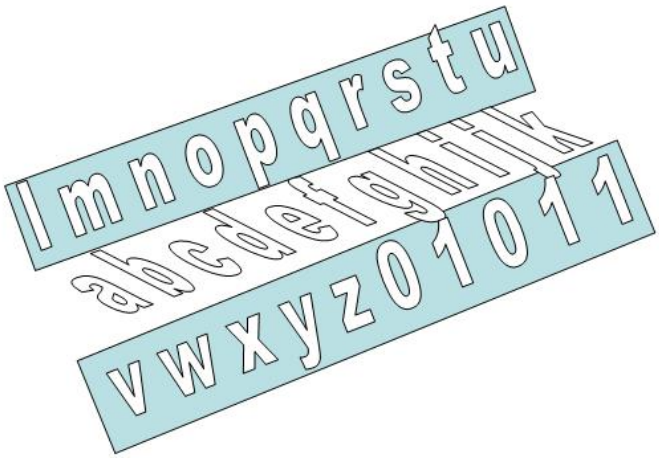
Uma Introdução à Criptografia



Nilton Cezar Ferreira
Glen Cezar Lemos

ÍNDICE

Introdução

O objetivo deste trabalho é dar uma idéia geral sobre criptografia e sua utilização, mostrando que ela é uma aplicação da matemática, em particular da álgebra. Tentaremos também motivar e dar fundamentos para os interessados em estudos posteriores. Mostraremos a diferença entre codificar e criptografar dando exemplos de sistemas criptográficos simples e códigos. Faremos uma breve introdução a sistemas criptográficos, sobretudo à block ciphers e stream ciphers, dando exemplo de um sistema utilizando atualmente.

1. INTRODUÇÃO A SISTEMAS CRIPTOGRÁFICOS

A necessidade de manter algumas mensagens secretas existe a milhares de anos. Porém a sociedade moderna vem se tornando cada vez mais dependente de meios seguros e precisos para transmissão e armazenamento de dados.

Usualmente, o principal objetivo é transmitir dados de forma rápida e barata, contudo, existem situações onde as informações são condidenciais; nestes casos os comunicantes devem ocultar e proteger o conteúdo de suas mensagens. O objetivo dos sistemas criptográficos é ocultar o conteúdo da mensagem, transformando-o antes de transmití-lo, para garantir que somente o destinatário seja capaz de entender a mensagem.

Estabeleceremos a seguir a terminologia necessária.

2. TERMINOLOGIA

Citaremos alguns termos técnicos que devem ser conhecidos antes de se iniciar uma leitura sobre sistemas criptográficos:

Texto em claro: Informação a ser ocultada.

Cifrar: Operação feita afim de ocultar a mensagem.

Criptograma: Mensagem depois de cifrada.

Emissor: Pessoa que envia a mensagem cifrada.

Algorítmo: Conjunto de regras usadas para cifrar a mensagem.

Chave: Parâmetro do qual o algorítmo depende. O objetivo do sistema criptográfico é não permitir que, sem o conhecimento desse dado, alguém consiga obter o texto em claro a partir do criptograma, ou seja, a segurança do sistema depende dela.

Decifrar: Processo de obter o texto em claro a partir do texto cifrado.

Receptor: Pessoa que recebe o criptograma, e a qual com o uso da chave vai decifrá-lo.

Código: Em sentido mais geral refere-se a qualquer transformação de informação de uma forma para outra.

Interceptor: Qualquer pessoa que consiga interceptar a mensagem que está sendo transmitida do emissor para o receptor.

Criptografia: È a ciência de se projetar sistemas criptográficos.

Criptógrafo: Pessoa que projeta sistema criptográficos.

Criptanálise: É o processo de se deduzir a mensagem a partir do criptograma sem conhecer a chave.

Criptanalista: Pessoa que tem como objetivo quebrar o sistema criptográfico, ou seja, fazer a criptanálise.

Canal seguro: Meio de comunicação utilizado para enviar a chave, sem que a mesma seja interceptada.

Criptologia: É a ciência que engloba a criptografia e a criptanálise.

Definição 1: Codificar é a arte de transformar mensagem escrita em outra mensagem sem o objetivo de dificultar o entendimento da mesma.

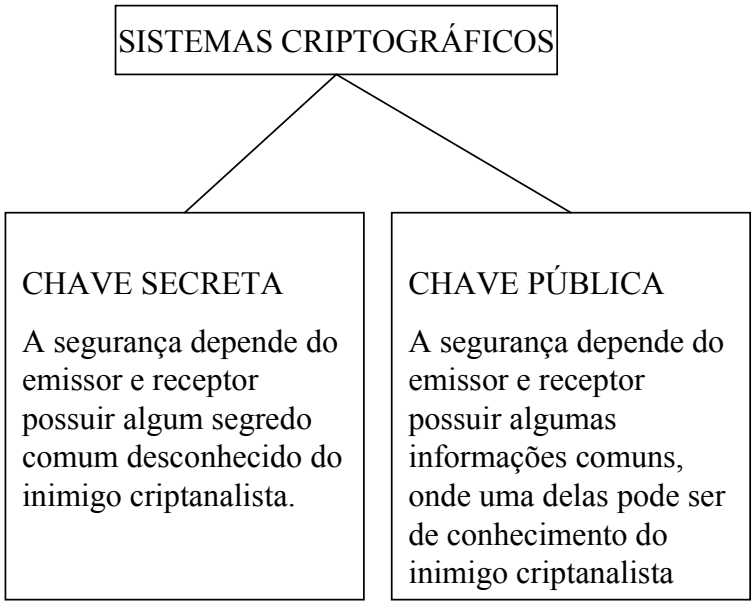
Exs: código morse, taquigrafia, idioma, códigos binários, etc.

Definição 2: Criptografar é a arte de transformar mensagem escrita, clara, em outra também escrita mas cifrada, ou seja, inintelegível para outrem que não o destinatário, conhecedor da convenção empregada na cifragem.

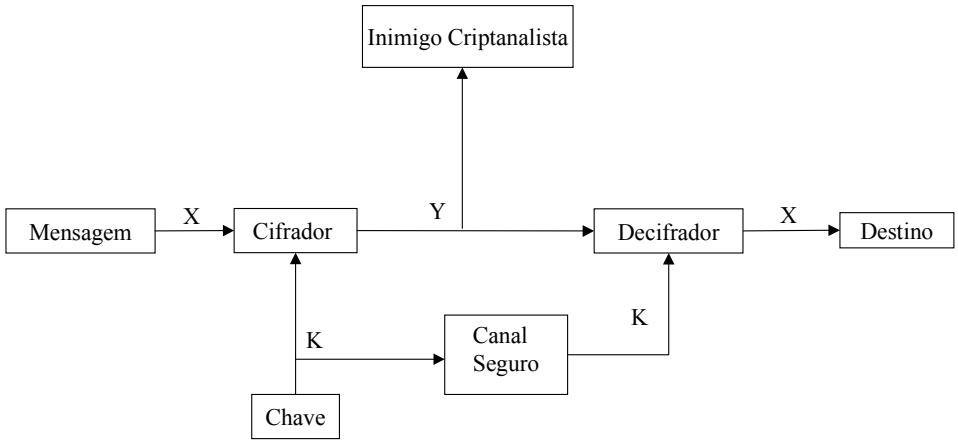
Exs: Cifra de CESAR, DES, IDEA.

3. SISTEMAS CRIPTOGRÁFICOS

Sistemas criptográficos se dividem em siméticos (chave secreta) e assimétricos (chave pública).



Nesse trabalho abordaremos os sistemas de chave secreta, que podem ser representados pelo seguinte diagrama.



A chave secreta depois de gerada deve ser enviada ao receptor sem que o inimigo criptanalista, tome conhecimento dela. O emissor fornece a sequência

input (texto em claro) $X = X_1, X_2, ..., X_m$ ao cifrador, o qual fazendo uso da chave $K = K_1, K_2, ..., K_n$ gera o criptograma $Y = Y_1, Y_2, ..., Y_s$ que será enviado ao decifrador, que fornecerá o texto em claro ao receptor.

Citaremos alguns exemplos de sistemas criptográficos simples.

3.1 - Sistemas de substituição monoalfabética

São aqueles onde cada letra do texto em claro é substituida, sempre, por uma mesma letra de um *Alfabeto Cifra*, segundo uma chave bem definida.

O exemplo mais antigo de que se tem notícia é conhecido como "Caesar Cipher" (CIFRA DE CESAR), usado pelo general e estadista Romano Julio Cesar (49 - 44 a.c) em sua campanha pela Gália e nas correspondências com seus amigos. Neste sistema, cada letra de a a w é representada pela terceira letra após, no alfabeto e x, y, z por A, B, C, respectivamente.

Alfabeto em claro: *a b c d e f g h j k l m n o p q r s t u v w x y z*
Alfabeto cifrado: 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.

Ex: Decifrar a seguinte mensagem, sabendo que ela foi criptografada usando a Cifra de Cesar.

Texto cifrado: D FULSWRORJLD VH GLYLGH HP FULSWRJUDILD H FULSWDQDOLVH.

Texto claro: **"A criptologia se divide em criptografia e criptanálise".**

3.1.1 Sistemas monoalfabéticos aditivos

O algorítimo do sistema monoalfabético aditivo consiste em adicionar ao

valor que representa a letra no alfabeto o valor da chave, indicando a posição da letra Cifra. Considerando o nosso alfabeto, o valor da chave pode variar de 0 a 25.

Definição 3: Sejam $a, b \in \mathbb{Z}$. Chamaremos adição de a e b módulo 26 o inteiro γ, que seja o resto da divisão de $(a + b)$ por 26. Denotaremos $\gamma = a \oplus b$.

Definição 4: Seja $a \in \mathbb{Z}$. O inteiro b, tal que $a \oplus b = 0$ é denotado o oposto de a.

Se fizermos corresponder a cada letra do alfabeto um número de 0 a 25, podemos representar o algoritmo usado nos sistemas aditivos, usando a adição módulo 26, da seguinte forma $x \rightarrow x \oplus k = y$, onde x representa o texto em claro, k a chave e y o criptograma.

Por exemplo:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

r	s	t	u	v	w	x	y	z
18	19	20	21	22	23	24	25	0

Logo, para decifrar um criptograma, o receptor, deverá usar a chave inversa, ou seja, o oposto de k.

Como existem apenas 26 chaves, concluímos que o sistema é inseguro e serviu apenas como ponto de partida para a criptografia por substituição.

Ex.1: Cifra de Cezar, onde a chave é 3.

Ex.2: Decifrar o criptograma abaixo, sabendo apenas que foi utilizado o sistema monoalfabético aditivo e o texto claro está em português.

Criptograma: D FLIUD GH FHCDU H R PDLV DQWLJR VLVWHPD FULSWRJUDILFR.

Analisando o criptograma notamos que as letras D,H e R aparecem isoladas, logo, em português temos apenas três possibilidades:

$$o \leftrightarrow D \text{ ou } a \leftrightarrow D \text{ ou } e \leftrightarrow D.$$

Se $o \leftrightarrow D$ então, $4 = 15 \oplus k \Rightarrow k = 15$. Logo, H corresponde a s e R a c.
Se $a \leftrightarrow D$ então, $4 = 1 \oplus k \Rightarrow k = 3$. Logo, H corresponde a e e R a o.
Se $e \leftrightarrow D$ então, $4 = 5 \oplus k \Rightarrow k = 25$. Logo, H corresponde a i e R a s.

Portanto, concluímos que na língua portuguesa só tem sentido se a chave usada for $k = 3$ e obtemos o seguinte texto em claro:

"A cifra de Cezar é o mais antigo Sistema Criptográfico".

3.1.2 Sistemas monoalfabéticos Multiplicativos

Quando estamos trabalhando com sistemas criptográficos devemos sempre tomar o cuidado de que cada criptograma determine um único texto em claro.

Def.5: Sejam $a, b \in \mathbb{Z}$. Chamaremos produto de a por b módulo 26, o inteiro γ, que seja o resto da divisão de $(a \cdot b)$ por 26. E denotaremos por $\gamma = a \otimes b$.

Def. 6: Seja $a \in \mathbb{Z}$. O inteiro b tal que $a \otimes b = 1$, é denominado inverso multiplicativo de a.

Por analogia ao sistema anterior, a operação soma é substituida pela multiplicação módulo 26. O algoritmos deste sistema pode ser representado da seguinte forma:

$x \rightarrow x \otimes k = y$,

onde x representa o texto em claro, k a chave, y criptograma.

Para decifrar o criptograma, o receptor deverá usar a chave inversa, ou seja o inverso de k. Por exemplo, se usarmos a chave $k = 2$, então para

$x_1 = a \Rightarrow y_1 = 1 \otimes 2 = 2 \leftrightarrow B$

$x_2 = n \Rightarrow y_2 = 14 \otimes 2 = 2 \leftrightarrow B$

Isto é, um único caracter do criptograma corresponde a dois cracteres do texto em claro, logo $k = 2$ não pode ser chave do sistema multiplicativo.

Consequentemente devemos escolher como chave todos os inteiros de 0 a 25 que são relativamente primos com 26. Logo k pode assumir os seguinte valores:

$k = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$

$k^{-1} = \{1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25\}$

Portanto, temos somente 12 possibilidades para a chave o que faz com que este sistema seja fraco como o anterior.

Ex. Decifrar o seguinte criptograma sabendo que foi usado o sistema multiplicativo, cuja chave é $k = 9$, na língua portuguesa.

Criptograma: **MIXSMIXCAI S GMI ACSVACI**

Vamos obter a chave inversa de $k = 9$. Seja x o inverso de 9 mod 26, então:

$9x \equiv 1(\mathrm{mod}\, 26) \Leftrightarrow \exists\ y \in \mathbb{Z}$ tal que $9x - 26y = 1$, resolvendo a equação diofantina, encontramos a solução particular $x = 3$, portanto $k^{-1} = 3$ é a chave inversa de $k = 9$, então:

$y_1 = M \Rightarrow x_1 = 13 \otimes 3 = 13 \longleftrightarrow m$

$y_2 = I \Rightarrow x_2 = 9 \otimes 3 = 1 \longleftrightarrow a$

$y_3 = X \Rightarrow x_3 = 24 \otimes 3 = 20 \longleftrightarrow t$

$y_4 = S \Rightarrow x_4 = 19 \otimes 3 = 5 \longleftrightarrow e$

$y_5 = C \Rightarrow x_5 = 3 \otimes 3 = 9 \longleftrightarrow i$

$y_6 = A \Rightarrow x_6 = 1 \otimes 3 = 3 \longleftrightarrow c$

$y_7 = G \Rightarrow x_7 = 7 \otimes 3 = 21 \longleftrightarrow u$

$y_8 = V \Rightarrow x_8 = 22 \otimes 3 = 14 \longleftrightarrow n$

Texto em Claro: **Matemática é uma Ciência.**

3.1.3 Sistemas monoalfabéticos Afins

É o sistema obtido usando a adição e a multiplicação módulo 26 simultâneamente, cujo algoritmo representamos por:

$x \rightarrow (x \otimes k_1) \oplus k_2 = y$

onde, x representa o texto em claro, k_1 e k_2 chaves e y o criptograma. Este sistema também é representado por: $[k_1, k_2]$.

Logo, o número total de chaves é 312.

Ex. Decifrar GLZOXA sabendo que foi usado o sistema afim [7,4].

$y_1 = G \Rightarrow 7 = (x_1 \otimes 7) \oplus 4 \Rightarrow 3 = x_1 \otimes 7 \Rightarrow x_1 = 19 \longleftrightarrow s$

$y_2 = L \Rightarrow 12 = (x_2 \otimes 7) \oplus 4 \Rightarrow 8 = x_2 \otimes 7 \Rightarrow x_2 = 16 \longleftrightarrow p$

$y_3 = Z \Rightarrow 0 = (x_3 \otimes 7) \oplus 4 \Rightarrow 22 = x_3 \otimes 7 \Rightarrow x_3 = 18 \longleftrightarrow r$

$y_4 = O \Rightarrow 15 = (x_4 \otimes 7) \oplus 4 \Rightarrow 11 = x_4 \otimes 7 \Rightarrow x_4 = 9 \longleftrightarrow i$

$y_5 = X \Rightarrow 24 = (x_5 \otimes 7) \oplus 4 \Rightarrow 20 = x_5 \otimes 7 \Rightarrow x_5 = 14 \longleftrightarrow n$

$y_6 = A \Rightarrow 1 = (x_6 \otimes 7) \oplus 4 \Rightarrow 23 = x_6 \otimes 7 \Rightarrow x_6 = 7 \longleftrightarrow g$

Texto em Claro: **Spring**.

3.1.4 - Sistemas monoalfabéticos Genéricos

Nestes sistemas chave é uma letra ou frase associada a um letra especial. O alfabeto do texto em claro deverá ser escrito normalmente e a frase chave será escrita abaixo do alfabeto do texto em claro, sem repetir letras e depois o alfabeto de substituição será completado escrevendo as letras que faltam em ordem alfabética.

Portanto o número de chaves possíveis é $26! \cong 4 \times 10^{26}$.

Se considerarmos que um criptanalista usará apenas o método de exaustão (testar todas as chaves possíveis) e seu computador demora um micro segundo para testar cada chave, a pesquisa exaustiva de todas as chaves consumiria $1,7 \times 10^4$ anos (50.000 vezes a idade estimada da Terra).

Do ponto de vista do número de chaves, este sistema é absolutamente seguro, porém apresenta fraquezas que neutralizam esta vantagem.

Existe um tamanho mínimo, acima do qual uma solução única é matematicamente possível, este tamanho foi definido por SHANNON em 1948, "Communication Theory of Secrecy Systems" ("Paper" confidencial da Bell System). A esta medida ele chamou de distância de unicidade.

$$U = \frac{H(k)}{D}$$

onde: H(k) é entropia da chave, que é o logaritmo na base 2 de todas as chaves possíveis

D é a redundancia da liguagem.

$$D = 1 + \frac{\sum_{\lambda=A}^{Z} P \log P_\lambda}{\log_2 26}, \text{ sendo D Espanhol} = 0{,}245 \text{ e H(k)} = 4 \times 10^{26}$$

Concluimos que: $U = \log \frac{4.10^{26}}{0{,}245} \cong 360$.

Isso quer dizer que um texto em espanhol deve ter pelo menos 360 caracteres para se obter solução única.

Ex. Palavra chave: **RELIGIÃO**
alfabeto claro: *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*
alfabeto cifra: R E L I G A O B C D F H J K M N P Q S T U V W X Y Z.

4. STREAM CIPHERS

Sistemas criptográficos são usualmente classificados em:

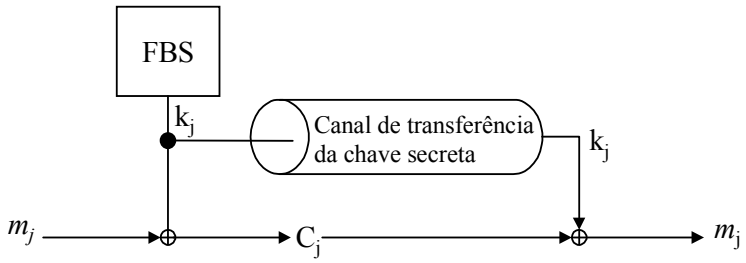
Block Ciphers (sistemas que cifram em blocos)**-** Neles o texto claro é dividido em blocos, normalmente de tamanhos fixos, e opera sobre eles com uma transformação não variante e chaves fixas.

Stream Ciphers (sistemas que cifram caracteres)**-** Nesse tipo de esquema o texto claro (mensagem) é transformado em caracteres e operado com uma transformação tempo-variante, governada pelo estado interno do sistema, ou seja, cada estado depende do estado anterior. Assim duas ocorrências do mesmo caracter no texto em claro normalmente não resulta em duas ocorrências de um mesmo caracter no texto cifrado (criptograma).

O mais notável de todos os sistemas criptográficos é o "one-time-pad" (Sistema de chave única), pois ele é totalmente seguro. Nele o texto cifrado

é a soma bit a bit módulo 2 da chave secreta, uma sequência aleatória, usada uma única vez. Como em $GF(2)$ a adição e subtração é a mesma, a decifração é obtida da mesma forma que a cifração. O princípio básico do "one time pad" é a independência estatística entre o texto cifrado e o texto em claro pelo fato da chave ser uma sequência verdadeiramente aleatória. O dispositivo que emite tal sequência aleatória, i.é, uma sequência onde cada bit tem igual probabilidade de ser 0 ou 1 independentemente dos dígitos anteriores é chamado de *fonte binária simétrica* (FBS).

A figura 4.1 ilustra o "one time pad".



m_j é um bit do texto em claro, k_j um bit da chave secreta e C_j um bit do texto cifrado.

*Fig.*4.1

Assumindo o ataque que tem como conhecidos somente textos cifrados, Shannon (1949) provou que o criptanalista nunca poderá separar o verdadeiro texto em claro dos outros possivelmente verdadeiros. Um sistema criptográfico é dito completamente seguro se a dependência entre o texto cifrado e em claro é zero, independentemente do tamanho da mensagem. Então, interceptando-se o texto cifrado o criptanalista não consegue obter nenhuma informação sobre o texto em claro.

Um ponto crucial do "one time pad" é que a única maneira de reproduzir a chave secreta utilizada pelo emissor é que ela seja gravada e uma cópia deve ser enviada ao receptor para decifrar a mensagem. Este sistema geralmente é impraticável devido aos problemas de geração das chaves (tais como, necessidade de se produzir uma quantidade ilimitada de chaves, necessidade de um local totalmente seguro para guardar a cópia que será enviada ao receptor da mensagem, etc.) e aos problemas de distribuição das mesmas (por exemplo, grande volume de chaves a serem distribuídas e se a chave for interceptada não é possível decifrar a mensagem). Por esta razão a aplicabilidade dele é quase que exclusivamente direcionada para fins de espionagens ou situações semelhantes, onde a completa segurança é exigida.

As desvantagens operacionais do "one time pad" levaram ao desenvolvimento dos stream ciphers síncronos, onde o processo de cifração do texto em claro é o mesmo do "one time pad" exceto pela geração da sequência aleatória. A segurança dos stream ciphers depende então da "aleatoriedade" da chave.

De um modo geral um stream cipher consiste de um mecanismo chamado *gerador de key streams* (usaremos a abreviatura em inglês KSG), o gerador de chaves, que produz sequências pseudo-aleatórias, denominadas "key streams", as quais são operadas caracter a caracter com os caracteres do texto em claro, resultando na mensagem cifrada.

As sequências produzidas pelos geradores de key streams seguem uma determinada regra; sendo assim elas não são verdadeiramente aleatórias. Em criptografia o que realmente necessitamos, é que estas sequências sejam imprevisíveis (simulem sequências aleatórias); ou seja, se o criptanalista inter-

ceptar parte da sequência ele não tem como completá-la. Estes tipos de sequências são chamadas de pseudo-aleatórias.

Os projetistas de stream ciphers têm como principal objetivo a produção, de forma eficiente, de sequências pseudo-aleatórias.

Distinguimos quatro métodos principais utilizados na construção de stream ciphers, que diferem nas hipóteses da capacidade e oportunidade do criptanalista, na definição de sucesso criptanalístico e na noção de segurança.

A construção de stream ciphers do ponto de vista teórico tem dois objetivos: um é garantir propriedades que teoricamente são exigidas tais como: período, complexidade linear e distância de estruturas lineares.

O segundo objetivo é estudar os princípios criptanalísticos e desenvolver critérios para tornar ataques baseados nestes princípios impossíveis. Por exemplo:
a) Substituição e aproximação; preferencialmente por componentes lineares.
b) Limitar e controlar o espaço das chaves.
c) Exploração das deficiências estatísticas, tal como dependência entre símbolos.

Para prevenir a criptanálise alguns critérios gerais necessários mas não suficientes são empregados na construção de geradores de key streams :

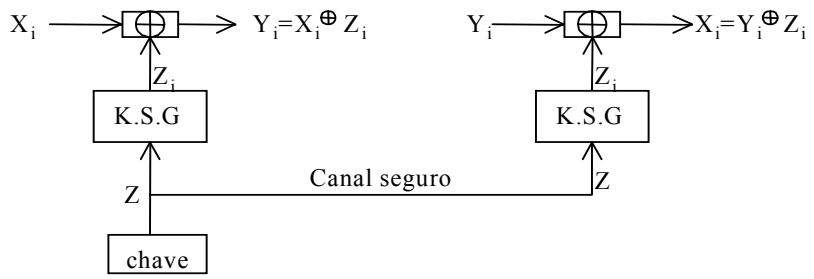
1) Longos períodos sem repetição;
2) Critérios de complexidade linear: Alta complexidade linear (i.é., próxima do período), perfil da complexidade linear, complexidade linear local, etc;
3) Critérios estatísticos. (Por exemplo, em sequências binárias o equilíbrio entre 0's e 1's; o equilíbrio do número de pares ordenados $00, 10, 01, 11$; *etc.*)

4) Confusão: A dependência estatística entre a chave, o texto claro e o texto cifrado é feita de maneira tão complexa que ela se torna inútil ao criptanalista;
5) Difusão: Cada dígito do texto em claro deve influenciar vários dígitos do texto cifrado e cada dígito da chave secreta também deve ter influência sobre vários dígitos do texto cifrado;
6) Critérios de não linearidade de funções booleanas, distância entre as funções lineares, etc.

Todo gerador seguro de chaves deve satisfazer estes critérios, i.e, do ponto de vista teórico os sistemas são projetados satisfazendo estes critérios. O DES apesar de não ser um stream cipher é um bom exemplo de projeto do sistema teórico, pois, com uma chave finita de 56 bits (a qual é muito menor que o tamanho necessário das chaves de outros sistemas) resistiu à criptanálise por mais de 15 anos. (Até que por volta de 1990 quando *Adi Shamir e Eli Biham* criaram a criptanálise diferencial, que é uma poderosa técnica de ataque a sistemas em blocos.)

Na prática, pode ocorrer que um gerador satisfazendo todos os critérios teóricos seja inseguro.

Exemplo 1 : Stream cipher binário aditivo. (Ilustrado na fig.4.2 abaixo)



X_i é um bit da mensagem, Z_i um bit da chave e Y_i um bit do criptograma

*Fig.*4.2

Nesse exemplo o gerador de chaves produz uma key stream cujos elementos pertencem ao corpo de Galois GF(2), que é adicionada módulo 2 aos bits de um texto em claro, obtendo assim o texto cifrado. O processo de decifração é idêntico ao processo de cifração, pois a adição e subtração em GF(2) é a mesma.

Como o processo de cifração e decifração é o mesmo, podemos dizer que o gerador de key streams usado pelo emissor e pelo receptor estão sincronizados. Assim, sempre que se perder este sincronismo é impossível decifrar a mensagem enviada.

Ex. Cifrar a palavra **MATEMATICA**, utilizando a chave K=1001110 obtida do *Linear Feedback Shift Register (LFSR)* < x^3+x+1, 3 > com preenchimento inicial (100).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

W	X	Y	Z	!	&	;	-	.	*
23	24	25	26	27	28	29	30	31	32

Vamos representar o texto em claro no corpo GF(2⁵)

M ↔13 = (01101)
A↔ 1 = (00001)
T ↔20 = (10100)
E ↔5 = (00101)
I ↔9 = (01001)
M ↔13 = (01101)
C ↔3 = (00011)

Texto claro : **MATEMATIC**A
Texto claro codificado: 01101 00001 10100 00101 01101 00001 10100 01001 00011 00001

Texto cifrado codificado: 11110 10101 01001 00010 00100 11011 11010 11010 10111 11100

Texto cifrado: - u i b d . z z w &

5. O SISTEMA EM BLOCOS IDEA

5.1 - INTRODUÇÃO

O sistema em blocos *IDEA (International Data Encryption Algorithm)* foi baseado no sistema em blocos *PES (Proposed Encryption Standard)*. Em ambos os casos o texto em claro e o texto cifrado são blocos de 64 bits, enquanto a chave secreta é longa, 128 bits. Ambos os sistemas são baseados no novo processo de "mistura de diferentes operações de grupos". A confusão necessária é obtida pelo uso sucessivo de três operações de grupos incompatíveis em pares de subblocos de 16 bits e a estrutura do sistema se encarrega de fornecer a difusão necessária. A estrutura do sistema também facilita a implementação, tanto em software quanto em hardware. O sistema IDEA é uma versão melhorada do PES e foi desenvolvido para aumentar a segurança contra a criptanálise diferencial.

5.2- PROCESSO DE CIFRAR

O IDEA é um sistema iterado que consiste de 8 iterações seguidas por uma transformação de saída. A primeira iteração completa e a transformação de saída são mostrados na figura 5.1

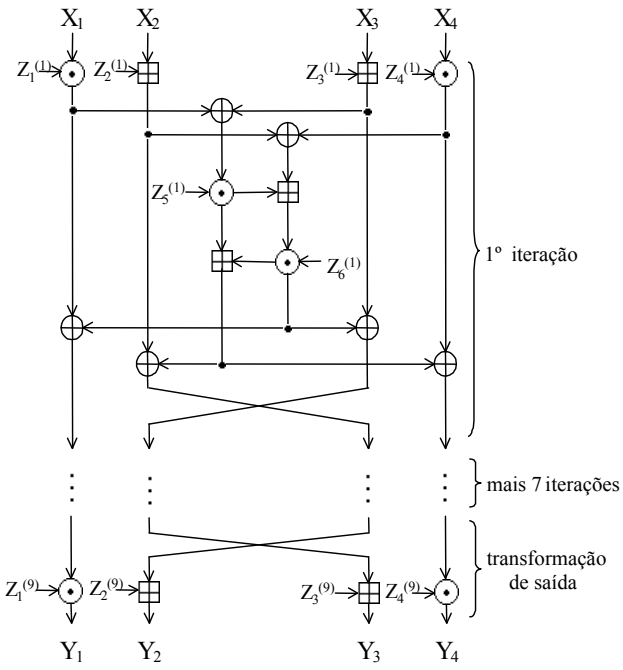


figura 5.1

X_i : Subbloco do texto em claro com 16 bits

Y_i : Subbloco do texto cifrado com 16 bits

$Z_i^{(r)}$: Subbloco da chave secreta com 16 bits

$\oplus$: Soma bit a bit módulo 2, entre subblocos de 16 bits

$\boxplus$: Adição de inteiros módulo 2^{16}

$\odot$: Multiplicação de inteiros módulo $2^{16}+1$, onde o subbloco nulo corresponde ao inteiro 2^{16}.

No processo de cifração mostrado na figura 5.1 são usados três diferentes operações de grupos entre pares de subblocos de 16 bits:

- Soma bit a bit módulo 2, denotado por $\oplus$;

- Adição de inteiros módulo 2^{16}, onde os subblocos de 16 bits são usadas

como números inteiros e representados na base 2; essa operação é denotada por $\boxplus$;

- Multiplicacão de inteiros módulo $2^{16}+1$, operação feita de maneira análoga a adição, onde o subbloco nulo será tratado como o inteiro 2^{16}; essa operação é denotada por $\odot$.

Exemplificando:

$(0,0,...,0) \odot (0,1,0,...,0) = (1,1,0,...,0,1)$, pois
$2^{16} \cdot 2^{14} \bmod (2^{16}+1) = 2^{15} + 2^{14} + 1$

O bloco de texto em claro X de 64 bits é particionado em quatro subblocos de 16 bits X_1, X_2, X_3 e X_4, isto é, $X = (X_1, X_2, X_3, X_4)$. Os quatro subblocos do texto em claro são então transformados em quatro subblocos de 16 bits Y_1, Y_2, Y_3 e Y_4, isto é, o bloco de texto cifrado será $Y = (Y_1, Y_2, Y_3, Y_4)$, sob o controle de 52 subchaves de 16 bits que são formadas a partir dos 128 bits da chave secreta; são usadas 6 subchaves em cada uma das 8 iterações. Para $r = 1, 2, ..., 8$ as seis subchaves usadas na r-ésima iteração é denotada por $Z_1^{(r)}, ..., Z_6^{(r)}$. Quatro subchaves de 16 bits são usadas na transformação de saída. Estas subchaves serão denotadas por $Z_1^{(9)}$, $Z_2^{(9)}$,$Z_3^{(9)}$ e $Z_4^{(9)}$.

5.3 - PROCESSO DE DECIFRAR

O processo de decifrar é essencialmente o mesmo de cifrar (C/D similar), com exceção de que as subchaves $K_i^{(r)}$ usadas para se decifrar não são as mesmas usadas para cifrar, porém as subchaves $K_i^{(r)}$ podem ser obtidas através das subchaves de cifrar da seguinte forma:

$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = (Z_1^{(10-r)^{-1}}, -Z_3^{(10-r)}, -Z_2^{(10-r)}, Z_4^{(10-r)^{-1}})$

para $r = 2, 3, ..., 8$

$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = (Z_1^{(10-r)^{-1}}, -Z_2^{(10-r)}, -Z_3^{(10-r)}, Z_4^{(10-r)^{-1}})$

para $r = 1$ ou $r = 9$ $(K_5^{(r)}, K_6^{(r)}) = (Z_5^{(r)}, Z_6^{(r)})$ para $r = 1, 2, ..., 8$

Aqui Z^{-1} denota o inverso multiplicativo (módulo $2^{16} + 1$) de Z, isto é, $Z \odot Z^{-1} = 1$; e -Z denota o inverso aditivo (módulo 2^{16}) de Z, ou seja, $-Z \boxplus Z = 0$.

5.4 - PROGRAMAÇÃO DAS SUBCHAVES

As 52 subchaves de 16 bits usadas no processo de cifrar são geradas a partir da chave de 128 bits selecionada inicialmente, da seguinte forma, os 128 bits da chaves são usados como as primeiras 8 subchaves, onde a ordenação das subchaves são definidas da seguinte forma:

$Z_1^{(1)}$,$Z_2^{(1)}$...$Z_6^{(1)}$, ...,$Z_1^{(2)}$, ...,$Z_6^{(2)}$, ...,$Z_1^{(8)}$, ...,$Z_6^{(8)}$,$Z_1^{(9)}$,$Z_2^{(9)}$,$Z_3^{(9)}$,$Z_4^{(9)}$. As próximas 8 subchaves são construídas de 128 bits, deslocando-se ciclicamente em 25 bits, para a esquerda, os 128 bits da chave secreta. continuamos com este processo até obtermos todas as 52 subchaves necessárias.

5.5 - OPERAÇÕES DE GRUPOS E SUAS ITERAÇÕES

O sistema IDEA é baseado no processo de mistura de operações de diferentes grupos com o mesmo número de elementos. Operações de grupos foram escolhidas pelo fato da relação estatística de três variáveis aleatórias U,V,W relacionadas por uma operação de grupo $*$, com $W = U * V$ prover segurança perfeita, no sentido que, se alguma das três variáveis é escolhida independentemente das outras, e ter igual probabilidade em um grupo, en-

tão as outras duas variáveis são estatísticamente independentes. A iteração de dife-rentes operações de grupos contribui para a *confusão* necessária à segurança do sistema, como será explicada nas próximas duas seções.

A iteração de diferentes operações de grupos será agora considerada em termos de *isotopismos* de *quasigrupos* e em termos de expressões polinomiais. Para generalizar a discussão além do caso de subblocos de 16 bits, seja n um dos inteiros 1,2,4,8 ou 16. Então $2^n + 1$ é um primo. Seja $\mathbb{Z}_{2^n}$ o anel dos inteiros módulo 2^n e seja $(\mathbb{Z}^*_{2^n+1}, \cdot)$ o grupo multiplicativo do corpo $\mathbb{Z}_{2^n+1}$. Consideremos também $(\mathbb{Z}_{2^n}, +)$, o grupo aditivo do anel $\mathbb{Z}_{2^n}$ e $(GF(2)^n, \oplus)$, o grupo de n-uplas sobre $GF(2)$ com a operação soma bit a bit módulo 2.

5.5.1 - As Três Operações Como Operações de Quasigrupos

Existe incompatibilidade nas propriedades dos três grupos $(GF(2)^n, \oplus)$, $(\mathbb{Z}_{2^n}, +)$ e $(\mathbb{Z}_{2^n+1}, \cdot)$ onde $n \geq 2$. Ou seja, o grupo $(GF(2)^n, \oplus)$, visto como um quasigrupo, não é isotópico a $(\mathbb{Z}_{2^n}, +)$ e nem a $(\mathbb{Z}_{2^n+1}, \cdot)$ e apesar de existir um isotopismo entre os dois últimos, este isotopismo é obrigatoriamente um logaritmo discreto, isto é, uma função muito complexa para se estabelecer uma ligação entre estes grupos. Essa incompatibilidade é muito importante para a segurança do sistema IDEA, pois isso dificulta, e muito, obter informações da operação, que acabou de atuar em um determinado subbloco, através da outra operação.

5.5.2 - Expressões Polinomiais Para Multiplicação e Adição

Uma aplicação direta d é uma aplicação de $\mathbb{Z}^*_{2^n+1}$ em $\mathbb{Z}_{2^n}$ definida por, $d(i) = i$ se $i \neq 2^n$ e $d(2^n) = 0$.

No processo de cifrar do IDEA, a multiplicação módulo $2^n + 1$ e a adição módulo 2^n estão relacionadas pela aplicação direta d e sua inversa d^{-1}. Mais precisamente, multiplicação módulo $2^n + 1$ induz a função

$g : \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n} \longrightarrow \mathbb{Z}_{2^n}$ definida por

$g(x, y) = d[(d^{-1}(x) \cdot d^{-1}(y)) \bmod (2^n + 1)]$ para todo $x, y \in \mathbb{Z}_{2^n}$

Note que $g(x, y)$, para $n = 16$, é a operação que nós denotamos por $\odot$ na seção 4.2. Similarmente, a adição módulo 2^n (a operação $\boxplus$) induz a função $f^* : \mathbb{Z}^*_{2^n+1} \times \mathbb{Z}^*_{2n+1} \longrightarrow \mathbb{Z}^*_{2^n+1}$ definida como

$f^*(x, y) = d^{-1}[(d(x) + d(y)) \mod (2^n + 1)]$ para todo $x, y \in \mathbb{Z}^*_{2^n+1}$

podemos estender a função f^* para a função $f : \mathbb{Z}_{2^n+1} \times \mathbb{Z}_{2^n+1} \longrightarrow \mathbb{Z}_{2^n+1}$ como

$$f(x, y) = \begin{cases} d^{-1}[(d(x) + d(y)) \bmod 2^n] & \text{para todo } x, y \in \mathbb{Z}^*_{2^n+1} \\ 0 \qquad \text{caso contrário} \end{cases}$$

Por exemplo, quando $n = 1$, a função f induzida pela adição módulo 2 é $f(x, y) = 2xy \mod 3 \qquad \forall x, y \in \mathbb{Z}_3$.

Análogamente, a função g induzida pela multiplicação módulo 3 é

$g(x, y) = x + y + 1 \mod 2$ para todo $x, y \in \mathbb{Z}_2$

TEOREMA 5.1: Seja $n \in \{2, 4, 8, 16\}$. Para cada $a \in \mathbb{Z}_{2^n+1} - \{0, 2^n\}$ a função $f(a, y)$ é um polinômio em y sobre o corpo $\mathbb{Z}_{2^n+1}$ de grau $2^n - 1$. Analógamente, para cada $a \in \mathbb{Z}_{2^n+1} - \{0, 2^n\}$, a função $f(x, a)$ é um polinômio em x sobre $\mathbb{Z}_{2^n+1}$ de grau $2^n - 1$.

Prova:

Seja $\mathbb{F}$ =$GF(q)$. Para cada $\alpha \in \mathbb{F}^* = GF(q) - \{0\}$.

$$(-\alpha)\prod_{\beta\in\mathbb{F}^*-\{\alpha\}}(x-\beta) = \begin{cases} 1, \text{ se } x = \alpha \text{ ou } x = 0 \\ 0, \text{ caso contrário} \end{cases}$$

Isso decorre do fato de que o produto de todos os elementos não nulos de um corpo finito é -1.

$$(-\alpha)\prod_{\beta\in\mathbb{F}^*-\{\alpha\}}(x-\beta) = \prod_{\beta\in\mathbb{F}^*}\beta = -1$$

Assim, cada função $h(.)$ de $\mathbb{F}$ em $\mathbb{F}$, pode ser escrita como um polinômio sobre $\mathbb{F}$ de grau no máximo $q-1$ como segue:

$$h(x) = \sum_{\alpha\in\mathbb{F}^*} h(\alpha)(-\alpha)\prod_{\beta\in\mathbb{F}^*-\{\alpha\}}(x-\beta) + (x^{q-1}-1)[\sum_{\alpha\in\mathbb{F}^*} h(\alpha) - h(0)] \qquad (1)$$

Note que $f(0,y) = 0$ para todo y em $\mathbb{F}$, e que $f(a,\cdot)$ para cada $a \neq 0$ é uma bijeção de $\mathbb{F}^*$ em $\mathbb{F}^*$;então $\sum_{\alpha\in\mathbb{F}^*} f(a,\alpha) = \sum_{\alpha\in\mathbb{F}^*} \alpha = 0$, para todo $a \neq 0$ e $\mathbb{F} \neq GF(2)$.

Da definição de $f(x,y)$ a função $f(a,y)$ pode ser escrita para cada $a \neq 0$ como

$$f(a,y) = \begin{cases} a+y, \text{ se } 1 \leq y \leq 2^n - a \\ a+y+1, \text{ se } 2^n - a < y \leq 2^n \end{cases} \qquad (2)$$

De (1) e (2) temos,

$$f(a,y) = \sum_{i=1}^{2^n}(a+i)(-i)\prod_{\substack{j\neq i \\ 1\leq j\leq 2^n}}(y-j) + \sum_{i=2^n-a+1}^{2^n}(-i)\prod_{\substack{j\neq i \\ 1\leq j\leq 2^n}}(y-j)$$

Concluimos então que $f(a,y)$ é um polinômio em y sobre $\mathbb{F}$ com grau menor ou igual a $2^n - 1$. Na realidade, o coeficiente de y^{2^n-1} em $f(a,y)$ é

$$\sum_{i=1}^{2^n}(a+i)(-i) + \sum_{i=2^n-a+1}^{2^n}(-i) = -a\sum_{i=1}^{2^n} i - \sum_{i=1}^{2^n} i^2 + \sum_{i=2^n-a+1}^{2^n}(-i) =$$
$$\sum_{i=-a}^{-1}(-i) = \sum_{i=1}^{a} i = \frac{a(a+1)}{2} \neq 0 \qquad \blacksquare$$

Exemplo: Para $n = 2$, a função

$$f(x,y) = 3(x^3y^2 + x^2y^2) + 3(x^3y + xy^3) + 2x^2y^2 + 4(x^2y + xy^2).$$

Lema 5.1: Se $p(x)$ é um polinômio sobre $\mathbb{Z}_{2^n}$ então, para todo $\beta \in \mathbb{Z}_{2^n}$,

$$P(2\beta) \mod 2 = P(0) \mod 2$$

Prova:

Seja $P(x) = a_k x^k + a_{k-1}x^{k-1} + ... + a_1 x + a_0 \in \mathbb{Z}_{2^n}[x]$.

Então para todo $\beta \in \mathbb{Z}_{2^n}$

$P(2\beta) = a_k(2\beta)^k + a_{k-1}(2\beta)^{k-1} + ... + a_1(2\beta) + a_0$, portanto, $P(2\beta) \bmod 2 = a_0 \bmod 2 = f(0) \bmod 2 \qquad \blacksquare$

Teorema 5.2: Se $n \in \{2, 4, 8, 16\}$ então, para cada $a \in \mathbb{Z}_{2^n} - \{0, 1\}$, a função $g(a, x) = a \odot x = x \odot a$ não pode ser escrita como um polinômio em x sobre o anel $\mathbb{Z}_{2^n}$.

Prova:

Seja $n > 1$, então para cada inteiro a, $1 < a < 2^n$, existe um inteiro $x_0 \in \{1, 2, ..., 2^n\}$ tal que as três inequações são satisfeitas:

$$2^n + 1 < 2ax_0 < 2(2^n + 1) \tag{3}$$

$$0 \leq 2a(x_0 - 1) < 2^n + 1 \tag{4}$$

$$0 \leq 2x_0 \leq 2^n \tag{5}$$

(3) é equivalente a $0 < 2ax_0 - (2^n + 1) < 2^n + 1$ com a condição de $2ax_0 - (2^n + 1)$ ser ímpar. Usando (5) e a definição da função g, temos que

$g(a, 2x_0) = a \odot (2x_0) = 2ax_0 - (2^n + 1)$. Sejam $\beta_1 = x_0$ e $\beta_2 = x_0 - 1$. Então,

$g(a, 2\beta_1) = g(a, 2x_0) \bmod 2 = (2ax_0 - (2^n + 1)) \bmod 2 = 1$.

Por outro lado,

$g(a, 2\beta_2) = g(a, 2(x_0 - 1)) \bmod 2$ e (4) implica que $2(x_0 - 1)$ é um inteiro par em {1,2,...,2^n}.

Logo $g(a, 2\beta_2) = 0 \bmod 2$.

Como $g(a, 2\beta_1) \neq g(a, 2\beta_2)$; pelo lema 4.1 concluímos que $g(a, x) = a \odot x$ não é um polinômio sobre $\mathbb{Z}_{2^n}$. ■

5.6 - SEGURANÇA NO IDEA

Confusão: A confusão necessária para segurança no IDEA é feita pela mistura de três operações de grupos incompatíveis. No esquema mostrado na figura 5.1, no processo de cifração do IDEA, as três diferentes operações de grupos são organizadas de modo que o output de uma operação de um tipo, nunca é usada como input de uma operação do mesmo tipo.

As três operações são incompatíveis no sentido de que:

1. Nenhum par obtido das três operações satisfaz a lei distributiva. Por exemplo, para as operações $\odot$ e $\boxplus$, existem a,b e c em $\mathbb{F}^{16}$, tais que,

$$a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c).$$

Só para ilustrar, quando $a = b = c = 1 = (0, 0, ..., 0, 1)$, o lado esquerdo da desigualdade dá $2 = (0, 0, ..., 0, 1, 0)$, enquanto o lado direito é igual a

$4 = (0, ..., 0, 1, 0, 0)$.

2. Nenhum par das três operações satisfaz a lei "associativa generalizada". Por exemplo, para as operações $\boxplus$ e $\oplus$ existem $a, b, c \in \mathbb{F}^{16}$, tais que,

$$a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c$$

Ilustrando, para $a = b = c = 1 = (0, ..., 0, 1) \in \mathbb{F}^{16}$, o lado esquerdo da desigualdade acima vale $1 = (0, ..., 0, 1)$, enquanto que o lado direito vale $3 = (0, .., 0, 1, 1)$. Assim, não se pode arbitrariamente mudar a ordem das operações para simplificar a análise do algoritmo.

3. As três operações estão ligadas pela aplicação direta d e sua inversa, que inibe isotopismos. O significado criptográfico deste fato é que, se existisse isotopismo entre duas operações, uma poderia ser obtida da outra através de uma aplicação bijetora. $(\mathbb{Z}^*_{2^n+1}, \odot)$ e $(GF(2)^n, \oplus)$ não são isotópicos e $(GF(2)^n, \oplus)$ e $(\mathbb{Z}_{2^n}, \boxplus)$ também não são isotópicos. Além disso, todo isotopismo de $(\mathbb{Z}^*_{2^n+1}, \odot)$ sobre $(\mathbb{Z}_{2^n}, \boxplus)$ é essencialmente um logaritmo discreto. Mais ainda, o logaritmo discreto é geralmente considerada uma função complexa.

4. É possível considerar que as operações $\odot$ e $\boxplus$ agem no mesmo conjunto (ou no anel $\mathbb{Z}_{2^n}$ ou em $\mathbb{Z}^*_{2^n+1}$). Contudo, para isso, deve-se analisar algumas funções altamente não lineares, no sentido de que a multiplicação módulo $2^{16}+1$, que é uma função bilinear sobre $\mathbb{Z}_{2^n+1}$, corresponde a uma função não polinomial sobre $\mathbb{Z}_{2^{16}}$ como foi mostrado no teorema 5.2. A adição módulo 2^{16}, que é uma função afim em cada argumento sobre $\mathbb{Z}_{2^{16}}$, corresponde a um polinômio em duas variáveis sobre em $\mathbb{Z}_{2^{16}+1}$ (de grau no máximo $2^{16} - 1$ em cada variável).

Difusão: Uma verificação direta mostra que a função de iteração é completa, isto é, que cada bit de saída da primeira iteração depende de cada bit do texto claro e de cada bit da chave secreta usada na iteração. Esta difusão é produzida no sistema IDEA por uma transformação chamada estrutura multiplicação-adição (MA), mostrada na figura 5.2. A estrutura MA transforma dois subblocos de 16 bits em dois subblocos de 16 bits controlados por dois subblocos de 16 bits de chave secreta. Esta estrutura tem as seguintes propriedades

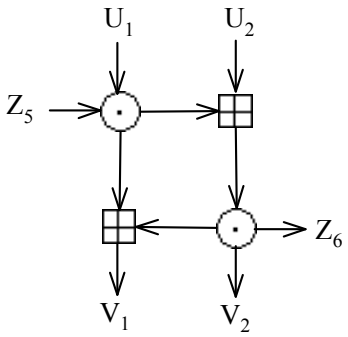


figura 5.2

a) Para toda escolha das subchaves Z_5 e Z_6, $MA(\cdot, \cdot, Z_5, Z_6)$ é uma transformação inversível; para toda escolha de U_1 e U_2, $MA(U_1, U_2, \cdot, \cdot)$ é também uma transformação inversível.
b) Esta estrutura tem uma "difusão completa" no sentido que cada subbloco de saída depende de todos os subblocos de entrada.
c) Esta estrutura usa o menor número de operações (quatro) requerida para realizar tal difusão completa.

Para dar um prova formal destas propriedades, necessitamos das seguintes

definições:

Definição 5.1: Um *grafo orientado (ou dígrafo)* denotado por D(V,E) é um conjunto finito não vazio V (cujos elementos são ditos vértices), e um conjunto E (os elementos de E são dito arestas) de pares ordenados de elementos distintos de V.

Denotamos cada aresta $e \in E$ por um par de vértices e = (v,w). Os vértices v, w são os *extremos (ou extremidades)* da aresta e, sendo v o *extremo de saída* e w o *extremo de entrada*. Num dígrafo cada aresta (v,w) possui uma única direção, de v para w. Um mesmo vértice v pode ser extremo de entrada de uma aresta e_1 e extremo de saída de uma aresta e_2.

Um algoritmo para uma função computacional (por exemplo o da fig. 5.1) determina um grafo orientado, onde os extemos de entradas são as arestas de entradas do algoritmo e os extremos de saídas são as saídas do algoritmo.

Considere a função tendo a forma:

$$(Y_1, Y_2) = E(X_1, X_2, Z_1, Z_2), \quad X_i, Y_i \in GF(2)^n, Z_i \in GF(2)^k, \; i = 1, 2 \quad (6)$$

e tal que, cada escolha de (Z_1, Z_2), $E(\cdot, \cdot, Z_1, Z_2)$ é inversível. Tal função será chamada um sistema de 2 blocos. Um sistema de 2 blocos tem *difusão completa* se para cada uma das variáveis de saída depende de cada variável de entrada.

Lema 5.2: Se um sistema de 2 blocos da forma (6) tem difusão completa, então o grafo computacional determinado por qualquer algoritmo que calcula a função que cifra contém no mínimo 4 operações.

Prova:.

Seja $Y_1 = E_1(X_1, X_2, Z_1, Z_2)$ e $Y_2 = E_2(X_1, X_2, Z_1, Z_2)$. Como E_1 tem

difusão completa, seu grafo computacional tem pelo menos 3 operações por que esta função tem quatro variáveis de entrada. Suponha que E_1 contém exatamente 3 operações. A inversibilidade do sistema de dois blocos implica que $E_2 \neq E_1$ e a difusão completa requer que E_2 não seja igual a algum resultado intermediário que aparece em E_1. Assim pelo menos uma operação que não aparece em E_1 e requerida no grafo computacional de E_2. Isto demonstra o lema.

5.6.1 - Segurança Perfeita do "One-time key"

A segurança perfeita no sentido de Shannon é obtida em cada iteração da função de cifrar se cada chave é usada um única vez.("one-time key".) Na realidade, tal segurança perfeita é alcançada na transformação de entrada na primeira iteração porque cada operação é uma operação de grupo. Mais ainda, para cada escolha (p_1, p_2, p_3, p_4) e (q_1, q_2, q_3, q_4) em $GF(2)^{64}$, existem exatamente 2^{32} diferentes escolhas para as subchaves $(Z_1, ..., Z_6)$ de modo que a função que cifra na primeira iteração transforma $(p_1, ..., p_4)$ em $(q_1, ..., q_4)$.

5.6.2 - Implementação do Sistema IDEA

O sistema IDEA pode ser facilmente implementado em software porque usa somente operações básicas entre pares de bolcos de 16 bits no processo de cifrar. O sistema foi implementado na linguagem de programação TURBO PASCAL versão 7.0, em um computador pessoal com processador pentium 200, e alguns dados de amostra foram usados para checar a exatidão da implementação.

A estrutura regular modular do sistema facilita também a implementação

em hardware. A similaridade entre o processo de cifrar e decifrar do IDEA, mostrado na próxima seção, torna possível o uso do mesmo algoritmo em ambos os processos de cifrar e decifrar. Um algoritmo para calcular a operação $\odot$ é descrito na seção 5.6.4.

5.6.3 - Similaridade Entre o Processo de Cifrar e Decifrar

O processo para decifrar é essencialmente o mesmo processo de cifrar; existe somente uma diferença nos subblocos de chaves secretas usadas; assim, o mesmo algoritmo pode ser usado em ambos os processos, de cifrar e decifrar. O único trabalho extra está no processo de criar as subchaves (para cifrar e decifrar) a partir da chave inicial de 128 bits. A seguir mostraremos que a função de cifrar do IDEA é constituída de um sistema de uma operação de grupo, uma involução e uma permutação involuntória que é um automorfismo do grupo $(GF(2)^{64}, \otimes)$. O sistema IDEA apresenta similaridade entre o processo de cifrar e decifrar.

Para o processo de cifrar do IDEA mostrado na figura 5.1, definamos

$$X \otimes Z_A = (X_1 \odot Z_1, X_2 \boxplus Z_2, X_3 \boxplus Z_3, X_4 \odot Z_4).$$

Podemos ver facilmente que $(GF(2)^{64}, \otimes)$ é um grupo.

Seja $P_I(X)$ a permutação em X que permuta os subblocos X_2 e X_3 de $X = (X_1, X_2, X_3, X_4)$ no fim de cada iteração. Então P_I é uma involunção e $P_I(X \otimes Z_A) = P_I(X) \otimes P_I(Z_A)$, de modo que P_I é um automorfismo no grupo $(\mathbb{F}_2^{64}, \otimes)$.

Resta mostrar que a função $I_n(\cdot, Z_B)$, mostrada na figura 5.3, com os 64 bits de entrada (S_1, S_2, S_3, S_4) e os 64 bits de saída (T_1, T_2, T_3, T_4) controlados pelos 32 bits da chave $Z_B = (Z_5, Z_6)$, é uma involução. Isto é, para todo

Z_B fixo, o inverso da função $I_n(\cdot, Z_B)$ é ela mesma. Esta propriedade é consequência do fato que a soma módulo 2 de (S_1, S_2) e (S_3, S_4) é igual a soma módulo 2 de (T_1, T_2) e (T_3, T_4).

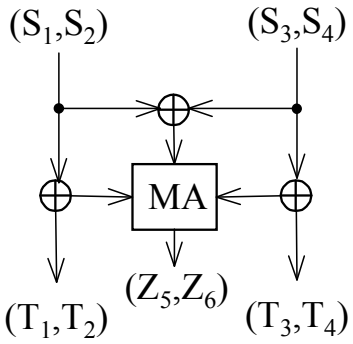


figura 5.3

5.6.4 - Algoritmo da Multiplicação

A maior dificuldade para implementação do sistema IDEA é a implementação da multiplicação módulo $(2^{16}+1)$. Ela pode ser implementada através da sugestão dada pelo lema 5.3.

Obs.: $(ab\ div\ 2^n)$ denota o quociente da divisão de ab por 2^n. Note que $(ab \bmod 2^n)$ corresponde aos n bits menos significantes de ab e $(ab\ div\ 2^n)$ é justamente o deslocamento para a direita dos n bits de ab.

Lema 5.3: Sejam a, b dois n-bits não nulos de inteiros em $\mathbb{Z}_{2^n+1}$. Então

$$ab \bmod (2^n+1) = \begin{cases} (ab \mod 2^n) - (ab\ div\ 2^n), \text{ se } (ab \mod 2^n) \geq (ab\ div\ 2^n) \\ (ab \mod 2^n) - (ab\ div\ 2^n) + 2^n + 1, \text{ caso contrário} \end{cases}$$

Prova:

Quaisquer que sejam $a, b \in \mathbb{Z}^*_{2^n+1}$, existem inteiros q e r (únicos) tais que

$ab = q(2^n+1) + r, \qquad 0 \leq r < 2^n + 1, \qquad 0 \leq q < 2^n.$

Mais ainda, $2^n \leq q + r < 2^{n+1}$. Observe que $r = ab \mod (2^{n+1})$. Temos

$$ab \; div \; 2^n = \begin{cases} q, \text{ se } q + r < 2^n \\ q + 1, \text{ se } q + r \geq 2^n \end{cases}$$

e

$$ab \mod 2^n = \begin{cases} q + r, \text{ se } q + r < 2^n \\ q + r - 2^n, \text{ se } q + r \geq 2^n. \end{cases}$$

Assim,

$$r = \begin{cases} (ab \mod 2^n) - (ab \; div \; 2^n), & \text{se } q + r < 2^n \\ (ab \mod 2^n) - (ab \; div \; 2^n) + 2^n + 1, \text{ se } q + r \geq 2^n. \end{cases}$$

Mas, $q + r < 2^n$ se, e somente se $(ab \mod 2^n) \geq (ab \text{ div } 2^n)$.■

Exemplo: $n = 2 \Rightarrow \mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ $a, b \in \mathbb{Z}_5^*$, para $a = 2 = (1, 0)$ e $b = 3 = (1, 1) \Rightarrow ab \bmod 5 = 1$, usando o lema 5.3

$$\begin{cases} ab \mod 4 = 2 \\ ab \text{ div } 4 = 1 \end{cases} \Rightarrow ab \bmod 5 = (ab \bmod 4) - (ab \; div \; 4) = 1$$

para $a = 1 = (1, 0, 0)$ e $b = 4 = (1, 0, 0)$

$$\begin{cases} ab \mod 5 = 4 \\ ab \text{ div } 4 = 1 \\ ab \mod 4 = 0 \end{cases} \Rightarrow ab \bmod 5 = (ab \bmod 4) - (ab \; div \; 4) + 5 = 4.$$

BIBLIOGRAFIA

[1] Cusick, T. W, Ding, C. and Renvall, A., *Stream Ciphers and Number Theory,* Elsevier Science B.V, Amsterdam, 1998.

[2] Dénes J. and Keedwell A.D. I. N., *Latin Squares and Their Applications,* Akdémiai Kiadó, *Budapest,* 1974.

[3] *Federal Information Processing Data Encryption Standart,* Federal Register, August 1, 1975.

[4] Ferreira, N.C., "*IDEA- Um Sistema Criptográfico em Blocos"* IME-UFG, Goiânia, dissertação de mestrado, 2000.

[5] Konheim A. G. *Cryptography: A Primer*, New York: Wiley - Interscience, 1981.

[6] Knudsen L. R. *Block Ciphers - Analysis, Desing and Applications,* DAIMI PB - PB - 485, 1994.

[7] Lai X. *On the Design and Security of block ciphers*, ETH Series in Information Processing vol.1, 1992.

[8] Lai X. and Massey J. L. *A Proposal for a New Block Encryption Standard*, Advances in Cryptology - EUROCRYPT´90, Proceedings, LNCS 473, pp. 389-404, Springer-Verlag, Berlin, 1991.

[9] Lemos, G.C., "*Uma Análise da Estrutura e Complexidade Linear de Sequências Binárias Produzidas por Geradores não Lineares*" IME-UFG, Goiânia, dissertação de mestrado, 1999.

[10] Massey J. L. *Applied Digital Information Theory II,* notas de curso, ETH - Zentrum, Zürich, 1993.

[11] Massey, J. L. and Serconek, S., *A Fourier Transform Approach to the Linear Complexity of Nonlinearly Filtered Sequences,* Advances in Cryptology-Crypto'94 (Ed. Yvo G. Desmedt), Lecture Notes in Computer Science, Vol. 839, Springer-Verlag, Berlin, Heidelberg, 1994.

[12] Rivest R. L., Shamir A. and Adleman L. *A Method for Obtaining Digital Signatures and Public-Key Criptosystems,* Communications of the ACM, Vol 21, No 2, 1978.

[13] Rueppel, Rainer A., *Analysis and Design of Stream Ciphers*, Springer-Verlag, Berlin, Heidelberg, 1986.

[14] Shannon C. E. *Communication Theory of Secrecy System*, Bell System Technical Journal, Vol. 28, pp. 656-715, Oct. 1949.

[15] Simmons, G. J., *Contemporary Cryptology, IEEE, Inc.,* pp. 67-133, New York, 1992.

[16] Szwarcfiter J. L. *Grafos e Algorimos Computacionais*, Campus, Rio de Janeiro 1988.