CiberSegurança

Módulo 1 - 03 CIFRAS SIMÉTRICAS

MEET, MEIC, MEIM

2021–2022

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Princípio de Kerckhoffs:

Os métodos para cifrar e decifrar devem ser de domínio público.

(The Enemy Knows the System)

A confidencialide de um sistema depende de manter secreta a chave para o decifrado.

Um sistema de cifra diz-se **simétrico** se é (*computacionalmente*) possível obter a chave para decifrar a partir da chave para cifrar.

Um sistema de cifra diz-se **assimétrico** se é (*computacionalmente*) impossível obter a chave para decifrar a partir da chave para cifrar.

⇝ Todas as cifras históricas apresentadas anteriormente, como as cifras de Cesar ou de Belaso-Vigenère, são cifras simétricas.

Prós das cifras simétricas mais frequentes (vs assimétricas):

- rápidas, permitem cifrar grandes quantidades de dados (baseadas em operações com bits) com menos recursos computacionais;
- heuristicamente seguras

e contras ...

$\rightsquigarrow$ o problema da distribuição das chaves, porque a chave deve ser mantida secreta...

$\rightsquigarrow$ heuristicamente seguras ...

Tipos de cifras simétricas

- **cifras stream (cifra fieira)**;
- **cifras por blocos**.

Uma cifra por *blocos de comprimento n* é um sistema de cifrado em que o texto claro está dividido em sequências de *n* elementos, chamados *blocos*, e a cifra é aplicada bloco a bloco.

Exemplo: Cifra que consiste em dividir o texto claro em blocos de dois elementos e permutá-los entre si.

Por exemplo, considerando o alfabeto estandar:

Texto claro:	ex	em	pl	od	eu	ma	ci	fr	ap	or	bl	oc	os
Texto cifrado:	XE	ME	LP	DO	UE	AM	IC	RF	PA	RO	LB	CO	SO

A mesma cifra aplicada a *arrays* de bits:

Texto claro:	01	10	00	11	01	01	10
Texto cifrado:	10	01	00	11	10	10	01

Cifras de substituição por blocos de *bits*: $\boxplus_n$

A adição módulo 2^n permite definir cifras de sustituição em *arrays* de n-bits, que denotamos por $\boxplus_n$.

Exemplo: A adição módulo 2^4 define um cifrado de sustituição por blocos de 4 bits.

Texto claro:	0000	0011	1111
Chave :	0001	0001	0001
Texto cifrado:	0001	0100	0000

$\rightsquigarrow$ Note-se que cifras com diferentes comprimento de blocos podem ser combinadas entre si, por exemplo, podemos combinar duas $\boxplus_4$ e uma $\boxplus_8$ para definir uma sustituição em arrays de 16 *bits*.

Cifra de Hill (1929)

A cifra de Hill por blocos com n caracteres consiste em dividir o texto claro em blocos de n caracteres, identificar cada caracter com um elemento de $\mathbf{Z}_{26}$ e multiplicar cada bloco de n caracteres por uma matriz quadrada K de ordem n, com K uma matriz invertível em $\mathbf{Z}_{26}$

Mais precisamente, dado um bloco de texto claro de comprimento n, $m = m_1 m_2 \cdots m_n$, com os caracteres identificados com elementos de $\mathbf{Z}_{26}$ e o bloco de texto cifrado com o mesmo tamanho, $c = c_1 c_2 \cdots c_n$, os métodos para cifrar e decifrar são:

$$\begin{array}{rcl} c & = & e_K(m) \\ m = & = & d_{K^{-1}}(m) \end{array} \qquad K \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \qquad K^{-1} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix}$$

$\rightsquigarrow$ A matriz quadrada K, invertível em $\mathbf{Z}_{26}$ é a **chave** do cifrado. Para decifrar, é preciso multiplicar os blocos do texto cifrado pela matriz K^{-1}.

Exemplo: Cifra de Hill

Considerando blocos de comprimento 4 e a matriz chave

$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 5 & 3 \\ 1 & 1 & 7 & 3 \\ 0 & 0 & 1 & 5 \end{bmatrix},$$

Texto claro:	e x e m	p l o d	e u m a	c i f r
(mod 26):	4 23 4 12	15 11 14 3	4 20 12 0	2 8 5 17

Cifrado do primeiro bloco:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 5 & 3 \\ 1 & 1 & 7 & 3 \\ 0 & 0 & 1 & 5 \end{bmatrix} \begin{bmatrix} 4 \\ 23 \\ 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 39 \\ 79 \\ 91 \\ 64 \end{bmatrix} \overset{\mathrm{mod}\,26}{=} \begin{bmatrix} 13 \\ 1 \\ 13 \\ 12 \end{bmatrix}$$

(mod 26):	13 1 13 12	3 12 3 3	24 2 4 12	1 6 18 12
Texto cifrado:	N B N M	D M D D	Y C E M	B G S M

A cifra de Hill módulo 2

O método da cifra de Hill pode ser modificado para cifrar blocos de *bits* de comprimento n.

Dada uma matriz quadrada K de ordem n, com entradas em $\mathbf{Z}_2$, invertível em $\mathbf{Z}_2$, cada bloco de texto claro de n bits é cifrado multiplicando por K

$$\begin{array}{ccc} c & = & e_K(m) \\ m = & = & d_{K^{-1}}(m) \\ m & = & m_1 m_2 \cdots m_n \\ c & = & c_1 c_2 \ldots c_n \\ & & (m_i, c_i \text{ bits}) \end{array} \qquad K \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \qquad K^{-1} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix}$$

$\rightsquigarrow$ Uma matriz com entradas em $\mathbf{Z}_2$ será invertível se o seu determinante é $1 \bmod 2$.

Exemplo: cifra de Hill módulo 2

Cifra de Hill por blocos de 4 bits, com matriz chave

$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para cifrar o texto limpo 000000111111 dividimos o texto em blocos de 4-bits:

Texto claro: 0000 0011 1111

Multiplicamos cada bloco, módulo $\mathbf{Z}_2$, pela matriz chave:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Texto cifrado: 0000 1101 1001

Exemplo: cifra de Hill módulo 2

Se alterarmos a **matriz chave** mas mantemos o método de cifra de Hill, o cifrado dos blocos de 4 bits poderá ser diferente.

Por exemplo, considerando o mesmo texto limpo 000000111111, e uma nova chave

$$K' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Os blocos de 4 bits 0000, 0011 e 1111 cifram do seguinte modo:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Texto claro:	0000	0011	1111
Texto cifrado:	0000	1011	0111

Matrizes inversas módulo n?

É possível trabalhar com matrizes com coeficientes módulo n, de modo bastante parecido às matrizes com coeficientes reais.

De facto, dada uma matriz K coeficientes em $\mathbf{Z}_n$, K será invertível se e só se o seu determinante, módulo n, é invertível (ou seja, co-primo com n).

Neste caso, a inversa pode ser calculada usando **condensação de matrizes**, ou através da fórmula usual:

$$K^{-1} = (\det K)^{-1}\mathrm{adj}(K)$$

com $\mathrm{adj}(K)$ a matriz adjunta de K.

$\rightsquigarrow$ Uma matriz K, com coeficientes em $\mathbf{Z}_2$ é invertível quando o seu determinante, $\det K$ é invertível em $\mathbf{Z}_2$, mais precisamente, quando $\det K$ é impar.

Exemplos:

$K_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ é invertível mod 2, $K_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ não é invertível mod 2

$\rightsquigarrow$ Uma matriz K, com coeficientes em $\mathbf{Z}_{26}$ é invertível quando o seu determinante, $\det K$ é invertível em $\mathbf{Z}_{26}$, mais precisamente, quando $\det K$ é primo com 26, ou seja, não divisível por 2 ou por 13.

Exemplos:

$K_1 = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}$ é invertível mod 26, $K_2 = \begin{bmatrix} 5 & 2 \\ 1 & 3 \end{bmatrix}$ não é invertível mod 26

Cifras compostas: os *rounds*

As cifras de tipo *round* são cifras **compostas** que usam normalmente combinações de cifras do tipo de substituição com cifras de transposição, e repetem o procedimento várias vezes (*rounds*).

As cifras *round* mais conhecidas são a cifra de Feistel (base do DES - Data Encryption Standard, usado largamente mas em desuso atualmente) e a cifra de Rijndael (base do AES - Advanced Encryption Standard, cifra por blocos estandarizada pelo NIST- National Institute of Standards of Technology, provavelmente a mais usada na atualidade em cifras simétricas).

O AES - Advanced Encryption Standard

- Cifra por blocos de 128 *bits*, com chave simétrica de tamanho 128,192 ou 256 *bits*;

  A partir da chave inicial são obtidas as chaves de cada round, as *roundkeys*.

- N_r *rounds* em função do tamanho da chave, mais precisamente, $N_r = 10, 12, 14$ *rounds*, para os tamanhos de chave 128, 192 ou 256, respetivamente;

- Descrição geral do algoritmo:

  1. inicializa com um XOR-*bitwise* do texto limpo com a *roundkey* inicial;
  2. até o *round* $N_r - 1$, ao texto obtido no *round* anterior são aplicadas consecutivamente quatro cifras por blocos (S-box, permutação de linhas, mistura colunas, XOR-*bitwise* com a *roundkey*);

     A cifras de tipo S-box usam no AES operações no corpo finito $\mathbf{F}_{2^8}$.
  3. no último round, aplica três cifras por blocos (S-Box, permutação de linhas, e XOR-*bitwise* com a *roundkey*).

$\rightsquigarrow$ **Consultar os detalhes, por exemplo, em N. Smart, "Cryptography , An Introduction"**

Em geral, as cifras por blocos podem ser usadas de diferentes modos para cifrar uma sequência de dados, o que se costumam chamar **modos de operação.**

Os cinco modos de operação mais frequentes são:

- modo ECB - *Electronic CodeBook Mode*;
- modo CBC - *Cipher Block Chaining Mode*;
- modo OFB - *Output Feedback Mode*;
- modo CFB - *Cipher FeedBack Mode*;
- modo CTR - *Counter Mode*.

Modo ECB - Electronic CodeBook

Uma mensagem é particionada em blocos de comprimento n bits e estes são encriptados separadamente.

$$\begin{aligned} m &= m_0 m_1 \cdots m_r \\ c_i &= e_K(m_i) \\ c &= c_0 c_1 \ldots c_r \end{aligned}$$

Em particular, no modo de operação ECB, blocos de texto claro idênticos com a mesma chave resultam em blocos cifrados idênticos. Trata-se do modo mais simples de operar com cifras de blocos.

$\rightsquigarrow$ Todos os exemplos apresentados anteriormente de cifras por blocos usavam este modo de operação.

Cifrado da sequência 000000111111 usando o cifrado de Hill, módulo 2, multiplicando pela matriz chave

$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e o modo de operação ECB.

Dividimos o texto claro em blocos

0000 0011 1111

e aplicamos a cada bloco o cifrado:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Em resumo

Texto claro:	0000	0011	1111
Texto cifrado:	0000	1101	1001

Modo CBC - Cipher Block Chaining Mode

Em cada etapa, o bloco limpo é combinado com o bloco cifrado anterior, geralmente através do *XOR* e aplicada a cifra de bloco ao resultante desta combinação.

O modo de operação CBC precisa de um vetor de inicialização (bloco inicial) denotado usualmente por IV.

$$
\begin{array}{rcl}
m & = & m_0 m_1 \cdots m_r \\
c_0 & = & e_K(m_0 \oplus IV) \\
c_i & = & e_K(m_i \oplus c_{i-1}), \;\; i \geq 1 \\
c & = & c_0 c_1 \ldots c_r
\end{array}
$$

Cifrado da sequência 000000111111 usando o cifrado de Hill, módulo 2, multiplicando pela matriz chave

$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e o modo de operação CBC com vetor de inicialização $IV = 1000$

Dividimos o texto claro em blocos de 4 bits

0000 0011 1111

e combinamos com XOR cada bloco com o resultado do cifrado do anterior (o primeiro bloco com o bloco de inicialização $IV = 1000$)

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \oplus 1 \\ 0 \oplus 0 \\ 0 \oplus 0 \\ 0 \oplus 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \oplus 1 \\ 0 \oplus 0 \\ 1 \oplus 0 \\ 1 \oplus 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \oplus 0 \\ 1 \oplus 1 \\ 1 \oplus 0 \\ 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

Em resumo

Texto claro:	0000	0011	1111
Texto cifrado:	1000	0101	1111

Modo CTR - Counter Mode

É seleccionado um bloco inicial IV, que serve como *contador* e para cada cifrar o bloco i, é usada a cifra de bloco para cifrar $IV + i$ e depois é realizado um XOR como bloco i-ésimo da mensagem.

$$\begin{aligned} m &= m_0 m_1 \cdots m_r \\ c_i &= m_i \oplus e_K(IV + i) \\ c &= c_0 c_1 \ldots c_r \end{aligned}$$

Cifrado da sequência 000000111111 usando o cifrado de Hill, módulo 2, multiplicando pela matriz chave

$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e o modo de operação CTR com vetor de inicialização $IV = 1000$

Dividimos o texto claro em blocos de 4 bits: 0000 0011 1111, calculamos o contador (*counter*) $IV = 1000$, $IV + 1 = 1001$, $IV + 2 = 1010$, que ciframos usando a cifra escolhida (Hill mod2, matriz K):

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Finalmente, aplicamos um XOR-bitwise ao resultado com o bloco respetivo:

Texto claro:	0000	0011	1111
(XORwise cifra do CTR):	1000	0011	1110
Texto cifrado:	1000	0011	0001

Métodos de preenchimento (*padding*)

Em criptografia os métodos de *padding*, são métodos que introduzem numa mensagem **informação irrelevante mas com um objetivo concreto**.

Por exemplo, nas cifras de susbtituição clássicas, usavam-se caracteres sem significado (*nulls*) para dificultar o análise de frequências.

No caso das cifras por blocos de comprimento n, o *padding* é usado para conseguir aplicar essas cifras a sequências de caracteres com comprimento arbitrário.

O preenchimento **OneAndZeroes** adiciona, no final de uma sequência de bits, um bit com valor 1 e depois adiciona bits com valor 0 até conseguir uma sequência cujo comprimento seja mútiplo de n.

Salienta-se que, se o texto claro já for um múltiplo de n, este padding adiciona um bloco extra, **fundamental** para detectar o início do *padding*,

Exemplo: *OneAndZeroesPadding* para 4-bits:

0100 110	--→	0100 1101
1100 1	--→	1100 1100
0100 1111	--→	0100 1111 1000

Exemplo: *OneAndZeroesPadding* para 8-bits

01001111 01	--→	01001111 01100000
01001	--→	01001100
01001111	--→	01001111 10000000

O método de padding *OneZeroesPadding*, quando usado em blocos de *n*-bytes, é designado por **ISO/IEC 7816-4**.

Exemplo: *ISO/IEC 7816-4 padding* para blocos de 6 bytes:

AB AB AB AB		--→	*AB AB AB AB* 80 00
AB AB AB AB AB AB	*AB*	--→	*AB AB AB AB AB AB* *AB* 80 00 00 00 00
AB AB AB AB AB AB		--→	*AB AB AB AB AB AB* 80 00 00 00 00 00

(Recorde-se que a notação hexadecimal do byte 1000 0000 é precisamente 80)

O preenchimento **TBC** adiciona, no final de uma sequência de bits que termina em 0, bits com valor 1 até atingir o comprimento múltiplo de n e, se a sequência termina em 1, adiciona bits com valor 0 até atingir o comprimento múltiplo de n.

Exemplo: TBC para 4-bits

0100 110	⇢	0100 1101
1100 1	⇢	1100 1000
0100 1111	⇢	0100 1111 0000

PKCS7 padding - blocos de *n*-bytes

O preenchimento **PKCS7** adiciona, no final de uma sequência de bytes, o número N de bytes necessário para que a sequência tenha comprimento múltiplo de n, cada um deles com valor exatamente N.

Se a sequência inicial já tem comprimento múltiplo de n, adiciona-se um novo bloco completo com n bytes, com valor n.

Exemplo: PKCS7 para blocos de 4-bytes :

$$
\begin{array}{lcl}
AB\,AB\,AB\,AB\quad AB & \dashrightarrow & AB\,AB\,AB\,AB\quad AB\ \color{red}{03\ 03\ 03} \\
AB\,AB & \dashrightarrow & AB\,AB\ \color{red}{02\ 02} \\
AB\,AB\,AB\,AB & \dashrightarrow & AB\,AB\,AB\,AB\quad \color{red}{04\ 04\ 04\ 04}
\end{array}
$$

Exemplo: PKCS7 para blocos de 6-bytes:

$$
\begin{array}{lcl}
8A\,B1\,ED & \dashrightarrow & 8A\,B1\,ED\,\color{red}{03\,03\,03} \\
8A\,B1\,ED\,ED & \dashrightarrow & 8A\,B1\,ED\,ED\,\color{red}{02\,02} \\
8A\,B1\,ED\,ED\,04\,FF & \dashrightarrow & 8A\,B1\,ED\,ED\,04\,FF\quad \color{red}{06\,06\,06\,06\,06\,06}
\end{array}
$$

No caso em que se consideram blocos de 8-bytes, o sistema de preenchimento PKCS7 é denotado por PKCS5 (trata-se de um caso particular do anterior).

Exemplo: PKCS5, blocos de 8 bytes:

AB AB AB AB AB AB	--→	*AB AB AB AB AB AB* 02 02
AB AB AB AB AB AB AB AB AB AB	--→	*AB AB AB AB AB AB AB AB AB AB* 06 06 06 06 06 06
AB AB AB AB AB AB AB AB	--→	*AB AB AB AB AB AB AB AB* 08 08 08 08 08 08 08 08

Neste *padding*, o último byte do preenchimento indica o número de bytes adicionados no *padding* e todos os outros bytes do *padding* são zeros.

Exemplo: ANSI X9.23 *padding* para blocos de 8-bytes :

AB AB AB AB AB AB	--→	*AB AB AB AB AB AB* 00 02
AB AB AB AB AB AB AB AB AB AB	--→	*AB AB AB AB AB AB AB AB AB AB* 00 00 00 00 00 06
AB AB AB AB AB AB AB AB	--→	*AB AB AB AB AB AB AB AB* 00 00 00 00 00 00 00 08

Uma cifra **stream** (cifra fieira) é um sistema de cifra simétrica tal que, após cada cifra de um símbolo do texto claro, a transformação do cifrado pode mudar.

A chave usada numa cifra *stream* costuma ser uma *cadeia* de chaves chamada **keystream**.

Exemplo:

A Cifra de Vernam é uma cifra de tipo *stream*:

keystream	0	0	0	1	1	1	0	0	0	0	0	1	0	1
XOR	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
texto limpo	1	1	1	0	0	0	0	1	1	1	0	1	0	0
texto cifrado	1	1	1	1	1	1	0	1	1	1	0	0	0	1

Esta cifra simétrica pode considerar-se uma cifra **stream**, com uma *keystream* **periódica**.
mais precisamente, é uma adição módulo 26 do texto limpo e a *keystream*, caracter a caracter.

keystream	*B* 1	*E* 4	*L* 11	*L* 11	*A* 0	*S* 18	*O* 14	*B* 1	*E* 4	*L* 11	*L* 11	*A* 0	*S* 18	*O* 14
+ mod 26	+	+	+	+	+	+	+	+	+	+	+	+	+	+
texto limpo	*c* 2	*i* 8	*f* 5	*r* 17	*a* 0	*d* 3	*e* 4	*b* 1	*e* 4	*l* 11	*l* 11	*a* 0	*s* 18	*o* 14
texto cifrado	3 *D*	12 *M*	16 *Q*	2 *C*	0 *A*	21 *V*	18 *S*	2 *C*	8 *I*	22 *W*	22 *W*	0 *A*	10 *K*	2 *C*

$\rightsquigarrow$ A cifra de Vernam e a cifra de Bellaso-Vigenère, são casos particulares da cifra *stream* que consiste em considerar como alfabeto $\mathbf{Z}_n$ e cifrar realizando, termo a termo, a adição módulo n com uma sequência de elementos de $\mathbf{Z}_n$ (a *keystream*).

As cifras *stream*, em função do tipo de *keystream*, classificam-se em:

- cifras stream **síncronas**: a *keystream* é gerada num processo que não depende do texto claro ou do texto cifrado resultante;
- cifras stream **asíncronas** : a *keystream* depende, de algum modo, do texto claro ou do texto cifrado anterior.

  (*self-synchronizing stream ciphers, asynchronous stream ciphers* ou ainda, *ciphertext autokey (CTAK)*).

Exemplos:

A cifra de Vernan (ONE-TIME-PAD) é síncrona. Um exemplo de cifra asíncrona, que usa um sistema de "auto-chave", foi descrita por Blaise de Vigenère no seu tratado *"Traicte de Chiffres"*, em 1585.

A cifra com auto-chave de Vigenère

É uma cifra *stream* com *auto-chave*, baseada na *Tabula Recta* de Trithemius.

Chave inicial:	A
Keystream:	A C I F R A D E V I G E N E R
Texto claro:	c i f r a d e v i g e n e r e
Texto cifrado:	C K N W R D H Z D O K R R V V

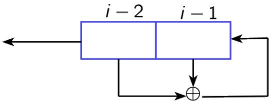
Ou equivalentemente, usando a identificação do alfabeto estándar com $\mathbf{Z}_{26}$:

Chave inicial:	0														
Keystream:	0	2	8	5	17	0	3	4	21	8	6	4	13	4	17
Texto claro:	2	8	5	17	0	3	4	21	8	6	4	13	4	17	4
Texto cifrado:	2	10	13	22	17	3	7	25	3	14	10	17	17	21	21

A seguranças das cifras simétricas requer que as **keystream** sejam o mais aleatórias possível (**keystreams "pseudo-aleatórias"**.)

Um modo eficaz de obter *keystream* pseudo-aleatórias é combinar *keystream* obtidas a partir dos chamados *Linear Feedback Shift Register* - LFSR.

Um LFSR - *linear feedback shift register* é um circuito que armazena uma sequência de bits (o registo, **register**) e, em cada ciclo, gera um novo bit combinando linearmente os bits armazenados, guardando o novo bit e deslocando (**shift**) a sequência.



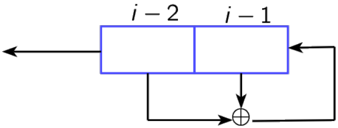
Algebricamente, um LFSR gera a partir de um um **registo** de bits de comprimento de s, $(k_{i-s}, \ldots, , k_{i-2}, k_{i-1})$, um novo bit através de uma relação de recorrência (módulo 2) :

$$k_i = c_1 k_{i-1} + c_2 k_{i-2} + \cdots + c_s k_{i-s}.$$

Exemplo:

LFSR definido pela relação:

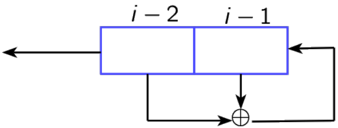
$$k_i = k_{i-1} + k_{i-2}$$



Exemplo:

Dado o LFSR definido pela relação:

$$k_i = k_{i-1} + k_{i-2}$$



Considerando o registo inicial $(k_0, k_1) = (1, 1)$, obtemos os registos:

$$(k_0, k_1) = (1, 1) \rightarrow (k_1, k_2) = (1, 0) \rightarrow (k_2, k_3) = (0, 1) \rightarrow (k_3, k_4) = (1, 1)...$$

visto que, aplicando a relação de recorrência, verifica-se:

$$\begin{aligned} k_2 &= k_1 + k_0 = 1 + 1 = 0 \\ k_3 &= k_2 + k_1 = 0 + 1 = 1 \\ k_4 &= k_3 + k_2 = 1 + 0 = 1 \\ &\dots \end{aligned}$$

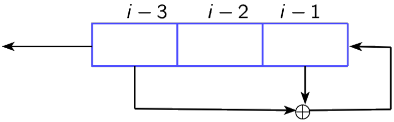
A *keystream* é 110110110..... (*keystream* de periodicidade 3).

⇝ A partir dos valores iniciais $k_0 = 0$, $k_1 = 0$, $k_2 = 0$, a *keystream* seria constante e igual a 0.

Exemplo: LFSR $k_i = k_{i-3} + k_{i-1}$

Exemplo:

Dado o LFSR definido pela relação:



$$k_i = k_{i-3} + k_{i-1}$$

e registo inicial $k_0 = 0,\ k_1 = 1,\ k_2 = 1$, obtemos

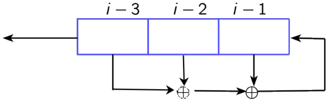
$$\begin{tikzcd}
 & (1,1,1) \arrow[r] & (1,1,0) \arrow[r] & (1,0,1) \arrow[dd] \\
(k_0, k_1, k_2) = (0,1,1) \arrow[ur] & & & \\
 & (0,0,1) \arrow[ul, dotted] & (1,0,0) \arrow[l] & (0,1,0) \arrow[l]
\end{tikzcd}$$

Observe-se que o primeiro registo repetido aparece quando $(k_7, k_8, k_9) = (0,1,1) = (k_0, k_1, k_2)$, o que determina uma *keystream* periódica de período 7, 0111010 0111010....

Exemplo: LFSR $k_i = k_{i-3} + k_{i-2} + k_{i-1}$

Exemplo:

Dado o LFSR definido pela relação:



$$k_i = k_{i-3} + k_{i-2} + k_{i-1}$$

e registo inicial $k_0 = 0,\ k_1 = 1,\ k_2 = 1$, obtemos

$$(k_0, k_1, k_2) = (0, 1, 1) \longrightarrow (1, 1, 0) \longrightarrow (1, 0, 0) \longrightarrow (0, 0, 1)$$

O primeiro registo repetido aparece quando

$$(k_4, k_5, k_6) = (0, 1, 1) = (k_0, k_1, k_2),$$

o que determina uma *keystream* periódica de período 4, 0110 0110....

- Os LFSR geram sempre sequências que a partir de um estado r_k são periódicas;

  Por exemplo, dado um LFSR de comprimento 3, em cada estado antes da geração do *bit* seguinte, temos um registo de três bits

  $$(k_0, k_1, k_2), (k_1, k_2, k_3), (k_2, k_3, k_4), \ldots\ldots$$

  Os eventuais estados não nulos deste LFSR são $2^3 - 1$ pelo que, no máximo, após $2^3 - 1$ iterações, o estado obtido será igual a algum dos estados anteriores. A partir da aí a *keystream* é periódica.

- Um LFSR com valores iniciais nulos gera uma *keystream* constante e igual a 0.

Periodicidade de um LFSR

Os LFSR geram sempre sequências que, a partir de um estado r são periódicas.

O caso ótimo, para usos criptográficos dos registos gerados por LFSR, ocorre quando o período da keystream é o máximo possível.

O período máximo possível para um LFSR de comprimento s *bits* é $2^s - 1$. Acontece quando os registos de comprimento s criados por uma relação de recorrência:

$$k_i = c_s k_{i-s} + \cdots + c_2 k_{i-2} + c_1 k_{i-1}$$

percorrem as $2^s - 1$ possibilidades de arrays de s-*bits* não nulos.

Como detetar os LFSR com período máximo?

O polinómio de conexão de um LFSR

A cada LFSR de comprimento s é associado um polinómio, com coeficientes em $\mathbf{Z}_2$, de grau inferior ou igual à s. Mais precisamente, se a relação de recorrência do LFSR é:

$$k_i = c_1 k_{i-1} + c_2 k_{i-2} + \cdots + c_s k_{i-s},$$

então o polinómio associado é:

$$c(X) = 1 + c_1 X + c_2 X^2 + \ldots c_s X^k$$

$\rightsquigarrow$ Veremos que certas propriedades algébricas dos polinómios de conexão, permitem determinar quais os LFSR que geram *keystream* com períodos máximos.

O polinómio de conexão de um LFSR

1. O polinómio de conexão do LFSR definido pela relação de recorrência $k_i = k_{i-1} + k_{i-2}$ é

$$c(X) = 1 + X + X^2$$

2. O polinómio de conexão do LFSR definido pela relação de recorrência $k_i = k_{i-1} + k_{i-3}$ é

$$c(X) = 1 + X + X^3$$

3. O polinómio de conexão do LFSR definido pela relação de recorrência $k_i = k_{i-1} + k_{i-2} + k_{i-3}$ é

$$c(X) = 1 + X + X^2 + X^3$$

Periodicidade de um LFSR

Caraterização de LFSR com período máximo

Se o polinómio de conexão $C(X)$ associado a um LFSR de comprimento s é um polinómio irredutível de grau s em $\mathbf{Z}_2[x]$, então todo o registo inicial não nulo produce uma sequência de chaves periódica com período igual ao menor valor N tal que $C(X)$ divide a $1 + X^N$.

Em particular, se $C(X)$ divide a $1 + X^{2^s-1}$ e não divide a nenhum polinómio da forma $1 + X^N$, com $N < 2^s - 1$, então o LFSR tem período máximo igual a $2^s - 1$.

Exemplos

- O polinómio de conexão

$$c(X) = 1 + X + X^2,$$

verifica as condições anteriores, pelo que o LFRS definido por $k_i = k_{i-1} + k_{i-2}$ gera uma *keystream* com período $3 = 2^2 - 1$;

- O polinómio de conexão

$$c(X) = 1 + X + X^3$$

verifica as condições anteriores, pelo que o LFRS definido por $k_i = k_{i-1} + k_{i-3}$ gera uma *keystream* com período $7 = 2^3 - 1$;

- O polinómio de conexão

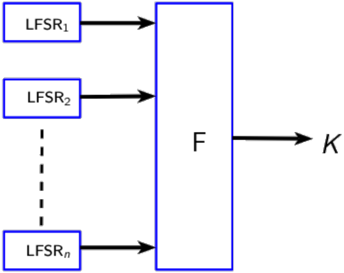
$$c(X) = 1 + X + X^2 + X^3$$

não é um polinómio irredutível. O LFSR definido por $k_i = k_{i-1} + k_{i-2} + k_{i-3}$ não gera uma *keystream* com período máximo ($4 \neq 2^3 - 1$).

Exemplos de polinómios de conexão associados a LFSR que geram *keystreams* com o máximo período possível:

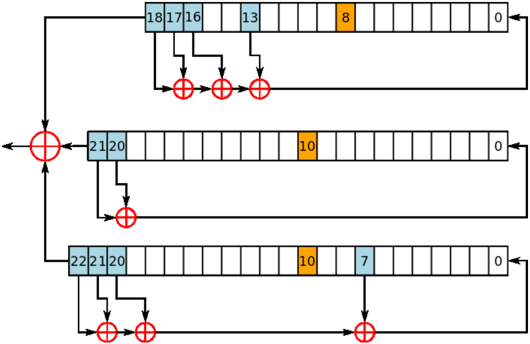
bits (s)	Polinómio conexão	Período
2	$x^2 + x + 1$	3
3	$x^3 + x^2 + 1$	7
4	$x^4 + x^3 + 1$	15
5	$x^5 + x^3 + 1$	31
6	$x^6 + x^5 + 1$	63
7	$x^7 + x^6 + 1$	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	255

Os *keystream* mais simples de implementar combinam diferentes LFRS usando funções de complexidade *não linear*:



Exemplos de geradores de *keystream* a partir de LFRS : E0, A5/1

O algoritmo de cifrado A5/1, que faz parte dos protocolos incluídos no GSM (Global System for Mobile Communications, 2G) para as comunicações entre telemóveis, usa um gerador de *keystream* que combina de modo não linear três LFRS.



Os LFSR usados em A5/1 estão associados aos seguintes polinómios de conexão :

1. $\text{LFSR}_1 : x^{19} + x^{18} + x^{17} + x^{14} + 1$ (registo de comprimento 19 *bits*)
2. $\text{LFSR}_2 : x^{22} + x^{21} + 1$ (registo de comprimento 22 *bits*)
3. $\text{LFSR}_3 : x^{23} + x^{22} + x^{21} + x^{8} + 1$ (registo de comprimento 23 *bits*)

Os LFSR não são atualizados em cada ciclo, a atualização depende do chamado *cloking bit* (nas posições os *bits* 8, 10 e 10, respetivamente).

A atualização dos registos é realizada seguindo a chamada *regra majoritária*: são observados os três *clocking bits*, estabelece-se qual o *bit* majoritário (o valor do *bit* que aparece mais vezes) e atualizam-se os registos cujo *clocking bit* é igual ao *bit* majoritário.