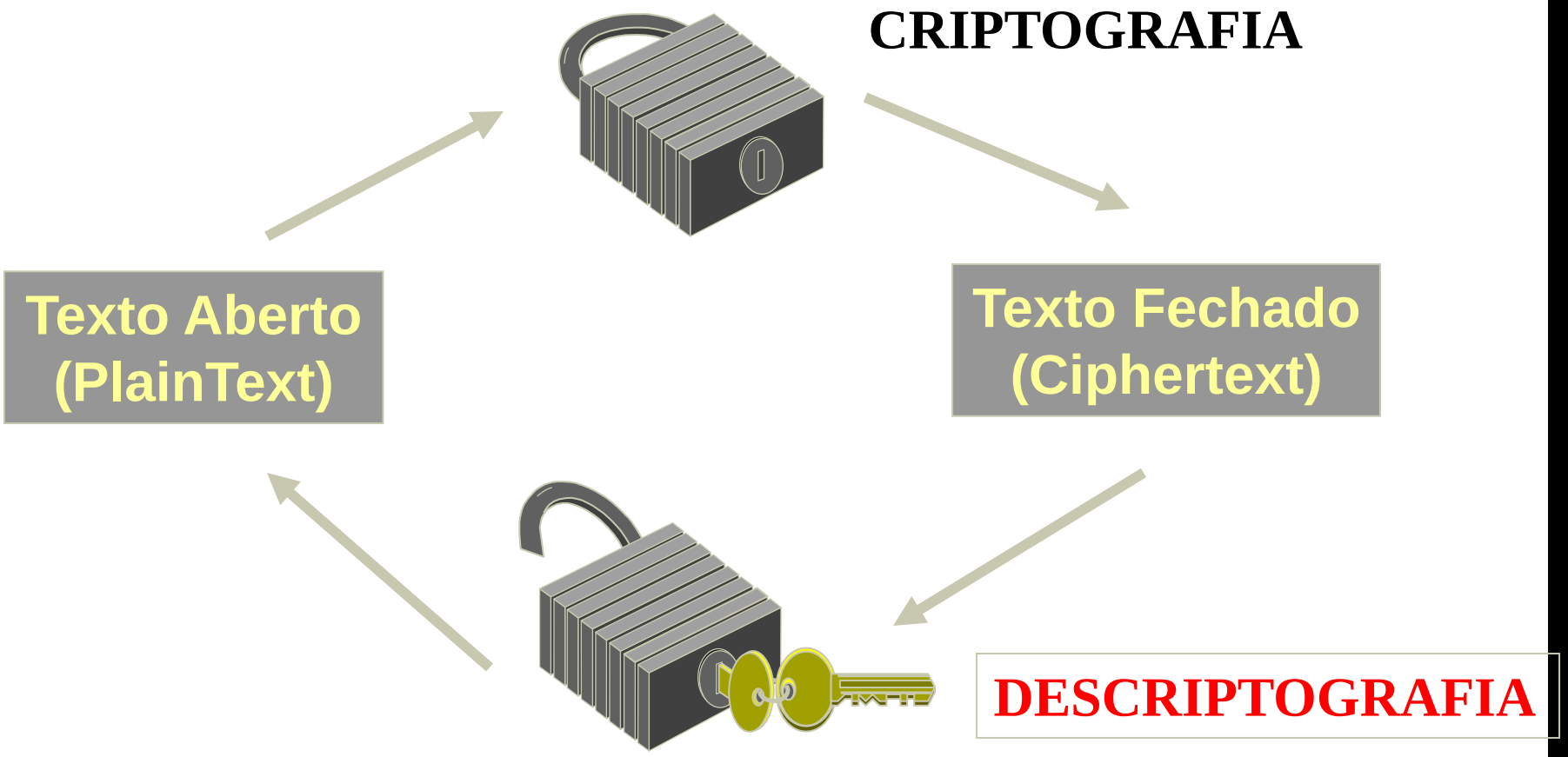
CRIPTOGRAFIA SIMETRICA E ASSIMÉTRICA

Professor

Edgard Jamhour

Criptografia e Descriptografia

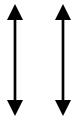


Sistema de Criptografia Simples

Caesar Cipher: Shift Cipher

Substituição de letras pelas letras deslocadas de N.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z



D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Nada de novo no front. —N = 3→ **Qdgd gh qryr qr iurqw.**

Nada de novo no front. —N = 4→ **Rehe hi rszs rs jvstx.**

Espaço das Chaves (KeySpace)

Uma chave é um valor específico do espaço de chaves (keyspace).

No exemplo anterior:

- Keyspace = 26
- N = 3, é a chave específica.

Algoritmos modernos:

- Chaves binárias: 128, 256, 1024, 2048 bits

Tipos de Criptografia:

- Simétrico: Keyspace $\cong 2^{\text{tamanho da chave}}$
- Assimétrico: Keyspace $<<<\ 2^{\text{tamanho da chave}}$

Marie-Antoinette and Axel von Fersen

A	AB	CD	EF	GH	IK	LM	NO	PQ	RS	TU	XY	Z&
B	AC	BK	DU	EI	FL	GN	HO	MY	PS	QX	RT	Z&
C	AD	BG	CZ	EK	FM	HT	IX	LR	NP	OQ	S&	UY
D	AE	BZ	CT	DK	FI	GS	HY	LQ	MX	NR	O&	PU
E	AF	BL	CI	DH	EU	GK	MT	NQ	OR	P&	SX	YZ
F	AH	BF	CL	DG	EQ	IY	KP	MU	NS	O&	RX	TZ
G	AG	BI	CL	DN	ER	FP	HT	KU	M&	OX	QY	SZ
H	AI	BT	CS	DO	EL	F&	GH	KM	NQ	PR	UY	XZ
I	AK	BT	CS	DX	EI	FL	GZ	HY	M&	NP	OQ	RU
K	AL	BO	CP	DG	ER	FS	HU	IX	KY	MZ	N&	QT
L	AM	BZ	CD	EG	FI	HK	LN	OR	PS	QU	TY	X&
M	AN	BO	CP	DQ	ER	FS	GT	HU	IX	KY	LZ	M&
N	AO	BC	DM	EP	FS	GN	HY	IU	KT	LQ	R&	XZ
O	AP	BL	CK	DQ	ES	FU	GX	HZ	I&	MO	NR	TY
P	AQ	BX	CU	DZ	ES	FO	GY	HT	IN	KR	L&	MP
Q	AR	BZ	CT	DH	EU	FQ	GO	IL	KN	MP	SY	X&
R	AS	BN	CQ	DT	EU	FY	G&	HO	IP	KR	LX	MZ
S	AT	BP	CQ	DR	E&	FS	GU	HX	IY	KZ	LN	MO
T	AU	BY	CM	DX	E&	FH	GQ	IR	KZ	LS	NP	OT
U	AX	BL	CO	DQ	ES	FU	GT	HY	IN	KZ	M&	PR
X	AY	B&	CZ	DE	FX	GU	HI	KT	LS	MR	NP	OQ
Y	AZ	BU	CG	DH	EX	FY	IO	K&	LN	MP	QS	RT

S. Tomokiyo

Cipher between Marie-Antoinette and Fersen (1791-1792)

l e r o i e t l a r e i n e

D E P U I S D E P U I S D E

Q U K C E & C B Q P I Y R U

Quiz 1

Qual o espaço de chaves do algoritmo utilizado por Maria Antoinette?

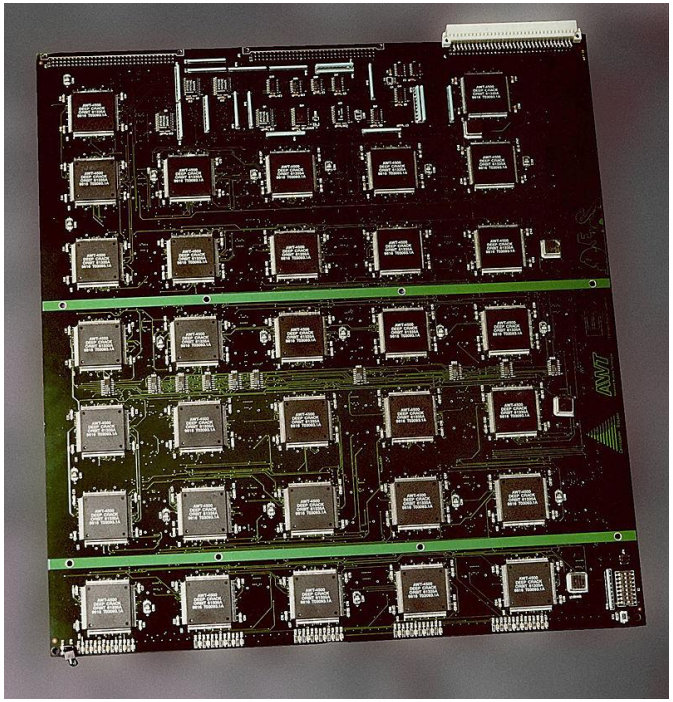
A. O mesmo do Caesar Cipher, pois é baseado no alfabeto.
B. É 2^{26}, pois o alfabeto tem 26 letras, e o algoritmo não diferencia maiúsculas de minúsculas
C. É exatamente 5^{26}.
D. É maior que 5^{26}.
E. É igual ao número de palavras no dicionário Francês.

Quebra de Criptografia

A quebra da criptografia utilizando força bruta é inviável para espaço de chaves acima de 128 bits.

- 2^{128} = 69 bilhões de vezes a massa da Terra medida em gramas

Keyspace = 2^{56}	Computador: Deep Crack (1856 chips) 72 quatrilhões de chaves 90 bilhões de chaves por segundo Tempo para encontrar uma chave: **56 horas**
Keyspace = 2^{128}	Computador: Hipotético 1 quintilhão de chaves por segundo Tempo para testar todas as chaves: **100 milhões de bilhões de anos**.



Quanto é 2^{128}?

Massa do Planeta Terra em Kg:

- 5.972×10^{24} kg

Massa do Planeta Terra em g:

- 5.972×10^{27} kg

Número de átomos no planeta Terra:

- 1×10^{50}

Valor numérico de 2^{128} em decimal:

- 340.000.000.000.000.000.000.000.000.000.000.000

Massa da Terra em gramas em decimal:

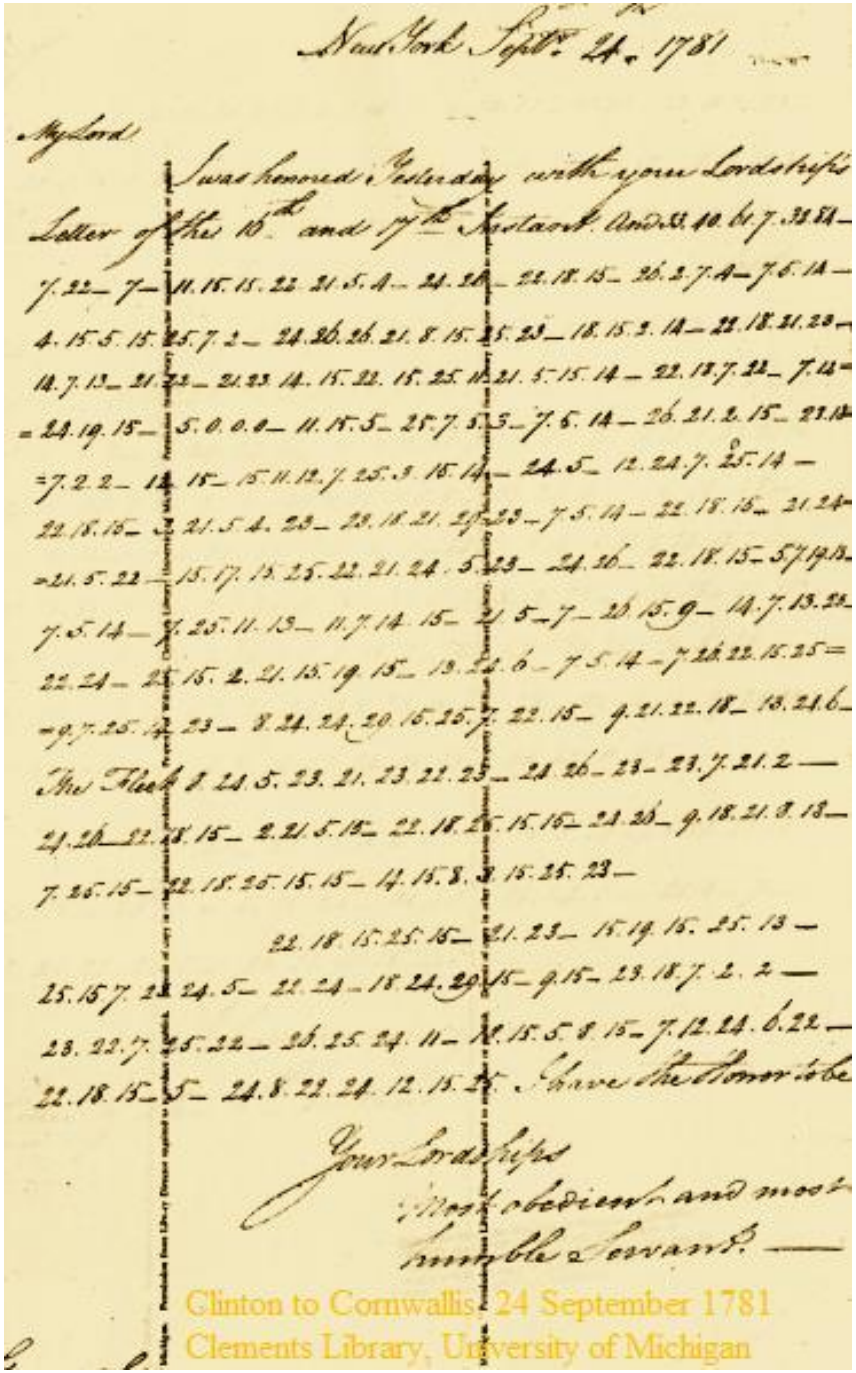
- 5.972.000.000.000.000.000.000.000

Número de átomos na Terra em decimal:

- 133.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000

Análise de Frequência

15 (87 times)	12 (11 times)
7 (56 times)	6 (11 times)
22 (54 times)	4 (9 times)
24 (42 times)	29 (8 times)
21 (41 times)	9 (8 times)
23 (40 times)	3 (7 times)
25 (40 times)	19 (5 times)
5 (38 times)	0 (3 times)
18 (35 times)	1 (1 time)
2 (26 times)	17 (1 time)
14 (23 times)	32 (1 time)
26 (18 times)	33 (1 time)
11 (14 times)	40 (1 time)
13 (13 times)	61 (1 time)
8 (12 times)	84 (1 time)



Análise de Frequência

Letras mais usadas em inglês: ETAONISH

- 15=E

Palavra mais usada em inglês: THE

A sequência: 22 18 15 é a que se repete mais vezes (12)

- 22=T, 18=H e 15=E

Outras deduções:

T H 25 E E = THREE; T H 25 E = THERE

- 25 = R

Analisando a sequência:

"a 26 t e r = 9 a r 14 23 - 8 24 24 29 e r a t e - 9 21 t h - y 24 6“

“a F t e r = W a r D S C O O P e r a t e W I t h y O U

Criptografia Simétrica e Assimétrica

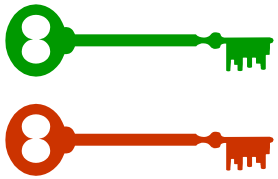
Dois sistemas de criptografia são usados atualmente:

Sistemas de chave secreta (**secret-key**)

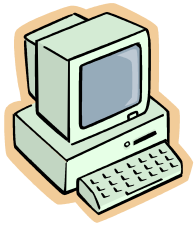
- Também denominados simétricos
- Trabalha com uma única chave, denominada SECRETA.

Sistemas de chave pública (**public-key**)

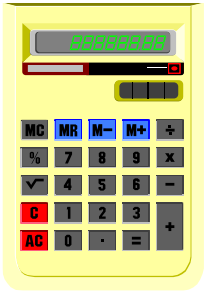
- Também denominado assimétrico
- Trabalho com um par de chaves
  - CHAVE PÚBLICA
  - CHAVE PRIVADA



Chave Secreta (Criptografia Simétrica)



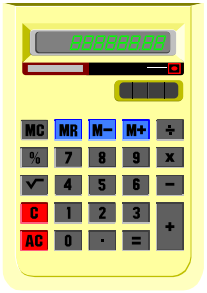
Algoritmo de Criptografia

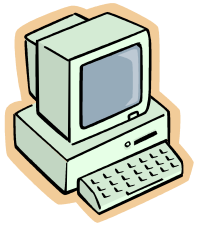


Texto Simples (plaintext)

Texto Codificado (ciphertext)

Algoritmo de Decriptografia





Texto Simples (plaintext)

Chave Secreta

=

Chave Secreta

Criptografia Simétrica e Chave de Sessão

A **chave de simétrica** ou **secreta** não é sempre a mesma

A troca de chaves é feita em cada nova **sessão**

A troca de chaves pode ser feita durante uma sessão usando como condição para a troca o tempo de uso de uma mesma chave ou o volume de tráfego já criptografado

Reutilizar uma chave ou usar sempre a mesma chave é considerado uma **falha de segurança**

- Se uma chave for descoberta, tudo que foi que foi feita por ela fica desprotegido.

Como a chave simétrica muda a cada sessão, ela é as vezes referenciada como **chave de sessão**

Quiz 2

Do ponto de vista do uso da criptografia, o que é uma sessão? Indique todas que aplicarem.

A. Todos os pacotes trocados com um mesmo computador de destino.
B. Todas as páginas acessadas em um mesmo servidor Web.
C. Todas as mensagens trocadas após a autenticação do usuário.
D. Todas as mensagens trocadas após um certo intervalo de tempo.
E. Todas as mensagens trocadas até que um volume de tráfego seja atingido.

Block Ciphers vs Stream Ciphers

Block Ciphers dividem a mensagem em blocos de tamanhos iguais

- 64-bits ou 128-bits

As operações de criptografia são feitas sobre os blocos e o resultado pode levar em conta também o resultado da criptografia de outros blocos

Stream Ciphers criptografam a mensagem como um fluxo contínuo de bits ou bytes

- Dados enviados em fluxo contínuo, como Video ou Audio.
- Precisa garantir que a chave de criptografia nunca seja reusada

Block Ciphers são mais eficientes quando o tamanho da mensagem a ser criptografada é previamente conhecido

XOR Cipher

Algoritmos do tipo stream utilizam frequentemente uma simples operação lógica de XOR para criptografar os dados.

Esses algoritmos são chamados de ciphers aditivos, e tem as seguintes propriedades:

$$A \oplus 0 = A,$$

$$A \oplus A = 0,$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C),$$

$$(B \oplus A) \oplus A = B \oplus 0 = B,$$

Exercício:

Utilizando o Shell do Python, execute as seguintes operações e anote o resultado:

1) 65 ^ 0

2) 65 ^ 65

3) 65 ^ 64

4) 65 ^ 64 ^ 97

5) 97 ^ 64 ^ 65

6) 65 ^ 64 ^ 64

7) 65 ^ 64 ^ 65

8) 65 ^ 64 ^ 97 ^ 64

9) 65 ^ 97

Quiz 3

Em relação ao exercício anterior, se ‘A’ for o dado e ‘@’ for a chave, o que significam as operações 6 e 7?

A. São exemplos de dupla criptografia.

B. A operação 6 é descriptografia e a operação 7 é dupla criptografia.

C. Ambas as operações não fazem sentido, pois o XOR é sempre entre dois elementos apenas.

D. A operação 6 é descriptografia e a operação 7 mostra como é simples descobrir a chave de criptografia.

Exemplo:

"Wiki" (01010111 01101001 01101011 01101001 in 8-bit ASCII)

Suponha que foi escolhida a chave: 11110011

Criptografia:

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
-----------------------------------
= 10100100 10011010 10011000 10011010
```

Descriptografia:

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
-----------------------------------
= 01010111 01101001 01101011 01101001
```

Exercício

Escreva um programa em Python que implementa um XOR cipher, definindo a função **xor_cipher** que está faltando no código ao lado.

Observação:

ord(caracter) => código

chr(código) => caractere

```
def xor_cipher(plain, key):
    return “esqueci de fazer o exercicio”

while True:
    key = input('entre com a chave: ')
    if not key:
        print("saindo ... ")
        break
    else:
        key = int(key)
    plain = input('entre com a mensagem: ')
    cipher = xor_cipher(plain, key)
    print("cripto:", cipher)
    print("decripto:", xor_cipher(cipher, key))
```

Problemas com o XOR Cipher: Colisão do PlainText e CipherText

Suponha que duas mensagens A e B sejam criptografadas com a mesma chave X:

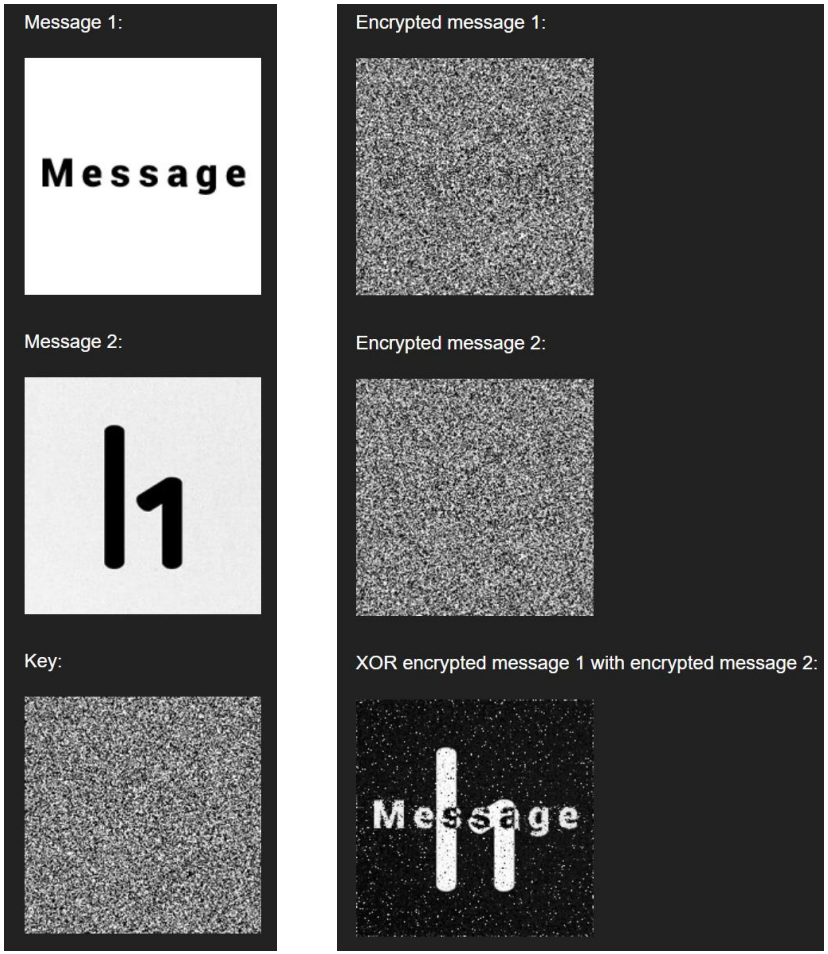
- E(A) = A XOR X
- E(B) = B XOR X

De acordo com as propriedades do XOR a seguinte relação é equivalente:

- E(A) XOR E(B) = A XOR B

Veja as consequências desse problema em exemplo1 e exemplo2.

A maneira de se proteger é nunca repetir a chave. Isso é feito combinando a chave com um outro valor na forma de um contador, por exemplo.



Exercício

Utilize as funções em Python fornecidas para fazer a verificação da propriedade:

A XOR B = E(A) XOR E(B)
e suas consequências.

```
def xor_cipher(plain, key):

    hkey = []
    result = []

    for i in key:
        hkey.append(ord(i))
    i = 0
    for c in plain:
        result.append(ord(c)^hkey[i])
        i += 1
        if i == len(key):
            i = 0
    return result

def to_ascii(result):
    cipher = ''
    for i in result:
        cipher += chr(i)
    return cipher
```

Problemas com o XOR Cipher: Tampering

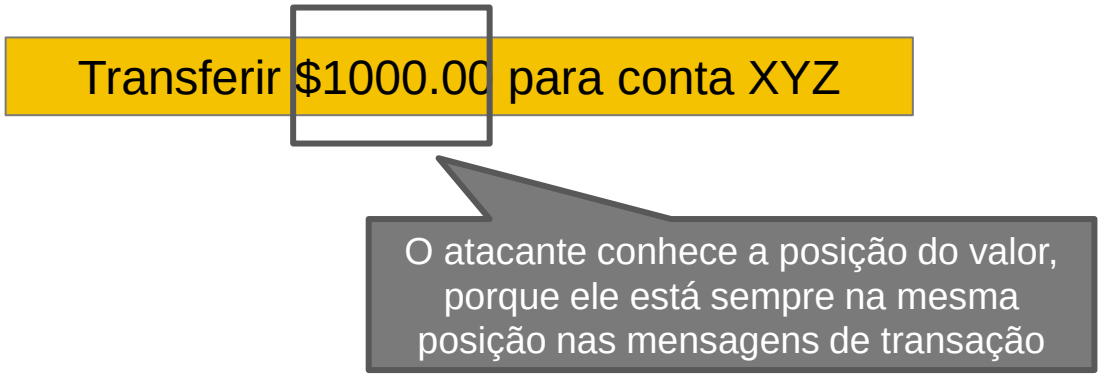
Ataque BIT FLIPPING: falsificar partes da mensagem sem conhecer a chave.

- Também definido com Replay e Man-in-The-Middle

Suponha que a comunicação entre Alice e Bob foi interceptada, mas está criptografada com a chave C(K)

- CHAVE xor Transferir $1000.00 para conta XYZ

Caso o atacante conheça a posição do valor $1000.00, ele pode alterar o valor sem conhecer a chave



Problemas com o XOR Cipher: BIT FLIPPING e ICV

A alteração pode ser feita com o seguinte procedimento:

- (CHAVE xor "$1000.00") xor ("$1000.00" xor "$9500.00") =
- CHAVE **xor "$1000.00“xor "$1000.00"** xor "$9500.00" =
- C(K) xor "$9500.00"

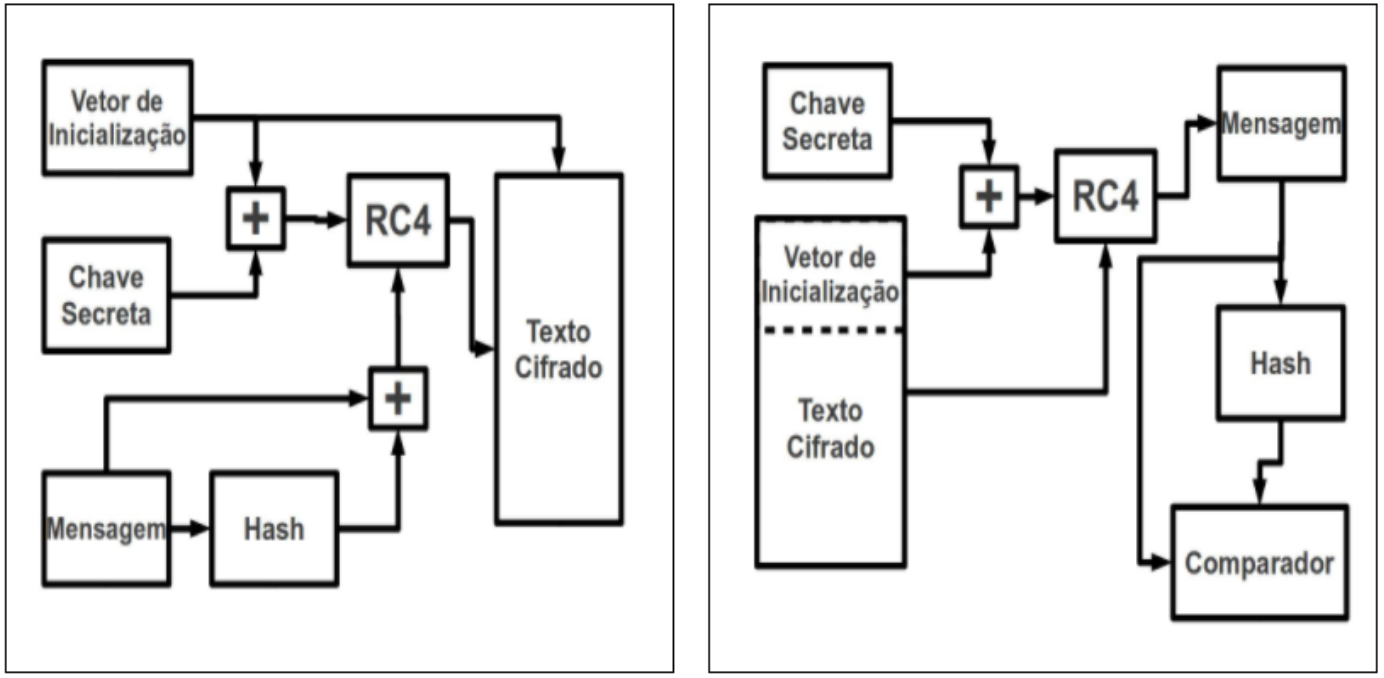
A maneira de proteger-se contra esse ataque é incluir um código de verificação de integridade na mensagem.

Esse código é denominado MAC ou ICV:

- MAC: Message Authentication Code
- ICV: Integrity Check Value

Vetor de Inicialização: IV

Uma maneira de contornar o problema é evitar que a mesma chave seja reutilizada utilizando um vetor de inicialização transmitido em PlainText junto com a mensagem.



(a) Encriptação (b) Decriptação

Figura 2: Diagrama de Blocos do Protocolo WEP

Confusion and Diffusion

Confusion significa que cada bit da mensagem criptografada depende de vários bits da chave.

- Esconde o relacionamento entre a chave e a mensagem criptografada
- Torna difícil encontrar a chave por aproximações sucessivas

Diffusion indica que pequenas mudanças no plaintext ou na chave devem gerar grandes mudanças na mensagem criptografada.

- Um bit modificado na chave ou no plaintext deve alterar aproximadamente metade dos bits do ciphertext.
- Esconde o relacionamento entre o plaintext e o ciphertext

A qualidade de um algoritmo de criptografia é fortemente dependente dessas duas características.

Como saber?

Confusion

Se você criptografar a mesma mensagem com uma chave levemente diferente, o resultado muda completamente.

- Mesma mensagem
- Chaves diferentes

Diffusion

Se você criptografar um mensagem levemente diferente com a mesma chave, o resultado muda completamente.

- Mensagens diferentes
- Mesma chave

Quiz 4

Quais propriedades o algoritmo de XOR cipher simples, sem vetor de inicialização, satisfaz?

A. Apenas Confusion.

B. Apenas Difusion.

C. Confusion e Difusion.

D. Nenhuma das duas.

E. Depende do tamanho da chave.

Algoritmo RC4 em Python

Para executar esse Código é necessário instalar a biblioteca cryptography:

pip install cryptography ou python -m pip install cryptography

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
chave = b'12345'
algorithm=algorithms.ARC4(chave)
cipher=Cipher(algorithm,mode=None)
encryptor=cipher.encryptor()
ciphertext=encryptor.update(b'Special charactors are not allowed')
print(ciphertext)
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext)
print(plaintext)
```

Quiz 5

Utilize o cógido anterior verifique quais propriedades o algoritmo RC4 satisfaz?

A. Apenas Confusion.

B. Apenas Difusion.

C. Confusion e Difusion.

D. Nenhuma das duas.

E. Depende do tamanho da chave.

Problema do STREAM CIPHER

Observe que é difícil (ou IMPOSSÍVEL) obter a propriedade de difusão se um byte é criptografado de cada vez.

Para conseguir difusão é preciso conhecer o conjunto de bytes que precisa ser criptografado antes de iniciar a criptografia.

Na prática isso limitaria muito o uso da criptografia na transmissão de dados.

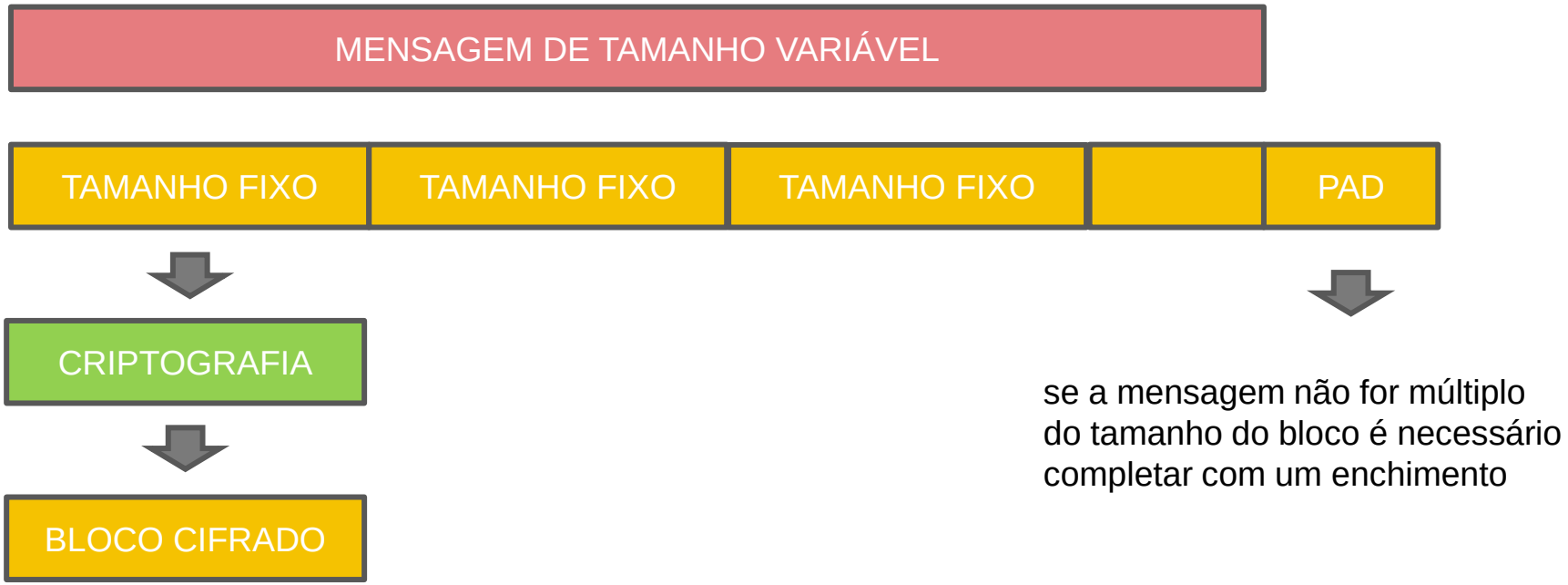
Então optou-se por uma solução intermediária;

DECOMPOR A MENSAGEM EM BLOCOS

Block Ciphers

Permite melhorar a segurança através de operações simples de permuta e substituição aplicadas sucessivas vezes sobre um conjunto de bytes fixos.

- Alta difusão
- Maior imunidade a inserção de símbolos no plaintext sem detecção



Difusão no Block Cipher

Se um único bit for alterado, ele precisa mudar completamente o resulltado da criptografia.

Dois blocos com apenas este bit diferente precisam gerar blocos criptografados com muitos bits diferentes.

ABCDEFGH → ¢¤¤£¢®¬ª

BBCDEFGH → ÁßÜØ»¶A*

E a maneira como isso acontece precisa depender da chave

E precisa parecer aleatória, para evitar ciptoanálise diferencial

CipherModes (Block Cipher Mode of Operation)

ECB: Electronic Codebook Mode

- Deprecated - considerado inseguro

CBC: Cipher Block Chaining

- Concatena cada bloco com o anterior
- **Pipeline Delays**: não pode ser processado em paralelo

Counter (CTM ou CTR ou ainda CM)

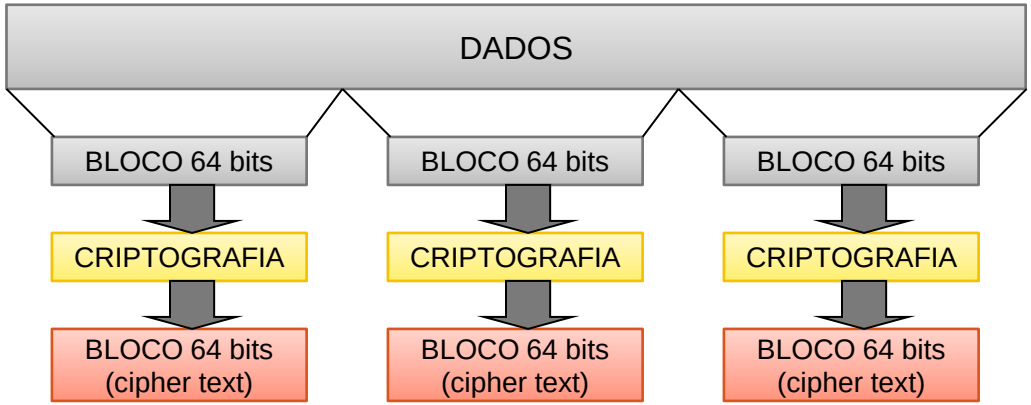
- Cria chaves diferentes para cada bloco como um **stream cipher**
- Pode ser processado em **paralelo**

Galois/Counter Mode (GCM)

- Combina CTM com **Galois Mode of authentication**, inserindo um hashing nos blocos
- Provê autenticidade (integridade) e confidencialidade

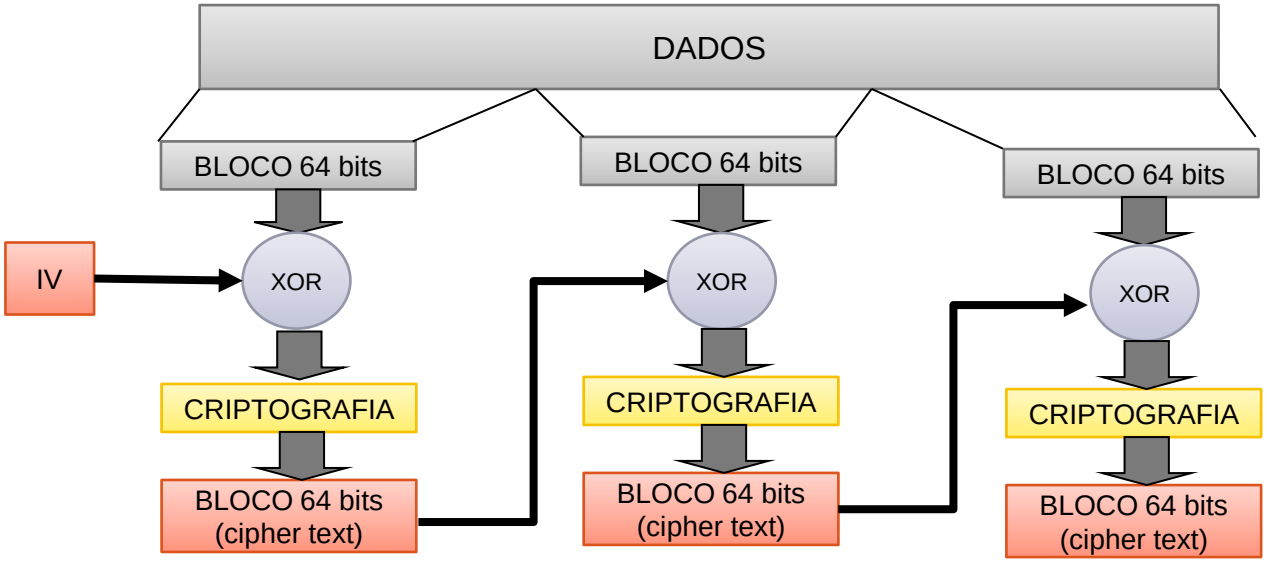
MODO ECB

O Modo ECB divide a mensagem em blocos de 64 bits, e criptografa cada bloco de maneira independente. Deprecated.



MODO CBC

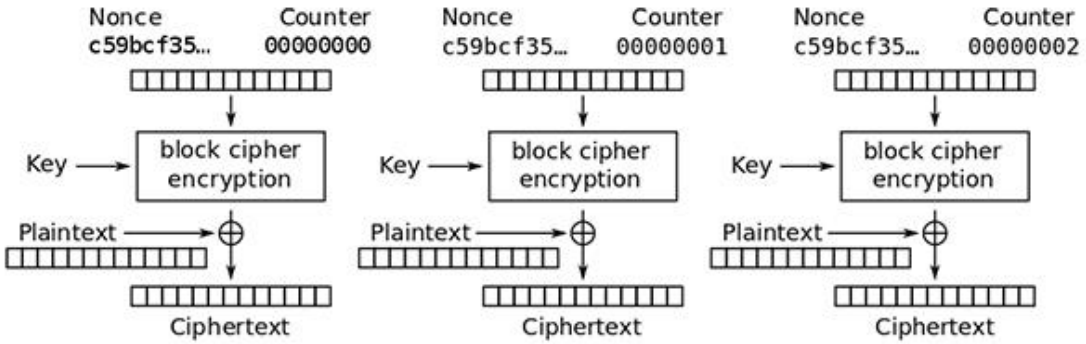
Garante que blocos de PLAINTEXT idênticos gerem blocos de CIPHERTEXT diferentes.
Tem a desvantagem de não suportar paralelismo.



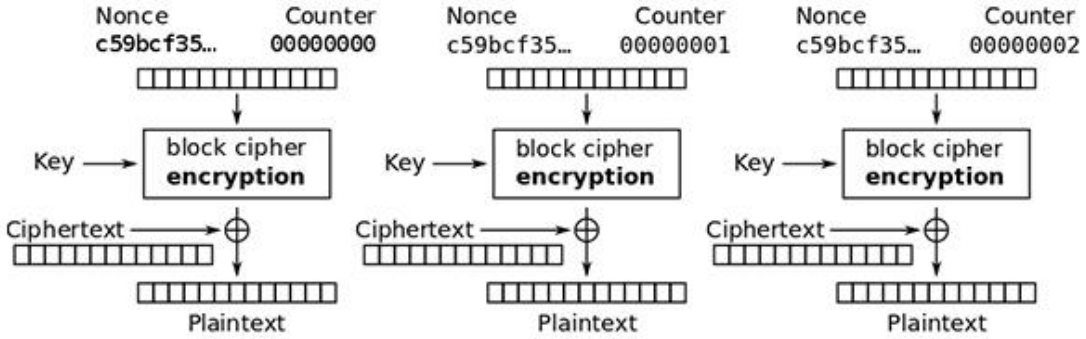
CTM/CTR (Counter Mode)

Faz um XOR de cada bloco com um contador sequencial combinado com um Nonce.

Permite que cada bloco seja processado de forma independente.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

NONCE = Number Only Used Once
- Número randômico que precisa ser diferente cada vez que é gerado

GCM (Galois/Counter Mode)

conteúdo informativo

Esse modo é usado nas versões modernas de WiFI.

Tem a vantagem de combinar criptografia com AUTENTICAÇÃO.

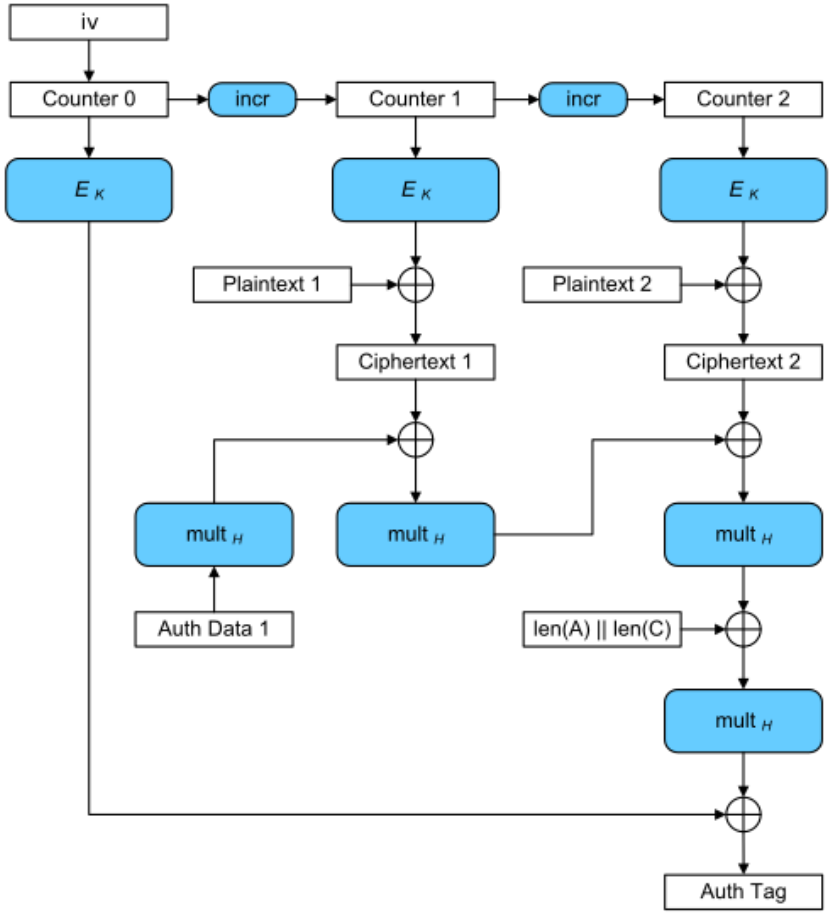
Galois significa campo de tamanho finito, geralmente obtido com expressões do tipo (inteiro mod primo)

Fornece criptografia autenticada com alto desempenho.

Os blocos tem tamanho fixo de 128 bits e são numerados sequencialmente combinados com um IV.

Ek= criptografia com a chave K

multH = multiplicacação com a Hash Key H



Algoritmos Simétricos (Quadro Resumo)

Algorithm	Type	Method	Key Size
AES	Symmetric encryption	128-bit block cipher	128-, 192-, or 256-bit key
3DES	Symmetric encryption	64-bit block cipher	56-, 112-, or 168-bit key
Blowfish	Symmetric encryption	64-bit block cipher	32- to 448-bit key
Twofish	Symmetric encryption	128-bit block cipher	128-, 192-, or 256-bit key
RC4*	Symmetric encryption	Stream cipher	40- to 2,048-bit key
DES*	Symmetric encryption	64-bit block cipher	56-bit key

RC4 e DES estão deprecated (e não fazem parte dos objetivos do COMPTIA)

Algoritmos Simétricos Usados Atualmente

AES: Advanced Encryption Standard

- Derivado dos algoritmos Rijndael
- Publicado pelo NIST em 2001
  - NIST: National Institute of Standards and Technology
- Adotado pelo governo Americano
- Baseado em SPN: Substitution-Permutation Network (Similar ao DES)

Criptografa os dados em blocos de 128 bits

Variantes com chaves de 128, 192 e 256-bits

Mais rápido e com menos consumo de recursos que o DES

Outros Algoritmos

3DES (Triple DES)

- Objetivo: dar sobrevida ao DES
- Pode aplicar o DES com até 3 chaves diferentes ($E_{K3}(D_{K2}(E_{K1}(plaintext))$
- Chaves de 168 (três chaves diferentes), 112 (K3=K1) e 56 bits (k1=K2=K3)

RC4

- Stream Cipher desenvolvido por Ron Rivest
- Pode usar chaves entre 40 e 2048 bits
- Passou a ser substituído pelo AES a partir de 2013

Outros Algoritmos

Blowfish

- Usa blocos de 64-bits e chaves entre 32 e 448 bits
- Projetado por Bruce Schneier para substituir o DES
- Pode ser mais rápido que o AES para alguns tamanhos de chave (256)

Twofish

- Usa blocos de 128-bits e chaves de 128, 192 ou 256 bits
- Foi um dos finalistas escolhidos pelo NIST mas foi derrotado pelo AES

DES – Data Encryption Standard

Um dos algoritmo de chave secreta mais difundido é o DES.

- Originalmente desenvolvido pela IBM (baseado no algoritmo de Host Feistel)
- Aprovado pelo NSA (National Security Agency) e padronizado NBS (atual NIST) em 1997.
- Deprecated

DES criptografa blocos de 64 bits com chaves de 56 bits.

- DES utiliza técnicas baseadas em substituição e permutação de bits (funções Feistel).

1998:	DES-cracker da Electronic Frontier Foundation (EFF) 1850 chips desenvolvidos especialmente para quebrar o código Custo de US$250.000, quebrou o algoritmo em 2 dias.
2008:	COPACOBANA RIVYERA (128 Spartan-3 5000's), Custo de US$10.000, quebrou o algoritmo em menos de um dia.

Cifras baseadas em Feistel

Dado:

Sequência de chaves: K0, ..., Kn

Bloco a ser criptografado:

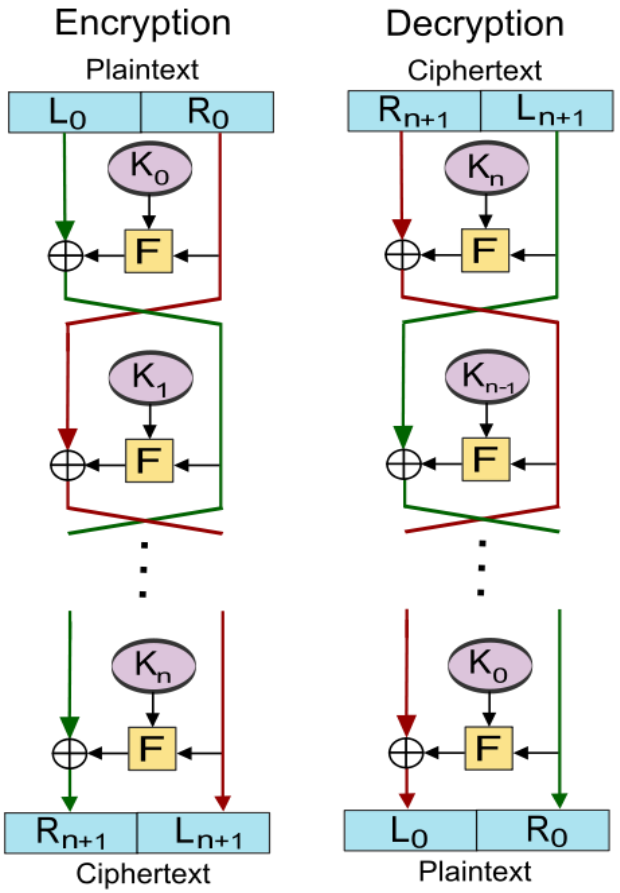
- Bloco é dividido em 2 partes iguais

Criptografia: n+1 rounds

- Li+1 = Ri
- Ri+1= Li ⊕ F(Ri,Ki)

Descriptografia:

- mesma operação com chaves na ordem reversa
- Ri = Li+1
- Li= Ri+1 ⊕ F(Li+1,Ki)



Exemplo: Criptografia Usando F = XOR

PRIMEIRO ROUND:

1. Gere a função F_0 usando R_0 e K_0:
   - $F_0 = \text{xor}(R_0, K_0)$
2. Calcule os novos lados direito R_1 e esquerdo L_1:
   - $R_1 = \text{xor}(F_0, L_0)$
   - $L_1 = R_0$

SEGUNDO ROUND:

1. Gere a função F_1 usando R_1 e K_1:
   - $F_1 = \text{xor}(R_1, K_1)$
2. Calcule os novos lados direito R_1 e esquerdo L_1:
   - $R_2 = \text{xor}(F_1, L_1)$
   - $L_2 = R_1$

Exemplo: Descriptografia Usando F = XOR

Inicialize: $L_0 = R_2$ e $R_0 = L_2$

PRIMEIRO ROUND:

1. Gere a função F_0 usando R_0 e K_1:
   - $F_0 = \text{xor}\ (L_0, K_1)$
2. Calcule os novos lados direito R_1 e esquerdo L_1:
   - $R_1 = \text{xor}\ (F_0, L_0)$
   - $L_1 = R_0$

SEGUNDO ROUND:

1. Gere a função F_1 usando R_1 e K_0:
   - $F_1 = \text{xor}\ (R_1, K_0)$
2. Calcule os novos lados direito R_1 e esquerdo L_1:
   - $R_2 = \text{xor}\ (F_1, L_1)$
   - $L_2 = R_1$

Quiz 6

Quais propriedades a cifra de Feistel satisfaz (use o código em Python fornecido no Blackboard)?

A. Apenas Confusion.
B. Apenas Difusion.
C. Confusion e Difusion.
D. Nenhuma das duas.
E. Depende da função de Feistel utilizada.

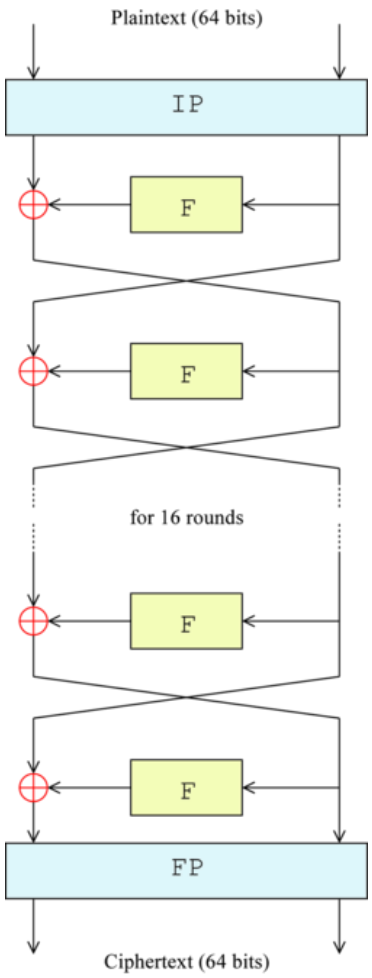
DES - Estrutura

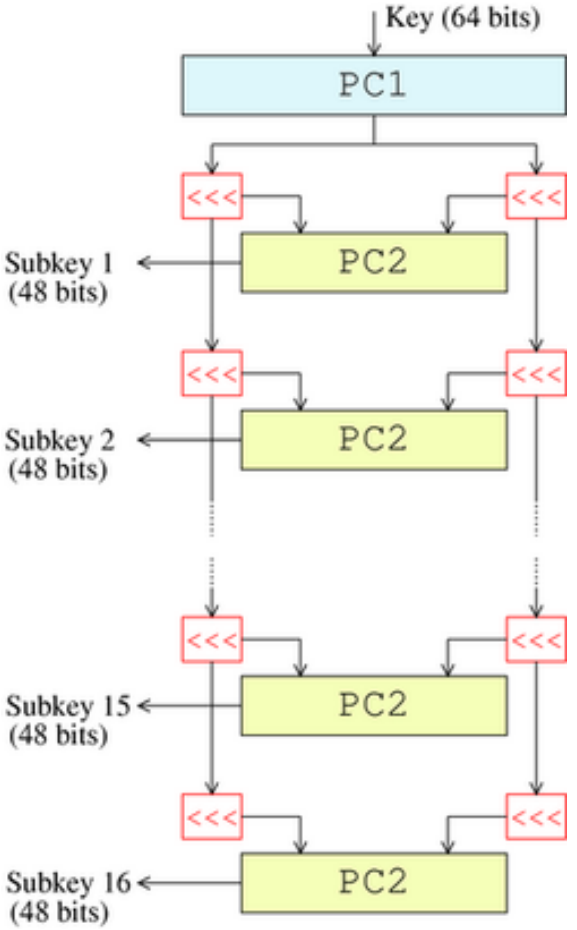
conteúdo informativo

1. Uma permutação inicial (IP)
2. 16 rounds de processamento (função de Feistel –F)
3. Uma permutação final (FP)

16 chaves de 48 bits diferentes são geradas a partir da chave original de 56 bits usando operações de “shift left” e “permutation choice”

Chave: sequência de 16 chaves



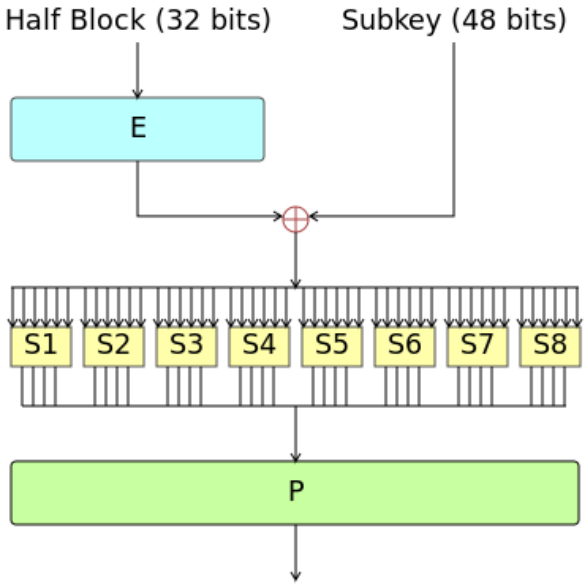


Função de Feistel

conteúdo informativo

É composto de 4 operações:

- **Expansion**: expande o bloco de 32 bits em 48, duplicando 16 bits.
- **Key Mixing**: efetua um XOR entre o bloco expandido e uma subchave
- **Substitution** : divide o bloco em partes de 6-bits e aplica o (S-Boxes = Substitution Boxes).
  - Cada um dos 8 S-boxes substitui uma parte de 6 bits por outra de 4 bits através de uma operação tabelada.
- **Permutation** : efetua uma permutação nos 32 bits para espalhar os bits de cada S-box em pelo menos 4 S-boxes diferentes para o próximo round.



Bloco de 32 bits

S-BOX = Substitution BOX

Técnica usada por algoritmos simétricos para dificultar a criptoanálise diferencial, escondendo a relação entre a chave e o ciphertext.

Essa é a tabela S5. O DES utiliza 8 tabelas diferentes

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

011011 → 1001

111010 → 0011

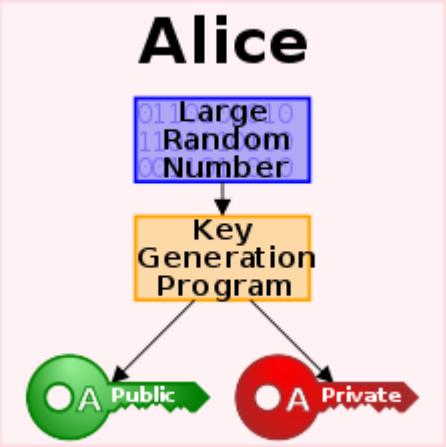
Difusão: espalhamento do efeito de um bit nos demais

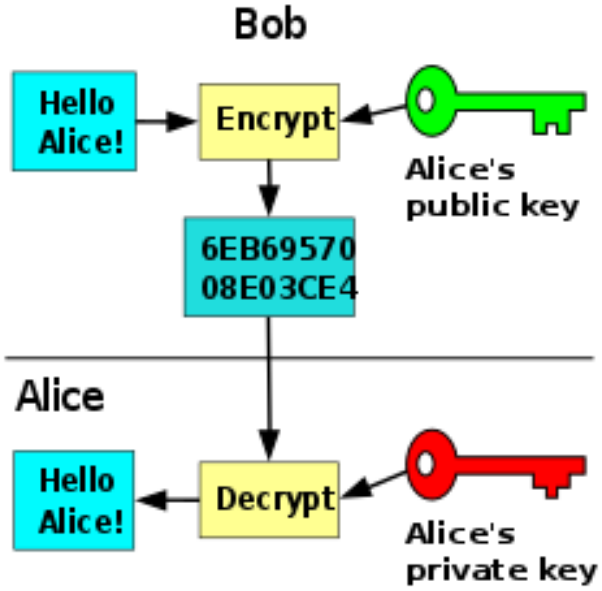
Chave Pública = CRIPTOGRAFIA ASSIMÉTRICA

Sistema de Criptografia Assimétrica = PAR DE CHAVES

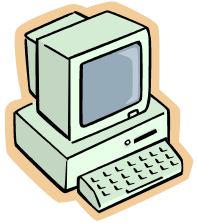
- A chave **publica** (não é secreta) criptografa a mensagem.
- A chave **privada** (é secreta) descriptografa a mensagem.

A chave pública deve ser distribuída para os transmissores para garantir comunicação segura com o receptor.



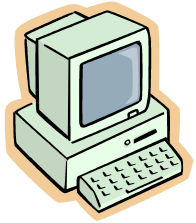


Chave Pública (Criptografia Assimétrica)

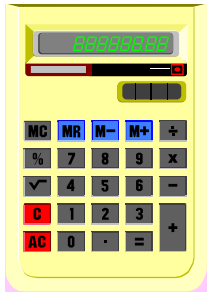


Algoritmo de Criptografia

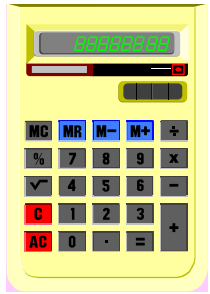
Algoritmo de Descriptografia



Texto Simples (plaintext)



Texto Codificado (ciphertext)



Texto Simples (plaintext)

Chave Pública

≠

Chave Privada

TRAPDOOR FUNCTION

Criptografia de chaves públicas precisam encontrar funções do tipo TRAPDOOR

Essas funções são fáceis de calcular em uma direção, mas muito difíceis no sentido reverso

Exemplo:

- Dado x, y = f(x) é fácil de calcular
- Mas dado y, x = $f^{-1}(x)$ é um problema muito complexo

Nas aplicações de criptografia:

- X = Chave Privada
- Y = Chave Pública

Algoritmos de Chave Pública

RSA: Rivest–Shamir–Adleman

- **Princípio: fatoração de números primos**
- **Patente: 1977**

ECC: Eliptic-Curve Cryptograpy

- **Princípio: curvas elípticas sobre campos finitos**
- **Conhecido a muito tempo, mas explorado a partir de 2015**

Diffie-Hellman Key Exchange

- **Princípio: cria um segredo compartilhado usando chaves públicas**
- **Publicado pelos autores em 1977, logo após o RSA**

RSA (Rivest, Shamir, Adleman)

Sejam **p**, **q** números primos (> 512 bits).

- **n** = **p***q**
- escolher **e co-primo** com n tal que: 1 < **e** < (**p**-1)(**q**-1)
- encontrar **d** tal que: **e***d** % (**p**-1)(**q**-1) = 1

n é o valor máximo do dado criptografado

As chaves são definidas da seguinte maneira:

- Chave pública: (**n**,**e**)
- Chave privada: (**n**,**d**)

Para criptografar uma mensagem "**m**" efetua-se a operação:

- **s** = $\mathbf{m}^{\mathbf{e}}$ mod **n**

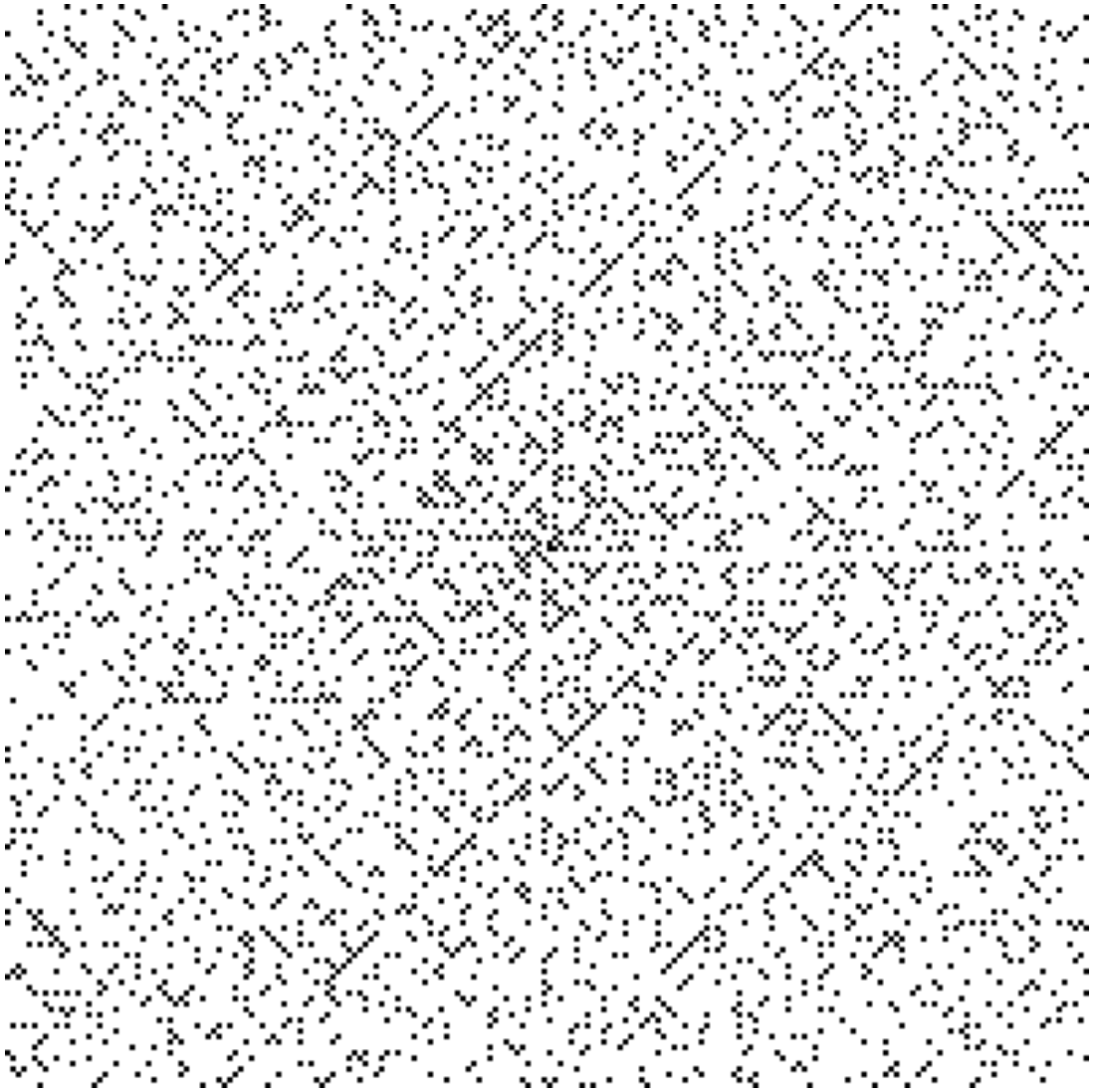
mod n faz com que o resultado seja sempre menor que n

Para descriptografar, efetua-se a operação:

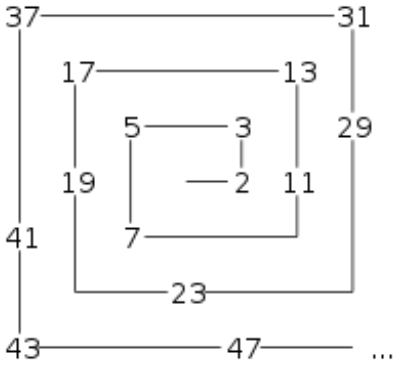
- **m** = $\mathbf{s}^{\mathbf{d}}$ mod **n**

Números Primos

RSA Factoring Challenge - Wikipedia



Ulam spiral
200 X 200



RSA Algorithm Example

1. Escolher: p = 3 e q = 11
2. Calcular: n = p * q = 3 * 11 = 33
3. Calcular: φ = (p - 1) * (q - 1) = 2 * 10 = 20
4. Escolher: e tal que 1 < e < φ e e e n são co-primos

solução possível: e = 7

5. Calcular: d tal que (d * e) % φ(n) = 1

solução possível: d = 3
pois (3 * 7) % 20 = 1

6. Escolher: chave pública = (e, n) ⇒ (7, 33)
7. Escolher: chave privada = (d, n) ⇒ (3, 33)

8. Criptografar: $c = m^e \bmod n$: se m = 2 ⇒

cipher text: $c = 2^7 \% 33 = 29$

9. Descriptografar: $m = s^d \bmod n$: se c = 29 ⇒

plain text: $m = 29^3 \% 33 = 2$

Por que mod n (% n)?

Mod n é uma forma de limitar um resultado em um valor máximo.

- Essa operação também é muito usada em operações de HASHING

No RSA o valor máximo de um dado criptografado é n (n = p * q)

Suponha a chave privada (n = 231, e = 3)

Se o resultado de m^e for menor que 231, o dado não muda:

- m = 2: m^e = 8
  8 % 231 = 8

Se o resultado de m^e for maior que 231, o dado é ajustado:

- m = 7: m^e = 343
  343 % 231 = 112

O que são coprimos?

Um número primo é um número divisível apenas por 1 e por ele memo

Coprimo é um conceito que se aplica a dois números.

- Coprimos são números primos entre si
- Isto é não tem divisor em comum (além do 1)

Exemplo: 12 e 15 não são coprimos

- Divisores do 12: 2, 3, 4, 6, 12
- Divisores do 15: 3, 5, 15

Exemplo: 15 e 14 são coprimos

- Divisores do 15: 3, 5, 15
- Divisores do 14: 2, 7, 14

Exercício

Utilize as funções fornecidas no próximo SLIDE para testar o algoritmo RSA

- Use: **p = 61** e **q = 53**
- Verifique se **e = 17** é uma escolha possível
- Calcule o valor de **d**
- Determine os valores da chave pública
- Determine os valores da chave privada
- Criptografe o número **123** com a chave pública
- Recupere o número original descriptogrando com a chave privada

Funções usadas no RSA

```
def find_primos(lower, upper):
    primos = [ ]

    for num in range(lower, upper + 1):
       if num > 1:
           for i in range(2, num):
               if (num % i) == 0:
                   break
           else:
               primos.append(num)
    return(primos)


def find_divisores(num):
    divisores = [ ]

    for i in range(2, num+1):
        if (num % i) == 0:
            divisores.append(i)

    return(divisores)
```

```
def testa_coprimo(a,b):
    x = find_divisores(a)
    y = find_divisores(b)
    if len(list(set(x) & set(y))) == 0:
        return (True)
    else:
        return (False)


def encontra_d(e, F):
    for i in range(0,10000):
        if (i * e) % F == 1:
            break
    if(i == 9999): print('nao achei')
    return (i)
```

Quiz 7

O algoritmo RSA é unidirecional, porque ele só oferece confidencialidade se as chaves forem usadas em um modo específico. Que modo é esse?

A. Quando a criptografia é com a chave privada e a descriptografia com a pública.
B. Quando a criptografia é com a chave pública e a descriptografia com a chave privada.
C. Quando a criptografia e descriptografia são feitas com a chave privada.
D. Quando a criptografia e descriptografia são feitas com a chave pública.
E. Quando as chaves pública e privadas nunca forem transmitidas em aberto pela rede .

RSA

Ron Rivest, Adi Shamir, and Leonard Adleman publicaram o algoritmo pela primera vez em 1978.

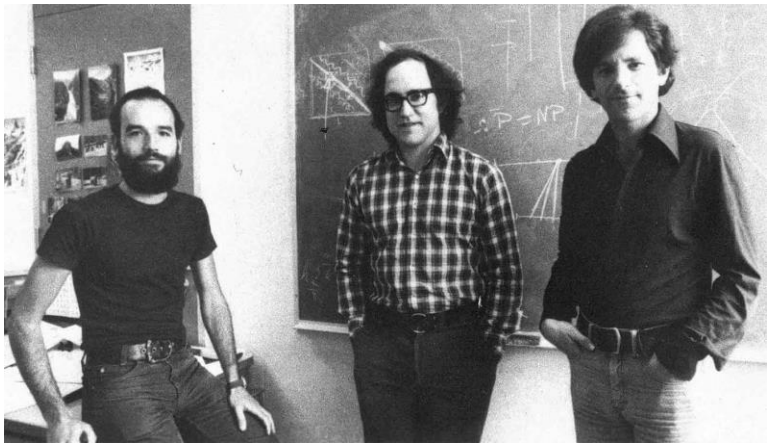
O mesmo algoritmo era conhecido pelo governo britânico desde 1973 (Clifford Cocks), mas só foi tornado público em 1997.

O algoritmo RSA é muito mais lento que o DES, pois os cálculos efetuados são complexos.

Por utilizar números primos, o RSA precisa de chaves muito grandes para reproduzir o mesmo grau de segurança do AES ou de outro algoritmo simétrico.

As chaves em RSA são em geral da ordem de 2048 bits.

Por isso o RSA não é usado para criptografar dados.



Críticas ao RSA

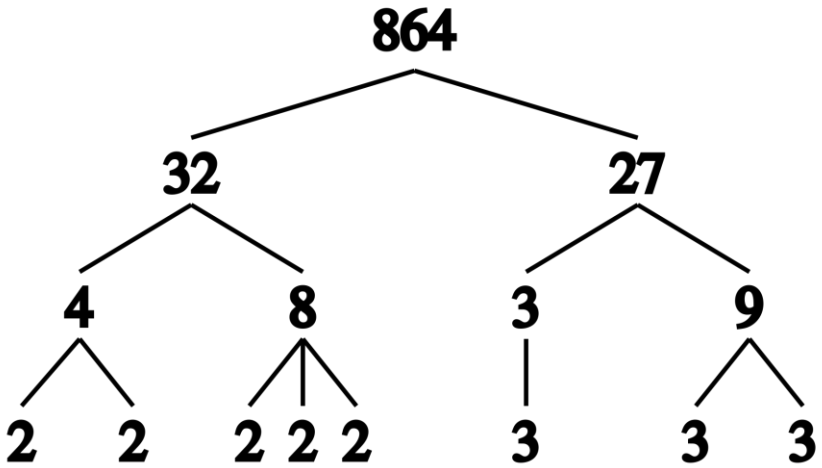
RSA é baseado em conceitos bem estabelecidos.

- A fatoração de grandes números primos é considerado um problema difícil
- Veja a página do Wikipedia: RSA Factoring Challenge
- Uma chave de 797 precisa de 900 anos usando um core (2019)

RSA não escala bem

- A dificuldade não aumenta proporcionalmente ao tamanho da chave
- A partir de 2030, as chaves RSA deverão ter tamanho de 3072 bits ou superior.

PRIME FACTORIZATION
é decompor
números inteiros
em números primos



ECC: Eliptic Curve Cryptography

ECC é um outro tipo de algoritmo de chave pública, que oferece um alto nível de segurança contra força bruta, mas com chaves muito menores que o RSA.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

2030

TOP SECRET

RSA vs ECC

$$y^2 = x^3 + ax + b$$

OPCIONAL: Veja as figuras nos tutoriais da cloudflare e fang-pen

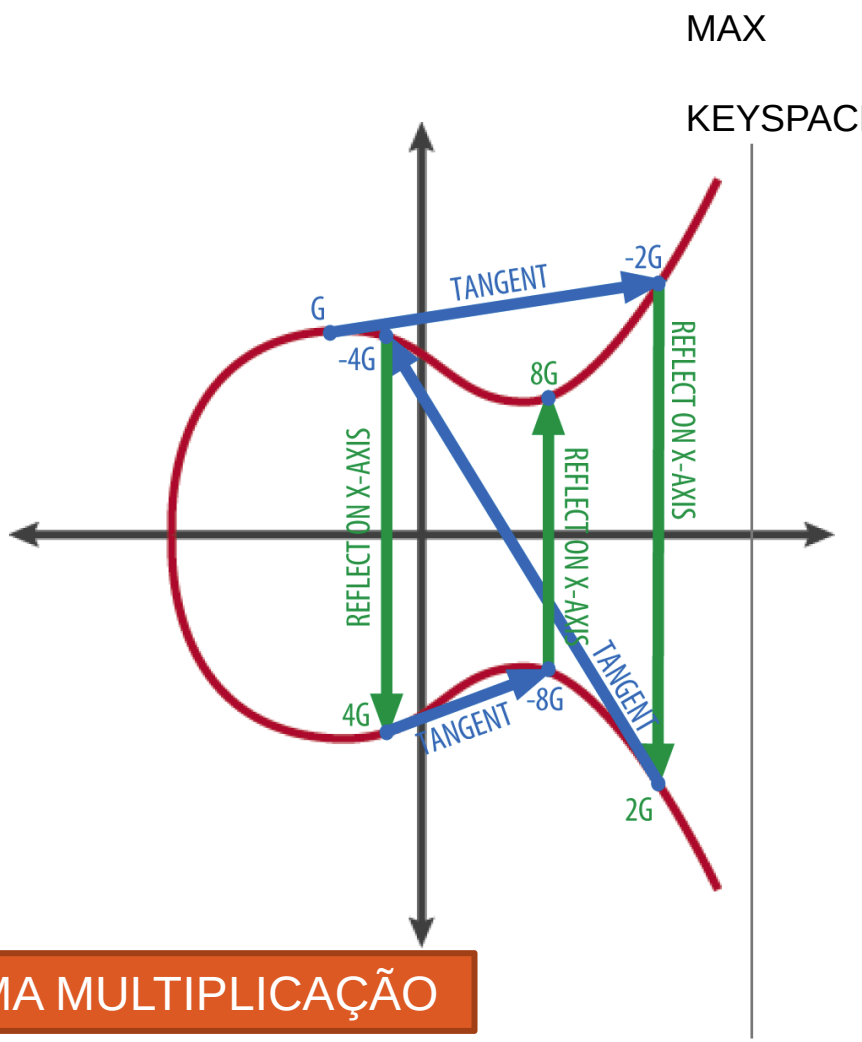
ECC: Eliptic Curve Cryptography

ECC define uma operação de “MULTIPLICAÇÃO” sobre um campo finito, em uma curva elíptica.

Campo finito (Galois Filed) é um conceito matemático que permite definir operações clássicas como adição e multiplicação sobre um conjunto limitado de valores.

A operação de multiplicação do ponto G por N é mostrado na figura.

- G: valor compartilhado
- N*G: chave pública
- N: chave privada



* NÃO É UMA MULTIPLICAÇÃO

É muito difícil determinar N conhecendo apenas G e NG.

Diffie-Hellman Key Exchange

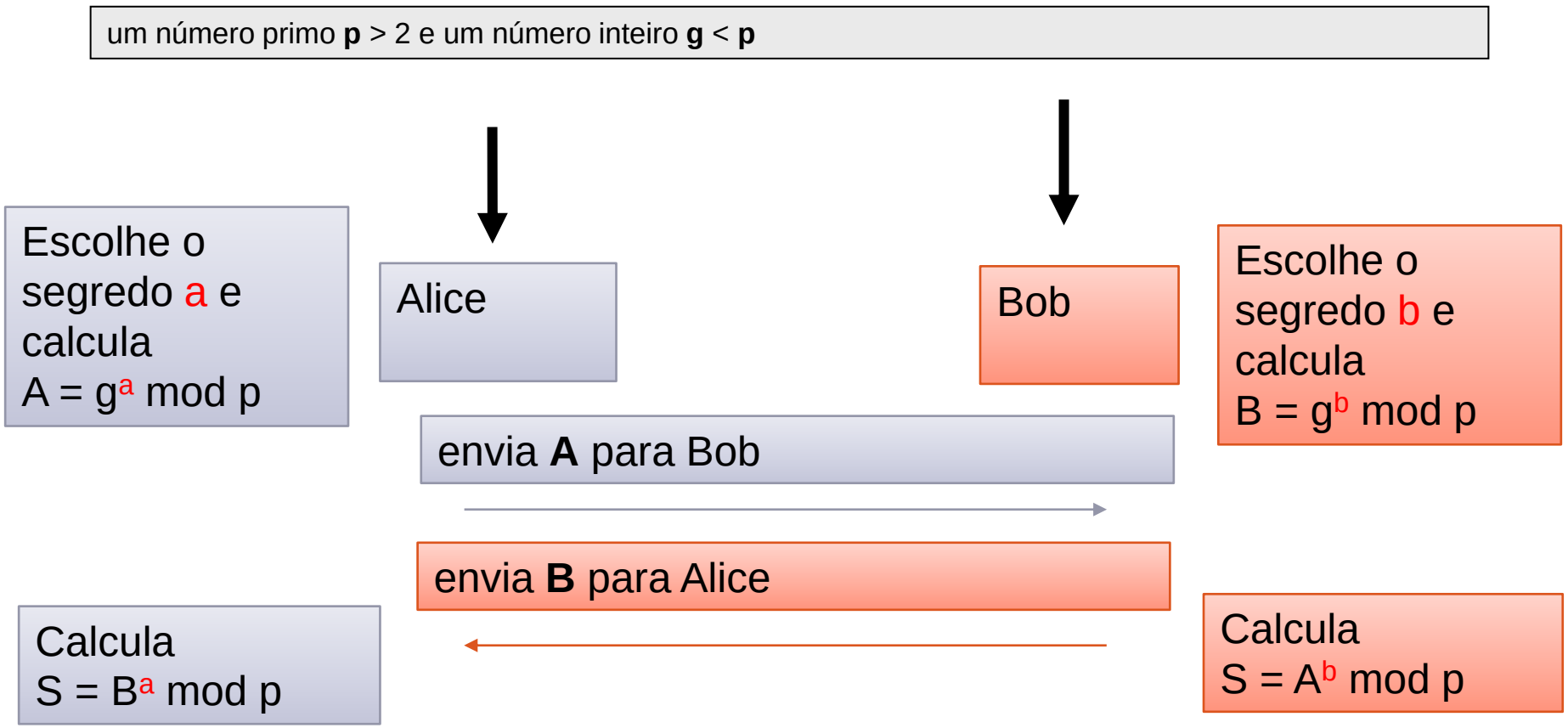
Diffie-Hellman define uma forma de combinar um segredo entre duas partes usando chaves públicas.

Existem muitas muitas variantes. A mais simples, é mostrada nos próximos slides

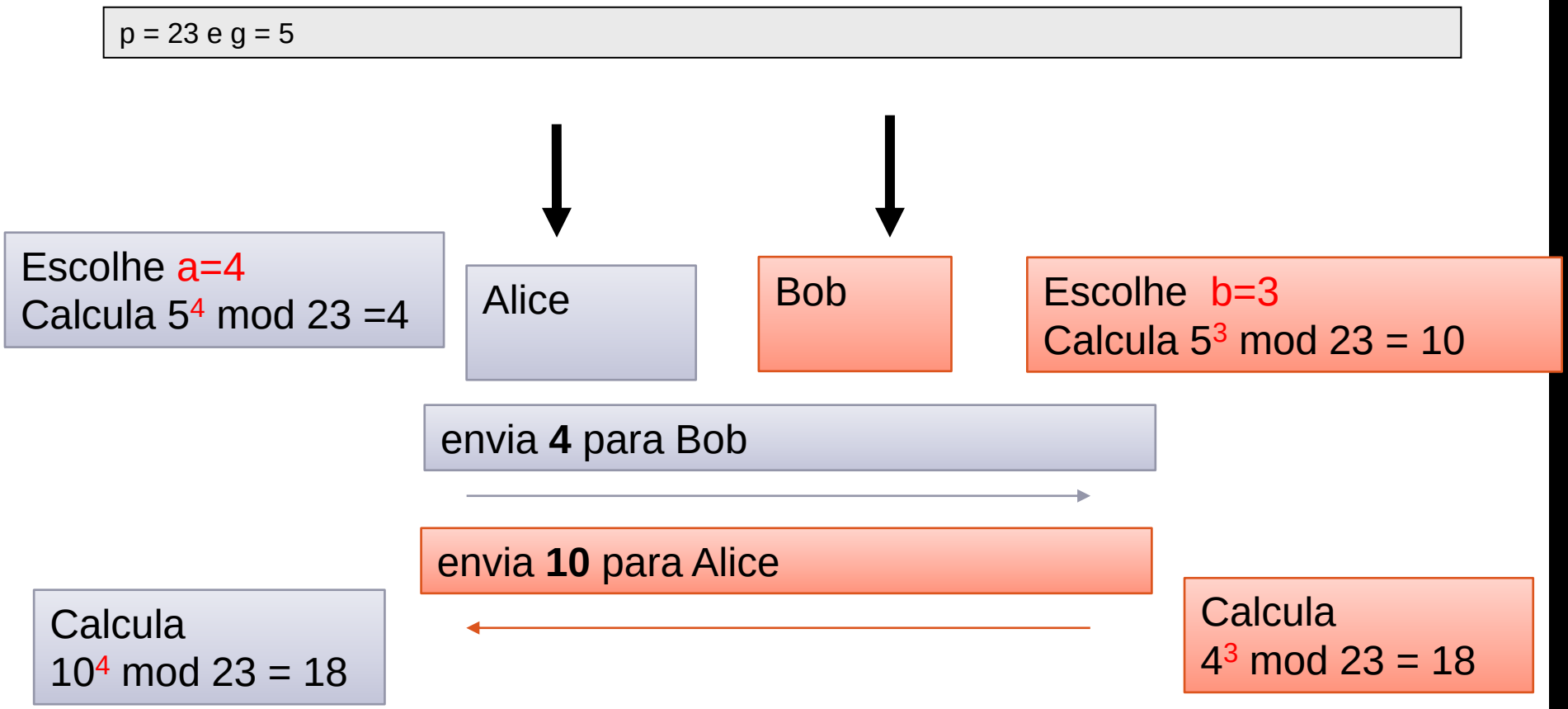
A força da chave é determinada através da escolha de um grupo pelos participantes:

group 1 - 768 bit modulus (CORRA)
group 2 - 1024 bit modulus (FUJA)
group 5 - 1536 bit modulus (EVITE)
group 14 - 2048 bit modulus (NO LIMITE)
group 20 - 384 bit elliptic curve (AGORA SIM)
group 21 - 521 bit elliptic curve (VAI DURAR)
group 24 - modular exponentiation group
with a 2048-bit modulus and 256-bit prime order subgroup
(OPA !!!)

Diffie-HellMan: Variante Básica (modulo)



Diffie-HellMan: Exemplo Módulo



ECC Diffie-Hellman

ECC Diffie-Hellman é uma variante do Diffie-Hellman onde as chaves públicas e a chave secreta são geradas por operações com curvas Elípticas

A maneira como as chaves são geradas está ilustrada no próximo slide.

A vantagem dessa abordagem é reduzir o tamanho das chaves públicas geradas, mantendo uma alto nível de segurança.

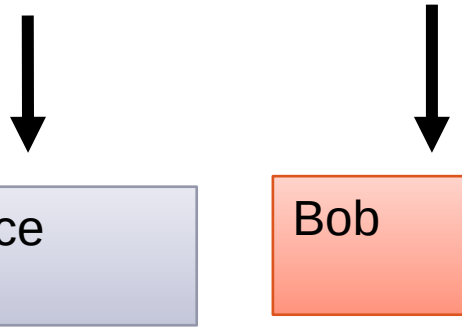
Veja diferença entre os grupos Diffie-Hellman 19 e 14!!!

ECC Diffie-Hellman é um método muito usado em Certificados Digitais, que será visto na sequência da disciplina.

Diffie-HellMan: ECC

Lembre-se que *
NÃO É UMA MULTIPLICAÇÃO CONVENCIONAL

Alice e Bob concordam em usar o ponto inicial G



Escolhe a
Calcula A = a * G

Alice

Bob

Escolhe b
Calcula B = b * G

envia **A** para Bob

envia **B** para Alice

Calcula
S = a * B

Calcula
S = b * A

S = a * (b * G)

S = b * (a * G)

CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA

FIM