

# Funções de hash

**Segurança da Informação**

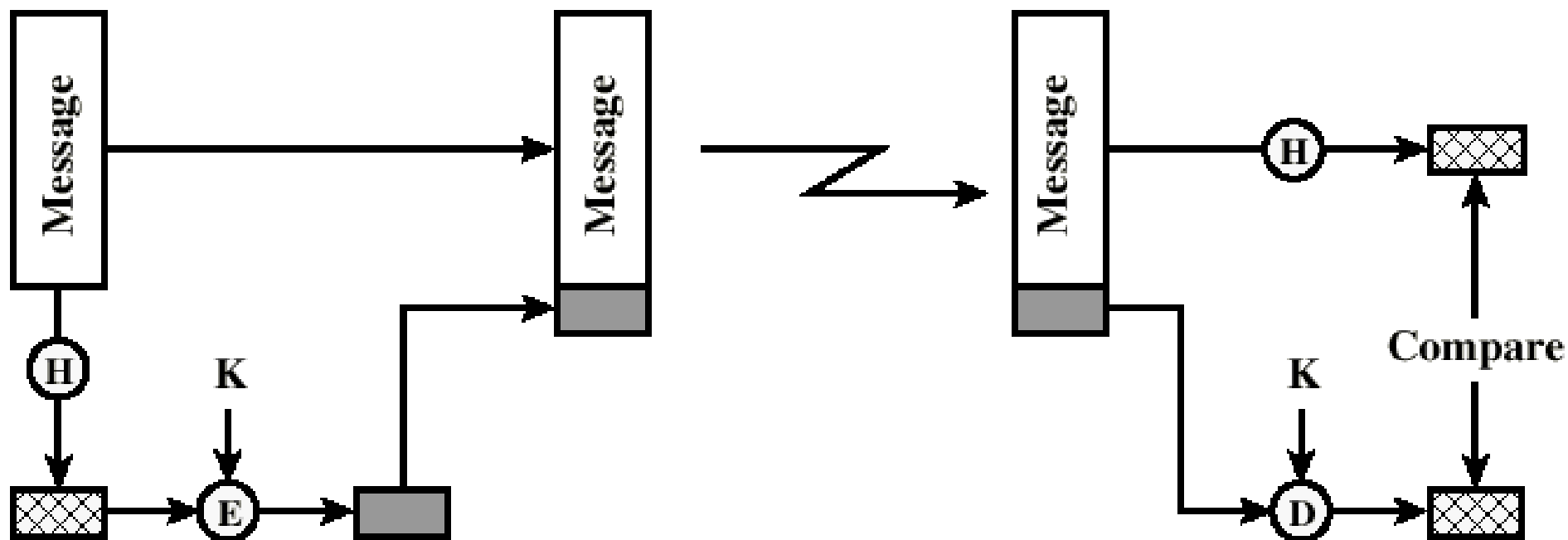
**Curso Técnico em Redes de Computadores**

**ETER – FAETEC - Rio de Janeiro - RJ**

# O que são as funções de hash?

- As funções de hash são fundamentais para a segurança, envolvendo cifragem na comunicação em redes.
- Seu uso principal é para garantir a **integridade** da informação.
- São funções que, a partir de uma mensagem, composta de uma sequência de bits de tamanho variável, gera um número em base64 que é uma “*impressão digital*” dessa mensagem.
- Alguns chamam também de funções de resumo, ou MDF (*message digest functions*).

# Como funciona uma função hash?



# Propriedades das funções de hash.

- **Condensação:** As funções devem gerar um hash de tamanho fixo, independente do tamanho da mensagem original. Afinal, o conjunto de mensagens é muito maior do que o conjunto de hashes possíveis.
- **Efeito avalanche:** Se mudar algo na mensagem original – ainda que seja um bit – o novo hash é algo completamente diferente.

# Vantagens das funções de hash sobre a cifragem.

- Cifragem é lenta, geração do hash é mais rápida.
- Hardware focado em cifragem é caro e otimizado para trabalhar com grandes conjuntos de dados.
- Os algoritmos de cifragem podem estar protegidos por uma patente.

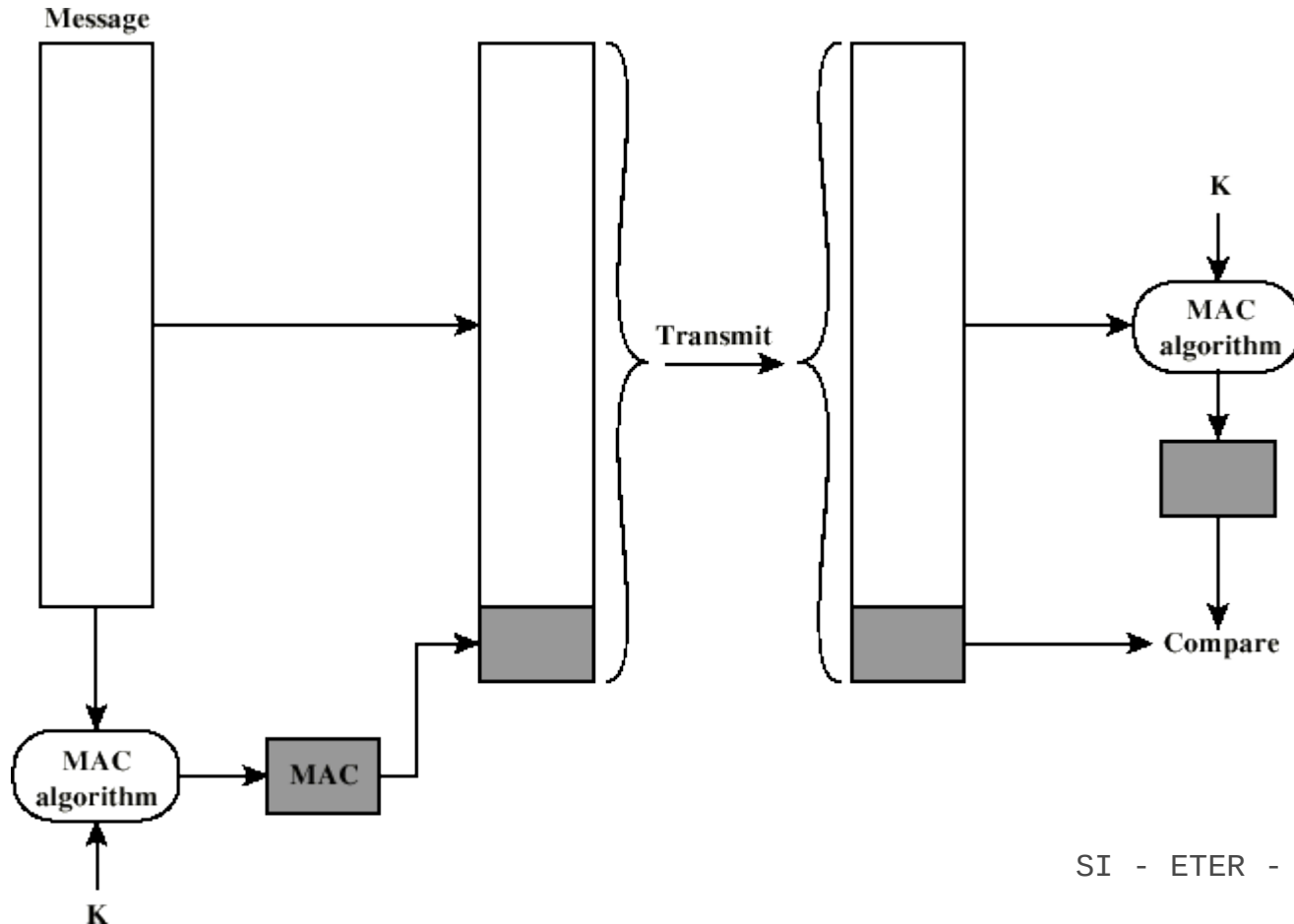
# Propriedades das funções de hash (resistências).

- **Unidirecionalidade, ou resistência à 1ª inversão:** É fácil e rápido gerar o hash a partir da mensagem, mas recompor a mensagem a partir do hash é computacionalmente inviável.
- **Resistência à 2ª inversão:** Se você tem uma mensagem e seu hash, é inviável encontrar uma outra mensagem que gere o mesmo hash.
- **Resistência à 3ª inversão:** É muito difícil (inviável computacionalmente) encontrar duas mensagens quaisquer que gerem o mesmo hash.

# Message Authentication Code (MAC)

- Algoritmo usado para autenticação de mensagens.
- Um pequeno bloco de dados é anexado à mensagem.
- O MAC é gerado usando uma chave secreta, e assume que ambas as partes (A e B) conhecem essa chave.
- A mensagem é transmitida junto com o MAC. No destino, o algoritmo é reexecutado, e os MACs são comparados. Se forem iguais, significa que a mensagem é autêntica.

# Message Authentication Code (MAC)



# Algoritmos SHS (Secure Hash Signature).

- São criações da NSA, agência estadunidense de segurança.
- Existem algumas “famílias” de algoritmos, mas a mais conhecida é a SHA-1 (Secure Hash Algorithm 1). Trabalha com mensagens com no máximo  $2^{64}$  bits, gera hashes fixos de 160 bits e usa blocos de 512 bits da mensagem por vez.
- Ele é tido como o menos seguro, e desde 2010 ele não é mais considerado seguro. Existem outras famílias, como a SHA-2 (hashes de até 512 bits) e a SHA-3 (semelhante à SHA-2, mas com estrutura interna bem diferente).

# Algoritmos SHS (Secure Hash Signature).

Algoritmo	Tamanho da mensagem (bits)	Tamanho do bloco (bits)	Tamanho da palavra (bits)	Tamanho do hash (bits)
SHA-1	Menor do que $2^{64}$	512	32	160
SHA-224	Menor do que $2^{64}$	512	32	224
SHA-256	Menor do que $2^{64}$	512	32	256
SHA-384	Menor do que $2^{64}$	1024	64	384
SHA-512	Menor do que $2^{64}$	1024	64	512

- No Linux, existem vários utilitários que implementam os algoritmos SHA, como o sha1sum, o sha1hmac e outros. No Debian, eles estão no pacote **coreutils**.

# Algoritmos MD (Message Digest).

- Esta família de algoritmos também é bastante usada hoje em dia para geração de hashes. Seu modo de funcionamento é diferente do modo das famílias SHA, e conforme muda o tipo do algoritmo, aumenta a segurança e aumenta também a lentidão.
- No Linux, temos o utilitário md5sum, parte do pacote **coreutils** (no Debian), que implementa o protocolo MD5.

# Algoritmos MD (Message Digest)


- A tendência é o MD5 dominar o mercado, visto que o aumento do poder computacional facilita a adoção desse algoritmo.

# Alguns sites para experimentar as funções de hash.

- MD2: <https://codebeautify.org/md2-hash-generator>
- Nesse site também há geradores de hash online para as funções MD, SHA1, SHA2 e muitos outros.
- Por exemplo, a frase **Turma de SI**, em MD5, gera o seguinte hash:  
e7e72dd2bb1c8292e2e6da57d6873ebc.
  - Em SHA-1, gera o seguinte hash:  
92d0f2873a074d693831d36051595ebfc50c9012.

# O HMAC.

- O HMAC usa um MAC, gerado por uma função hash, como o SHA-1, o MD5 ou qualquer outra.
- É um padrão da Internet, descrito no RFC 2104.
- Motivações:
  - Funções hash são mais rápidas do que algoritmos de criptografia, como o DES.
  - Há muito código-fonte com as implementações das funções hash.
  - Não há restrições de exportação a partir dos EUA.

# O CMAC.

- O Cypher-based Message Authentication Code é baseado nas cifras de bloco, e são amplamente usados na indústria e nos governos.
- Há uma limitação de tamanho da mensagem, mas pode trabalhar com duas chaves e preenchimento (padding).

# Ataques às funções hash.

- **Paradoxo do aniversário:** Definida uma data, é necessário um grupo de 253 pessoas para que haja chances maiores que 50% de pelo menos uma delas fazer aniversário na data definida. Mas se você precisa que duas pessoas façam aniversário em uma mesma data qualquer com probabilidade maior do que 50%, o grupo pode ser reduzido para 23 pessoas.
- Logo, o **ataque do aniversário** consiste em atacar a resistência à 3ª inversão, que é encontrar um par de mensagens que gere o mesmo hash. O atacante obtém a 1ª mensagem (a que se quer ter acesso), e cria uma 2ª mensagem. Ele manipula a 2ª até que obtenha o mesmo hash. Tendo a obtido, ele troca a mensagem legítima pela manipulada e a envia.

# Modos de operação.

- Como vimos, em alguns algoritmos o tamanho do bloco é fixo (128, até 256 bits). E se a mensagem tiver um tamanho maior?
- Para isto existem os **modos de operação**. Os modos de operação do protocolo independem do protocolo escolhido.

# Modo de operação ECB.

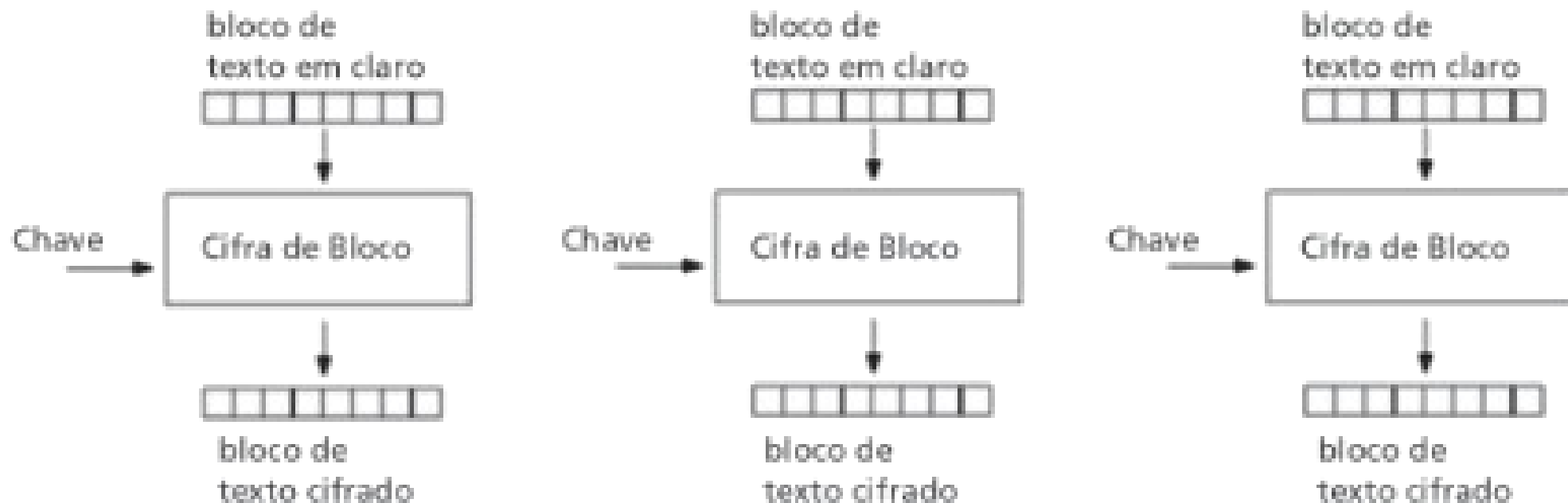
- Electronic Codebook.
- Cada bloco é cifrado independente do outro. Não há relação entre um bloco de texto cifrado e o próximo.
- Uma vantagem clara é que este algoritmo é simples e é **paralelizável**: Se você tem vários núcleos de processamento (coisa comum hoje em dia), cada núcleo pode cifrar um conjunto de blocos, simultaneamente.
- Se um bit foi corrompido, isto afetará o bloco da mensagem recuperada, não a mensagem toda.

# Modo de operação ECB.

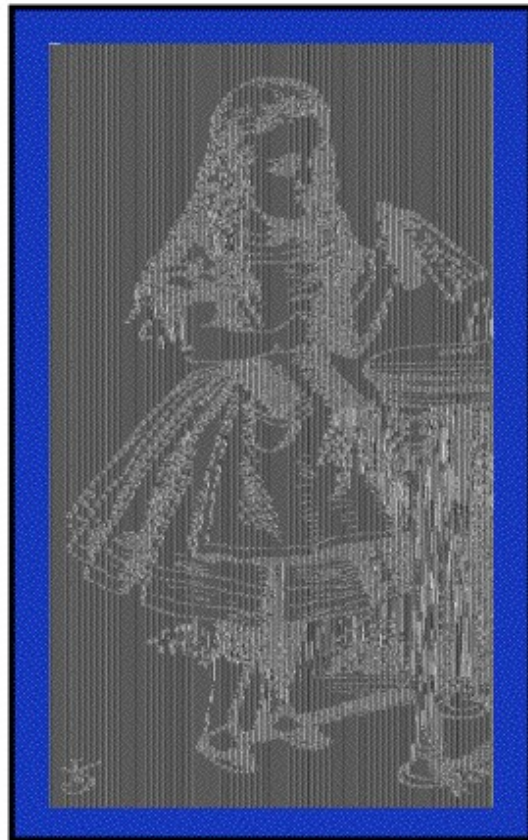
- Uma desvantagem é que os padrões de cifragem se repetem: “xyz” sempre gerará “%a0”, por exemplo. Algoritmos assim são susceptíveis a **ataques de repetição**, principalmente se não tiverem informação de temporalidade presente.
- E como este modo usa blocos de tamanho fixo, a mensagem deve ter um tamanho que é múltiplo do tamanho dos blocos. Se isto não ocorre (muito comum), o algoritmo tem que “encher” o final da mensagem com bits aleatórios (padding) que serão descartados no destino. E isto gera processamento extra indesejado.

# Modo de operação ECB

## CIFRAGEM



# Exemplo de cifragem usando ECB.



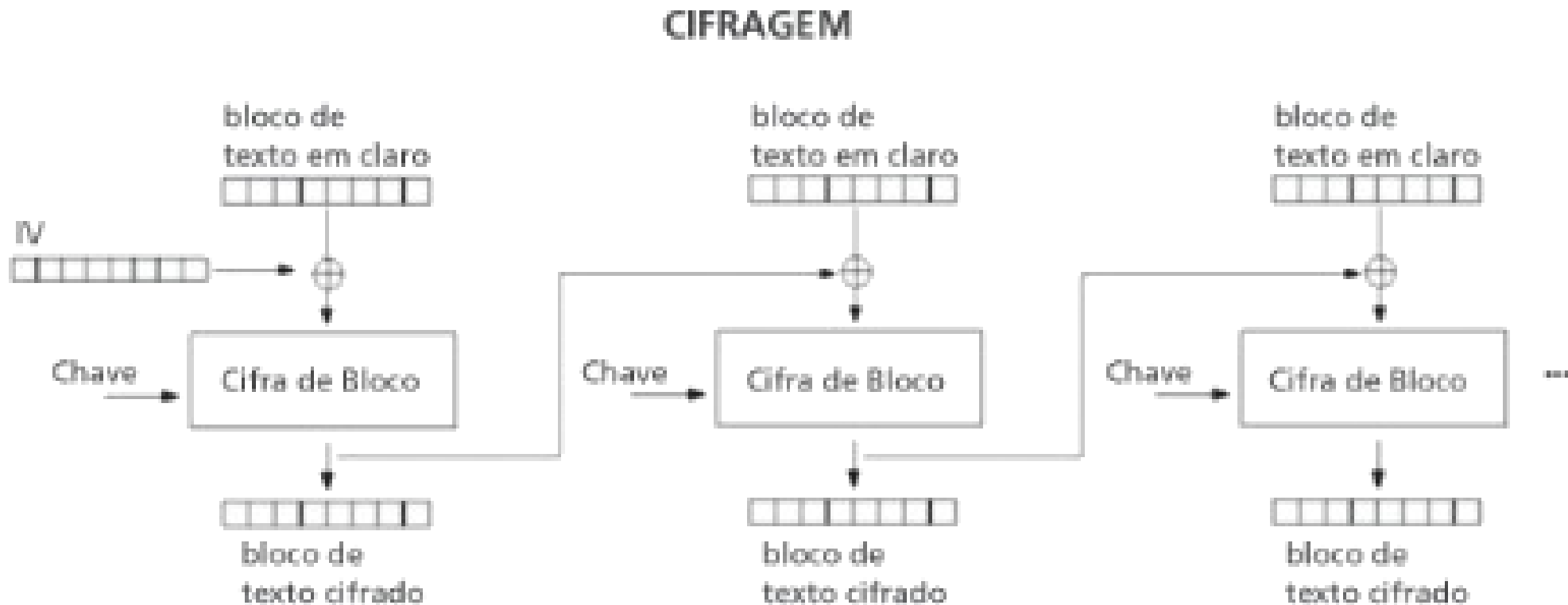
# Modo de operação CBC

- Cipher Block Chaining.
- Este método é mais complexo que o anterior, pois há uma relação entre o bloco cifrado e o próximo bloco, com texto em claro, de forma que um alimenta o outro.
- Uma vantagem clara é que, aqui, o ataque por repetição é mitigado: O texto em claro “xyz” pode gerar “%a0” ou “a3u”, por exemplo.
- Este modo de operação também trabalha com blocos de tamanho fixo, então precisa de preenchimento (padding), o que gera processamento extra.

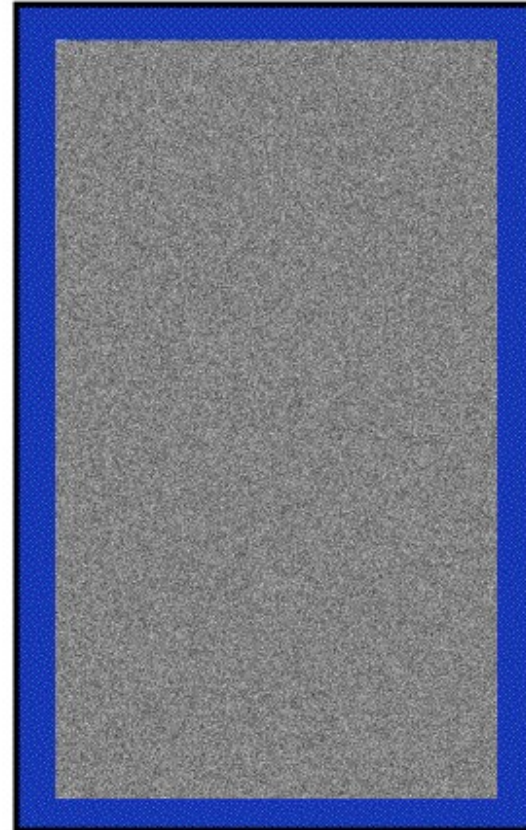
# Modo de operação CBC

- Uma desvantagem clara é que este modo não é paralelizável: Já que um bloco depende do outro, seu processamento é **sequencial**.
- Outra desvantagem é que, se um bit da mensagem cifrada for corrompida na transmissão, isto afetará toda a mensagem, pois **este erro será propagado aos outros blocos**.

# Modo de operação CBC



# Exemplo de cifragem usando CBC.



# Outros modos de operação.

- Existem outros modos, como o CTR (Counter), o CFB (Cipher Feedback) e o OFB (Output Feedback). Estes modos trabalham com fluxos contínuos de bytes, **sem ter que lidar com blocos de tamanho fixo.**
- Métodos como o CRC (verificação cíclica de redundância) é usado para detecção de erros, e pode também ser usado para garantir a integridade da mensagem.

# Conclusão e resumo.

- As funções de hash são usadas para garantir a integridade da mensagem.
- As funções de hash são “só de ida”. Não é possível decifrar uma informação após a sua cifragem usando um algoritmo de hash.
- As propriedades são: unidirecionalidade, condensação, efeito avalanche e resistência à 1ª, à 2ª e à 3ª inversões.
- Há modos de operação, como o ECB e o CBC, que lidam com blocos de texto de tamanho fixo. Ambos tem vantagens e desvantagens. Podem ser empregados pela mesma função de hash, se assim for desejado.
- Outros modos de operação são o CTR, o CFB e o OFB, que trabalham com fluxos contínuos de texto.