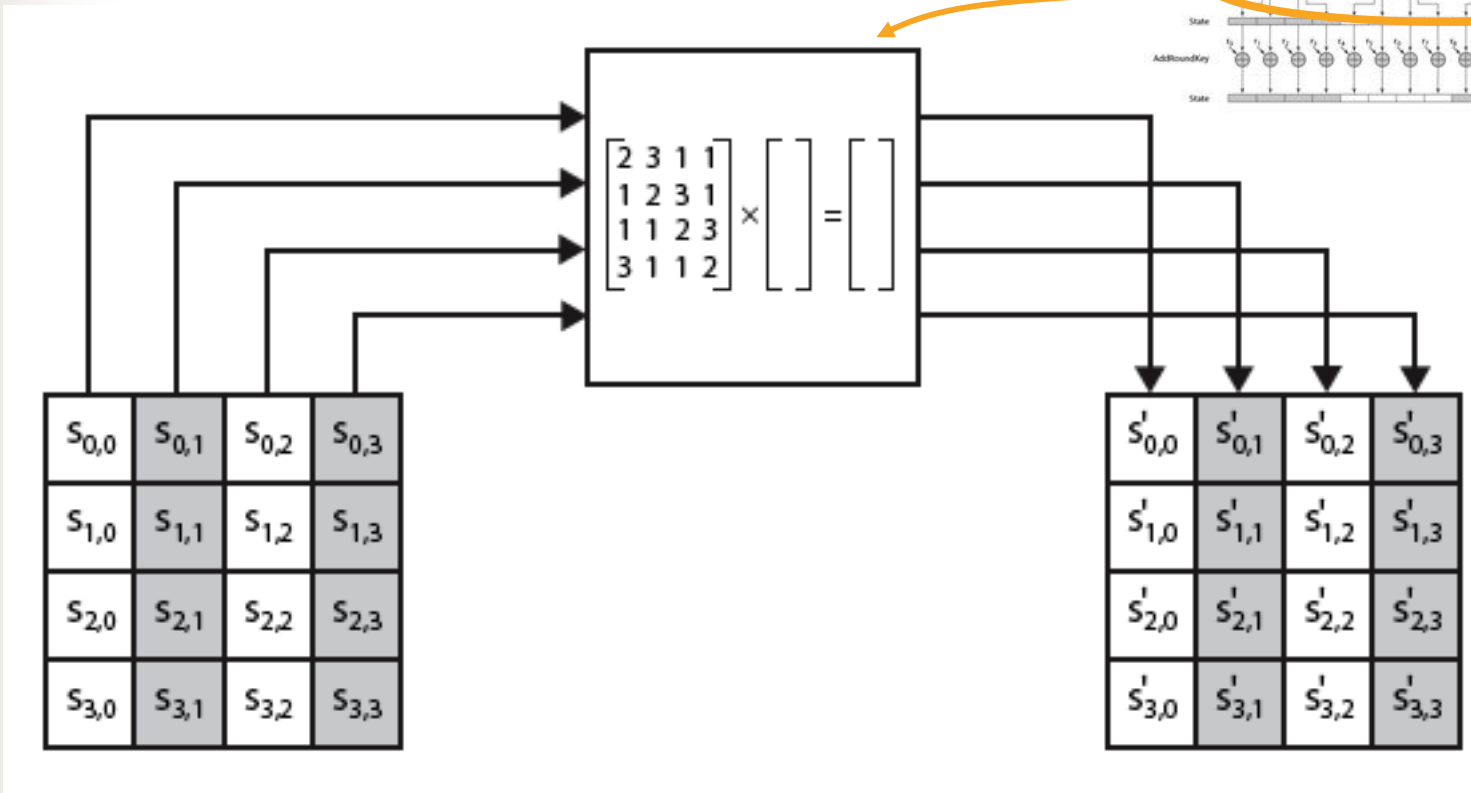


Deslocamento de Linhas

- Um deslocamento circular de byte em cada linha
  - 1ª linha não muda
  - 2ª linha desloca 1 byte circular a esquerda
  - 3ª linha desloca 2 bytes circular a esquerda
  - 4ª linha desloca 3 bytes circular a esquerda
- Para decifrar inverte-se usando deslocamento a direita
- Uma vez que o vetor de estado é processado por coluna, este passo permuta bytes entre colunas.
  - Causa o espalhamento dos bytes entre colunas.

Embaralhamento de Colunas





Embaralhamento de Colunas

- Cada coluna é processada separadamente
- Cada byte é trocado por um fator dependente de todos os 4 bytes na coluna
- Efetivamente uma multiplicação de matriz em GF(2^8) usando polinômio primo $m(x) = x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Embaralhamento de Colunas

- Pode expressar cada coluna como 4 equações
  - Derivar cada novo byte na coluna
- Decifrar requer usar uma matriz inversa
  - Com coeficientes maiores, sendo um pouco mais difícil
- Tem uma característica alternativa
  - Cada coluna é um polinomial de 4 termos
  - Com coeficientes em $GF(2^8)$
  - E polinômios multiplicados módulo (x^4+1)

Adição de Chave da Rodada

- Estado XOR com 128-bits de chave de rodada
- Inversão para decriptografia idêntica
  - Uma vez que XOR é seu próprio inverso, com as chave em ordem inversa
- Projetada para ser tão simples quanto possível
  - forma de cifra de Vernam sobre chave expandida
  - requer outros estágios para inserir complexidade / segurança

Add Round Key

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

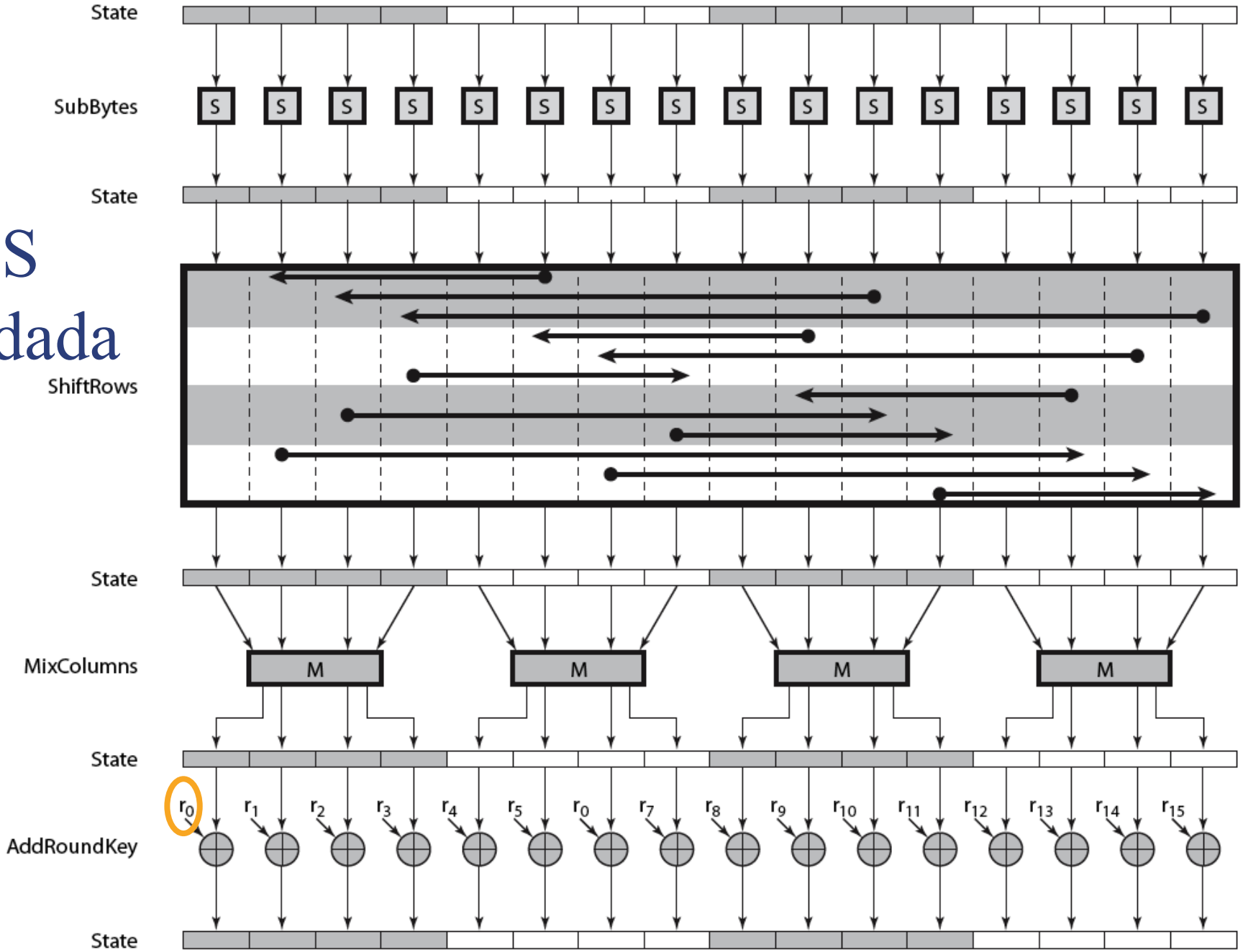
$\oplus$

w_i	w_{i+1}	w_{i+2}	w_{i+3}

$=$

$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s'_{1,0}$	$s'_{1,1}$	$s'_{1,2}$	$s'_{1,3}$
$s'_{2,0}$	$s'_{2,1}$	$s'_{2,2}$	$s'_{2,3}$
$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

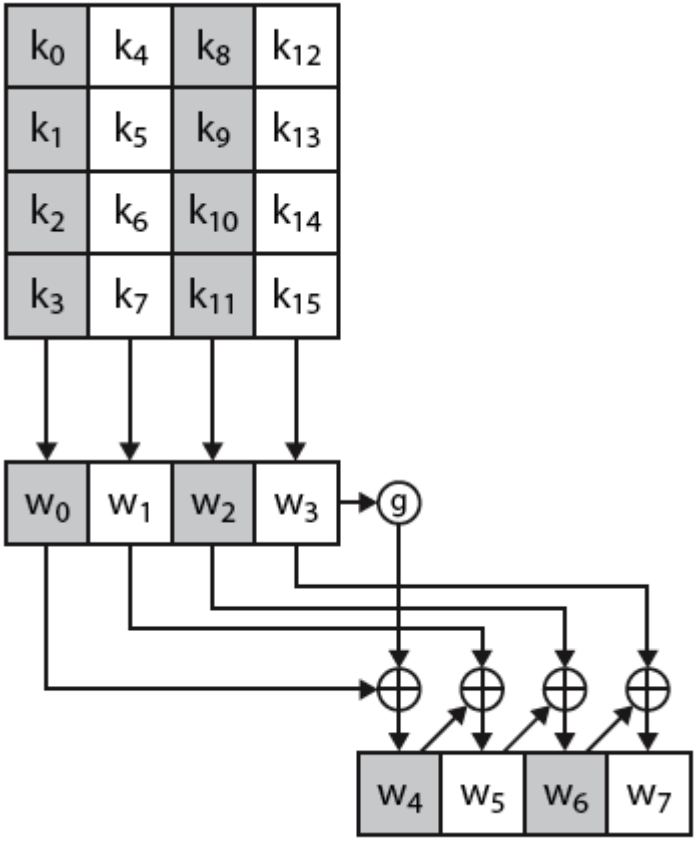
AES Rodada


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0
r1
r2
r3
r4
r5
r0
r7
r8
r9
r10
r11
r12
r13
r14
r15
State

Expansão de AES

- Expande 128-bits (16-byte) de chave em vetor de 44/52/60 palavras de 32-bits
- Inicia copiando as 4 primeiras palavras
- A cada rodada cria palavras que dependem das palavras da chave anterior e posições anteriores (W_{i-1})
  - A 1ª palavra tem operação g {rotação + S-box + XOR sobre constante de rodada} sobre (W_{i-1}), antes de fazer XOR com palavra da chave anterior (W_{i-4})
  - Nos demais casos, apenas faça XOR de $(W_{i-4}) \oplus (W_{i-1})$

Expansão de AES



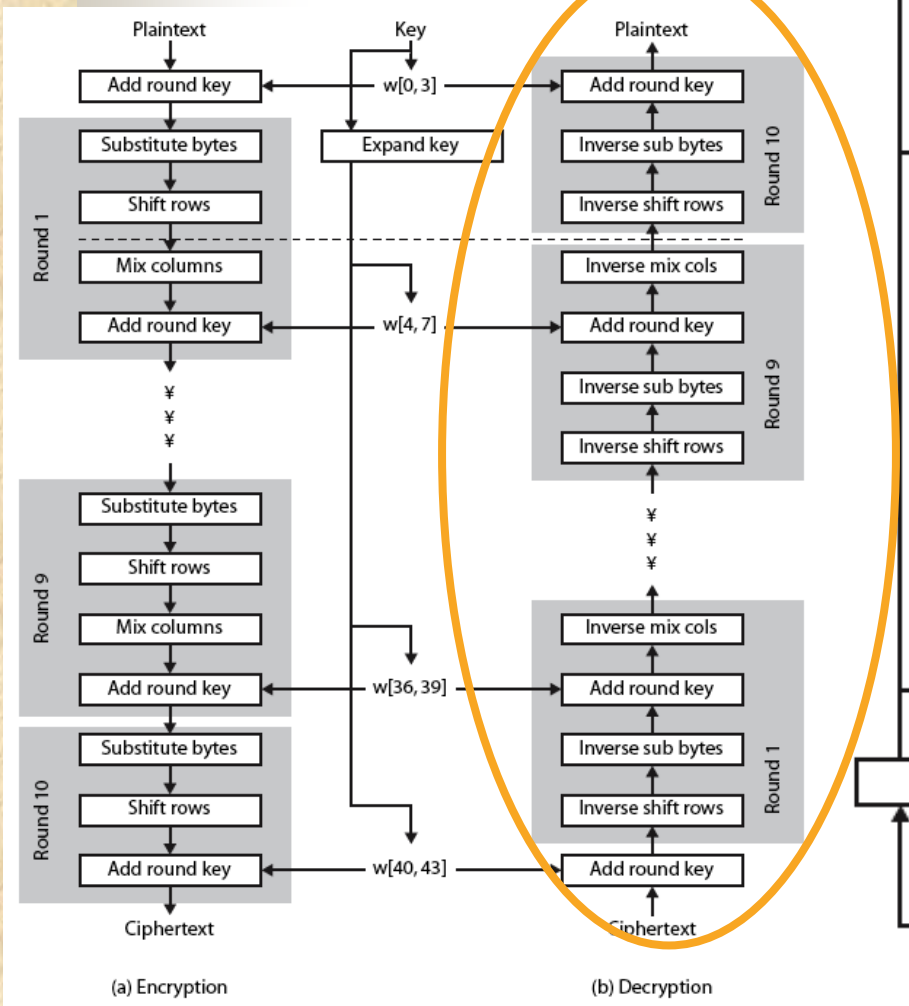
Expansão de AES

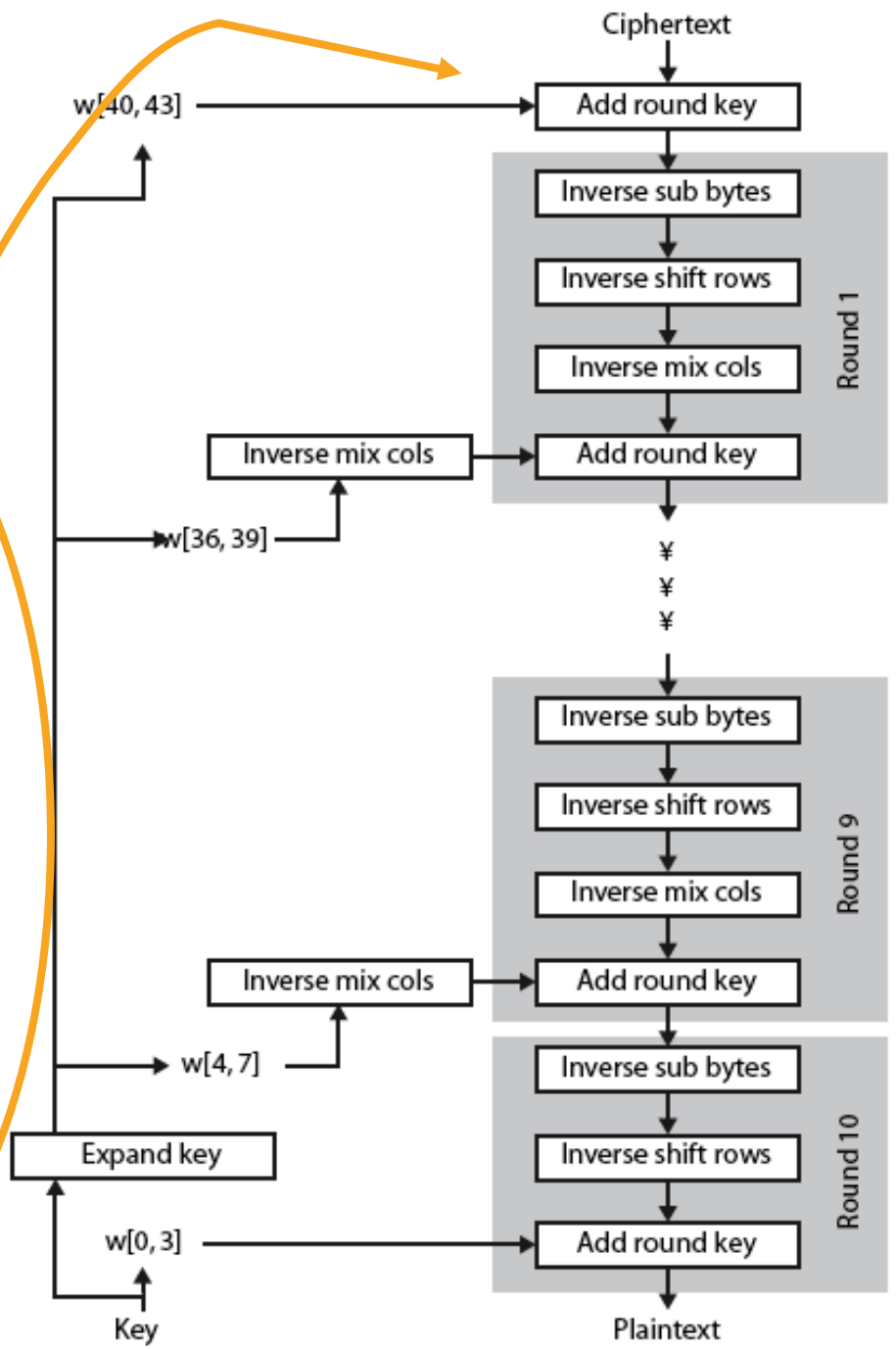
- Projetado para resistir a ataques conhecidos
- Critério de projeto incluiu
  - Conhecer parte da chave é insuficiente para saber mais partes
  - Transformação inversível
  - Rápido sobre diversas CPU's
  - Usa constante de rodada pra quebrar simetria
  - Os bits da chave são espalhados na chave de rodada
  - Não linearidade suficiente para impedir análise
  - Descrição simples

Decriptografia AES

- Não é idêntica a criptografia, uma vez que os passos são invertidos
- Mas pode-se fazer um decifrador inverso equivalente com passos do cifrador
  - mas com funções inversas em cada passo
  - e diferente escalonamento de chave
- Baseia-se em trocas sem alterar o resultado
  - Troca Subst. bytes & Deslocamento de linha
  - Troca Emb. coluna & Adição chave (alterada)

Decriptografia AES





Aspectos de Implementação

- Implementação eficiente para CPUs de 8-bits
  - Usando tabelas de substituição de 256 entradas
  - Deslocamento e XOR são operações simples
  - Embaralhamento de colunas requer multiplicação de matriz em GF(2^8), que pode ser simplificado pelo uso de tabelas de busca e XOR's

Aspectos de Implementação

- Implementação eficiente para CPUs de 32-bits
  - Redefine passos para usar palavras de 32-bits
  - Pode précomputar 4 tabelas de 256 palavras
  - Cada coluna de rodada pode ser computada usando 4 tabelas de busca e 4 XORs
  - Com custo de 4Kb para armazenar tabelas
- Projetistas acreditam que implementação eficiente foi um fator chave para a seleção do cifrador AES