Aula 9 - Cifras de bloco

Prof. Gabriel Rodrigues Caldas de Aquino

Instituto de Computação
Universidade Federal do Rio de Janeiro
gabrielaquino@ic.ufrj.br

Compilado em:
September 19, 2025

Cifras de Fluxo vs Cifras de Bloco

Cifras de Fluxo:

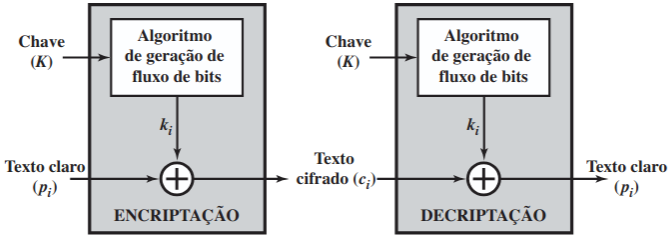
- Encriptam dados um **bit** ou **byte** por vez.
- Exemplos clássicos: Vigenère autochaveada e Vernam.
- Ideal: One-time pad (inquebrável se o fluxo de chaves for verdadeiramente aleatório).
- Limitação prática: o fluxo de chaves precisa ser compartilhado previamente via canal seguro.
- Solução prática: gerar o fluxo de bits algoritmicamente, controlado por chave, garantindo que partes futuras do fluxo sejam imprevisíveis.

Cifras de Bloco:

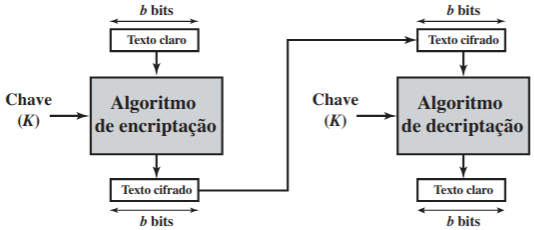
- Tratam um **bloco** de texto claro como um todo, normalmente de 64 ou 128 bits.
- Usuários compartilham uma chave simétrica.
- Modos de operação permitem uso semelhante ao das cifras de fluxo.
- Mais analisadas e amplamente utilizadas em aplicações de criptografia simétrica.

Cifra de fluxo vs. Cifra de bloco

Figura 3.1 Cifra de fluxo e cifra de bloco.



(a) Cifra de fluxo usando gerador algorítmico de fluxo de bits



Confusão e Difusão segundo Claude Shannon

Contexto histórico:

- Claude Shannon desenvolveu a ideia de **cifras de produto**, alternando funções de confusão e difusão [**?**].
- A estrutura da **cifra de Feistel**, baseada na proposta de Shannon de 1945, é utilizada em muitas cifras de bloco simétricas atuais.

Conceitos de Shannon:

- **Difusão**: espalhar a influência de cada bit do texto claro por muitos bits do texto cifrado, dificultando deduções estatísticas.
- **Confusão**: tornar a relação entre chave e texto cifrado complexa, de modo que conhecer parte da saída não revele informações sobre a chave.

Objetivo: impedir criptoanálise baseada em estatísticas do texto claro, como frequências de letras ou palavras prováveis.

Difusão em Criptografia

A difusão busca dissociar a estrutura estatística do texto claro das estatísticas do texto cifrado.

Princípio: Cada dígito do texto claro deve afetar muitos dígitos do texto cifrado, espalhando a informação.

Exemplo matemático: Para uma mensagem $M = m_1, m_2, m_3, \ldots$:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

onde k letras sucessivas do texto claro influenciam cada letra do texto cifrado.

Efeito:

- Frequências de letras e dígrafos no texto cifrado se aproximam de uma distribuição uniforme.
- Em cifras de bloco binárias, difusão é alcançada por permutações seguidas de funções de transformação, garantindo que bits de diferentes posições contribuam para cada bit cifrado.

Difusão e Confusão em Cifras de Bloco

Cada cifra de bloco transforma um bloco de texto claro em texto cifrado, dependendo da chave.

Difusão:

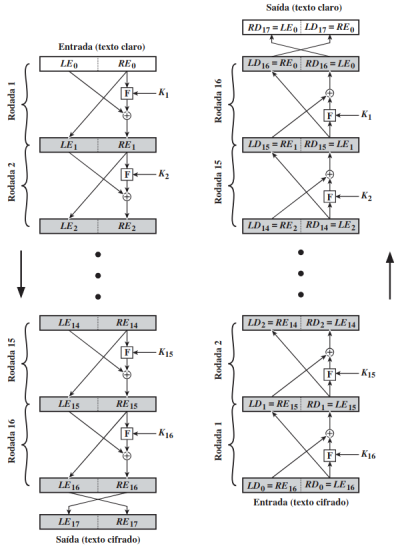
- Complica o relacionamento estatístico entre o texto claro e o texto cifrado.
- Faz com que cada bit do texto claro influencie muitos bits do texto cifrado.
- Frustra tentativas de deduzir padrões no texto claro.

Confusão:

- Complica o relacionamento entre o texto cifrado e a chave de encriptação.
- Utiliza funções de substituição complexas para dificultar a descoberta da chave.
- Funções lineares simples resultariam em pouca confusão e vulnerabilidade.

Cifra de Feistel

Figura 3.3 Encriptação e decriptação de Feistel (16 rodadas).



Estrutura da Cifra de Feistel

A estrutura de Feistel permite criar cifras de bloco seguras a partir de funções de encriptação simples.

Descrição:

- Entrada: bloco de texto claro de 2w bits e chave K.
- O bloco é dividido em duas metades: L_0 e R_0.
- Os dados passam por n rodadas de processamento.
- Cada rodada i recebe L_{i-1}, R_{i-1} e uma subchave K_i derivada de K.
- As subchaves K_i são normalmente diferentes entre si e da chave original.
- Após n rodadas, as metades são combinadas para gerar o bloco cifrado.

Exemplo: 16 rodadas, mas qualquer número de rodadas pode ser implementado.

Rodadas da Cifra de Feistel

Todas as rodadas têm a mesma estrutura básica:

Processamento em cada rodada:

1. Aplica-se uma função de substituição F à metade direita dos dados R_{i-1}, parametrizada pela subchave K_i.
2. Realiza-se a operação XOR entre a saída de F e a metade esquerda L_{i-1}:

$$L_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

3. Executa-se a permutação trocando as metades:

$$R_i = R_{i-1}, \quad L_i \leftrightarrow R_i$$

Observações:

- F é uma função de w bits de R_{i-1} e y bits de K_i, produzindo w bits de saída.
- Estrutura geral é repetida em todas as rodadas, formando uma rede de substituição-permutação (SPN) como proposta por Shannon.

Fatores que influenciam a segurança da cifra de Feistel

A execução de uma rede de Feistel depende de diversos parâmetros de projeto, sendo um dos principais:

Tamanho de bloco:

- Blocos maiores proporcionam maior segurança, mantendo os demais fatores constantes.
- Blocos maiores reduzem a velocidade de encriptação/decriptação para um dado algoritmo.
- Maior segurança é obtida por meio de maior difusão.
- Tradicionalmente, o tamanho de bloco de 64 bits era considerado adequado para cifras de bloco.
- O AES moderno utiliza tamanho de bloco de 128 bits.

Fatores que influenciam a segurança da cifra de Feistel

- **Tamanho da chave:**
  - Chave maior $\Rightarrow$ maior resistência a ataques de força bruta e maior confusão.
  - Impacto: pode reduzir a velocidade de encriptação/decriptação.
  - 64 bits ou menos: inseguros.
  - 128 bits: tornou-se padrão comum.
- **Número de rodadas:**
  - 1 rodada: segurança inadequada.
  - 16 rodadas: valor típico que garante segurança aceitável.
- **Algoritmo de geração de subchaves:**
  - Quanto mais complexo, mais difícil a criptoanálise.
- **Função F:**
  - Função mais complexa $\Rightarrow$ maior resistência a ataques.

Considerações adicionais no projeto da cifra de Feistel

- **Encriptação/Decriptação rápidas em software:**
  - Muitas vezes a implementação ocorre em software, não em hardware.
  - Desempenho do algoritmo passa a ser um fator crítico.
- **Facilidade de análise:**
  - Algoritmos claros e concisos são mais fáceis de avaliar contra ataques.
  - Transparência aumenta a confiança na robustez criptográfica.
  - Exemplo: o DES não possui estrutura de fácil análise.

Algoritmo de Decriptação de Feistel

O processo de decriptação com uma cifra de Feistel é basicamente o mesmo da encriptação. A regra é a seguinte: use o texto cifrado como entrada para o algoritmo, mas aplique as subchaves K_i em ordem reversa.

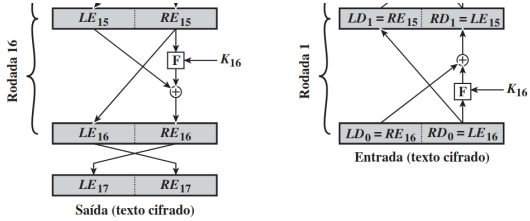
Ou seja:

- Use K_n na primeira rodada, K_{n-1} na segunda, ..., até K_1 na última rodada.

Observação: Isso permite que o mesmo algoritmo seja usado tanto para encriptação quanto para decriptação.

Validação do Algoritmo de Feistel com Ordem de Chaves Invertida

Observando a última rodada de encriptação da Cifra de Feistel e a primeira rodada da decriptação

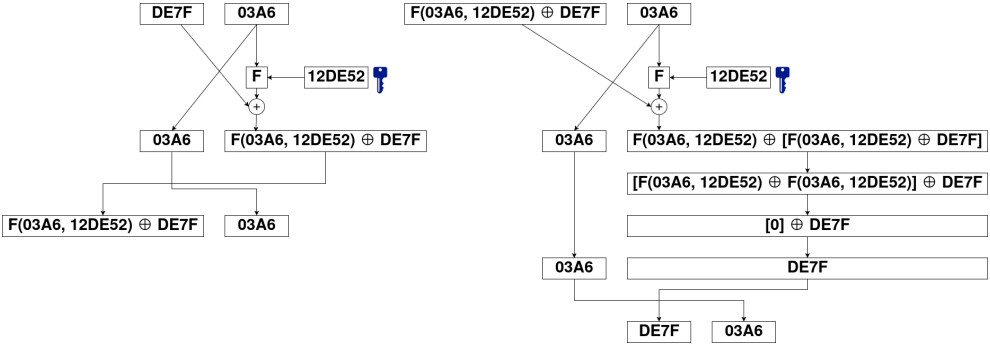


E sabendo que a operação lógica ou-exclusivo tem as seguintes propriedades:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

Validação do Algoritmo de Feistel com Ordem de Chaves Invertida

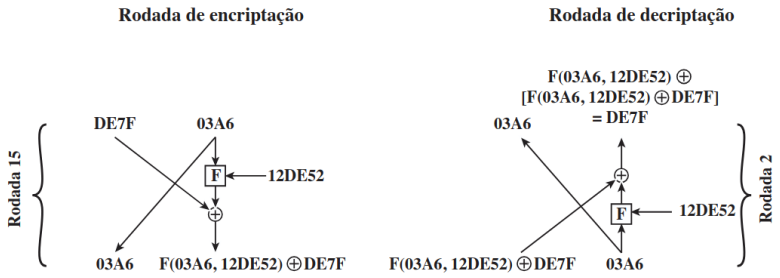
Temos:



Resumindo

Temos:

Figura 3.4 Exemplo Feistel.



Data Encryption Standard (DES)

Até a introdução do **Advanced Encryption Standard (AES)** em 2001, o DES era o esquema de encriptação mais utilizado.

Histórico:

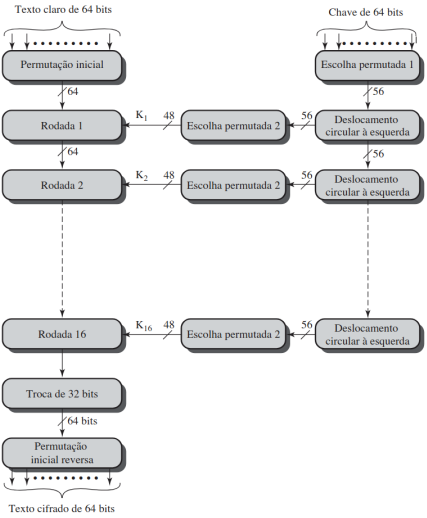
- Adotado em 1977 pelo **NIST**.
- Também conhecido como **Data Encryption Algorithm (DEA)**.
- Encripta dados em blocos de 64 bits usando uma chave de 56 bits.
- O mesmo algoritmo e chave são usados para encriptação e decriptação.

Evolução:

- Em 1994, o NIST reafirmou o DES para uso federal, recomendando restrição a informações não confidenciais.
- Em 1999, o NIST indicou que o DES seria apenas para sistemas legados e recomendou o **Triple DES**.
  - Triple DES repete o algoritmo DES três vezes usando duas ou três chaves diferentes.
- Hoje recomendamos o uso do **AES**

Esquema do DES

Figura 3.5 Representação geral do algoritmo de encriptação DES.



Funcionamento DES

No esquema de encriptação DES existem duas entradas na função: **texto claro** (64 bits) e **chave** (56 bits).

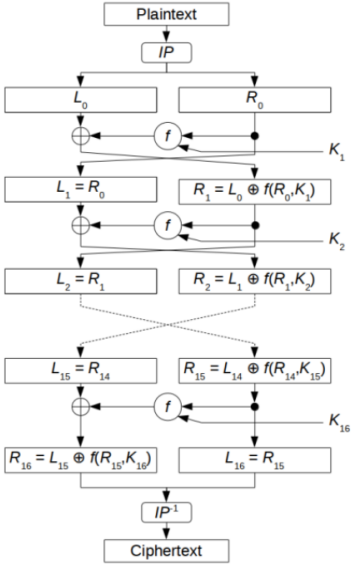
Processamento do texto claro:

- **Permutação Inicial (IP)**: reorganiza os bits do texto claro para produzir a entrada permutada.
- **16 rodadas de função Feistel**: cada rodada envolve permutação e substituição dos bits.
- **Troca das metades esquerda e direita** para gerar a pré-saída.
- **Permutação Final (IP^{-1})**: inverso da permutação inicial, produzindo o texto cifrado de 64 bits.

Data Encryption Standard (DES)

- Cifra de bloco que processa dados em blocos de 64 bits.
- Entrada: bloco de texto claro de 64 bits.
- Saída: bloco de texto cifrado de 64 bits.
- Algoritmo simétrico: mesma chave e mesmo algoritmo para encriptação e decriptação (com pequenas diferenças no escalonamento de chaves).

Esquema detalhado - DES



Chaves no DES

- Comprimento efetivo da chave: 56 bits.
- A chave é representada como um número de 64 bits:
  - A cada byte, o bit menos significativo é usado para verificação de paridade.
  - Esses 8 bits de paridade não participam da criptografia.
- Existem algumas chaves fracas, mas são facilmente evitáveis.
- Toda a segurança do DES depende da chave utilizada.

Funcionamento DES

Processamento da chave:

- Chave de 56 bits passa por uma permutação inicial.
- Para cada uma das 16 rodadas, gera-se uma subchave (K_i) usando:
  - Deslocamento circular à esquerda.
  - Permutação fixa.
- Cada rodada utiliza uma subchave diferente, derivada da chave original.

Decriptação DES: Assim como qualquer cifra de Feistel, a decriptação usa o mesmo algoritmo da encriptação, exceto que a aplicação das subchaves é invertida. Além disso, as permutações inicial e final são invertidas.

Outline of the DES Algorithm

- O DES opera sobre blocos de **64 bits**.
- Etapas principais:
  - **Permutação inicial (IP)** sobre o bloco de entrada.
  - Divisão em duas metades:
    - Lado esquerdo (32 bits).
    - Lado direito (32 bits).

Rodadas e Finalização do DES

- São realizadas **16 rodadas** de operações idênticas.
- Em cada rodada:
  - A função f combina dados com subchaves derivadas da chave principal.
- Após a 16ª rodada:
  - As metades esquerda e direita são reunidas.
  - Aplica-se a **permutação final**, inversa da permutação inicial.

DES Round Function f

- Em cada rodada:
  - A chave de 56 bits é **deslocada** e **reduzida** para 48 bits.
  - O lado direito (32 bits) é **expandido** para 48 bits.
  - Os 48 bits resultantes são combinados com a subchave via **XOR**.
  - O resultado passa por **8 S-boxes**, produzindo 32 bits.
  - Esses 32 bits sofrem uma **permutação**.
- A saída da função f é combinada com o lado esquerdo via XOR.
- Após isso, ocorre a **troca de metades**.

Equações de uma Rodada no DES

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

- L_i, R_i: metades esquerda e direita no final da rodada i.
- K_i: subchave de 48 bits da rodada i.
- f: função que realiza expansão, XOR, substituições e permutação.
- Repetido por **16 rodadas**.

Uma rodada do DES

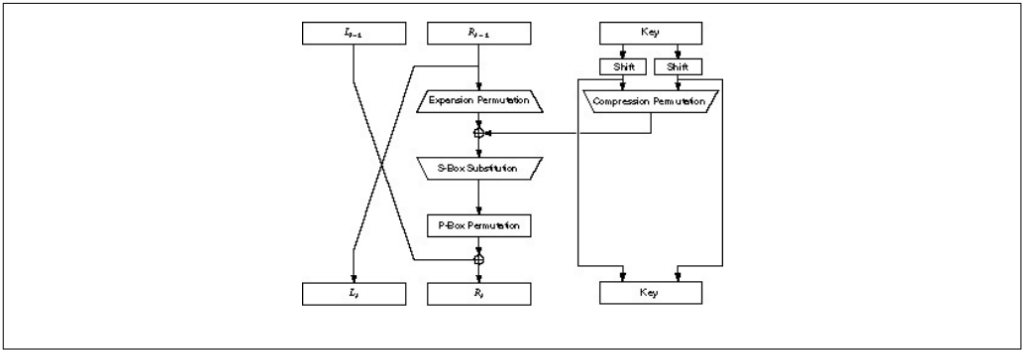


Figure 12.2 *One round of DES.*

Exemplo do DES

Exemplo prático do DES:

- **Objetivo**: observar os padrões em hexadecimal que surgem de uma etapa para outra, sem necessariamente replicar o cálculo manualmente
- Texto claro escolhido é um **palíndromo hexadecimal**.

Texto claro:	`02468aceeca86420`
Chave:	`0f1571c947d9e859`
Texto cifrado:	`da02ce3a89ecac3b`

Exemplo do DES

- Evolução do algoritmo ao longo das rodadas.
- Primeira linha: valores de 32 bits das metades esquerda (L) e direita (R) **após a permutação inicial (IP)**.
- Linhas seguintes (1 a 16): resultado após cada rodada, incluindo:
- Última linha: valores de L e R após a permutação inicial inversa (IP^{-1}).
- A concatenação desses dois valores gera o **texto cifrado**.

Tabela 3.2 Exemplo de DES.

Rodada	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Nota: as subchaves DES são apresentadas como oito valores de 6 bits no padrão hexa.

Efeito Avalanche

- Propriedade desejável em algoritmos de encriptação.
- Uma pequena mudança no texto claro ou na chave deve causar uma grande alteração no texto cifrado.
- **Objetivo**: Alterar apenas **um bit** no texto claro ou na chave deve modificar **muitos bits** no resultado.
- **Resultado esperado**: Caso a mudança fosse pequena, seria possível reduzir o espaço de busca de textos claros ou de chaves.
- O efeito avalanche aumenta a segurança, dificultando ataques de análise e padrões previsíveis.

Exemplo do Efeito Avalanche no DES

- Texto claro inicial modificado no 4º bit.
- Tabela de acompanhamento mostra:
  - Coluna 1: rodadas do algoritmo.
  - Coluna 2: valores intermediários de 64 bits no final de cada rodada.
  - Coluna 3: número de bits diferentes entre os dois textos.
- Após apenas 3 rodadas: diferença de **18 bits**.
- Texto cifrado final: diferença de **32 bits**.
- Demonstra claramente o **efeito avalanche** do DES.

Exemplo do Efeito Avalanche no DES

Tabela 3.3 Efeito avalanche no DES: mudança no texto claro.

Rodada		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Efeito Avalanche com Alteração da Chave

Experimento: Texto claro mantido constante mas com duas chaves utilizadas:

- Chave original modificada **com diferença no 4º bit**

Resultados:

- Diferença significativa no texto cifrado final — cerca de metade dos bits são diferentes.
- O efeito avalanche já aparece após poucas rodadas do DES.

Demonstra: Robustez do algoritmo contra pequenas alterações na chave.

Exemplo do Efeito Avalanche no DES - Mudança na chave

Tabela 3.4 Efeito avalanche no DES: mudança na chave.

Rodada		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

A Força do DES

- Desde sua adoção como padrão federal, surgiram preocupações sobre a segurança do DES.
- Principais áreas de preocupação:
  - **Tamanho da chave**: 56 bits pode ser vulnerável a ataques de força bruta.
  - **Natureza do algoritmo**: estrutura e procedimentos internos do DES podem influenciar sua resistência a ataques criptográficos.
- Essas questões motivaram o desenvolvimento de algoritmos mais seguros, como o Triple DES (3DES) e o AES.

Uso de Chaves de 56 Bits no DES

- Chave de 56 bits $\rightarrow 2^{56} \approx 7,2 \times 10^{16}$ chaves possíveis.
- Ataque de força bruta **pareceria** impraticável:
  - Pesquisando metade do espaço de chaves
  - Uma máquina realizando 1 encriptação por microssegundo levaria mais de 1000 anos.
- No entanto, Diffie e Hellman (1977) propuseram tecnologia paralela:
  - Máquina com 1 milhão de dispositivos, cada um realizando 1 encriptação/µs.
  - Tempo médio de busca: cerca de 10 horas.
  - Custo estimado: US$ 20 milhões (em 1977).
- Demonstra que, apesar do tamanho da chave, o DES não é invulnerável a ataques especializados.

Impacto da Tecnologia Moderna na Segurança do DES

- Atualmente, nem é necessário hardware especial para ataques de força bruta contra o DES.
- Processadores comerciais modernos já oferecem velocidades que ameaçam a segurança do DES:
  - Computadores multicore atuais: 10^9 combinações de chaves por segundo [SEAG08].
  - Testes em máquinas Intel multicore: 5×10^8 encriptações por segundo [BASU12].
  - Supercomputadores contemporâneos: 10^{13} encriptações por segundo.
- Instruções de hardware recentes para AES também aceleram operações criptográficas, mostrando que ataques de força bruta se tornam cada vez mais viáveis.
- Conclusão: o DES, com chave de 56 bits, é vulnerável diante da tecnologia atual.

Força Bruta e Tamanho da Chave

- Tabela 3.5 mostra o tempo necessário para ataques de força bruta em diferentes tamanhos de chave.
- Exemplos:
  - Um único PC moderno pode quebrar o DES (56 bits) em 1 ano.
  - Vários PCs em paralelo reduzem drasticamente o tempo.
  - Supercomputadores contemporâneos: 1 hora para descobrir a chave do DES.
- Chaves de 128 bits ou mais são efetivamente inquebráveis por força bruta:
  - Mesmo com aceleração de 10^{12} vezes, ainda seriam necessários 100 mil anos.
- Alternativas mais seguras ao DES: **AES** e **Triple DES**

Exemplo do Efeito Avalanche no DES - Mudança na chave

Tabela 3.5 Tempo médio exigido para uma busca exaustiva no espaço de chaves.

Tamanho de chave (bits)	Cifra	Número de chaves alternativas	Tempo exigido a 10^9 decriptações/s	Tempo exigido a 10^{13} decriptações/s
56	DES	$2^{56} \approx 7{,}2 \times 10^{16}$	2^{55} ns = 1,125 ano	1 hora
128	AES	$2^{128} \approx 3{,}4 \times 10^{38}$	2^{127} ns = $5{,}3 \times 10^{21}$ anos	$5{,}3 \times 10^{17}$ anos
168	Triple DES	$2^{168} \approx 3{,}7 \times 10^{50}$	2^{167} ns = $5{,}8 \times 10^{33}$ anos	$5{,}8 \times 10^{29}$ anos
192	AES	$2^{192} \approx 6{,}3 \times 10^{57}$	2^{191} ns = $9{,}8 \times 10^{40}$ anos	$9{,}8 \times 10^{36}$ anos
256	AES	$2^{256} \approx 1{,}2 \times 10^{77}$	2^{255} ns = $1{,}8 \times 10^{60}$ anos	$1{,}8 \times 10^{56}$ ano
26 caracteres (permutação)	Monoalfabético	$2! = 4 \times 10^{26}$	2×10^{26} ns = $6{,}3 \times 10^{9}$ anos	$6{,}3 \times 10^{6}$ anos

Resumo - Projeto de cifra de bloco

Número de Rodadas no DES

- Maior número de rodadas $\rightarrow$ mais difícil a criptoanálise, mesmo se a função F for relativamente fraca.
- Critério de projeto: número de rodadas suficiente para que criptoanálises conhecidas exijam mais esforço do que um ataque de força bruta.
- No DES:
  - 16 rodadas
  - Criptoanálise diferencial: $2^{55,1}$ operações
  - Força bruta: 2^{55} operações
  - Se houvesse 15 ou menos rodadas, criptoanálise diferencial exigiria menos esforço que a força bruta.
- Critério facilita comparar a força de diferentes algoritmos.
- Sem descobertas revolucionárias em criptoanálise, a força de um algoritmo que satisfaz este critério é julgada principalmente pelo tamanho da chave.

Resumo - Projeto de cifra de bloco

Projeto da Função F no DES

- A função F é o núcleo da cifra de bloco de Feistel e é responsável pela **confusão**.
- Critérios importantes para o projeto de F:
  - **Não linearidade**: quanto menos linear, mais difícil a criptoanálise.
  - **Efeito avalanche**: mudança em 1 bit da entrada altera muitos bits da saída.
  - **Strict Avalanche Criterion (SAC)** [WEBS86]:
    - Qualquer bit de saída muda com probabilidade 1/2 quando qualquer bit de entrada é invertido.
    - Originalmente definido para S-boxes, mas aplicável à função F como um todo.
  - **Bit Independence Criterion (BIC)** [WEBS86]:
    - Bits de saída mudam independentemente quando qualquer bit de entrada é invertido.
- Aplicar SAC e BIC fortalece a eficácia da função de confusão e aumenta a segurança do algoritmo.

Resumo - Projeto de cifra de bloco

Algoritmo de Escalonamento de Chave

- Em cifras de bloco de Feistel, a chave principal é usada para gerar uma **subchave** para cada rodada.
- Objetivo do escalonamento de chave:
  - Maximizar a dificuldade de deduzir subchaves individuais.
  - Tornar mais difícil recuperar a chave principal a partir das subchaves.
- Não existe, até o momento, um princípio geral universalmente aceito para o projeto do algoritmo de escalonamento de chaves.
- O cuidado no escalonamento é crucial para garantir a segurança global do algoritmo.

Triple DES (3DES)

- Primeiro padronizado em 1985 pela ANSI para aplicações financeiras (X9.17).
- Incorporado ao DES como parte do padrão FIPS 46-3 em 1999.
- Utiliza **três chaves** e três execuções do algoritmo DES.
- Sequência de operação: **Encrypt-Decrypt-Encrypt (EDE)**:

$$C = E(K_3, D(K_2, E(K_1, P)))$$

  - C = texto cifrado
  - P = texto claro
  - K_1, K_2, K_3 = chaves
- A abordagem EDE garante compatibilidade com DES original quando $K_1 = K_2 = K_3$.

Diretrizes do FIPS 46-3 para 3DES

- 3DES é o algoritmo de cifra simétrica aprovado pelo FIPS para uso corrente.
- O DES original (chave de 56 bits) é permitido apenas para sistemas legados.
- Novas aquisições de sistemas devem suportar 3DES.
- Organizações governamentais com sistemas DES legados são incentivadas a migrar para 3DES.
- Espera-se que 3DES e AES coexistam como algoritmos aprovados pelo FIPS, permitindo uma transição gradual para AES.

Exemplo do Efeito Avalanche no DES - Mudança na chave

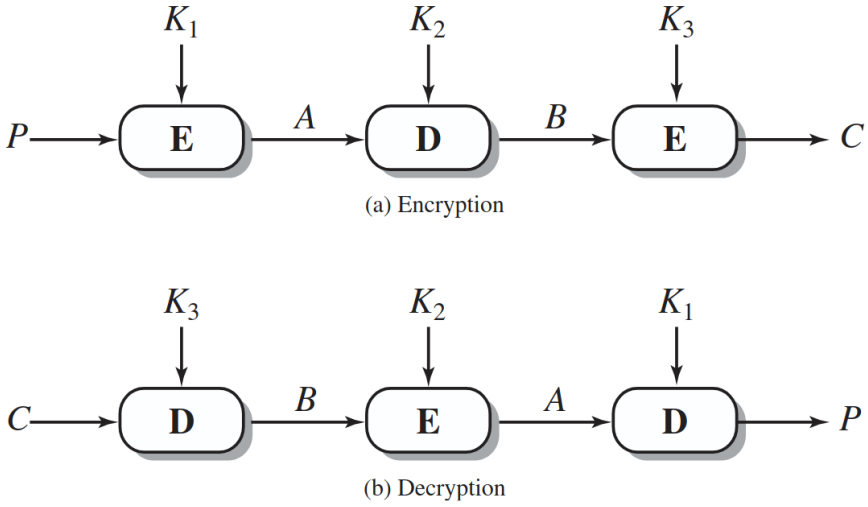


Figure 20.2 **Triple DES**

Compatibilidade do 3DES com DES

- Quando as três chaves são iguais ($K_1 = K_2 = K_3$), temos:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K, P)$$

- A operação do meio $D(K_1, E(K_1, P))$ se anula, resultando apenas na cifra DES original.
- Importância:
  - Permite que dados cifrados com DES possam ser decifrados por um sistema 3DES configurado em modo compatível.
  - Sistemas antigos que usam DES podem processar dados cifrados por 3DES no modo compatível.
- Essa retrocompatibilidade facilitou a adoção do 3DES sem necessidade de substituir todos os sistemas legados.

Chaves e Criptografia no 3DES

- O uso da decriptação na segunda etapa do 3DES não tem significado criptográfico; sua vantagem é permitir compatibilidade com o DES original:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K, P)$$

- Com três chaves distintas (K_1, K_2, K_3), o 3DES possui **168 bits efetivos de chave**.
- É possível usar apenas duas chaves $(K_1 = K_3)$, resultando em uma chave efetiva de **112 bits**.
- O padrão FIPS 46-3 fornece diretrizes formais para a utilização do 3DES com estas configurações.

Chaves e Criptografia no 3DES

- O uso da decriptação na segunda etapa do 3DES não tem significado criptográfico; sua vantagem é permitir compatibilidade com o DES original:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K, P)$$

- Com três chaves distintas (K_1, K_2, K_3), o 3DES possui **168 bits efetivos de chave**.
- É possível usar apenas duas chaves $(K_1 = K_3)$, resultando em uma chave efetiva de **112 bits**.
- O padrão FIPS 46-3 fornece diretrizes formais para a utilização do 3DES com estas configurações.

Advanced Encryption Standard (AES) e Rijndael

- Publicado pelo **NIST** em 2001 como padrão de cifra simétrica de bloco.
- Substitui o DES em diversas aplicações, oferecendo maior segurança e eficiência.
- AES é mais complexo que cifras de chave pública (ex.: RSA), sendo difícil de explicar de forma simplificada.
- Em 2000, o NIST selecionou a família de cifras de bloco **Rijndael** como vencedora do concurso AES.
- **Rijndael**: cifra de bloco escolhida como AES pelo NIST.
- **AES não usa Cifra de Feistel**
- Três variantes especificadas: AES-128; AES-192; AES-256

Tipos AES

Table 3. Key-Block-Round Combinations

	Key length		Block size		Number of rounds
	Nk	(in bits)	*Nb*	(in bits)	*Nr*
AES-128	4	128	4	128	10
AES-192	6	192	4	128	12
AES-256	8	256	4	128	14

Diferenças entre AES-128, AES-192 e AES-256

- As três variantes diferem em três aspectos principais:
  1. **Comprimento da chave** (128, 192 ou 256 bits).
  2. **Número de rodadas (Nr)**, que determina o tamanho do key schedule.
  3. **Especificação da recursão em KEY EXPANSION()**.
- Para cada algoritmo:
  - **Nr**: número de rodadas.
  - **Nk**: número de palavras da chave.
  - **Nb**: número de palavras do estado; no padrão AES, **Nb = 4**.
- Somente essas configurações de Rijndael estão conformes com o padrão AES.
- Veja a especificação FIPS 197 do AES

AES também suporta vários modos

Exemplos de modos de operação:

- AES-ECB (Electronic Codebook)
- AES-CBC (Cipher Block Chaining)
- AES-CTR (Counter Mode)
- AES-GCM (Galois/Counter Mode)
- Artigo: A Comparative Analysis of AES Common Modes of Operation
- Artigo: MODES OF OPERATION OF THE AES ALGORITHM

Electronic Code Book (ECB) – Considerações

- O modo ECB divide a mensagem em blocos $(P_1, P_2, \ldots, P_N)$ e cifra cada bloco separadamente com a mesma chave K.
- Se a mensagem não preencher o último bloco, ele é completado com *padding*.
- Vantagem: erros em um bloco não se propagam para os outros, permitindo decifrar blocos não corrompidos.
- Desvantagem: a criptografia é determinística; blocos de texto claro idênticos produzem blocos cifrados idênticos.
- Problemas adicionais:
  - Blocos idênticos ou mensagens com início igual são facilmente reconhecíveis.
  - A ordem dos blocos cifrados pode ser alterada sem que o receptor perceba.
- Conclusão: não recomendado para dados maiores que um bloco; alguns autores desaconselham completamente seu uso.

Modo ECB

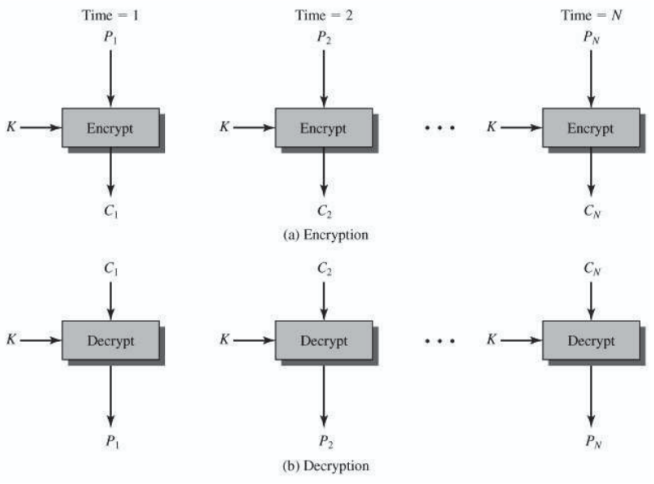


Figure 1: Scheme of the ECB mode of operation [2]

Cipher Block Chaining (CBC) – Considerações

- O modo CBC resolve o problema do ECB, reduzindo padrões repetidos no texto cifrado.
- Cada bloco de texto claro é XORado com o bloco cifrado anterior antes da encriptação.
- O primeiro bloco de texto claro é XORado com um *initialization vector* (IV).
- Benefício: mensagens longas com padrões repetidos podem ser tratadas de forma mais segura.
- Resultados de cifras distintas: mesmo texto claro cifrado múltiplas vezes gera diferentes textos cifrados devido ao IV.
- Desvantagem: requer mais tempo de processamento que o ECB devido ao encadeamento.
- Pode ser sincronizado para evitar propagação de erros causados por ruído no canal.
- O CBC não suporta paralelismo (**diferentemente do ECB)**

Modo CBC

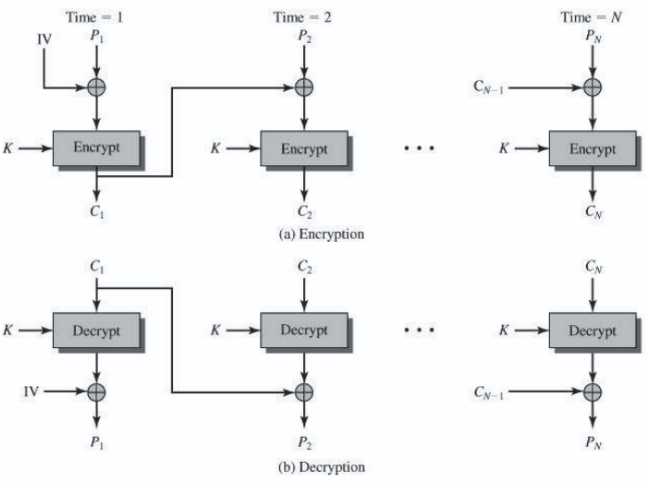
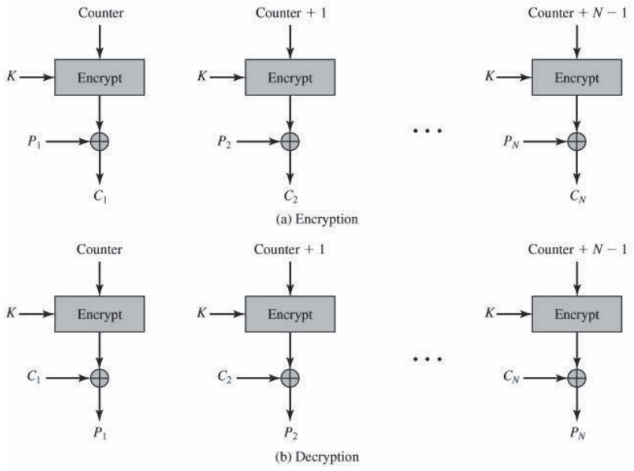


Figure 2: Scheme of the CBC mode of operation [2]

Counter Mode (CTR)

- Usa um contador como vetor de inicialização (IV), com tamanho igual ao do bloco.
- Cada bloco de texto claro é XORado com a saída da cifra aplicada ao contador.
- Não há necessidade de *padding* no último bloco.
- Os blocos são independentes, sem propagação de erro entre eles.
- Suporta paralelismo e pré-processamento, acelerando cifragem/decifragem.
- As operações de encriptação e decriptação são idênticas.
- É fundamental não reutilizar o mesmo contador com a mesma chave, sob risco de quebra completa da confidencialidade.
- Normalmente, o contador é inicializado com um valor único (ex.: 96 bits aleatórios + 32 bits incrementais).
- A chave deve ser trocada após $2^{n/2}$ blocos (onde n é o tamanho do bloco).
- Considerado um dos modos mais seguros e eficientes para AES.

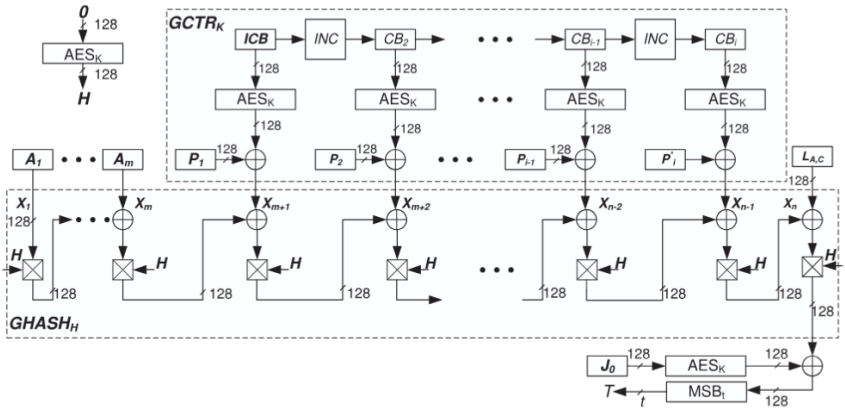
Modo CTR


Counter
Counter + 1
Counter + N − 1
K
Encrypt
P1
P2
PN
C1
C2
CN
(a) Encryption
Counter
Counter + 1
Counter + N − 1
K
Encrypt
C1
C2
CN
P1
P2
PN
(b) Decryption

AES – Galois/Counter Mode (GCM)

- GCM combina duas funções:
  - **Autenticação**: cálculo de um *tag* de integridade.
  - **Confidencialidade**: criptografia no modo *Counter*.
- O processo de confidencialidade é baseado no modo CTR (*Counter Mode*).
- A autenticação é realizada pela função **GHASH**, que usa multiplicações em $\mathrm{GF}(2^{128})$.
  - $GF(2^{128})$ é o campo de Galois com 128 bits: cada bloco de 128 bits é tratado como um polinômio de grau ≤ 127 com coeficientes 0 ou 1; a soma é XOR e a multiplicação é feita módulo
$$p(x) = x^{128} + x^7 + x^2 + x + 1.$$
- O *hash subkey* H é obtido aplicando AES sobre o bloco nulo.
- O *tag* de autenticação T é gerado a partir dos dados confidenciais e dos dados adicionais autenticados (AAD).
- Na decriptação autenticada, o *tag* é verificado para garantir integridade e autenticidade.
- Artigo: Modo GCM

Modo GCM


0
128
AES_K
128
H
GCTR_K
ICB
INC
CB_2
CB_{i-1}
INC
CB_i
128
AES_K
A_1
A_m
P_1
P_2
P_{i-1}
P*_i
L_{A,C}
X_1
X_m
X_{m+1}
X_{m+2}
X_{n-2}
X_{n-1}
X_n
GHASH_H
J_0
T
t
MSB_t

AES-GCM: Campo de Galois e GHASH

- O GCM usa o campo finito $GF(2^{128})$ para autenticação.
- Cada bloco de 128 bits é tratado como um polinômio de grau até 127:

$$b_0 + b_1 x + b_2 x^2 + \cdots + b_{127} x^{127}, \quad b_i \in \{0, 1\}$$

- A chave de hash H é gerada cifrando um bloco de zeros com AES.
- O GHASH combina cada bloco de dados ou AAD com H assim:

$$X_i = (X_{i-1} \oplus B_i) \cdot H \mod p(x)$$

- O polinômio irreducível usado é fixo pelo padrão NIST:

$$p(x) = x^{128} + x^7 + x^2 + x + 1$$

Fluxo de autenticação no AES-GCM (GHASH)

O fluxo de autenticação no AES-GCM (GHASH) funciona assim:

1. Pega o acumulador do bloco anterior X (ou zero no primeiro bloco).
2. Faz XOR com o bloco atual da mensagem P.
3. Multiplica o resultado pelo H, que é a chave de hash obtida cifrando um bloco de zeros com AES.
4. Reduz módulo o polinômio

$$p(x) = x^{128} + x^7 + x^2 + x + 1$$

para manter 128 bits.

Ou seja, cada passo é:

$$X_i = ((X_{i-1} \oplus P_i) \cdot H) \bmod p(x)$$

- O XOR encadeia os blocos e mistura os dados.
- O módulo garante que o resultado continue com 128 bits, pronto para o próximo bloco ou para gerar a Tag final.