# Criptografia com chaves simétricas

SEGA4 – Segurança da Informação

## Objetivos

- Apresentar os principais conceitos de criptografia com chaves simétricas.
- Apresentar os algoritmos DES, Triple DES e AES e RC4.

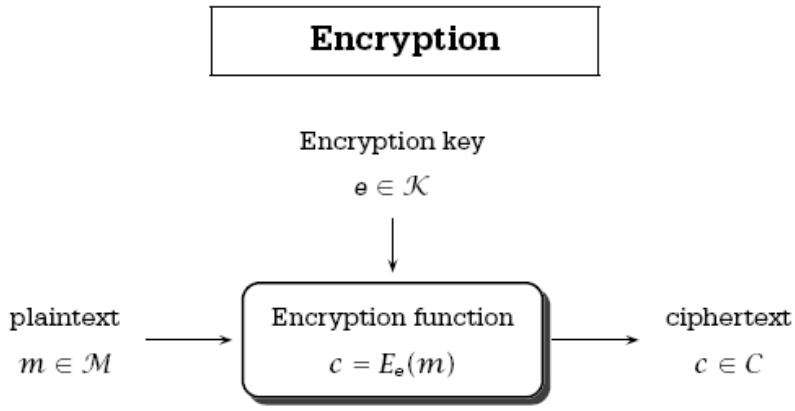
## Histórico

- A partir do desenvolvimento da computação digital, a criptografia e a cripto-análise se tornaram disciplinas da matemática.
- As operações de criptografia são operações sobre bits de informação e não mais operações sobre textos.

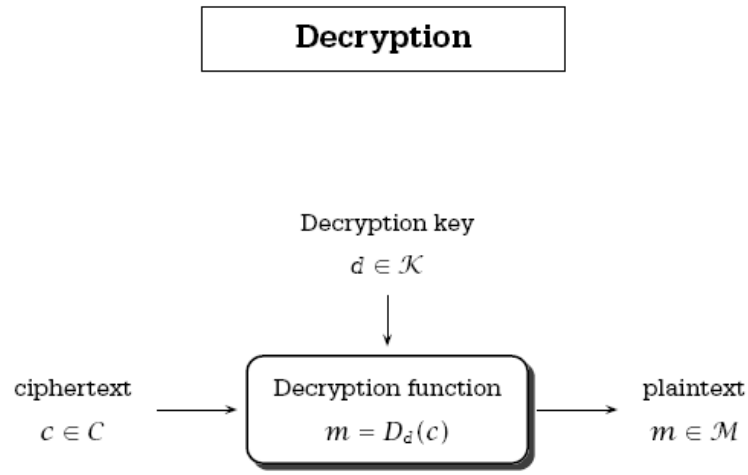
## Classificação de algoritmos de acordo com tipos de chaves criptográficas

- Sem chaves
  - Funções Hash
    - MD5, SHA1
  - Funções de uma única direção (one-way).
  - Sequências aleatórias.
- Chaves secretas ou simétricas
  - DES, 3DES, AES.
- Chaves públicas ou assimétricas
  - RSA, El-Gammal, DSA.

# Encriptação (Cifragem)




# Decriptação (Decifração)



## Definições

- Criptografia simétrica
  - As chaves ***e*** e ***d*** são essencialmente as mesmas.
- Criptografia assimétrica
  - Conhecido a chave ***e***, é computacionalmente inviável a descoberta da chave ***d***.
  - A chave ***e*** pode ser pública.

## Criptologia

- Criptologia é o ramo da matemática que engloba a criptografia e a criptoanálise.
  - Criptoanálise  aborda técnicas para decifrar informações cifradas sem o conhecimento da chave secreta.
  - Criptografia aborda técnicas para cifrar informações.

## Princípios de Design contra criptoanálise

- Para se proteger contra a criptoanálise, pelo menos uma das seguintes condições deve ser verdadeira:
  - Textos cifrados devem ter a menor regularidade estatística possível.
  - Para $C=E_e(M)$:
    - Conhecidos C e M deve ser muito difícil encontrar e.
    - Conhecido somente C, deve ser muito difícil recuperar M.

## Princípio de Kerckhoffs

- A implementação interna do sistema criptográfico deve ser completamente conhecida pelo atacante.
- Engenharia reversa pode facilmente identificar algoritmos.
- Um design seguro é aberto.

## Princípios de Shannon:

- Difusão (transposição ou permutação) – espalhar redundância.
- Confusão (substituição) – manter a relação (chave,texto cifrado) complexa.
- Idéia central: cifrar como se estivesse usando um one-time pad.

## Difusão

- Objetivo: Espalhar redundâncias de M em C.
- Benefícios:
  - É necessário um C muito longo para obter características estatísticas.
- Princípios de implementação:
  - Fazer cada $c_i$ depender de muitos $m_j$.
  - Usar transposição.

## Confusão

- Objetivo:
  - Tornar complexo o relacionamento entre a chave “e” com o texto cifrado C.
- Benefícios:
  - Torna difícil:
    - Limitar o espaço de busca de chaves observando C.
    - Obter partes de M a partir de partes da chave.
    - Obter a chave a partir de C e M conhecidos.
- Implementação:
  - Fazer cada $c_i$ dependente de todos os $e_j$ da chave E.
  - Garantir que C é não linear em partes da chave E.
- Efeitos negativos:
  - Velocidade, consumo de energia, consumo de memória, etc.

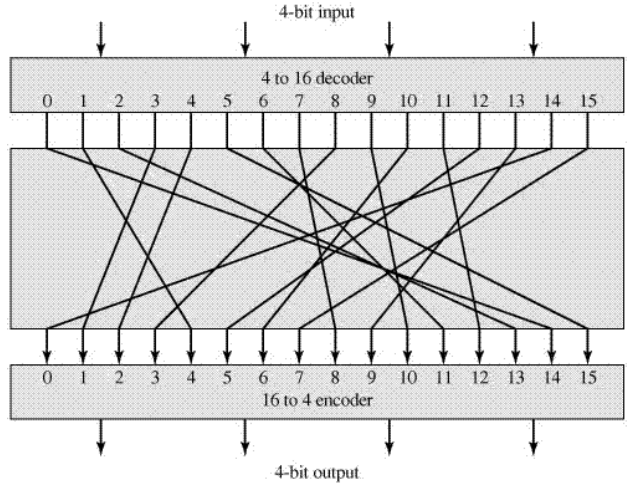
## Efeito avalanche

- Um efeito desejável da difusão e confusão é o efeito avalanche:
  - Uma pequena alteração na mensagem M ou na chave E resulta em uma alteração imprevísivel no texto cifrado C.
  - Um bit alterado na entrada causa vários bits alterados na saída (metade).

## Cifrador em bloco

- Um cifrador em bloco é uma função que tem como entradas uma chave de n-bits e um string B de m-bits e encripta B em um string B´ de m-bits.
- Os tamanhos de m e n variam para diferentes algoritmos. Para o DES temos: m=64 e n=56.
- Blocos devem ser maiores que 64 bits para evitar ataques por dicionário (manter pares (M,C) para chaves fixas).

## Cifrador de Bloco Ideal



# Cifrador de bloco ideal

- Um cifrador de bloco ideal aplica uma cifragem por substituição. Cada valor de chave define um mapeamento reversível de uma palavra de m bits para outra palavra de m bits.

| Plaintext | Ciphertext |
|---|---|
| 0000 | 1110 |
| 0001 | 0100 |
| 0010 | 1101 |
| 0011 | 0001 |
| 0100 | 0010 |
| 0101 | 1111 |
| 0110 | 1011 |
| 0111 | 1000 |
| 1000 | 0011 |
| 1001 | 1010 |
| 1010 | 0110 |
| 1011 | 1100 |
| 1100 | 0101 |
| 1101 | 1001 |
| 1110 | 0000 |
| 1111 | 0111 |

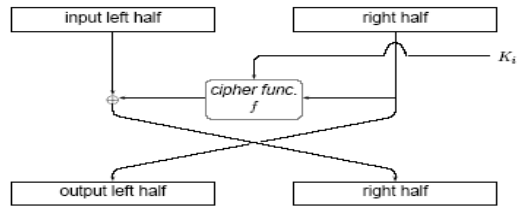
Para bloco de 4 bits a chave tem o tamanho de 4bits X16 linhas = 64 bits (número de bits da segunda coluna).

Para 64 bits a chave seria de $64 \times 2^{64}$ bits ~ $2^{70}$

# Cifrador de Feistel

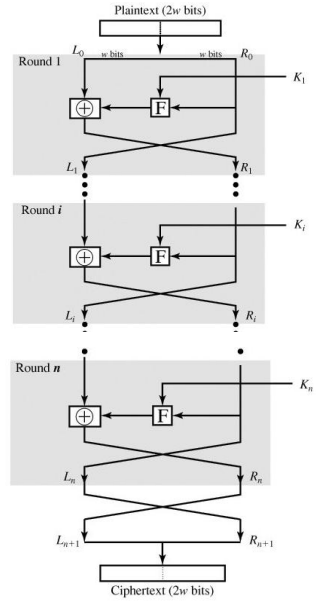
- Feistel propôs a criação de uma aproximação para o bloco ideal.

Each round $i$ treats a block in two halves:

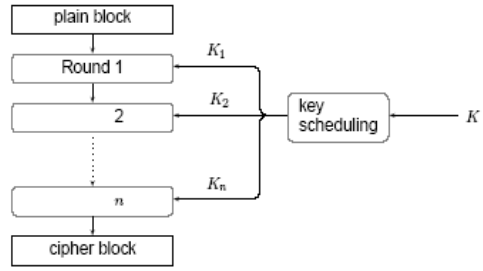



where

- *cipher function* $f$ is the same for every round.
- $f$ need not be bijective for the round to be a permutation.
- last round does not swap left and right half.



# Cifrador em blocos típico

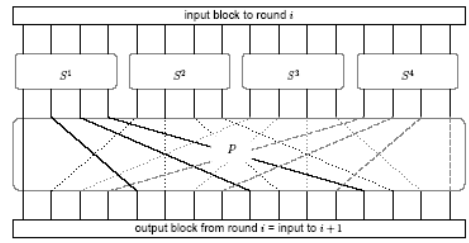



Each round $i$

- takes **subkey** $K_i$ derived from $K$ by **key scheduling**.
- contains
  - *Permutation-box* for diffusion by transposition,
  - *Substitution-box* for confusion.

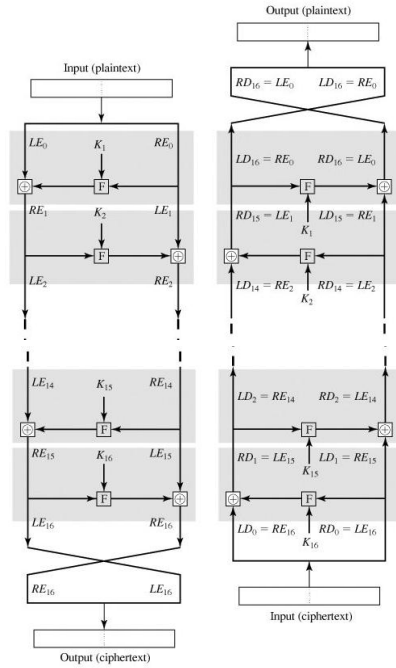
# Redes de substituição e permutação

Structure of each round $i$ (numbers here are for example only):




where

- S-boxes map 4-bit num to 4-bit num bijectively, providing confusion, nonlinearity.
- P-box transposes bits across 4-bit sub-block boundaries, providing diffusion.



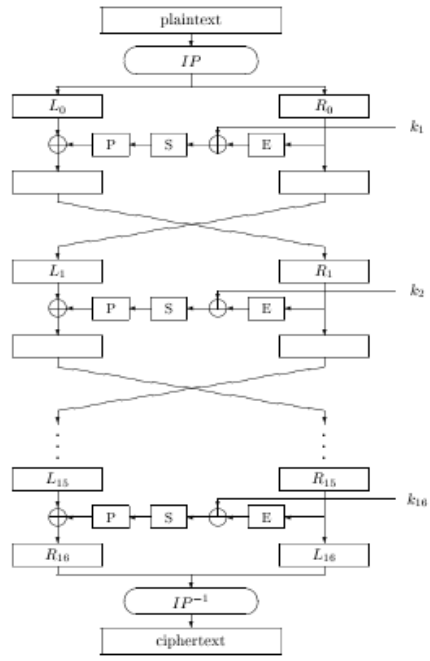

# DES – Data Encryption Standard

- Publicado em 1977 como padrão do governo americano.
- Utiliza a rede de Feistel.
- Adaptação do NSA do sistema Lucifer da IBM.
- Fundamentos foram segredos por mais de 20 anos.

## Características

- Blocos de 64 bits e chaves de 56 bits.
- Cifrador de Feistel
  - Permutação inicial.
  - Divisão em duas metades.
  - 16 iterações com operações idênticas.
  - Inversão da permutação inicial.

## DES



**(a) Initial Permutation (IP)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

**(b) Inverse Initial Permutation ($IP^1$)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

**(c) Expansion Permutation (E)**

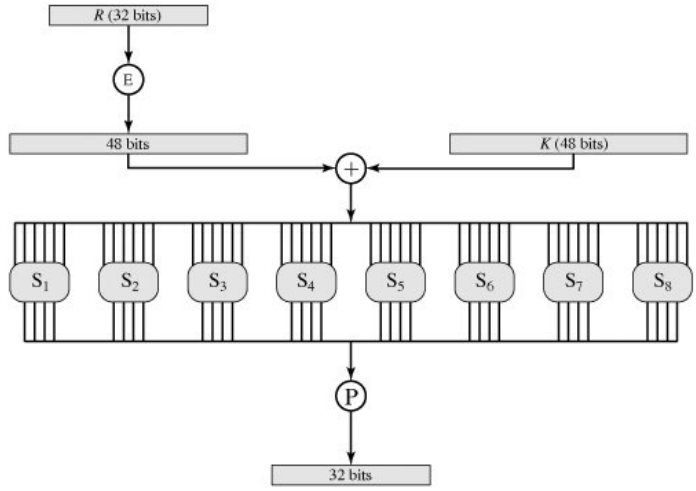
| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

**(d) Permutation Function (P)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 7 | 20 | 21 | 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 | 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 | 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 | 22 | 11 | 4 | 25 |

Figure 3.6. Calculation of F(R, K)

(This item is displayed on page 78 in the print version)




$S_1$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

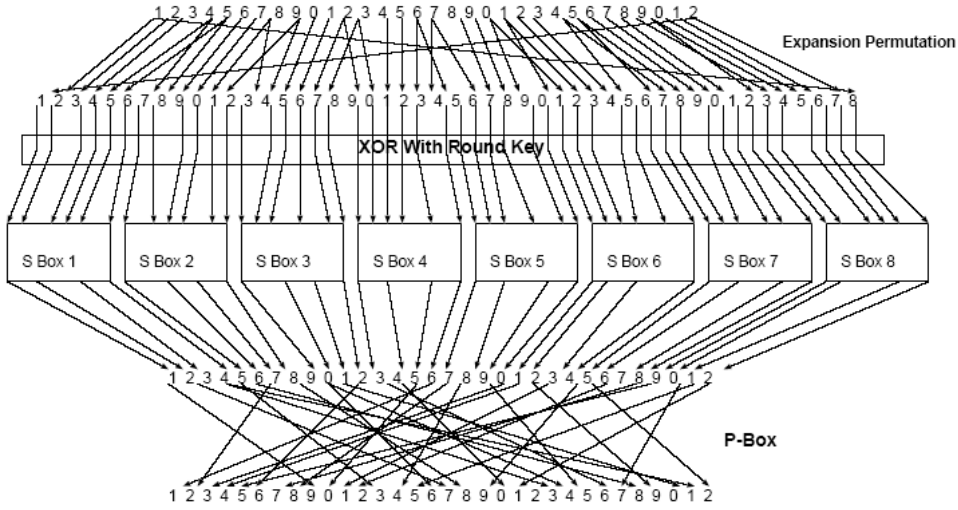
$S_2$

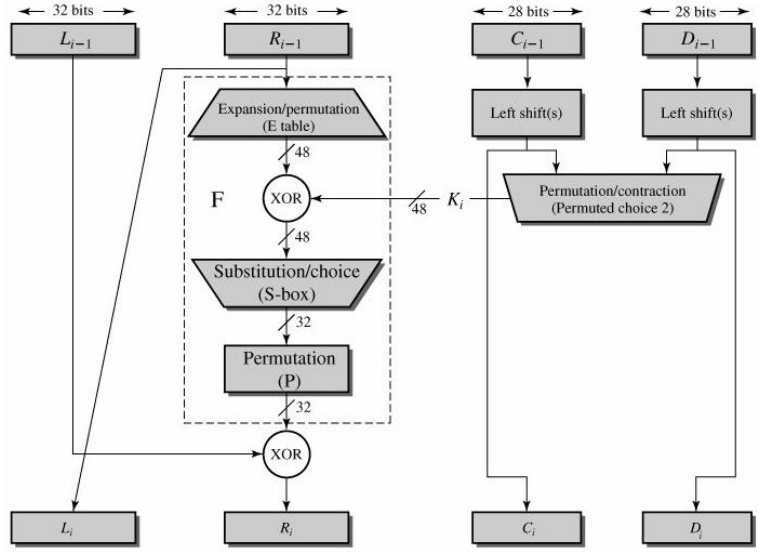
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

$S_3$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

# Iteração do DES



## Referência

- STALLINGS, William. **Criptografia e Segurança de Redes: Princípios e Práticas. 6. ed.** São Paulo: Pearson Education, 2014. ISBN 8543005892. Disponível em: https://ifsp.bv3.digitalpages.com.br/users/publications/9788543005898

## Segurança do DES

- Busca exaustiva de chaves:
  - O número de possibilidades é $2^{56}$.
- Para um Pentium 200MHZ
  - 1 PC:2000 anos.
  - 200 PCs: 10 anos.
  - 6000 PCs: 3 meses.
- Hardware (FPGA)
  - Menos de 2 anos.
- Hardware (ASIC)
  - 1 chave em 50 horas.
  - Por US$ 1 milhão -> 1 chave em meia hora.

## Cont.

- Para chaves de 40 bits (SSL, Lotus Notes, etc.)
  - 2 semanas em um PC.
  - 2 horas em uma LAN.
  - 10 minutos em um FPGA.

## Como melhorar o DES?

- Uma idéia é encriptar duas vezes com duas chaves diferentes:
  - $C= E_{k2}(E_{k1}(M))$.
- Problema:
  - $X=E_{k1}(M) = D_{k2}(C)$
  - Meet in the middle attack.

## Meet in the middle attack

- Conhecido um conjunto de pares $(M_1,C_1),(M_2,C_2)$.... Encriptados com o par de chaves $K_1,K_2$. Seguir os seguintes passos:
  - Encriptar $M_1$ com todas as $2^{56}$ chaves. Salvar todos os resultados em uma tabela.
  - Decriptar $C_1$ com todas as possíveis chaves. Para cada bloco decriptado verificar a existência na tabela anterior. Será identificado pares de chaves candidatos.
- Para cada par candidato verificar com $M_2,C_2$.

## Triple DES

- $C = E_{K1}(E_{K2}(E_{K3}(M)))$.
- Variações meet in the middle existentes, mas de análise complexa.
- Considerado seguro, mas três vezes mais lento que o DES.

## Avaliação do DES

- Durante 25 anos foi considerado seguro.
- A melhor forma de ataque é a força bruta. Hoje é um ataque viável, portanto o DES é considerado inseguro.
- Em 1997, o NIS  estabeleceu um concurso para estabelecer o AES (Advanced Encryption Standard).

## Critérios para o AES

- Cifrador em bloco: blocos de 128 bits e chaves de 128/192/256 bits.
- Resistência equivalente ao 3-DES.
- Desempenho muito superior ao 3-DES.
- Designers abdicam da propriedade intelectual.

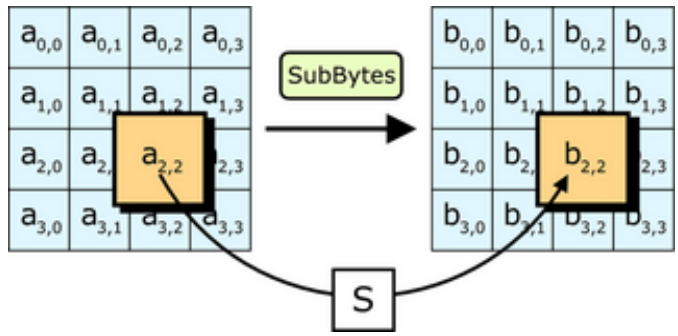
## Algoritmo Rijndael (raindau)

- Algoritmo criado por dois criptógrafos belgas: Joan Daemen e Vincent Rijmen.
- Baseado em redes de permutação e substituição.
- Alto desempenho e requer pouca memória.

## Passos do AES

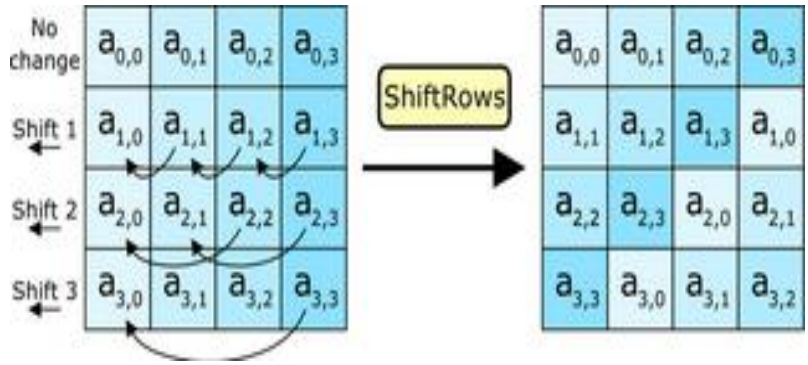
- Um bloco de 128 bits é dividido em um array de 16 bytes (Matriz 4 x 4). Em uma iteração é processado os seguintes passos:
  - Subbytes
  - ShiftRow
  - MixColum
  - AddRoundKey

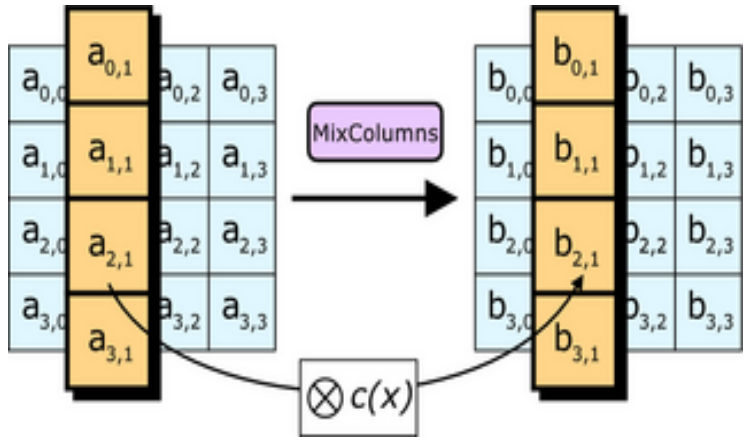
# Substituição de bytes

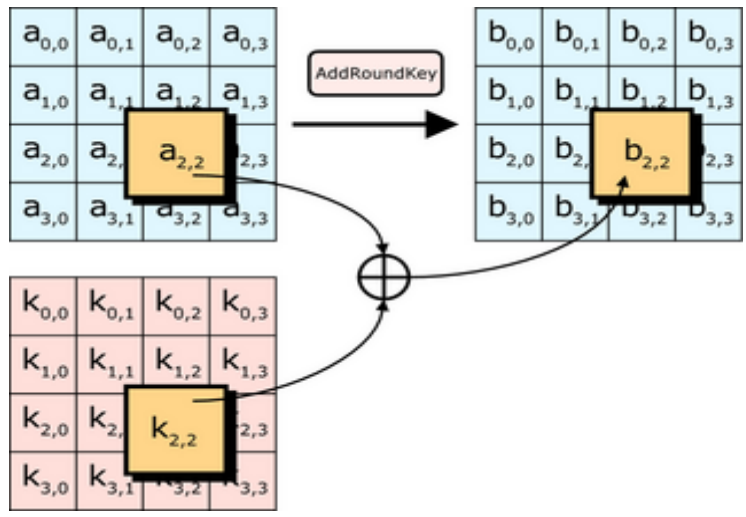



Cada byte da matriz de estados é substituída conforme uma tabela de conversão.

Um único S Box para o cifrador.



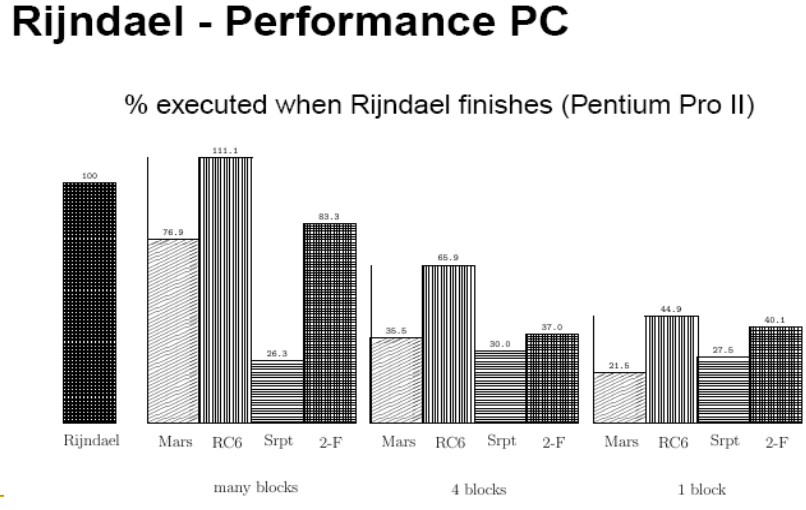

MixColumns
c(x)


AddRoundKey

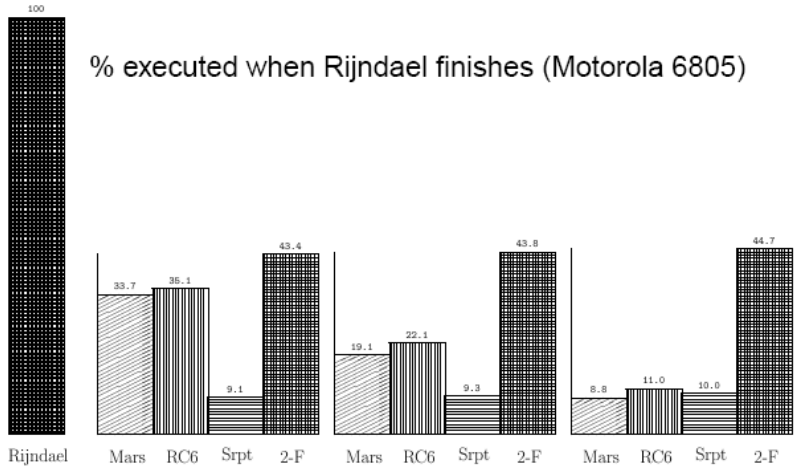
## Pseudo-código para 10 iterações

```
AddRoundKey(S,K[0]);
for (i = 1; i <= 9; i++)
  {
     SubBytes(S);
     ShiftRows(S);
     MixColumns(S);
     AddRoundKey(S,K[i]);
  }
SubBytes(S);
ShiftRows(S);
AddRoundKey(S,K[10]);
```

## Desempenho do Rijndael



## Rijndael - Performance Smartcard

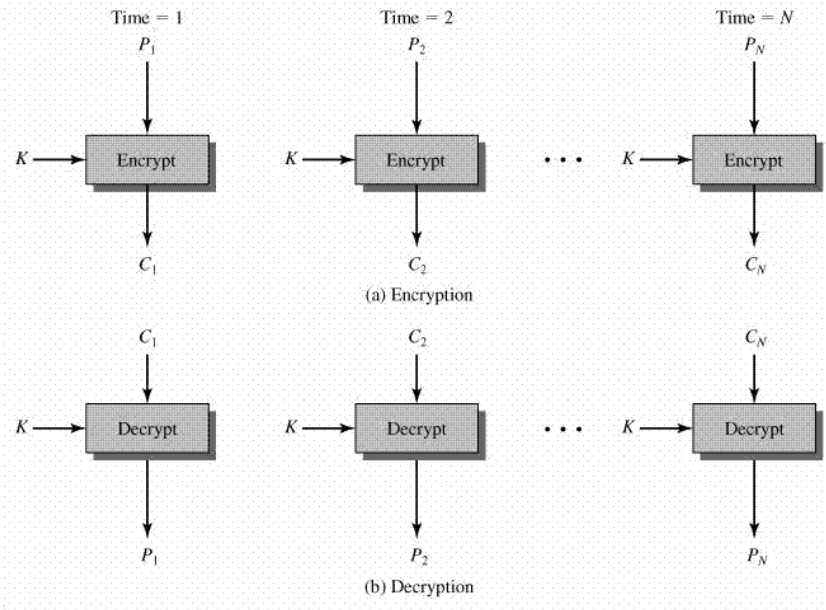



## Modos de operação

- Para mensagens maiores que o tamanho de bloco foram definidos quatro modos de operação que podem ser utilizados por qualquer cifrador de bloco.
  - ECB – Eletronic Code Book.
  - CBC – Cypher Block Chaining
  - OFB – Output Feedback
  - CFB – Cipher Feedback

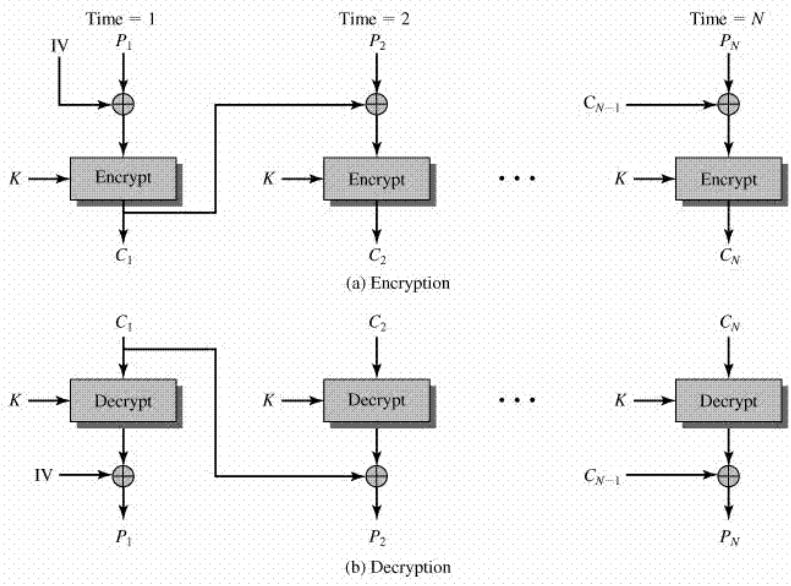
# ECB

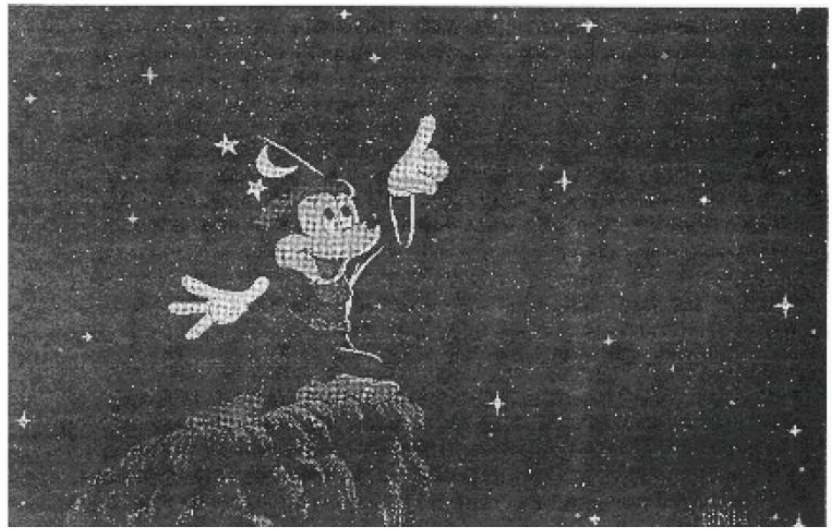
- Cada bloco de 64 bits do texto plano é codificado de forma independente com a mesma chave.
- Utilizado somente para mensagens curtas tais como transmissão de chaves.



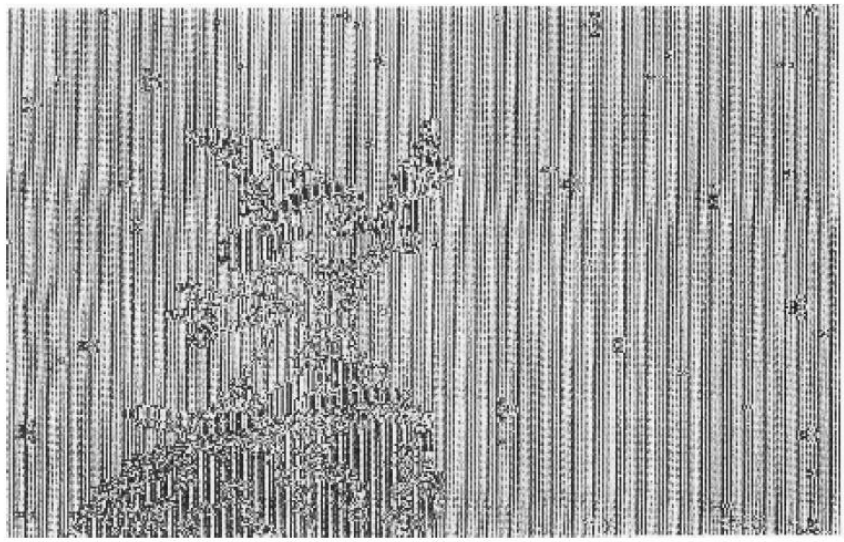
# CBC

- A entrada para o algoritmo de encriptação é o XOR do próximo conjunto de 64 bits da mensagem e os 64 bits do texto cifrado precedente.
- Aplicado para transmissão de mensagens em bloco e para autenticação.
- Um oponente que conhece partes do texto plano pode cortar e modificar partes da mensagem.

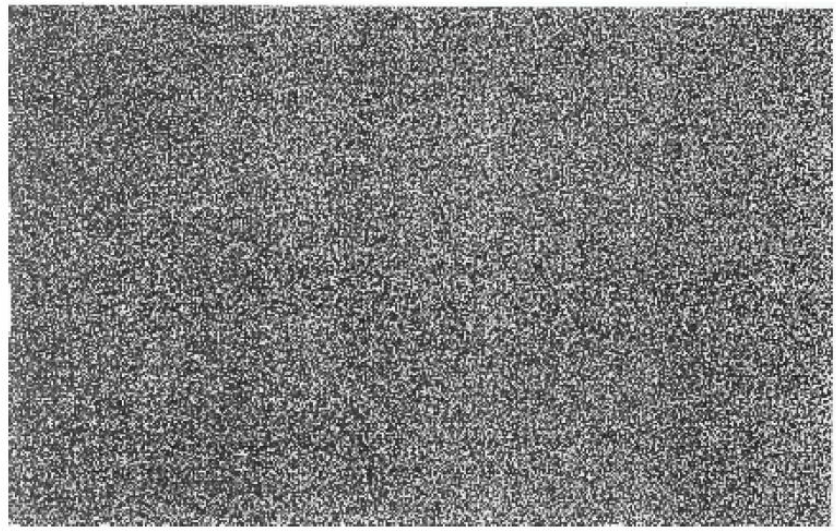




Plaintext : original picture



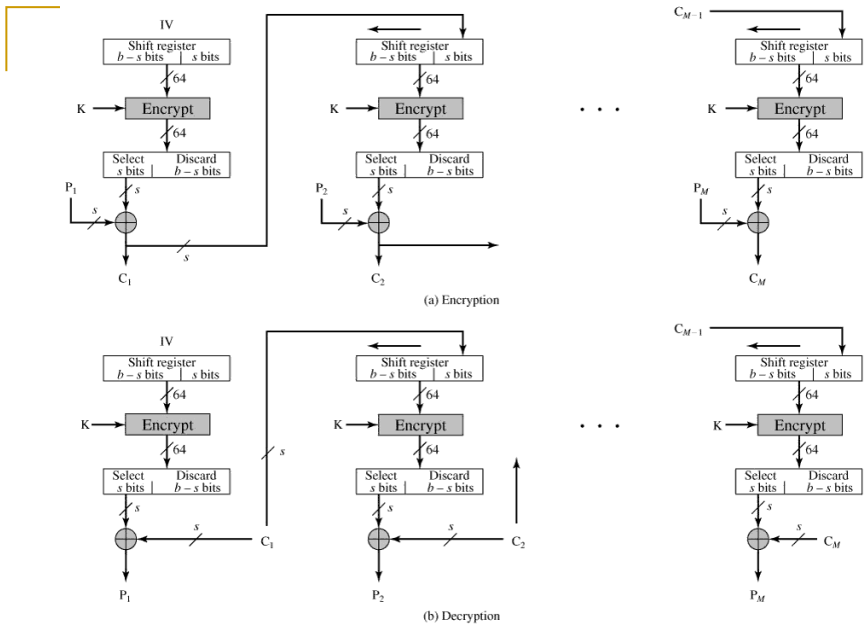
Ciphertext: ECB Encryption



Ciphertext: CBC Encryption

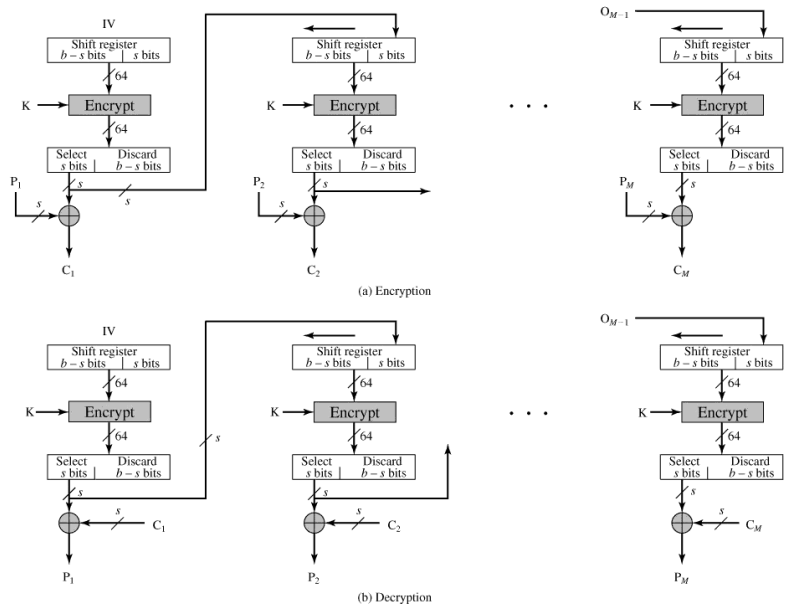
## CFB

- A entrada é processada j bits de cada vez. O texto cifrado precedente é utilizado para produzir uma saída pseudo-aleatória para uma operação XOR com o texto da mensagem para produzir o próximo conjunto de bits cifrado.
- A idéia é ter número de bits disponíveis de acordo com a disponibilidade de banda do meio de comunicação.



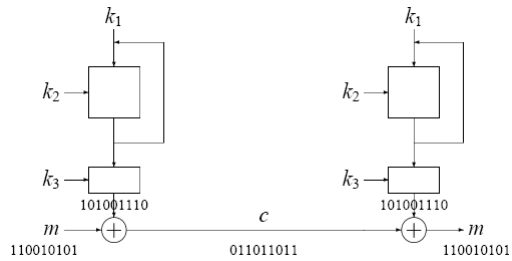

# OFB

- Similar ao CFB, exceto que a entrada para o algoritmo de encriptação  não possui efeito do texto da mensagem.
- Utilizado para transmissão de streams de mensagens em canais com ruídos (exemplo: comunicação por satélites).
- O ruído não se propaga para os bits subsequentes.




# Cifradores de streams

- Idéia básica:
  - Substituir a sequência aleatória do código de Vernam por uma sequência pseudo-aleatória.



## Considerações para o design de cifradores de streams

- A seqüência de encriptação, através de uma função semi-aleatória, deve ser longa.
- Os valores dos streams devem parecer aleatórios. Todos os bytes deverão aparecer de maneira uniforme.
- De modo a suportar ataques por força bruta, as chaves devem ser longas (128 bits).

## Algoritmos

- A5: utilizado em telefonia celular GSM.
- PKZIP
- RC4

## RC4

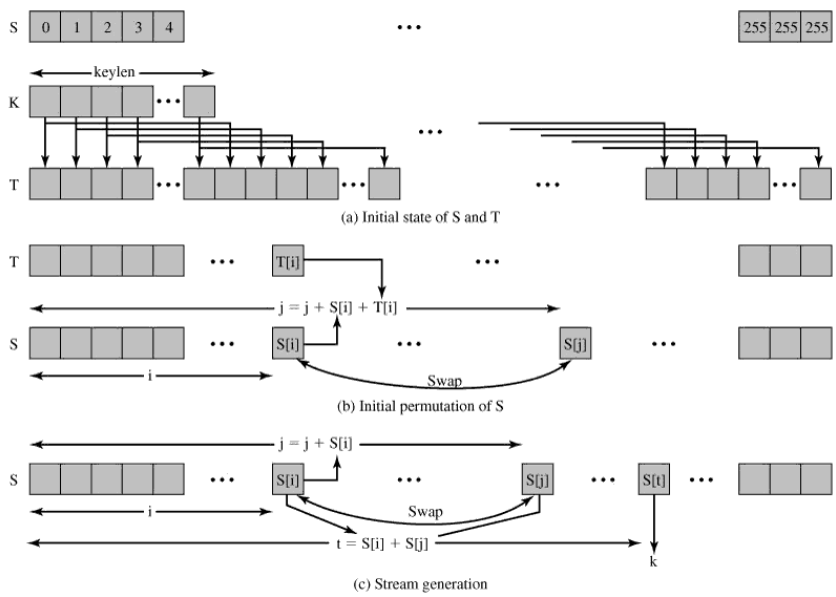
- Criado por Ron Rivest em 1987.
- O período da função é de cerca de $10^{100}$.
- Utilizado no protocolo SSL e WEP.
- Segredo até 1994.
- Vários trabalhos mostram possíveis ataques contra o RC4. Nenhum desses ataques são considerados praticáveis.

## Inicialização (KSA – Key Scheduling)

- /* Initialization */
- for i = 0 to 255 do
  - S[i] = i;
  - T[i] = K[i mod keylen];
- /* Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:*/
- /* Initial Permutation of S */
- j = 0;
- for i = 0 to 255 do j = (j + S[i] + T[i]) mod 256;
- Swap (S[i], S[j]);
- /* Because the only operation on S is a swap, the only effect is a permutation. S still contains all the numbers from 0 through 255.*/

## Geração do stream (PRGA- Pseudo Random Generation)

- /* Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S. After S[255] is reached, the process continues, starting over again at S[0]:*/
- /* Stream Generation */
- i, j = 0;
- while (true)
  - i = (i + 1) mod 256;
  - j = (j + S[i]) mod 256;
  - Swap (S[i], S[j]);
  - t = (S[i] + S[j]) mod 256;
  - k = S[t];
- /* To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of ciphertext. */



## Ataques contra a criptografia

- Ataques apenas com texto cifrado
  - Deduzir partes do texto original a partir de cópias de textos cifrados.
- Known-plaintext attack
  - Deduzir chaves a partir de partes do texto original conhecido.
- Chosen-plaintext attack
  - Deduzir chaves a partir de texto selecionado.
- Purchase key attack
  - Corrupção, violência, ....

## Análise linear e diferencial

- Observar pares de textos cifrados, em posições onde o texto plano apresenta diferenças interessantes.
- Por análise probabilística são deduzidas estruturas de chaves.
- DES e AES por enquanto não se mostraram vulneráveis para estes ataques.

## Ataques por análise de tempo e potência

- Observações de tempos de resposta podem levar a deduções sobre chaves.
  - Solução: Fazer com que todas as fases do algoritmo levem o mesmo tempo.
- Comportamento do consumo de energia pode levar deduções sobre chaves utilizadas.

## Conclusões

- Atualmente existem algoritmos rápidos e seguros para criptografia simétrica.
- Podem existir vulnerabilidades na implementação e formas de uso destes algoritmos.