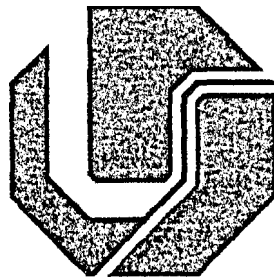


1101
621.5
F 266p
TES/ME

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



PROJETOS DE SISTEMAS CRIPTOGRÁFICOS UTILIZANDO
CÓDIGOS LINEARES

DIRBI/UFU



1000187001

JOSÉ LUIZ DE FREITAS JÚNIOR

JULHO

1998

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**PROJETOS DE SISTEMAS CRIPTOGRÁFICOS UTILIZANDO
CÓDIGOS LINEARES**

Dissertação apresentada por José Luiz de Freitas júnior à
Universidade Federal de Uberlândia, para obtenção do título
de Mestre em Engenharia Elétrica – aprovada em 10/07/98
pela Banca Examinadora:

Professor João Nunes de Souza, Dr. - UFU - Orientador

Professora Márcia Aparecida Fernandes, Dr^a. - UFU

Professor Gilberto Arantes Carrijo, Dr. - UFU

Professor Antônio Alfredo F. Loureiro, Dr. - UFMG

**PROJETOS DE SISTEMAS CRIPTOGRÁFICOS UTILIZANDO
CÓDIGOS LINEARES**

JOSÉ LUIZ DE FREITAS JÚNIOR

Dissertação apresentada por José Luiz de Freitas júnior à
Universidade Federal de Uberlândia visando obter o título de
Mestre em Engenharia Elétrica

Prof. Dr. João Nunes de Souza,
Orientador

Prof. Dr. Darizon Alves de Andrade
Coordenador do Curso de Pós Graduação

EPÍGRAFE

À Deus pela minha vida e oportunidades

Aos meus familiares

À minha esposa Edivânia e meu filho Rafael, pelo afago, diversão e apoio.

“O senhor ... Mire e veja que o mais importante e bonito do mundo é isto, que as pessoas não estão sempre iguais, não foram terminadas, mas que elas vão sempre mudando. Afinam ou desafinam – verdade maior. É o que a vida me ensinou. Isto me alegra, montão”. (Guimarães Rosa [Cun,1996])

AGRADECIMENTOS

À Universidade Católica de Goiás, pelo investimento e credibilidade, em especial ao professor e amigo José Roldão G. Barbosa, pelo seu constante incentivo e empenho em resolver os reveses com muita justiça, sabedoria e perspicácia; aos professores do Departamento de Ciência da Computação, pela prova de verdadeiro companheirismo.

À Caixa Econômica Federal, por viabilizar minha transferência durante o curso.

À Universidade Federal de Uberlândia, por me permitir vivenciar o curso de Mestrado, especialmente ao professor Dr. João Nunes de Sousa que com seu bom humor, capacitação e benevolência tornou possível, juntamente com a professora Dr. Márcia Aparecida Fernandes, a produção desta tese.

Ao professor Dr. Edilberto, pelo constante incentivo e orientação.

Ao professor Dr. Jaime Portugheis (Unicamp), pela disponibilidade e eficiência ao encaminhar a pesquisa, mesmo estando distante.

À professora e esposa Edivânia, por rever minha produção escrita, tornando-a mais compreensível.

Aos colegas mestrandos, pelo coleguismo e divertimento durante o curso.

A todas essas pessoas e instituições, espero retribuí-las de alguma forma, com a mesma dedicação e competência. Pretendo também, poder acrescentar algo de útil no tocante a criptografia e teoria algébrica dos códigos à todas as pessoas interessadas pelo tema.

“Não se trata de descobrir e percorrer sozinho uma única vez uma pista. Mas de traçar e de concluir, para uso de muitos, uma larga pista”. (Lebert [Cun,1996])

RESUMO

Faz-se neste trabalho um estudo dos fundamentos de criptografia clássica, de chave pública e de teoria dos códigos. Em seguida, utiliza-se códigos lineares e cíclicos para projetar sistemas criptográficos clássicos e de chave pública.

Tendo como referência os sistemas criptográficos DES e RSA, verifica-se a eficiência dos sistemas propostos.

ABSTRACT

This work has been done to study the foundations of classic cryptography, of public key and of code theories. It follows, to apply the linear and cyclic codes to project classical cryptographic systems and of public key.

Having as reference the DES and RSA system, we verify the efficiency of the proposed systems.

PROJETOS DE SISTEMAS CRIPTOGRÁFICOS UTILIZANDO CÓDIGOS LINEARES

SUMÁRIO

1	ESTUDO DA ARTE	01
1.1	INTRODUÇÃO.....	01
1.2	TRABALHOS RELACIONADOS COM O TEMA.....	05
1.3	ORGANIZAÇÃO DA DISERTAÇÃO	06
2	CRIPTOGRAFIA CLÁSSICA E DE CHAVE PÚBLICA.....	08
2.1	CRIPTOGRAFIA CLÁSSICA	08
2.1.1	INTRODUÇÃO.....	08
2.1.2	NOTAÇÕES.....	09
2.1.3	SISTEMA CLÁSSICO – DES (CIFRAÇÃO DE DADOS PADRÃO) ..	14
2.1.4	CRIPTOANÁLISE DO DES.....	25
2.2	CRIPTOGRAFIA DE CHAVE PÚBLICA.....	26
2.2.1	INTRODUÇÃO.....	26
2.2.2	FUNÇÕES UNIDIRECIONAIS.....	28
2.2.3	TEORIA DE COMPLEXIDADE.....	30

2.2.4	ALGUMAS VANTAGENS DAS CHAVES PÚBLICAS.....	32
2.2.5	O RSA	34
2.2.5.1	INTRODUÇÃO	34
2.2.5.2	DESCRIÇÃO	35
2.2.5.3	PROJETO	36
2.2.5.4	CRIPTOANÁLISE E FATORAÇÃO.....	41
2.3	CONCLUSÃO	44
3	CÓDIGO SEGURO PARA ARMAZENAMENTO E TRANSMISSÃO	
	DIGITAL.....	46
3.1	INTRODUÇÃO.....	46
3.2	SISTEMA TÍPICO DE TRANSMISSÃO (OU ARMAZENAMENTO).....	47
3.3	TIPOS DE CÓDIGOS	50
3.4	CÓDIGOS DE BLOCO LINEARES.....	51
3.4.1	INTRODUÇÃO.....	51
3.4.2	ESTRUTURA SISTEMÁTICA	54
3.4.3	MATRIZ DE VERIFICAÇÃO DE PARIDADE.....	55
3.4.4	SÍNDROME E DETECÇÃO DE ERROS	56
3.4.5	DISTÂNCIA MÍNIMA	58

3.4.6	CAPACIDADE DE DETECÇÃO E CORREÇÃO DE ERROS	
	ALEATÓRIOS	61
3.4.7	CONJUNTO PADRÃO E DECODIFICAÇÃO SÍNDROME.....	65
3.4.8	CÓDIGOS DE HAMMING	72
3.5	CONCLUSÃO.....	73
4	CÓDIGOS CÍCLICOS.....	75
4.1	INTRÔDUÇÃO.....	75
4.2	DEFINIÇÃO DE CÓDIGOS CÍCLICOS.....	75
4.3	PROPRIEDADES BÁSICAS DE CÓDIGOS CÍCLICOS.....	76
	4.3.1 ESQUEMA SINTÉTICO DAS PROPRIEDADES BÁSICAS.....	80
4.4	ESTRUTURA SISTEMÁTICA DE CÓDIGOS CÍCLICOS	80
4.5	MATRIZES GERADORA E DE VERIFICAÇÃO DE PARIDADE.....	82
	4.5.1 MATRIZ GERADORA (G).....	82
	4.5.2 MATRIZ DE VERIFICAÇÃO DE PARIDADE (H).....	83
	4.5.3 MATRIZES: GERADORA E DE VERIFICAÇÃO DE PARIDADE NA FORMA SISTEMÁTICA	85
4.6	CÁLCULO DA SÍNDROME E DETECÇÃO DE ERROS.....	85
4.7	DECODIFICAÇÃO DE CÓDIGOS CÍCLICOS.....	87
4.8	CONCLUSÃO.....	89

5	SISTEMAS CRIPTOGRÁFICOS CLÁSSICOS USANDO TEORIA ALGÉBRICA DOS CÓDIGOS LINEARES.....	90
5.1	INTRODUÇÃO.....	90
5.2	SISTEMA CRIPTOGRÁFICO CLÁSSICO USANDO CÓDIGOS LINEARES.....	90
5.2.1	CRIPTOANÁLISE DO SISTEMA.....	97
5.2.2	CONSIDERAÇÕES SOBRE ESTE SISTEMA.....	98
5.3	SISTEMA CRIPTOGRÁFICO CLÁSSICO COMPOSTO POR CÓDIGOS LINEARES E DES.....	100
5.3.1	CRIPTOANÁLISE DESTE SISTEMA.....	101
5.4	CONCLUSÃO.....	103
6	SISTEMA CRIPTOGRÁFICO DE CHAVE PÚBLICA USANDO TEORIA ALGÉBRICA DOS CÓDIGOS CÍCLICOS.....	104
6.1	INTRODUÇÃO.....	104
6.2	CONCEITOS FUNDAMENTAIS SOBRE MATRIZES.....	105
6.2.1	MATRIZ NÃO SINGULAR.....	105
6.2.2	MATRIZ DE PERMUTAÇÃO.....	105
6.3	SISTEMA CRIPTOGRÁFICO DE CHAVE PÚBLICA.....	106
6.4	CRIPTOANÁLISE DO SISTEMA.....	110

6.5	CONCLUSÃO.....	114
7	CONCLUSÕES.....	116
8	REFERÊNCIAS BIBLIOGRÁFICAS.....	119

LISTA DE FIGURAS

2.1	ELEMENTOS BÁSICOS PARA TRANSMISSÃO DIGITAL	09
2.2	NOTAÇÕES UTILIZADAS EM UM SISTEMA DE TRANSMISSÃO DE DADOS	10
2.3	UMA VISÃO MACROSCÓPICA DO DES	16
2.4	A PERMUTAÇÃO N.....	17
2.5	A PERMUTAÇÃO IP	17
2.6	A TRANSFORMAÇÃO T	19
2.7	A TRANSFORMAÇÃO G.....	20
2.8	A EXPANSÃO E	20
2.9	AS CONTRAÇÕES S	21
2.10	A PERMUTAÇÃO P.	22
2.11	A PERMUTAÇÃO PC-1.....	23

2.12	O VETOR R INDICA O NÚMERO DE POSIÇÕES DOS DESLOCAMENTOS CIRCULARES DE C E D	23
2.13	ESCOLHA PC-2. ENTRADA 56 BITS; SAÍDA: 48 BITS.....	24
2.14	REPRESENTAÇÃO DA COMPLEXIDADE DE UMA FUNÇÃO.....	28
2.15	TEMPO DE EXECUÇÃO DO ALGORITMO DA FATORAÇÃO DE SCHROEPPPEL	43
3.1	SISTEMA TÍPICO DE TRANSMISSÃO	47
3.2	SISTEMA TÍPICO DE TRANSMISSÃO REDUZIDO.....	49
3.3	FORMATO SISTEMÁTICO DE UMA PALAVRA CÓDIGO	54
3.4	CONJUNTO PADRÃO PARA UM CÓDIGO LINEAR	66
5.1	CAPACIDADE DE CORREÇÃO DE ERROS (E) X COMPRIMENTO DA MENSAGEM (K) DE UM CÓDIGO LINEAR DE TAMANHO N	97
5.2	POSSIBILIDADES DE MATRIZES DE VERIFICAÇÃO DE PARIDADE H.99	
5.3	BLOCO DE MENSAGEM DE COMPRIMENTO K E POSSÍVEIS BLOCOS DE K CIFRADOS	102
6.1	TENTATIVA DE ASSINATURA DIGITAL UTILIZANDO PALAVRAS COM K BITS	113
6.2	TENTATIVA DE ASSINATURA DIGITAL UTILIZANDO PALAVRAS COM N BITS	114

LISTA DE TABELAS

3.1	CÓDIGO DE BLOCOS BINÁRIO COM $K=4$ E $N=7$	51
3.2	UM CONJUNTO PADRÃO PARA O CÓDIGO LINEAR (7,3)	68
3.3	TABELA DE DECODIFICAÇÃO PARA O CÓDIGO LINEAR (7,4) DADO NA TABELA 3.1.....	71
4.1	CÓDIGO CÍCLICO (7,4) GERADO PELO POLINÔMIO $G(X) = 1+X+X^3$...	76
4.2	DECODIFICAÇÃO DO CÓDIGO CÍCLICO (7,4) GERADO POR $G(X) = 1+X+X^3$	89
5.1	REPRESENTAÇÃO BINÁRIA DAS LETRAS DO ALFABETO	91
5.2	SISTEMA CLÁSSICO USANDO CÓDIGOS LINEARES	93
5.3	CÓDIGO LINEAR (9,5).....	94
5.4	TABELA DE DECODIFICAÇÃO PARA O CÓDIGO LINEAR (9,5)	96
6.1	SISTEMA DE CHAVE PÚBLICA USANDO CÓDIGOS CÍCLICOS	108
6.2	TABELA DE DECODIFICAÇÃO SÍNDROME PARA O CÓDIGO CÍCLICO (7,4)	110

LISTA DE ABREVIATURAS E SÍMBOLOS

DES	DATA ENCRYPTION STANDARD
NSA	NATIONAL SECURITY AGENCY
PT	TEXTO PRINCIPAL
CT	CRIPOTEXTO
RSA	RIVEST, SHAMIR E ADLEMAN
MDC	MÁXIMO DIVISOR COMUM
NBS	NATIONAL BUREAU OF STANDARDS
ANSI	AMERICAN NATIONAL STANDARDS INSTITUTE

PROJETOS DE SISTEMAS CRIPTOGRÁFICOS

UTILIZANDO CÓDIGOS LINEARES

CAPÍTULO I

ESTUDO DA ARTE

1.1 - INTRODUÇÃO

Este trabalho tem por objetivo o estudo e desenvolvimento de sistemas criptográficos clássicos e de chave pública, utilizando teoria algébrica dos códigos lineares, com o intuito de conhecer melhor o assunto e verificar a viabilidade de tais sistemas em relação ao DES [Luc,1986] e RSA [Sal,1990].

O interesse por este tema, advém do fato de eu ser professor e economiário e atuar com disciplinas e tarefas que requerem conhecimentos relativos à segurança de dados; são várias as aplicações de sistemas criptográficos, tanto em ciência da computação quanto em engenharia, e, principalmente, em várias outras áreas como aplicações militares, diplomáticas, comércio eletrônico, no tocante à segurança de dados.

Técnicas de proteção criptográficas são necessárias para transmissão e armazenamento de informações que transitam em ambientes de comunicações de dados.

Para desenvolver tais sistemas criptográficos serão expostos bases teóricas imprescindíveis para produção e compreensão dos mesmos, tendo como referência as obras citadas na bibliografia, bem como as orientações dos professores orientadores.

A palavra *Criptografia* vem do grego (*kryptós* = escondido + *grapho* = grafia) – é, portanto, a arte ou ciência de escrever em cifra ou em código, a fim de tornar a

mensagem escrita compreensível apenas a seu destinatário, para decifrá-la, quase sempre requer o conhecimento de uma chave, uma informação secreta. É um dos mecanismos de segurança mais utilizados atualmente e surgiu da necessidade de enviar informações “sensíveis” através de meios de comunicação não confiáveis [Sal,1990] e [Luc,1986].

Mas, através da arte ou ciência chamada *criptoanálise*, do grego *kryptós* + *análisis* = decomposição; terceiros podem quebrar o sistema e determinar o texto original, mesmo desconhecendo a chave – de posse da mensagem cifrada. Da união entre criptografia e criptoanálise surgiu a *criptologia* (do grego *kryptós* = oculto + *lógos* = estudo) que é uma ciência usada desde a escrita hieroglífica dos Egípcios - há quase quatro mil anos, a mesma vem sendo muito utilizada, principalmente para fins militares e diplomáticos, como exemplo, pode-se citar sua utilização na Segunda Guerra, e a conseqüente quebra dos códigos alemão e japonês, que foi fundamental para o sucesso dos Aliados [Luc,1986].

Quanto ao tipo, a criptografia pode ser de:

- * **Chave secreta** – a qual utiliza a mesma chave para cifrar (método secreto de escrita, através do qual transforma-se o texto original em código) e decifrar (processo inverso de cifrar) uma mensagem. Neste caso, emissor e destinatário combinam a chave secreta a ser usada na transmissão, em conseqüência é grande a possibilidade de violação.
- * **Chave pública** – projetada por Diffie e Hellman [Sal,1990], ela dificulta a violação, através de duas chaves: a pública - de conhecimento de todos; e a privada - conhecida apenas pelo seu dono. Então, o emissor utiliza a chave pública do destinatário para cifrar a mensagem e a envia, o destinatário, por sua vez, utiliza sua chave privada para decifrar a mensagem. A criptografia de chave pública possui muitas vantagens em relação a de chave

privada, dentre elas, a verificação de assinaturas através de métodos de autenticação. Entretanto, a velocidade é uma desvantagem grande, devido às operações de cifragem e decifragem requererem cálculos com números muito grandes.

A concepção de criptosistemas de chave privada, baseados em códigos corretores de erros, tem atraído o interesse de pesquisadores que atuam na área de Teoria da Informação e, desde o surgimento do primeiro criptosistema deste tipo, em 1978 [Van,1988], até os dias de hoje, contribuições importantes têm sido dadas a criptografia através da concepção de novos esquemas de cifragem que empregam teoria dos códigos¹.

Teoria dos códigos começou em 1940 com o trabalho de Golay, Hamming e Shannon [Hol,1992], apesar do problema tratado ser de engenharia, desenvolveu-se através de técnicas matemáticas mais sofisticadas, dando origem a famílias de códigos – por exemplo, códigos de Hamming, Cíclicos e BCH, como também códigos mais avançados, tais como códigos Golay, Goppa, Alternant, Kerdock e Preparata [Hol,1992].

Códigos foram inventados para corrigir erros sobre o canal de comunicação com ruído [Hol,1992]. A transmissão/armazenamento de dados em um canal de comunicação ocorre apenas em uma direção, da origem para o destino. Logo, controles de erros para este tipo de sistema, devem ser realizados utilizando código de correção de erros que corrige automaticamente os erros detectados no destino.

Quanto ao tipo, os códigos podem ser: códigos de blocos e códigos convolucionais. Códigos de blocos lineares (códigos lineares) são uma subclasse dos códigos de blocos – objeto do presente estudo.

¹ O livro [Van,1988] apresenta o sistema MCELIECE, o qual utiliza códigos Goppa.

* **Códigos de blocos** - a codificação de um código de bloco divide a seqüência de informação em blocos de mensagem de k bits. Um bloco de mensagem é representado pela k -tupla binária $u = (u_1, u_2, \dots, u_k)$ chamada mensagem. A codificação transforma cada mensagem u em uma n -tupla $v = (v_1, v_2, \dots, v_n)$ de símbolos discretos, chamado de palavra código. Logo, correspondendo a 2^k possíveis mensagens diferentes, existem 2^k possíveis palavras códigos diferentes. Este conjunto de 2^k palavras códigos de tamanho n é chamado um código de blocos (n, k) . A razão $R = k/n$ é chamada razão do código e pode ser interpretada como o número de bits de informação entrando no canal de codificação por símbolo transmitido.

Em um código de bloco binário (código binário), cada palavra código v é também binária. Conseqüentemente, para um código binário ser útil (isto é, ter uma palavra código diferente associada a cada mensagem), $k \leq n$ ou $R \leq 1$. Quando $k < n$, $n-k$ bits de redundância podem ser adicionados a cada mensagem para formar uma palavra código. Estes bits de redundância permitem, eventualmente, a correção de erros provocados pelo canal.

* **Códigos lineares** - Um código de bloco de tamanho n e 2^k palavras código é denominado um código linear (n, k) , se e somente se, estas 2^k palavras códigos formar um subespaço de k -dimensão do espaço vetorial de todas as n -tuplas sobre o corpo $GF(2)$ [Lin,1983]. Um código de bloco binário é linear, se e somente se, a soma de duas palavras código em módulo-2 for também uma palavra código.

Um código linear C é denominado cíclico, se para toda palavra código $v = (v_0, v_1, \dots, v_{n-2}, v_{n-1}) \in C$, existir também uma palavra código $v^{(1)} = (v_{n-1}, v_0, v_1, \dots, v_{n-2}) \in C$ [Wic,1995].

Deve-se considerar como base da teoria da informação ligada às comunicações e às transmissões com sigilo: nas comunicações deve-se eliminar o ruído restaurando a informação original; na criptografia deve-se eliminar o ruído introduzido através de ciframento de forma a restaurar a informação original.

1.2 - TRABALHOS RELACIONADOS COM O TEMA

Conforme consulta à Internet, indica-se a seguir alguns “workshops” e “projetos” atuais em desenvolvimento/desenvolvidos, cujos assuntos se relacionam com o tema proposto.

* Visando satisfazer as exigências das novas tecnologias, foi entregue em abril/1998, ao governo dos Estados Unidos, o projeto de um criptosistema para substituir o DES, cujo nome é “Advanced Encryption Standard (AES)” desenvolvido pelo laboratório RSA - Professor Ronald L. Rivest (web: <http://www.rsa.com>).

* A equipe do Laboratório RSA tem criado inovações em tecnologia de cifragem de alta escala, dentre elas: “ the RC2, the standard for e-mail encryption; RC4, the standard for Secure Socket Layer (SSL); and the recently patented RC5 algorithm designed for today's high-speed data encryption (web: <http://www.rsa.com>).

* Os professores Lima, R. C. C. e Campello de Souza, R. M. do Departamento de Eletrônica e Sistemas da Universidade Federal de Pernambuco – Recife, estão desenvolvendo o projeto “Criptografia via codificação de canal”, tendo como objetivo o estudo e a construção de novos criptosistemas de chave privada que utilizam códigos lineares (web: <http://www.ufp.br> – fone: 081.271 8210).

* Robert Morelos-Zaragoza disponibilizou em Junho/97 um programa em linguagem C para codificação e decodificação de códigos BCH binários. Os métodos de codificação usados no programa foram baseados no livro: “Error Control Coding: Fundamentals and Applications” – Lin and Costello, Prentice Hall, 1983 (e-mail:r.morelos-zaragoza@ieee.org).

* Conferências internacionais do IMA (Institute for Mathematical and its Applications) sobre criptografia e códigos. A primeira conferência ocorreu em dezembro/1986; a quinta em dezembro/1995, Cirencester, UK, dentre os trabalhos expostos cita-se: Authentication Codes: na Area where Coding and Cryptology Meet; Coding and Cryptography for Speech and Vision; Constructions for Variable-Length Error-Correcting Codes, ..., [Boy,1995] e a sexta ocorreu em dezembro de 1997, Cirencester, UK. (<http://www.springer-ny.com>).

* Workshop on Coding Theory, Cryptography, and Computer Security, Lethbridge, Alberta. August 3-7, 1998. Organizer(s): H. Kharaghani, W. H. Holzmann - University of Lethbridge (<http://www.pims.math.ca/sections/activities/exthem98.11.html>).

1.3 - ORGANIZAÇÃO DA DISERTAÇÃO

Neste estudo serão abordados:

- 1- os sistema criptográficos DES e RSA
- 2 - códigos lineares e cíclicos;

Para o desenvolvimento dos sistemas serão utilizados vários conceitos que serão apresentados previamente nos capítulos que se seguem.

O capítulo II introduz os conceitos básicos de criptografia de chave secreta e apresenta o DES, um dos criptosistemas clássicos mais utilizados. Aborda-se os princípios gerais da construção dos criptosistemas de chave pública, suas principais vantagens em relação à criptografia clássica. Mostra-se um dos criptosistemas mais utilizados atualmente, o RSA. Ele é baseado em uma idéia simples sobre teoria dos números, enquanto é fácil multiplicar dois números primos da ordem de 100 dígitos, é extremamente difícil fatorar o seu produto da ordem de 200 dígitos.

O capítulo III introduz os conceitos básicos de código seguro para armazenamento e transmissão digital. Apresenta os códigos de blocos lineares e sua estrutura geral. Discute os conceitos de detecção e correção de erros e apresenta uma técnica de decodificação geral usando conjunto padrão e decodificação síndrome [Wic,1995].

O capítulo IV enfoca a teoria geral dos códigos lineares mais usados na prática, os códigos cíclicos, envolvendo suas propriedades básicas e descrição do polinômio gerador através da teoria de polinômios sobre corpos de Galois.

No capítulo V serão desenvolvidos sistemas criptográficos clássicos utilizando teoria algébrica dos códigos lineares, sendo um sistema utilizando códigos lineares e outro composto com o sistema DES.

O capítulo VI apresenta um sistema criptográfico de chave pública utilizando teoria algébrica dos códigos cíclicos, tendo como referência o sistema RSA, analisa-se a eficiência do sistema proposto. Finalmente, no capítulo VII são apresentadas as conclusões gerais desse trabalho.

CAPÍTULO II

CRIPTOGRAFIA CLÁSSICA E DE CHAVE PÚBLICA

2.1 - CRIPTOGRAFIA CLÁSSICA

2.1.1 - INTRODUÇÃO

A criptografia é uma arte/ciência que consiste de dois mundos. Existe o mundo das comunicações legais, tais como a troca de mensagens entre usuários legais de um banco de dados; e o mundo dos inimigos, que tentam ilegalmente interceptar mensagens e fazer qualquer tipo de invasão. Para as pessoas do mundo legal é desejável que os inimigos saibam muito pouco sobre as mensagens enviadas e recebidas. Por outro lado, os inimigos gostariam de ter livre acesso a essas mensagens.

Criptografia é a arte de “proteger e invadir” esses dois mundos. Por exemplo, um sucesso do inimigo leva o usuário legal a fortalecer seus métodos de defesa. Isso significa um novo desafio para o inimigo. E assim por diante.

Para expor esses dois mundos será explanado detalhadamente o mundo legal, alguns possíveis ataques inimigos, e os motivos pelos quais esses geralmente não são bem sucedidos.

Para tanto, será abordada a criptografia clássica, sua nomenclatura e principais características, enfocando o DES (Data Encryption Standard) [Sal,1990] por ser um dos criptosistemas clássicos mais utilizados e seguros no tocante aos ataques inimigos; da mesma forma e pelo mesmo motivo, ao elucidar a criptografia de chave pública será enfocada o RSA e a seguir, uma breve conclusão referente a este capítulo.

2.1.2 - NOTAÇÕES

Serão abordadas, a seguir, as principais notações utilizadas em criptografia, apesar da mesma ser variável em diferentes textos sobre o assunto.

O termo global para escrita secreta é criptografia.

O conjunto básico está descrito na Figura 2.1. Uma mensagem é inicialmente enviada através de um canal inseguro, onde ela pode ser interceptada por um espião. Não há como garantir a segurança do canal e portanto a interceptação é possível. O principal objetivo dos espiões inimigos é violar o segredo de comunicação e se beneficiar da informação secreta. Objetivos mais sofisticados podem também ser alcançados, como por exemplo, o inimigo pode alterar a mensagem, confundindo o receptor. Dessa forma, o inimigo confunde também o receptor sobre a identidade do remetente.

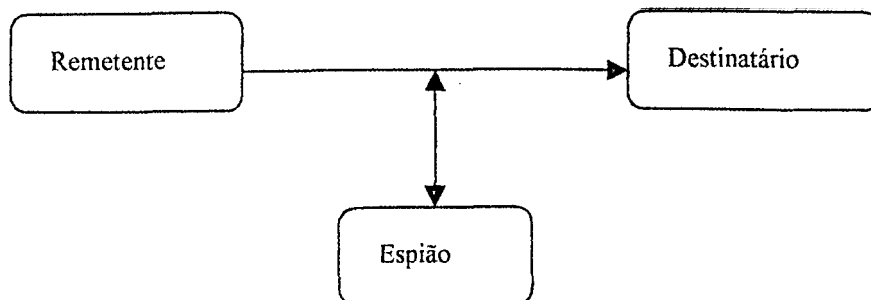


Figura 2.1 – Elementos básicos para transmissão de dados

A mensagem em sua forma original será chamada de texto principal. Então, o remetente cifra o texto principal. O resultado será referido como criptotexto, e este é então enviado via um canal inseguro e, finalmente, decifrado pelo receptor, obtendo assim o texto principal original.

Então, a função do remetente é cifrar o texto principal obtendo o criptotexto.

E a função do destinatário é o inverso: decifrar o criptotexto obtendo assim o texto principal.

Expressões simbólicas também podem ser usadas: $E(pt) = ct$ e $D(ct) = pt$. A Figura 2.2 ilustra essas notações:

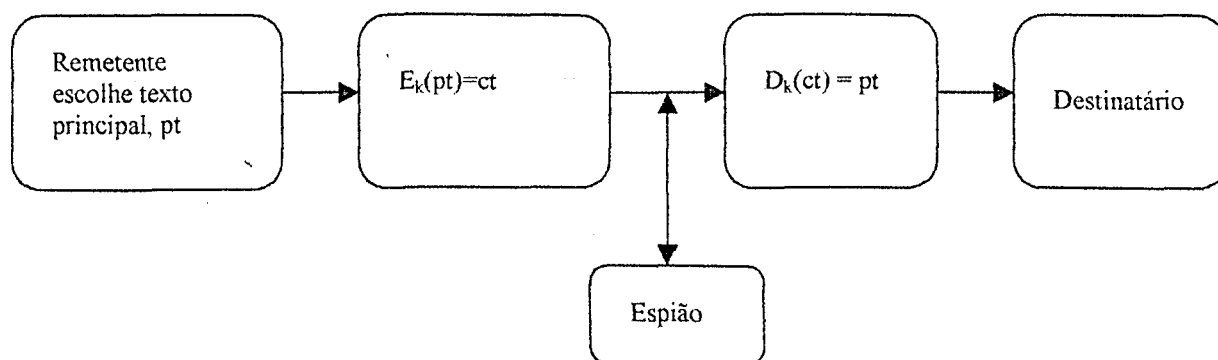


Figura 2.2 – Notações utilizadas em um sistema de transmissão de dados

Agora, será analisada a cifragem e posteriormente a decifragem. Estas translações acontecem dentro de uma estrutura chamada de criptosistema. Um criptosistema consiste nos seguintes itens:

(i) Um espaço para texto principal \underline{PT} , isto é, a coleção de todos os possíveis textos principais pt .

(ii) Um espaço para chave \underline{K} . Cada chave k em \underline{K} , determina um método de cifragem E_k e um método de decifragem D_k . Se E_k é aplicado em um texto principal pt , e D_k é aplicado logo após, então pt é obtido.

(iii) Um espaço para criptotexto \underline{CT} , isto é, a coleção de todos os possíveis criptotextos ct . Elementos de \underline{CT} resultam nos elementos de \underline{PT} aplicando o método de cifragem E_k , onde k pertence a \underline{K} .

Algumas noções básicas da linguagem, serão vistas para se entender a explanação teórica. Um conjunto finito não vazio Σ é denominado alfabeto. Os elementos de Σ são as letras. *Strings* finitas dos elementos de Σ são as palavras. A mesma letra pode ocorrer várias vezes em uma palavra. As *strings* que não possuem letras também são consideradas como uma palavra, a palavra vazia λ . O tamanho de uma palavra w é o número de letras em w , onde cada letra é contada independente do número de vezes que ela ocorre. O conjunto de todas as palavras sobre Σ é denotado por Σ^* . Subconjuntos de Σ^* são chamados de linguagens sobre Σ .

Retornando à noção de criptosistema, analisando os diferentes itens que foram dados, tem-se os seguintes conceitos: O texto principal \underline{PT} pode ser o conjunto Σ^* , ou então consistir de todas as expressões que possuem significado em uma linguagem natural. Estas duas possibilidades são essencialmente diferentes de muitos pontos de vista. Se o espaço para o texto principal é Σ^* , então toda letra na mensagem tem um significado: não existe "liberdade" no processo de decifragem. Por outro lado, toda linguagem natural é altamente redundante no sentido de que uma mensagem é corretamente entendida até mesmo se muitos caracteres individuais forem distorcidos. Isso é uma vantagem definitiva para o espião: Ele pode entender a mensagem corretamente, embora sua análise esteja errada em alguns pontos.

Na maioria dos casos, o espaço para chave é infinito, pois o inimigo não pode ter a chance de testar todas as chaves.

Cada chave k determina um função de cifragem E_k e um função de decifragem D_k e, além disso, E_k é o inverso de D_k .

O espaço para criptotexto é determinado pelos dois primeiros itens do criptosistema: todas as possíveis cifragens de todos os possíveis textos principais.

Para um criptosistema ser bom, Sir Francis Bacon [Sal,1990] propôs as seguintes condições:

(i) Dado E_k e pt , o cálculo de $E_k(pt)$ é fácil. Dado D_k e ct , o cálculo de $D_k(ct)$ é fácil.

(ii) Sem conhecer D_k , é impossível encontrar pt a partir de ct .

(iii) À primeira vista, não deve existir suspeitas no criptotexto.

Pode-se concordar com Francis Bacon, com as seguintes restrições: a condição (iii) não é considerada importante. A condição (i) diz que para os usuários legais o criptosistema não deve ser muito complicado. “Fácil” se refere aqui à complexidade teórica (a referência informal para problemas fáceis significa que os valores do polinômio que mede a complexidade são pequenos, pelo menos dentro dos limites considerados). Em (ii) “impossível” é substituído por “computacionalmente intratável”. Considera-se que tanto os usuários legais quanto os espões possuem conhecimento sobre computação.

Considerando o espião como sendo o criptoanalista, a diferença entre criptoanálise e decifragem é que o criptoanalista tem que descobrir a mensagem sem a chave de decifragem D_k .

O remetente (resp. destinatário) conhece E_k (resp. D_k). Por exemplo, as duas partes podem concordar sobre o assunto em uma prévia reunião. Os detalhes desse acordo

dependem do criptosistema usado. Este procedimento é a principal diferença entre criptosistema clássico e de chave pública.

Observe-se que, para qualquer chave k e texto principal pt :

$$D_k(E_k(pt)) = D_k(ct) = pt. \quad (2.1)$$

Observações gerais sobre criptoanálise:

Regra dourada para projetistas de criptosistemas: nunca subestimar o criptoanalista. A regra dourada deve ser aplicada em todas as atividades do criptoanalista, tais como espionando uma informação, inventando métodos de ataques, etc.. Além disso, supõe-se o seguinte: o criptoanalista conhece o criptosistema usado. Apesar disso, ele não conhece a chave. Entretanto, se o número de todas as possíveis chaves for pequeno, então todas as chaves podem ser testadas. Isso significa que um criptosistema com um pequeno número de chaves é inútil na prática.

A condição essencial para um criptosistema ser bom é que seja intratável descobrir o texto principal pt a partir do criptotexto ct sem conhecer o método de decifragem D_k .

Serão abordadas detalhadamente possíveis questões para o criptoanalista, supondo que ele conhece o criptosistema usado [Sal,1990].

(i) Apenas o criptotexto. Aqui a criptoanálise tem que ser baseada apenas em uma amostra do criptotexto. Para o criptoanalista é sempre melhor que a amostra seja grande. Eficientes métodos criptoanalíticos podem ser baseados em informações estatísticas relacionadas com a linguagem do texto principal, por exemplo, informação sobre a frequência de letras individuais em português, inglês, etc (considerando o idioma do texto principal).

(ii) Texto principal conhecido. Neste caso, o criptoanalista conhece alguns pares $(pt, E_k(pt))$. O conhecimento de tais pares, pode ajudar na análise do criptotexto, conhecendo ct.

(iii) Texto principal escolhido. O criptoanalista também conhece aqui alguns pares $(pt, E_k(pt))$. Entretanto, pt foi escolhido por ele. Em situações onde o criptoanalista tem que definir conjecturas sobre a chave, é claro que este caso é bem melhor que a questão (ii). Por outro lado, (iii) é provavelmente mais realista nos casos onde o criptoanalista tem a possibilidade de se passar por um usuário autorizado do sistema em questão.

(iv) Chave de cifragem. O criptoanalista conhece o método de cifragem E_k e tenta encontrar o correspondente método de decifragem D_k , antes efetivamente de receber qualquer amostra do criptotexto. Esta questão é típica de criptosistemas de chave pública. O método de cifragem E_k pode ter sido publicado muito antes, e pode levar meses para que E_k seja usado para cifrar importantes mensagens. Nesse caso, o criptoanalista tem muito tempo para o pré-processamento, ou seja, o criptoanalista tem bastante tempo para descobrir D_k , antes de se enviar uma mensagem.

2.1.3 – SISTEMA CLÁSSICO – DES

Dentre os principais criptosistemas clássicos, será focado o DES – Data Encryption Standard, pois este é o mais utilizado de todos os tempos [Sal,1990]. A história deste sistema começou em 1973, quando o National Bureau of Standards (NBS) americano solicitou propostas para algoritmos criptográficos. Em 1974, a IBM propôs em essência o DES, que foi desenvolvido conjuntamente pelo laboratório de pesquisa de Yorktown e o laboratório de desenvolvimento de Kingston. O projeto da IBM originou-se de outro

anterior, chamado LUCIFER e projetado por Feistel H. [Luc,1986]. Em 1977, o NBS adotou o DES como padrão de ciframento de dados para aplicações não ligadas à segurança nacional. O DES foi também adotado pelo ANSI (American National Standards Institute).

DES especifica um algoritmo, para ser implementado em projetos de hardwares, para cifrar e decifrar dados.

Definição:

Um sistema criptográfico DES padrão é um ciframento composto que cifra blocos de 64 bits em blocos de 64 bits, mediante uma chave de 64 bits (dos quais 8 são de paridade) [Luc,1986].

Assim, para cada chave, o DES é uma substituição monoalfabética¹ sobre o alfabeto de $2^{64} = 1.8 \times 10^{19}$ letras. Observe que as técnicas publicadas de quebra de substituições monoalfabéticas se aplicam apenas a alfabetos pequenos.

O DES é uma função resultante da composição de outras, que é descrita por:

$$DES = IP^{-1} \cdot T_{K_{16}} \cdot N.T_{K_{15}} \cdot \dots \cdot N.T_{K_2} \cdot N.T_{K_1} \cdot IP, \quad (2.2)$$

onde:

- IP é uma substituição fixa, chamada permutação inicial, de 64 bits em 64 bits.

¹ criptosistemas monoalfabéticos – são os sistemas em que o resultado das substituições permanecem inalteradas em todo o texto e Criptosistemas polialfabéticos são os sistemas em que o resultado das substituições variam em diferentes partes do texto principal.

- T é uma transformação, que depende de uma chave de 48 bits, e que preserva a metade direita.
- N é uma troca das duas metades de 32 bits cada uma.
- K_1, K_2, \dots, K_{16} são chaves de 48 bits, que são funções da chave original.

A função DES é representada pela figura 2.3 a seguir:

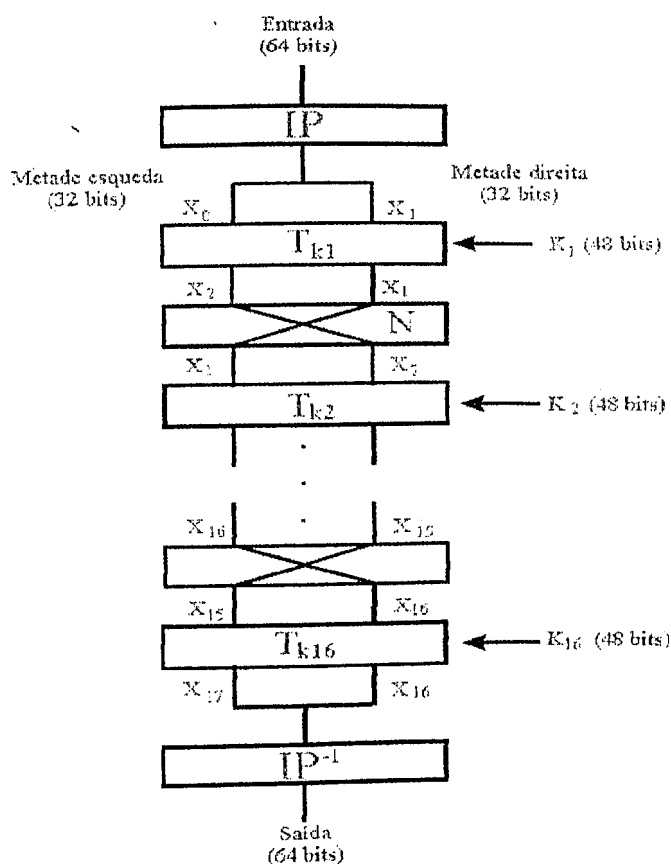


Figura 2.3 - Uma visão macroscópica do DES

A Figura 2.4 ilustra a permutação N . Neste caso, a função N apenas inverte as metades esquerda e direita dos bits que compõem a palavra de 64 bits da entrada.

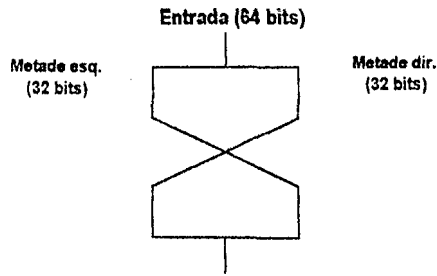
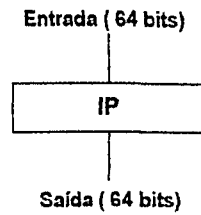


Figura 2.4 - A permutação N

A permutação IP, é descrita na figura 2.5.



57	49	41	33	25	17	9	1	
59	51	43	35	27	19	11	3	
61	53	45	37	29	21	13	5	
63	55	47	39	31	23	15	7	
56	48	40	32	24	16	8	0	
58	50	42	34	26	18	10	2	
60	52	44	36	28	20	12	4	
62	54	46	38	30	22	14	6	

Figura 2.5 - A permutação IP

Para executar IP, os bits de entrada são numerados de 0 a 63 da esquerda para a direita. O resultado desta execução considera a tabela da figura 2.5. Isto significa que o bit

mais à esquerda da saída é o bit número 57 da entrada, o segundo bit mais à esquerda da saída é o bit 49 da entrada e assim por diante.

Verifica-se que N é uma involução, ou seja, N^2 é a identidade. Vê-se a seguir que T_{K_i} , para $i=1, \dots, 16$ são evoluções. Como resultado destes fatos, a partir de (2.2), tem-se:

$$DES^{-1} = IP^{-1} \cdot T_{K_1} \cdot N \cdot T_{K_2} \cdot \dots \cdot N \cdot T_{K_{15}} \cdot N \cdot T_{K_{16}} \cdot IP. \quad (2.3)$$

Isto é, cifrar e decifrar em DES é essencialmente a mesma operação, invertendo-se apenas a ordem das chaves usadas em cada um dos 16 estágios.

Uma transformação T_{K_i} (Figura 2.6) é definida a partir de uma transformação G , que é aplicada à metade direita da entrada de 64 bits. Esta transformação G depende de uma chave de 48 bits. Na definição de T_{K_i} utiliza-se também um somador (“ou-exclusivo”) que soma à metade esquerda à saída de G . Observe que, T não altera a metade da direita da palavra de entrada. Independentemente do que G seja, T_{K_i} é uma involução. De fato, se denotado a metade esquerda da palavra de entrada por x , a metade direita por y e a entrada por (x, y) , tem-se:

$$T(x, y) = (x \oplus G(y), y) \quad (2.4)$$

$$T^2(x, y) = T(x \oplus G(y), y) = (x \oplus G(y) \oplus G(y), y) = (x, y) \quad (2.5)$$

Onde se utiliza a propriedade do ou exclusivo $G(y) \oplus G(y) = 0$.

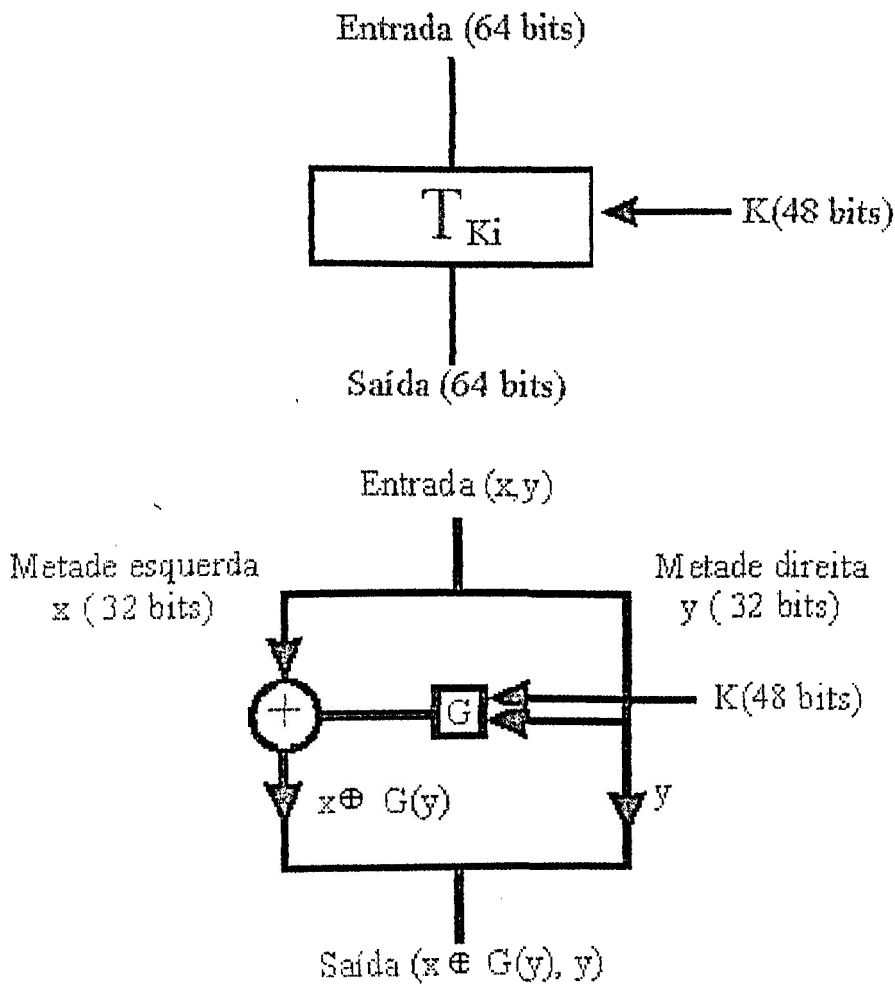


Figura 2.6- A transformação T

A transformação G consiste de:

- uma expansão E de 32 para 48 bits;
- uma soma (“ou-exclusivo”) com a chave K de 48 bits;
- uma contração de 48 para 32 bits obtida a partir de 8 contrações de 6 para 4 bits;
- uma permutação P de 32 bits.

Tais operações são descritas na figura 2.7.

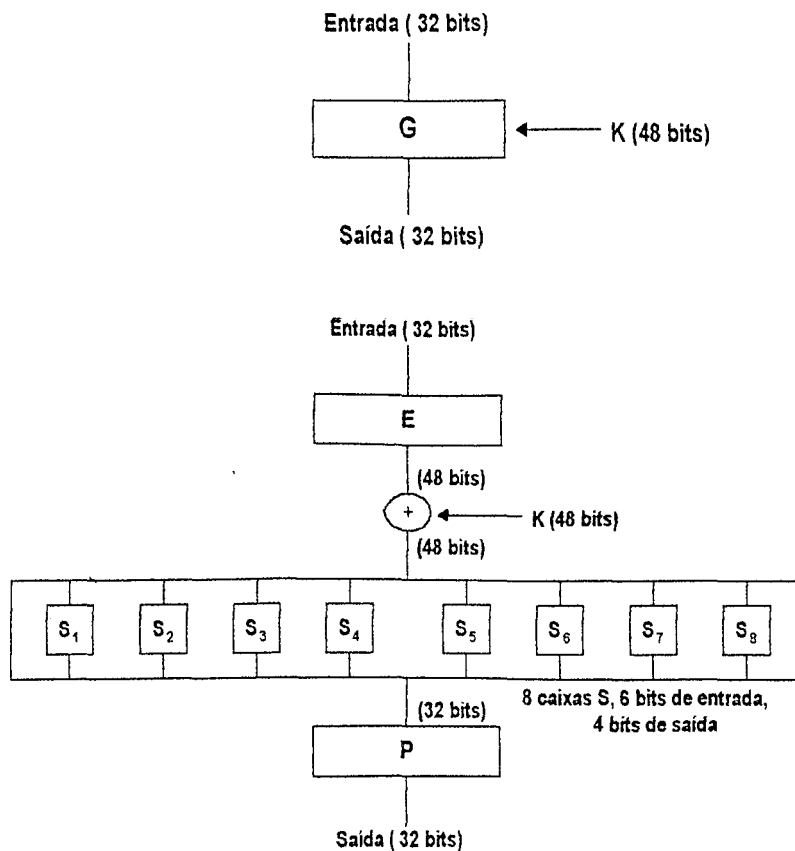


Figura 2.7 - A transformação G

Visando uma melhor compreensão da transformação G, serão expostas três figuras ilustrativas:

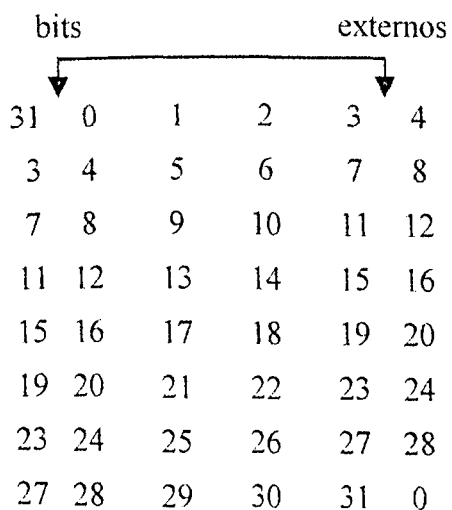


Figura 2.8 A expansão E

Na Expansão E os 32 bits de entrada são numerados de 0 a 31, da esquerda para a direita. A saída (48 bits) é obtida utilizando a tabela da figura 2.8. Ela consiste de uma seqüência de bits, onde o primeiro é o bit 31 da entrada, o segundo é o bit 0 da entrada. Nesta ordem, tem-se por exemplo que o bit 3 da entrada é o bit 4 e o bit 6 da saída, o bit 16 da entrada é o bit 23 e o bit 25 da saída, e assim por diante. Considerando a entrada como 8 blocos de 4 bits, no resultado da aplicação de E, os bits externos de cada bloco são duplicados.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S ₁	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S ₂	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	14	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S ₃	10	0	9	14	6	4	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S ₄	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	<u>14</u>	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S ₅	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S ₆	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S ₇	4	11	2	14	15	0	8	13	3	12	9	7	5	14	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S ₈	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figura 2.9 - As contrações S.

Nas contrações S_i , cada caixa S_i recebe 6 bits de entrada e envia 4 de saída.

Cada contração é definida a partir de uma tabela, conforme indicado na figura 2.9.

A partir de uma entrada $b_0 \dots b_5$, utiliza-se inicialmente os bits $b_0 b_5$, que é um número entre 0 e 3, para selecionar a linha da tabela S_i . Em seguida, o número determinado pelos bits $b_1 \dots b_4$, que está entre 0 e 15 é utilizado para determinar a coluna da tabela.

Assim, a contração S_4 , por exemplo, utiliza a tabela S_4 da figura 2.9. Neste caso, uma entrada igual a 110110, fornece a linha 2 e a coluna 11, determinando uma saída igual a 14 = 1110.

15	6	19	20
28	11	27	16
0	14	22	25
4	17	30	9
1	7	23	13
31	26	2	8
18	12	29	5
21	10	3	24

Figura 2.10. A permutação P.

Na permutação P, a entrada $b_0 b_1 \dots b_{31}$, os 32 bits são numerados da esquerda para a direita de 0 a 31. Neste caso, a saída $b_{15} b_6 b_{19} b_{20} b_{28} \dots b_3 b_{24}$ é dada pela tabela da figura 2.10, que utiliza os mesmos princípios descritos na definição de IP.

Para completar a descrição do DES, resta agora descrever as chaves K_1, \dots, K_{16} usadas nos estágios T_i , $i = 1, \dots, 16$. Cada uma dessas 16 chaves possuem 48 bits. A chave inicial do DES, $K = k_0 k_1 \dots k_{63}$ tem 64 bits, dos quais 8 são de paridade. Os bits de paridade são os bits de ordem congrua a 7 módulo 8, isto é, os bits 7, 15, 23, 31, 39, 47, 55, e 63. Os

demais bits são carregados, 28 bits cada vez, em dois registradores, C_0 e D_0 , de acordo com a permutação PC-1 definida pela tabela dada na Figura 2.11.

$$\begin{array}{r}
 C_0 = 56 \ 48 \ 40 \ 32 \ 24 \ 16 \ 8 \\
 \quad 0 \ 57 \ 49 \ 41 \ 33 \ 25 \ 17 \\
 \quad 9 \ 1 \ 58 \ 50 \ 42 \ 34 \ 26 \\
 \quad 18 \ 10 \ 2 \ 59 \ 51 \ 43 \ 35 \\
 D_0 = 62 \ 54 \ 46 \ 38 \ 30 \ 22 \ 14 \\
 \quad 6 \ 61 \ 53 \ 45 \ 37 \ 29 \ 21 \\
 \quad 13 \ 5 \ 60 \ 52 \ 44 \ 36 \ 28 \\
 \quad 20 \ 12 \ 4 \ 27 \ 19 \ 11 \ 3
 \end{array}$$

Figura 2.11 - A permutação PC-1

A partir da figura 2.11 observa-se, por exemplo, que o bit 56 da chave K é o primeiro bit de C_0 e assim por diante. O bit 62 de K é o primeiro bit de D_0 e assim por diante.

Em seguida, os blocos C_0 e D_0 sofrem deslocamentos circulares à esquerda (“shift left-circular”) de r_i posições, onde r_i é dado na Figura 2.12, Obtendo-se C_i e D_i , $i = 1, \dots, 16$.

$$\begin{array}{r}
 i \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \\
 r_i \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 1
 \end{array}$$

Figura 2.12 - O vetor r indica o número de posições dos deslocamentos circulares de C e D

A partir de C_i e D_i , obtém-se a concatenação $C_i D_i$.

Em seguida, A função PC-2 seleciona os 48 bits da chave K_i , $i = 1, \dots, 16$, a partir de $C_i D_i$ conforme a seleção determinada pela tabela da figura 2.13. Esta tabela determina que, por exemplo, o primeiro bit de K_i é o bit 13 de $C_i D_i$ e assim por diante.

13	16	10	23	0	4
2	27	14	5	20	9
22	18	11	3	25	7
15	6	26	19	12	1
40	51	30	36	46	54
29	39	50	44	32	47
43	48	38	55	33	52
45	41	49	35	28	31

Figura 2.13 - Escolha PC-2. Entrada 56 bits; saída 48 bits

Observando-se a Figura 2.12, verifica-se que, ao final do processo, $C_{16}=C_0$ e $D_{16}=D_0$. Portanto de (2.2), (2.3), e da Figura 2.12, conclui-se que o processo para decifrar é igual ao esquema para cifrar, exceto, que:

- o primeiro deslocamento não é executado ($r_1 = 0$)
- os deslocamentos são à direita e de valor r_i iguais aos da Figura 2.12 ($2 \leq i \leq 16$).

Após esta descrição do DES, considera-se a seguir uma análise sobre critérios do projeto.

O projeto do DES é baseado na interação ou composição de duas transformações:

- substituições não lineares, viáveis para blocos de tamanho pequeno
- permutações de bits, transformações lineares viáveis para qualquer tamanho de bloco.

As substituições não lineares são implementadas através das caixas S_i . Sua função é introduzir o que Shannon chamou de confusão [Sha,1949], fazendo com que nenhum bit de saída seja uma função linear dos bits da entrada e da chave.

As permutações de bits no DES são implementadas pelas transformações IP, N, PC-2 e pelas rotações da chave. Sua função é introduzir o que Shannon [Sha,1949] chamou de difusão, ou seja, o espalhamento da dependência de cada bit da saída num número maior de bits da entrada e da chave.

2.1.4 - CRIPTOANÁLISE DO DES

A questão fundamental aqui é: quão seguro é o DES? Tanto a IBM quanto a NSA (National Security Agency - a agência de segurança nacional americana) validaram o DES. Um dos métodos de validação usados pela IBM, consistiu na aplicação de testes X^2 para a determinação da não existência de correlação entre conjuntos pequenos de bits de chave, entrada e saída [Kon,1981].

No entanto, a maioria dos métodos de validação usados pela IBM e todos os métodos usados pela NSA foram classificados como secretos pela NSA.

Parece pouco provável que se consiga quebrar o DES por vias estatísticas, conforme indicam os testes X^2 efetuados. Analiticamente, o ataque também é inviável, uma vez que a não linearidade é alta. Resta a força bruta. Como a chave possui 56 bits, 2^{56} diferentes chaves existem e portanto são necessários $2^{56} = 10^{17}$ ciframentos para determinar a chave, ou seja, cerca de $5 \cdot 10^6$ chips dias, a $1\mu s$ por ciframento. Diffie e Hellman [Dif,1976] conjecturam a construção de uma máquina especial a 10^{12} ciframentos por segundo, que exauriria as possibilidades em um dia; a máquina conteria 10^6 chips, cada

chip examinando uma parte diferente do espaço de chaves (esta máquina custaria cerca de 20 milhões de dólares).

Em [Den,1982], DENNING apresenta a seguinte abordagem, cifrando o texto principal três vezes usando duas chaves k_1 e k_2 :

$$CT = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(Pt))).$$

O texto principal é restabelecido pelo processo reverso e aplicando a inversa das transformações:

$$DES_{k_1}^{-1}(DES_{k_2}(DES_{k_1}^{-1}(CT))) = Pt.$$

MERKLE e HELLMAN [Mer,1981] julga esta abordagem menos segura do que usando uma simples chave de 112 bits, mostrando que este ciframento pode ser quebrado com 2^{56} operações e 2^{56} chaves armazenadas em uma fita magnética. Portanto, a simples composição de dois sistemas DES não altera a complexidade final da criptoanálise.

O DES possui, também, uma característica desejável do ponto de vista de segredo: uma pequena mudança no texto principal ou na chave dá origem a uma grande mudança no texto cifrado [Sal,1990].

2.2 - CRIPTOGRAFIA DE CHAVE PÚBLICA

2.2.1 - INTRODUÇÃO

São sistemas no qual pode-se seguramente publicar seu método de cifragem. Isso significa que até o criptoanalista o conhecerá. Entretanto, ele será incapaz de decifrar seu criptotexto, pois o ciframento é feito através de uma função unidirecional. Isto é a base

da criptografia de chave pública, ou seja, um método de cifragem que pode tornar-se público, tal idéia foi apresentada por Diffie e Hellman [Sal,1990].

Para conseguir a unidirecionalidade desejada com segredo, lança-se mão de um problema computacionalmente intratável, que o destinatário sabe resolver. No caso de RSA (de Rivest, Shamir e Adleman), o problema intratável é a fatoração de inteiros; o destinatário calcula o produto de dois primos e publica-o: para decifrar as mensagens a fatoração desse produto é necessária, e somente o destinatário a conhece. O RSA é o criptosistema de chave pública mais estudado hoje em dia, o qual é baseado na teoria dos números (veja [Kob,1987]).

A teoria de complexidade oferece informações sobre a complexidade de vários métodos computacionais, por exemplo, qual o tempo computacional para execução de vários programas - informação crucial em criptografia [Sal,1990].

Observações: Em matemática, como na vida real, existem algumas ruas *one-way* (unidirecional). Isto é, na rua, para se ir de A para B é fácil. Mas é praticamente impossível ir de B para A. A cifragem é vista como a direção de A para B. Embora você seja capaz de ir nessa direção, isso não o torna capaz de ir na direção oposta; ou seja, a decifragem.

O método de cifragem é não determinístico. Muitos criptotextos originam do mesmo texto principal. Por outro lado, cada criptotexto dá origem a apenas um texto principal.

A seguir, apresenta-se os conceitos de funções unidirecionais (não levando em consideração os rigores das notações matemáticas), e algumas questões da teoria de complexidade - tópicos de fundamental importância em um sistema de chave pública.

Logo após serão elucidadas algumas vantagens dos sistemas de chave pública, enfocando o sistema RSA e sua criptoanálise, seguida de uma breve conclusão deste capítulo.

2.2.2 - FUNÇÕES UNIDIRECIONAIS

A idéia de criptografia de chave pública está relacionada com funções unidirecionais. Dado um valor qualquer x , é fácil calcular o valor da função $f(x)$, mas é intratável calcular x através de $f(x)$. Aqui “intratável” é visto no sentido de complexidade teórica (Um problema é dito ser computacionalmente intratável se não está em P, problemas tratáveis, isto é, problemas em P, possuem subclasses tais como: problemas com complexidade no tempo linear, quadrático, cúbico, etc). A figura 2.14 representa a complexidade de uma função.

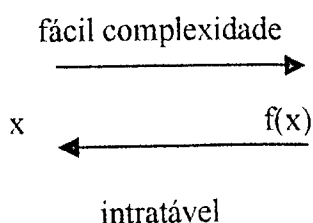


Figura 2.14 - Representação da complexidade de uma função

Popularmente, pode-se dizer que uma função é unidirecional se for computacionalmente viável computá-la e computacionalmente inviável calcular a sua inversa. Por exemplo: Imagine que se tem dois números primos da ordem de 10^{100} ; multiplicá-los é uma questão de segundos com a tecnologia atual; no entanto dado seu produto, da ordem de 10^{200} , o melhor algoritmo conhecido leva hoje 1 bilhão de anos para fatorar o produto dado [Luc,1986]. Assim, a função produto de dois números é unidirecional.

As funções unidirecionais podem ser com segredo (se existe uma informação, o “segredo”, que torna o cálculo da sua inversa viável) e sem segredo (caso contrário). Em um sistema criptográfico de chave pública, a função de ciframento é unidirecional com segredo, pois dada a chave de deciframento é fácil calcular a função de deciframento. A função de dois primos grandes é unidirecional sem segredo.

Há circunstância em que mesmo uma função unidirecional sem segredo é útil: um exemplo típico é na proteção de senhas (“passwords”). O arquivo de senhas as apresentam cifradas por uma função unidirecional sem segredo (e de inviável deciframento); quando um usuário inicia a sessão, fornece a senha, que é então cifrada e comparada com a senha cifrada armazenada: se forem iguais, o sistema aceita o usuário, caso contrário o rejeita. Desta maneira, exige-se apenas integridade do arquivo de senhas.

Referindo-se $f(x)$ como uma função, a figura anterior - 2.14 pode ser entendida em um amplo sentido que inclui também métodos de cifragem não determinísticos. Além disso, o cálculo de x a partir de $f(x)$, deve ser intratável apenas para o criptoanalista.

Em um típico criptosistema de chave pública, apenas a criptoanálise honesta está baseada no cálculo de x a partir de $f(x)$. Podem existir outros métodos criptoanalíticos onde este cálculo pode ser evitado. Dessa forma, o criptoanalista terá sucesso até mesmo se o cálculo de x a partir de $f(x)$ for intratável.

Definição: Um problema é dito intratável se não existir algoritmo para o problema operando em tempo polinomial. Se existir algum algoritmo, o problema é dito tratável.

Ao se referir a problemas “fáceis” significa que possuem um algoritmo operando em um tempo polinomial baixo, de preferência em um tempo linear. Problemas NP-completos são considerados intratáveis. Deve ser observado, que a teoria de

complexidade tradicional não é ideal do ponto de vista de criptografia, pois ela se refere ao pior caso de complexidade, por exemplo, “Qual o máximo grau de dificuldade que um problema pode ter?”

Uma função $f(x)$ unidirecional, significa que a transição de x para $f(x)$ é fácil, enquanto que a transição inversa de $f(x)$ para x é intratável.

2.2.3 – TEORIA DE COMPLEXIDADE

Denning [Den,1982] define a teoria de complexidade como sendo fornecedora de fundamentos para:

- Analisar os requisitos computacionais das técnicas criptoanalíticas;
- Estudo inerente a dificuldades de resolver cálculos;
- Estudos inerentes a dificuldades de provar propriedades de segurança sobre sistemas arbitrários;
- Analisar dificuldades computacional de proteção de dados confidenciais através de fórmulas estatísticas.

Em geral, acredita-se que $P \neq NP$. Isso implica que problemas NP-completos são intratáveis. Além disso, poder-se-ia tentar mostrar que a criptoanálise de um criptosistema de chave pública é NP-completo, tendo estabelecido sua intratabilidade. Entretanto, o argumento seguinte mostra que este não é sempre o caso.

A chave de cifragem é pública, combinando este fato à condição proposta para qualquer criptosistema, clássico ou de chave pública: uma vez que a cifragem é fácil, a chave de cifragem e o texto principal são conhecidos. Em qualquer criptosistema de chave

pública o problema da criptoanálise é NP. Dado um criptotexto, o criptoanalista primeiro supõe o texto principal, cifra-o e depois verifica se ele obteve o criptotexto dado. Até mesmo se o método de cifragem é não determinístico, todo o procedimento é ainda NP.

O problema da criptoanálise é também Co-NP (consiste de problemas cujo o “complemento” está em NP): se o método de cifragem for determinístico, então seu procedimento será exatamente como antes: verificar se um suposto texto principal conduz ou não ao criptotexto dado. No caso geral, independentemente do método de cifragem ser determinístico, tem-se a tripla: (w, k, c) onde w é o suposto texto principal, k é a chave de cifragem pública, e c é o criptotexto. Observe que, existe apenas um texto principal, caso contrário, a decifragem seria ambígua.

O algoritmo primeiro supõe o texto principal p , e verifica (em tempo polinomial não determinístico) se p dá origem a c de acordo com k . Apenas no caso de uma resposta positiva o algoritmo continua, comparando p e w letra por letra. Se uma diferença for encontrada, o algoritmo pára.

Observe que o problema da criptoanálise é encontrar o texto principal quando o criptotexto e a chave pública são conhecidos.

Então o problema da criptoanálise para um criptosistema de chave pública está em $NP \cap Co-NP$. Portanto, se o problema da criptoanálise fosse NP-completo, teria $NP = Co-NP$. Isso implica que é altamente improvável que o problema da criptoanálise para um criptosistema de chave pública seja NP-completo, ou ainda, mais alto no nível de complexidade.

2.2.4 – ALGUMAS VANTAGENS DAS CHAVES PÚBLICAS

As vantagens da criptografia de chave pública são imensas, visto que a idéia pode ser realizada sem muitos efeitos prejudiciais. A maior inovação das chaves públicas está relacionada ao gerenciamento de chaves: como lidar e enviar chaves.

A chave de cifragem para qualquer criptosistema clássico é a mesma da decifragem e portanto, não pode se tornar pública. Isso significa que duas partes legais (remetente e destinatário) têm que concordar antes sobre o método de cifragem. Isso pode acontecer tanto num encontro entre as duas partes ou então enviando a chave de cifragem, via algum canal absolutamente seguro.

Se um criptosistema de chave pública for usado, as duas partes não precisam se encontrar - elas não precisam se conhecer ou ter qualquer tipo de comunicação anterior. Esta é uma vantagem enorme, por exemplo, no caso de um grande banco de dados, onde existem vários usuários e alguns, querem se comunicar apenas com um determinado usuário.

Comparando os criptosistemas clássicos e os de chave pública em relação ao tamanho da chave: Se a chave for muito maior que o texto principal no criptosistema clássico, nada será realizado em termos de criptografia. Neste caso, é mais viável transmitir o texto principal, através de um canal seguro. No criptosistema de chave pública, o tamanho da chave de cifragem é irrelevante, pois ela se torna pública de qualquer maneira. Isso significa que o tamanho da chave de decifragem também é irrelevante: o destinatário apenas tem que armazená-la em um lugar seguro.

Agora, aborda-se uma questão importante: autenticação. Como saber se a mensagem enviada por um canal de comunicação ou sistema de informação é autêntica? Como se gera uma assinatura digital? Primeiro, explica-se melhor o que se quer.

Considere duas partes A e B com interesses contrários. Quando A envia uma mensagem para B, esta mensagem será assinada de tal forma que as duas partes obterão os seguintes tipos de proteção:

1 - A e B deverão ser protegidos contra mensagens endereçadas para B, que são distribuídas por uma 3ª pessoa C, que finge ser A.

2 - A (ou B) deve ser protegida contra mensagens falsificadas por B (ou A), que alega tê-las recebido de A (ou B), devidamente assinadas.

As condições (1) e (2) são um tanto contraditórias, e portanto, difíceis de serem simultaneamente satisfeitas. De acordo com (1), B deveria conhecer algo sobre a assinatura de A. De acordo com (2), B não deve saber nada sobre a assinatura de A.

Se um bom criptosistema clássico for usado, (1) pode ser satisfeito de uma forma razoável: A e B concordam sobre a chave de cifragem conhecida apenas por eles. Uma mensagem é assinada utilizando esta chave. A chave e o criptosistema têm que ser trocados constantemente. Tendo C encontrado esta chave, ele se torna capaz de enviar mensagens corretamente assinadas.

O item (2) é aparentemente mais difícil de ser satisfeito, pois neste tipo de criptosistema, B deveria conhecer a forma de como A gera a assinatura, mas sendo para B impossível reproduzi-la. Observe que, ao trabalhar com uma grande rede de comunicações (por exemplo, uma rede de correio eletrônico), se torna impraticável usar um método de assinatura distinto para cada par de usuário.

Agora, se um criptosistema de chave pública for usado, (1) e (2) podem ser satisfeitos, pelo menos no princípio. Como antes, denota-se por E_A, E_B, \dots e D_A, D_B, \dots as chaves de cifragem e decifragem usadas por A, B, ..., respectivamente. Primeiro, A envia uma mensagem w para B na forma $E_B(D_A(w))$. Então, B pode reaver $D_A(w)$ através da sua chave de decifragem D_B . A partir de $D_A(w)$, B pode recuperar w através da chave de cifragem E_A . Observe que, E_A e D_A são inversas.

Dessa forma, (1) e (2) são satisfeitos. Apenas A conhece D_A , e portanto, nem C nem B podem falsificar a assinatura.

Se apenas a assinatura (mas não a cifragem da mensagem) é importante, então é suficiente que A envie para B o par $(w, D_A(w))$. Dessa forma, as condições (1) e (2) também são satisfeitas.

2.2.5 – O RSA

2.2.5.1 – Introdução

O criptosistema de chave pública mais amplamente usado e testado foi criado por Rivest, Shamir e Adleman, sendo denominado de sistema RSA. Ele é baseado em uma simples idéia acerca da teoria dos números, e ainda tem sido capaz de resistir a todos ataques criptoanalíticos. A idéia é um inteligente uso do fato que, enquanto é fácil multiplicar dois números primos, é extremamente difícil fatorar o seu produto.

Então, o produto pode se tornar público e ser usado como chave de cifragem. Dessa forma, os números primos não podem ser descobertos através do produto. Por outro lado, os números primos são necessários para decifragem. Então, tem-se uma excelente estrutura para um criptosistema de chave pública.

Deve-se, enfatizar que conforme [Sal,1990] não existe prova formal para:

1 - a fatoração é intratável ou é intratável em casos especiais necessários para o RSA;

2 - a fatoração é necessária para a criptoanálise do RSA, isto é, não existe método criptoanalítico que evite a fatoração.

Existem inúmeras evidências empíricas de que (1) e (2) são verdadeiras.

2.2.5.2 - Descrição

A seguir, apresenta-se os detalhes do RSA. Seja p e q dois números primos grandes, distintos e aleatórios. Denota-se:

$$n = p \cdot q \quad \text{e} \quad \varphi(n) = (p-1)(q-1) \quad (2.6).$$

Escolha um número grande, aleatório, $d > 1$, tal que $(d, \varphi(n)) = 1$ e calcule o número e , $1 < e < \varphi(n)$, satisfazendo a congruência

$$ed \equiv 1 \pmod{\varphi(n)} \quad (2.7)$$

os números n , e e d são chamados de módulo, expoentes de cifragem e decifragem, respectivamente. Os números n e e constituem a chave de cifragem pública, enquanto que, os itens restantes p , q , $\varphi(n)$ e d constituem a chave de decifragem secreta. Observe que, estas informações não consistem de quatro itens independentes. Por exemplo, o conhecimento de p , imediatamente revela os três itens restantes.

Para cifrar, eleva-se o texto principal à potência e e reduz módulo n .

Para decifrar, eleva-se o criptotexto à potência d e reduz módulo n .

Mais especificamente, assume-se que o texto principal é codificado como um número decimal. O número é dividido em blocos de tamanho conveniente. Os blocos são cifrados separadamente. Um tamanho adequado para blocos é um único inteiro i satisfazendo a inequação $10^{i-1} < n < 10^i$.

Definição:

Seja w um bloco de texto principal, e c o bloco com o criptotexto correspondente. A cifragem é expressa em termos da seguinte equação

$$c = (w^e, \text{mod } n) \quad (2.8)$$

A seguir, mostra-se que a decifragem funciona como desejado. Para maiores referências veja [Sal,1990].

Considere w e c conforme a definição acima, isto é, $c = (w^e, \text{mod } n)$. Tem-se que:

$$w \equiv c^d \pmod{n} \quad (2.9)$$

Além disso, se a decifragem é única, $w = c^d \pmod{n}$.

2.2.5.3 - Projeto

A seguir, discute-se o projeto do criptosistema, isto é, como os diferentes itens são gerados. Em geral, quando se diz que um número aleatório é escolhido, ou que se seleciona algo aleatoriamente, então está usando um gerador de números aleatórios, por exemplo, um programa de computador que gera uma seqüência de dígitos que possui tantas propriedades estatísticas de seqüências aleatórias quanto possível. Não se discute aqui nenhum detalhe relativo a geradores de números aleatórios.

Para seleccionar dois números grandes, aleatórios, p e q , escolhe aleatoriamente um inteiro ímpar r de tamanho adequado e o mesmo é testado para ver se é primo. Em caso de uma resposta negativa, $r+2$ é testado e assim por diante. Conforme a teoria dos números primos, existem aproximadamente, $10^{100}/\ln 10^{100} - 10^{99}/\ln 10^{99}$ números primos com 100 dígitos. (Aqui, \ln refere-se ao logaritmo natural).

Quando este número é comparado com o número $(10^{100}-10^{99})/2$ que é o número de inteiros ímpares com 100 dígitos, visualizamos que a probabilidade de sucesso para um teste individual é aproximadamente 0.00868.

Portanto, usando computadores, não é difícil a determinação de um número primo com 100 dígitos. É claro que, essa determinação depende da eficiência do teste da primalidade.

Dado que p e q são escolhidos, determina-se em seguida, $\varphi(n) = (p-1)(q-1)$. A determinação de d tal que $(d, \varphi(n)) = 1$, considera o algoritmo de Euclides, testando valores para d até que se tenha $(d, \varphi(n)) = 1$. Obtendo-se d tal que $(d, \varphi(n)) = 1$, a cadeia de equações obtidas do algoritmo de Euclides fornece imediatamente e , tal que $ed \equiv 1 \pmod{\varphi(n)}$.

Uma operação necessária para cifrar e decifrar é a exponenciação modular, isto é, calcular $(a^r, \text{mod } n)$. Isto pode ser feito de uma maneira mais rápida do que multiplicar a repetidamente por a . O método referido é o do quadramento. Neste método, uma redução módulo n é feita após cada quadramento. Desta forma, números maiores do que n^2 nunca são encontrados.

Mais especificamente, considera-se a representação binária de r , como sendo:

$$r = \sum_{j=0}^k x_j 2^j, \quad x_j = 0, 1; \quad k = [\log_2 r] + 1 \quad (2.10)$$

Já que se conhece todos os números, tem-se:

$$(a^{2^j}, \text{ mod } n), \quad 0 \leq j \leq k \quad (2.11)$$

$(a^r, \text{ mod } n)$ pode ser calculado, pela formação de no máximo $k-1$ produtos, e reduzindo cada produto módulo n . Então, é suficiente calcular os números (2.11), o qual envolve k quadramentos modulares, e além disso, $k-1$ produtos modulares. Isto significa, calcular no máximo $2k-1$ produtos com ambos os fatores menores que n e reduzindo o produto módulo n . Se r for grande e $\varphi(n)$ é conhecido, então r pode ser primeiro reduzido módulo $\varphi(n)$, desde que $(a, n) = 1$.

Por exemplo, calcular $(7^{83}, \text{ mod } 61)$, nota-se que $7^{60} \equiv 1 \text{ mod } 61$ (61 é primo, $\varphi(61)=60$ e $(7, 61)=1$, logo $7^{\varphi(61)} \equiv 1 \text{ mod } 61$). Consequentemente, pode-se calcular, obtendo-se o mesmo valor: $(7^{23}, \text{ mod } 61)$ – pois $7^{83} \equiv 7^{23} \cdot 7^{60} \text{ mod } 61 \Rightarrow (7^{83}, \text{ mod } 61) = (7^{23}, \text{ mod } 61)$. Através de quadramentos sucessivos, obtêm-se as potências de 7 onde o expoente é uma potência de 2 :

J	0	1	2	3	4	mod 61
7^{2^j}	7	49	22	57	16	

Como $23 = 10111$, obtêm-se o resultado desejado:

$$(7^{23}, \text{ mod } 61) = (16(22(49 \cdot 7))), \text{ mod } 61 = 17$$

Agora, será apresentado um exemplo de ciframento usando o RSA, (considerando valores pequenos para p e q a título de ilustração). Considere $p=5$, $q=11$, $n=55$, $\varphi(n)=40$, $e=7$ e $d=23$. Considere que o texto principal são números no intervalo fechado $[1, 54]$. Além disso, precisa-se excluir números cujo o máximo divisor comum com

55 excede a 1, isto é, números divisíveis por 5 e por 11. Em geral, se $(w,n) > 1$ para alguns textos principais w , então pode-se fatorar n calculando o máximo divisor comum de n e a versão cifrada de w . Claro, neste exemplo pode-se fatorar n por qualquer caminho. Em geral a probabilidade de se ter um texto principal com um fator não trivial em comum com

n é menor que $\frac{1}{p} + \frac{1}{q}$. Assim a probabilidade é desprezível para p e q grandes.

Encontrar o texto cifrado é fácil, conforme mostra a tabela a seguir:

Texto principal	Texto cifrado	Texto principal	Texto cifrado
1	1	28	52
2	18	29	39
3	42	31	26
4	49	32	43
6	41	34	34
7	28	36	31
8	2	37	38
9	4	38	47
12	23	39	19
13	7	41	46
14	9	42	48
16	36	43	32
17	8	46	51
18	17	47	53
19	24	48	27
21	21	49	14
23	12	51	6
24	29	52	13
26	16	53	37
27	3	54	54

Esta tabela pode ser reorganizada para formar uma tabela de decifragem completa.

Texto cifrado	Texto principal	Texto cifrado	Texto principal
1	1	28	7
2	8	29	24
3	27	31	36
4	9	32	43
6	51	34	34
7	13	36	16
8	17	37	53
9	14	38	37
12	23	39	29
13	52	41	6
14	49	42	3
16	26	43	32
17	18	46	41
18	2	47	38
19	39	48	42
21	21	49	4
23	12	51	46
24	19	52	28
26	31	53	47
27	48	54	54

Este exemplo demonstra, o importante fato: criptografia de chave pública não funciona de forma adequada quando os espaços para os textos principais são pequenos. Um criptoanalista pode construir para o estágio de pré-processamento uma tabela de decifragem completa simplesmente pela cifragem de todos os possíveis textos principais e alterando o criptotexto resultante em uma ordem alfabética conveniente.

2.2.5.4 – Criptoanálise e fatoração

Segundo Salomaa [Sal,1990] não existe resultado formal afirmando que a fatoração de n seja necessária na criptoanálise do sistema RSA. É concebível que a criptoanálise possa ser realizada de diferentes maneiras. Entretanto, se algumas dessas diferentes maneiras descobrir alguns dos itens secretos, então elas também conduziriam a uma rápida fatoração de n . Isso será mostrado abaixo.

Lema 2.1:

Para fatorar n , é aplicável qualquer algoritmo que calcule $\varphi(n)$, sem alteração da complexidade (prova em [Sal,1990]).

Supondo ter um método para calcular o expoente de decifragem d , quer-se mostrar que esse método pode ser usado para fatorar n . O problema não é tão simples como no Lema 2.1. Além disso, o algoritmo resultante da fatoração será probabilístico. A probabilidade de falha, pode ser arbitrariamente pequena. A complexidade do novo algoritmo não é essencialmente maior ao daquele para calcular d . A complexidade do novo algoritmo depende da probabilidade fixada mas, para qualquer probabilidade, o novo algoritmo é executado em tempo polinomial, munido apenas do algoritmo para calcular d .

Teorema 2.1:

Qualquer algoritmo para calcular d , pode ser convertido em um algoritmo probabilístico para fatorar n (demonstração em [Sal,1990]).

O RSA é aplicado também em um meio, onde o módulo, bem como os expoentes de cifragem e decifragem, são distribuídos por um agente no qual confiam todas as partes envolvidas. Supondo que o agente publique um módulo n comum a todos, bem como os expoentes de cifragem e_A, e_B, \dots dos usuários A, B, \dots . Em adição, o agente

distribui aos usuários individualmente, os expoentes de decifragem d_A, d_B, \dots . Os primos p e q são conhecidos apenas pelo agente. Esta questão é vulnerável no sentido do seguinte teorema. O método é similar ao usado no teorema 2.1. O resultado pode ser visto como um exemplo de criptoanálise sem fatorar n .

Teorema 2.2:

Na questão descrita acima, qualquer usuário é capaz de encontrar em um tempo quadrático determinístico, o expoente de decifragem secreto do outro usuário (sem fatorar n) (demonstração em [Sal,1990]).

Embora no Teorema 2.2 o usuário B construa d_A sem fatorar n , o Teorema 2.1 pode ser usado para fatorar n .

Conforme Lucchesi [Luc,1986], evidentemente, um espião que dispuser de um algoritmo de fatoração eficiente poderá utilizá-lo para fatorar n , que é público, obtendo os primos p e q . De posse destes, o espião calculará $\varphi(n)$ e, conhecido e , que é público, determinará d . Assim, quebrar o RSA não é mais difícil do que fatorar inteiros.

Por outro lado, não se conhece nenhum algoritmo eficiente para fatorar. Na faixa de valores de n , da ordem de 10^{200} , o algoritmo mais eficiente conhecido (mas não publicado), de R. Schroepel [Luc,1986], leva cerca de 10^9 anos, a $1 \mu\text{s}$ por operação aritmética. A figura 2.15 apresenta uma tabela de tempos para várias faixas de valores de n no RSA.

Pode-se argumentar que o sistema será quebrado com o conhecimento de $\varphi(n)$, sem necessidade de conhecer p e q : de fato, dados e e $\varphi(n)$ o expoente secreto d pode ser calculado. No entanto, o conhecimento de $\varphi(n)$ (e de n) permite fatorar n rapidamente:

$$n - \varphi(n) + 1 = n - (p-1)(q-1) + 1 = p + q$$

e, portanto, p e q são as raízes da equação:

$$x^2 - (n - \varphi(n) + 1)x + n = 0.$$

Nº de algarismos decimais de $n=pq$	Tempo (1µs/operação aritmética)
50	3,9 horas
75	104 dias
100	74 anos
200	$3,8 \times 10^9$ anos
300	$4,9 \times 10^{15}$ anos
500	$4,2 \times 10^{25}$ anos

Figura 2.15 Tempo de execução do algoritmo da fatoração de Schroepel [RSA]

A teoria exposta permite concluir que fatorar n e determinar $\varphi(n)$ são problemas computacionalmente equivalentes, e que sucesso em qualquer um deles implica na quebra do RSA pela determinação do expoente secreto d .

Um outro enfoque seria tentar determinar d diretamente. De posse de d pode-se calcular $ed-1$, um múltiplo de $\varphi(n)$; de posse deste múltiplo de $\varphi(n)$ pode-se fatorar n eficientemente se forem observadas algumas hipóteses, conforme Lucchesi [Luc,1986].

Assim, parece que com os enfoques até agora adotados não é possível quebrar o RSA. De qualquer forma, deve-se tomar algumas precauções sugeridas pelos autores do RSA [Luc,1986]:

- Os números p e q devem diferir de alguns bits em comprimento (caso contrário p e q ficarão muito próximos de \sqrt{n}).

- Os números $p-1$ e $q-1$ devem conter fatores primos grandes.
- O $\text{mdc}(p-1, q-1)$ deve ser pequeno.

2.3 - CONCLUSÃO

A criptografia lida com dois mundos: o mundo da comunicação legal e o do criptoanalista. A fim de evitar que um se sobressaia ao outro, estabelecem-se regras a serem seguidas.

Um dos pontos chave do DES é que o mesmo evita a linearidade das substituições do algoritmo. Uma propriedade a ser evitada em substituições é a linearidade, pois ciframento lineares são extremamente frágeis e sucumbem rapidamente.

Não existe um procedimento global que seja recomendado para todas as tarefas criptoanalíticas. Entretanto, um criptoanalista deve sempre ser ativo: se um método falhar, outro deve ser tentado.

O RSA, um dos principais criptosistemas de chave pública foi apresentado, e observou-se que criptografia de chave pública não funciona de forma adequada quando os espaços para os textos principais são pequenos. Um criptoanalista pode construir para o estágio de pré-processamento, uma tabela de decifragem completa simplesmente pela cifragem de todos os possíveis textos principais, alterando o criptotexto resultante em uma ordem alfabética conveniente.

Foi enfatizada a idéia de que não existe resultado formal dizendo que a fatoração de n seja necessária para a criptoanálise de sistemas RSA.

É possível realizar a criptoanálise de outras maneiras, como por exemplo, através de uma informação parcial e conforme teorema 2.2, pode-se encontrar o expoente de decifragem secreto do outro usuário sem utilizar a fatoração.

Os criptosistemas de chave pública, são em geral, consideravelmente mais lentos que os clássicos, devido à complexidade dos cálculos com funções unidirecionais, que requerem muitos cálculos.

Os dois capítulos seguintes, tratam da teoria dos códigos lineares, visando um maior embasamento, para propor, nos capítulos V e VI, novos sistemas criptográficos.

CAPÍTULO III

CÓDIGO SEGURO PARA ARMAZENAMENTO E TRANSMISSÃO

DIGITAL

3.1 - INTRODUÇÃO

Teoria dos códigos começou em 1940 com o trabalho de Golay, Hamming e Shannon, apesar de eles terem os originais em um problema de engenharia, desenvolveu-se através de técnicas matemáticas mais sofisticadas, dando origem a famílias de códigos – por exemplo, códigos de Hamming, BCH e Cíclicos, como também códigos mais avançados, tais como códigos Golay, Goppa, Alternant, Kerdock e Preparata [Hol,1992].

Recentemente, há uma demanda crescente por sistemas eficientes de armazenamento e transmissão digital de dados seguros. Esta demanda tem sido acelerada pela emergência de grande escala, alta velocidade de dados em redes para troca, processamento e armazenamento de informações digitais nas esferas privadas, governamentais e militares [Lin,1983].

Para obter a informação original, através da teoria dos códigos ligadas às comunicações, transmissões e armazenamento de dados de forma segura, deve-se eliminar o ruído provocado pelo canal com métodos de decodificação eficientes.

Este capítulo, apresenta os elementos básicos de um sistema típico de transmissão e armazenamento de dados, os conceitos básicos de códigos de blocos e a subclasse, códigos de blocos lineares - somente com símbolos do corpo binário $GF(2)$ ¹, envolvendo suas propriedades básicas, estrutura sistemática, obtenção das matrizes:

¹ $GF(2)$ – Corpos de Galois com dois elementos (0,1), onde as operações de adição e multiplicação são feitas bit a bit (Definições de Corpos de Galois $GF(2)$ e construção de $GF(2^m)$ em [Lin,1983]).

geradora e de verificação de paridade, cálculo da síndrome e detecção de erros, distância mínima, correção de erros aleatórios e o esquema de decodificação síndrome.

3.2 - SISTEMA TÍPICO DE TRANSMISSÃO (OU ARMAZENAMENTO)

A transmissão e armazenamento de informações digitais tem muito em comum. Ambos transferem dados de uma origem para um destinatário (ou usuário). Um sistema típico de transmissão (ou armazenamento) pode ser representado pelo diagrama de bloco da figura 3.1 abaixo [Lin,1983], onde se tem os seguintes conceitos:

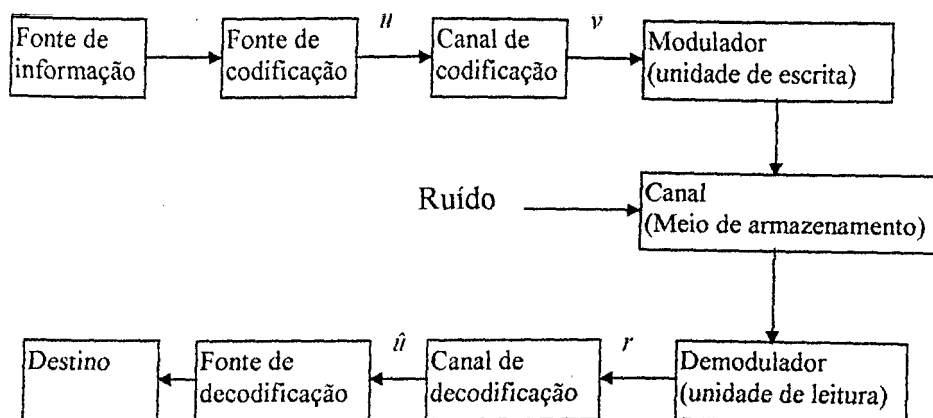


Figura 3.1 Sistema típico de transmissão (ou armazenamento).

O diagrama de bloco mostrado na figura 3.1 representa um sistema unidirecional. A transmissão ocorre apenas em uma direção, da origem para o destino. Controles de erros para sistema unidirecional devem ser realizados usando correção de erros de forma radical, isto é, empregando código de correção de erros que corrige automaticamente os erros detectados no destino. Na seção 3.4, descreve-se os códigos de blocos lineares e a maneira de codificar e decodificar corrigindo erros na decodificação automaticamente.

- 1- Fonte de informação: Pode ser uma pessoa ou uma máquina. (Ex.: Um computador digital). A saída da origem, que será comunicado ao destinatário, pode ser uma forma de onda contínua ou uma seqüência de símbolos discretos.
- 2- Fonte de codificação: Transforma o que sai da origem numa seqüência de dígitos binários (bits) chamada de seqüência de informação u . No caso de origem contínua, esta envolve conversão analógico para digital (A/D). A fonte de codificação é idealmente projetado de maneira que:
 - a) O número de bits por unidade de tempo requerido para representar o que sai da origem seja minimizado;
 - b) O que sai da origem pode ser reconstituído da seqüência de informação u sem ambigüidade.
- 3- Canal de codificação: Transforma a seqüência de informação u em uma seqüência discreta codificada v chamada palavra código. Na maioria das instância, v é também uma seqüência binária, embora em algumas aplicações, códigos não binários têm sido usado.
- 4- Modulador (unidade de escrita): Transforma cada símbolo que sai do canal de codificação em uma forma de onda de duração T segundos que é adequada para transmissão.
- 5- Canal (meio armazenamento): Esta forma de onda entra no canal e é corrompido pelo ruído. Ex.: Linha telefônica
- 6- Demodulador (unidade de leitura): Processa cada forma de onda recebida de duração T segundos e produz uma saída que pode ser discreta (quantizada) ou contínua (não quantizada). A seqüência que sai do demodulador, correspondendo à seqüência codificada v , é chamada de seqüência recebida r .

- 7- Canal de decodificação: Transforma a seqüência recebida r em uma seqüência binária \hat{u} chamada seqüência estimada. A estratégia de decodificação é baseada nas regras do canal de codificação e nas características de ruído do canal (ou médio armazenamento). Idealmente, \hat{u} será uma réplica da seqüência de informação u , embora o ruído possa causar alguns erros de decodificação.
- 8- Fonte de decodificação: Transforma a seqüência estimada \hat{u} em uma estimativa do que sai da origem e libera esta estimativa ao destinatário. Quando a origem é contínua, envolve conversão digital para analógico (D/A).

O propósito é utilizar os conceitos de teoria algébrica dos códigos aplicados a protocolos criptográficos. Neste sentido, o esquema da figura 3.1 é reduzido, conforme a figura 3.2 a seguir.

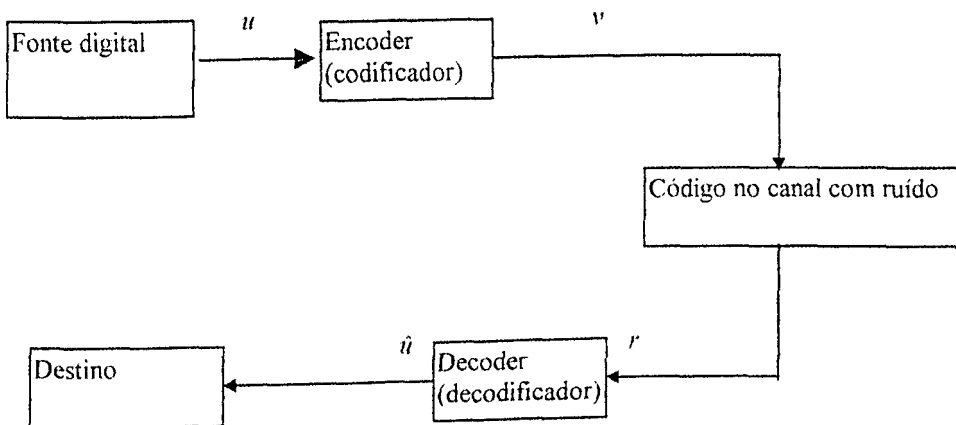


Figura 3.2 Sistema típico de transmissão reduzido

Além disso, tem-se como proposta neste trabalho, que a informação possa ser codificada pelo “encoder”, e transmitida o mais rápido possível (meio com ruído); No “decoder” tem-se a reprodução segura da informação.

Desta forma, como na teoria dos códigos, o enfoque principal do estudo é a análise da codificação e da decodificação.

3.3 TIPOS DE CÓDIGOS

Quanto ao tipo, os códigos podem ser: códigos de blocos e códigos convolucionais. Neste estudo será visto os códigos de blocos. A codificação de um código de bloco divide a seqüência de informação em blocos de mensagem de k bits. Um bloco de mensagem é representado pela k -tupla binária $u = (u_1, u_2, \dots, u_k)$ chamada uma mensagem (o símbolo u é usado para denotar uma mensagem de k -bits, mais precisamente, a seqüência de informação inteira). Existem um total de 2^k possíveis mensagens diferentes. A codificação transforma cada mensagem u independentemente em uma n -tupla $v = (v_1, v_2, \dots, v_n)$ de símbolos discretos, chamado de palavra código (o símbolo v é usado para denotar um bloco de n -símbolos, mais precisamente, a seqüência inteira codificada). Portanto, correspondendo a 2^k possíveis mensagens diferentes, existem 2^k possíveis palavras códigos diferentes para a saída do "encoder". Este conjunto de 2^k palavras códigos de tamanho n é chamado um código de blocos (n, k) . A razão $R = k/n$ é chamada razão do código e pode ser interpretada como o número de bits informação entrando no "encoder" por símbolo transmitido.

Em um código binário, cada palavra código v é também binária. Conseqüentemente, para um código binário ser útil (isto é, ter uma palavra código diferente associada a cada mensagem), $k \leq n$ ou $R \leq 1$. Quando $k < n$, $n-k$ bits de redundância podem ser adicionados a cada mensagem para formar uma palavra código. Estes bits de redundância fornecem ao código a capacidade de combate ao canal com ruído (capacidade de correção de erros provocados pelo canal). Exemplo de um código de bloco binário com $k=4$ e $n=7$ é mostrado na tabela 3.1.

Tabela 3.1 - Código de blocos binário com $k=4$ e $n=7$

Mensagem	Palavras código
0000	0000000
1000	1101000
0100	0110100
1100	1011100
0010	1110010
1010	0011010
0101	1000101
1110	0101110
0001	1010001
1001	0111001
0101	1100101
1101	0001101
0011	0100011
1011	1001011
0111	0010111
1111	1111111

3.4 CÓDIGOS DE BLOCOS LINEARES (CÓDIGOS LINEARES)

3.4.1 - INTRODUÇÃO

Na teoria dos códigos, para um código de bloco ser útil, as 2^k palavras códigos devem ser distintas. Portanto, deve ter uma correspondência um para um entre a mensagem u e sua palavra código v .

Para um código de bloco com 2^k palavras códigos e tamanho n , o instrumento de codificação deve ser proibitivamente complexo para k e n grandes, desde que tenha que armazenar as 2^k palavras códigos de comprimento n em um "dicionário". Portanto, restringi-se atenção a códigos de bloco que podem ser mecanizados de maneira prática. Um estrutura desejável para um código de bloco é a linearidade. Com esta estrutura em

códigos de bloco, a complexidade de codificação será grandemente reduzida, como será visto.

Definição:

Um código de bloco de tamanho n e 2^k palavras códigos é denominado um código linear (n,k) , se e somente se, estas 2^k palavras códigos forma um subespaço de k -dimensão do espaço vetorial de todas as n -tuplas sobre o corpo $GF(2)^2$.

Um código de bloco binário é linear, se e somente se, a soma de duas palavras código em módulo-2 for também uma palavra código. O código de bloco dado na tabela 3.1 é um código de bloco linear $(7,4)$. Por exemplo: a soma das duas palavras código $(1\ 1\ 0\ 1\ 0\ 0\ 0) + (0\ 1\ 0\ 0\ 0\ 1\ 1) = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ é uma palavra código.

Como o código linear $C(n,k)$ é um subespaço de k -dimensão do espaço vetorial V_n de todas as n -tuplas binárias, é possível encontrar k palavras códigos linearmente independente, g_0, g_1, \dots, g_{k-1} , em C tal que toda palavra código em C seja uma combinação linear destas k palavras códigos, isto é:

$$v = u_0g_0 + u_1g_1 + \dots + u_{k-1}g_{k-1}, \quad (3.1)$$

onde $u_i = 0$ ou 1 para $0 \leq i < k$. organizando estas k palavras códigos linearmente independentes como as linhas da matriz $k \times n$, temos:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix}, \quad (3.2)$$

onde $g_i = (g_{i0}, g_{i1}, \dots, g_{i,n-1})$ para $0 \leq i < k$. Se $u = (u_0 \ u_1 \ u_2 \ \cdots \ u_{k-1})$ é a mensagem a ser codificada, a palavra código correspondente pode ser obtida da seguinte maneira:

² Definições de espaço e subespaço vetorial em [Lin,1983].

$$v = u \cdot G = (u_0 \quad u_1 \quad u_2 \quad \dots \quad u_{k-1}) \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} \quad (3.3)$$

$$= u_0 g_0 + u_1 g_1 + \dots + u_{k-1} g_{k-1}.$$

Onde : u é a mensagem a ser codificada;

$G_{k,n}$ é denominada a matriz geradora do código.

Note que qualquer conjunto com k palavras códigos linearmente independentes de um código linear (n,k) pode ser usado para formar a matriz geradora do código. Observa-se que um código linear (n,k) é completamente especificado pelas k linhas da matriz geradora G . Portanto, o “encoder” tem que somente armazenar as k linhas de G e formar uma combinação linear destas k linhas baseado na mensagem de entrada $u = (u_0 \quad u_1 \quad u_2 \quad \dots \quad u_{k-1})$.

Exemplo: O código linear $(7,4)$ dado na tabela 3.1 tem a seguinte matriz geradora:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Se $u = (1 \ 1 \ 0 \ 1)$ é a mensagem a ser codificada, sua palavra código

correspondente será:

$$\begin{aligned} v &= 1 \cdot g_0 + 1 \cdot g_1 + 0 \cdot g_2 + 1 \cdot g_3 \\ &= 1 \cdot (1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0) + 1 \cdot (0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0) + 0 \cdot (1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0) + 1 \cdot (1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1) \\ &= (0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1) \end{aligned}$$

3.4.2 ESTRUTURA SISTEMÁTICA

Uma propriedade desejável para códigos lineares é a estrutura sistemática das palavras código (figura 3.3); onde a palavra código é dividida em duas partes: a parte da mensagem - consiste de k dígitos de informação não alteráveis (a mensagem) e parte de verificação de redundância - consiste de $n-k$ dígitos de verificação de paridade, que são somas lineares dos dígitos informação. Um código de bloco linear com esta estrutura é referenciado como código bloco linear sistemático como se vê na tabela 3.1, onde os quatro dígitos mais à direita de cada palavra código são idênticos aos dígitos informação correspondentes.

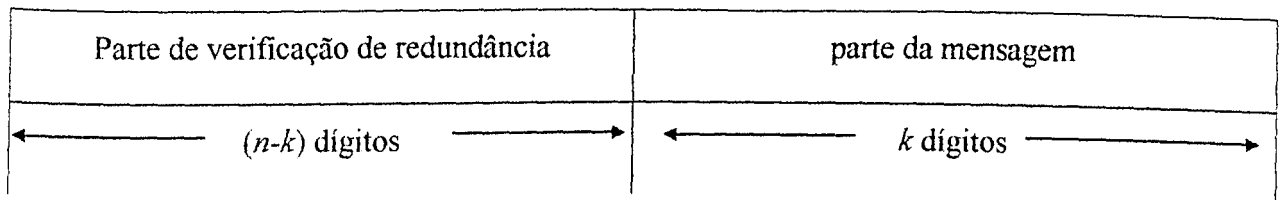


Figura 3.3 - Formato sistemático de uma palavra código

Um código linear (n,k) na forma sistemática é completamente especificado por uma matriz geradora $G_{k,n}$ da seguinte forma:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & \vdots & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & \vdots & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & \vdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & \vdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (3.4)$$

Matriz $P_{k,n-k}$

Matriz identidade k,k

Neste caso, $G=[P \ I_k]$, onde $P_{ij}=0$ ou 1. A submatriz P consiste de $k.(n-k)$ entradas. Como cada entrada P_{ij} pode ser 0 ou 1, então existem $2^{k.(n-k)}$ matrizes G 's distintas.

Considerando $u = (u_0, u_1, \dots, u_{k-1})$ a mensagem a ser codificada. A palavra código correspondente é

$$v = (v_0, v_1, \dots, v_{n-1}) = (u_0, u_1, \dots, u_{k-1}) \cdot G \quad (3.5)$$

De (3.4) e (3.5) os componentes de v são

$$v_{n-k+i} = u_i \quad \text{para } 0 \leq i < k \quad (3.6a)$$

$$e \ v_j = u_0 p_{0j} + u_1 p_{1j} + \dots + u_{k-1} p_{k-1,j} \quad \text{para } 0 \leq j < n-k \quad (3.6b)$$

Equação (3.6a) mostra que os k dígitos mais a direita de cada palavra código v são idênticos aos dígitos de informação u_0, u_1, \dots, u_{k-1} a serem codificados e (3.6b) mostra que os $n-k$ dígitos de verificação de redundância mais à esquerda são soma lineares dos dígitos informação. As $(n-k)$ equações dadas por (3.6b) são chamadas equações de verificação de paridade do código.

3.4.3 MATRIZ DE VERIFICAÇÃO DE PARIDADE

Existe uma outra matriz útil associada a todo código linear. Para qualquer matriz $G_{k \times n}$ com k linhas linearmente independentes, existe uma matriz $H_{(n-k) \times n}$ com $(n-k)$ linhas linearmente independentes, tal que, qualquer vetor no espaço linha de G é ortogonal às linhas de H e qualquer vetor que é ortogonal³ às linhas de H está no espaço linha de G . Conseqüentemente, pode-se descrever o código linear $(n-k)$ gerado por G em um caminho

³ Linhas linearmente independentes são abordadas em espaços vetoriais em [Lin, 1983].

alternativo, como segue: Uma n -tupla v é uma palavra código no código gerado por G , se e somente se, $v \cdot H^T = 0$. Esta matriz $H_{(n-k) \times n}$ é chamada matriz de verificação de paridade do código. Se a matriz geradora do código linear (n, k) está na forma sistemática de (3.4), H assume a seguinte forma:

$$H = [I_{n-k} \cdot P^T] = \begin{bmatrix} 1 & 0 & \dots & 0 & P_{00} & P_{10} & \dots & P_{k-1,0} \\ 0 & 1 & \dots & 0 & P_{01} & P_{11} & \dots & P_{k-1,1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & P_{0,n-k-1} & P_{1,n-k-1} & \dots & P_{k-1,n-k-1} \end{bmatrix} \quad (3.7)$$

Onde P^T é a transposta da matriz P .

Exemplo: Considere a matriz geradora do código linear $(7,4)$ dada no exemplo anterior. A matriz de verificação de paridade correspondente é:

$$H = [I_{7-4} \cdot P^T] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

3.4.4 SÍNDROME E DETECÇÃO DE ERROS

Foi visto na seção anterior que um código linear (n, k) é completamente definido por uma matriz geradora G ou sua matriz de verificação de paridade H . Seja $v = (v_0, v_1, \dots, v_{n-1})$ a palavra código que será transmitida sobre um canal com ruído. Seja $r = (r_0, r_1, \dots, r_{n-1})$ o vetor recebido pela saída do canal. Por causa do canal com ruído, r pode ser diferente de v , tendo

$$r = v + e,$$

onde e é o erro introduzido pelo canal de transmissão. Logo,

$$e = r + v \quad (3.8)$$

$e = (e_0, e_1, \dots, e_{n-1})$ é uma n -tupla onde $e_i = 1$ para $r_i \neq v_i$ e $e_i = 0$ para $r_i = v_i$. Esta n -tupla é chamada vetor erro (ou erro padrão). Os 1's em e são os erros de transmissão causados pelo canal com ruído. De (3.8), o vetor recebido r é o vetor soma da palavra código transmitida e o vetor erro, isto é, $r = v + e$. Em geral, o receptor não conhece v ou e . Sobre o vetor recebido r , o decoder deve primeiro determinar se r contém erros de transmissão. Logo, quando r é recebido, o decoder calcula a seguinte $(n-k)$ -tupla:

$$\begin{aligned} s &= r.H^T, \\ &= (s_0, s_1, \dots, s_{n-k}) \end{aligned} \quad (3.9)$$

que é chamada a síndrome de r . Tem-se que $s=0$, se e somente se, r for uma palavra código e $s \neq 0$, se e somente se, r não for uma palavra código. Portanto, quando $s \neq 0$, sabe-se que r não é uma palavra código e a presença de erros foi detectada. É possível que erros em certos vetores não sejam detectados, isto é, r contém erros mas $s = r.H^T = 0$. Isto acontece quando o erro padrão e é idêntico a uma palavra código diferente de zero. Neste caso r é uma palavra código resultante da soma de duas palavras códigos e conseqüentemente $r.H^T=0$. Erros padrões desse tipo são chamados erros padrões não detectáveis. Como existem 2^k-1 palavras códigos diferentes de zero, então existem $2^k - 1$ erros padrões não detectáveis. Quando um erro padrão não detectável ocorre, o decodificador produz uma decodificação errada. Como será visto adiante, para cifrar uma mensagem não serão usados erros padrões que são palavras códigos.

O cálculo da síndrome s do vetor recebido r realmente depende somente do erro padrão e e não depende da palavra código transmitida v . Como r é o vetor soma de v e e , de (3.9)

$$s = r \cdot H^T = (v + e) \cdot H^T = v \cdot H^T + e \cdot H^T.$$

Mas, $v \cdot H^T = 0$. Conseqüentemente, obtém-se a seguinte relação entre a síndrome e o erro padrão:

$$s = e \cdot H^T \tag{3.10}$$

3.4.5 DISTÂNCIA MÍNIMA

Considera-se a seguir uma análise de erros padrões não detectáveis. Este estudo se fundamenta na distância mínima de códigos lineares, que é definida a partir do peso de Hamming [Wic,1995].

3.4.5.1 PESO DE HAMMING (OU SIMPLEMENTE PESO)

Definição: Seja $v = (v_0, v_1, \dots, v_{n-1})$ uma n -tupla binária. O peso de v , denotado por $w(v)$, é o número de componentes diferentes de zero de v .

Exemplo: O peso de $v = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ é 4, ou seja, $w(v) = 4$.

3.4.5.2 DISTÂNCIA DE HAMMING (OU SIMPLEMENTE DISTÂNCIA)

Definição: Seja v e w duas n -tuplas. A distância entre v e w , denotada $d(v,w)$, é o número de posições onde eles diferem.

Exemplo: A distância entre $v = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ e $w = (0\ 1\ 0\ 0\ 0\ 1\ 1)$ é 3 ($d(v,w)=3$); eles difere nas posições zero, 1 e 3.

Considerando as definições de distância e adição módulo-2, a distância entre duas n -tuplas, v e w , é igual ao peso da soma de v e w , isto é:

$$d(v,w) = w(v+w) \quad (3.11)$$

Exemplo: A distância entre $v = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ e $w = (1\ 1\ 1\ 0\ 0\ 1\ 0)$ é 4 e o peso de $v + w$ é também 4.

Dado um código linear C , pode-se calcular a distância entre qualquer duas palavras códigos diferentes.

3.4.5.3 DISTÂNCIA MÍNIMA

Definição: Dado um código de bloco C . A distância mínima, denotada por d_{\min} , é:

$$d_{\min} = \min \{d(v,w) : v,w \in C, v \neq w\}, \quad (3.12)$$

ou seja, qualquer dois vetores códigos distintos de C difere em pelo menos d_{\min} lugares.

Se C é um código linear, então a soma de dois vetores é também um vetor código. Logo de (3.11), a distância entre dois vetores códigos em C é igual ao peso do terceiro vetor código em C . Portanto, considerando (3.12), tem-se:

$$\begin{aligned} d_{\min} &= \min \{w(v+w) : v,w \in C, v \neq w\}. \\ &= \min \{w(x) : x \in C, x \neq 0\}. \\ &= w_{\min} \end{aligned} \quad (3.13)$$

O parâmetro $w_{\min} = \min \{w(x) : x \in C, x \neq 0\}$ é chamado o peso mínimo do código linear C . Portanto, a distância mínima de um código de bloco linear é igual ao peso mínimo de suas palavras códigos diferentes de zero.

Exemplo: O código $(7,4)$ dado na tabela 3.1 tem peso mínimo 3; assim a distância mínima é 3.

A seguir, será mostrado um teorema que relaciona a estrutura peso de um código linear à sua matriz de verificação de paridade.

Teorema 3.1:

Seja C um código linear (n,k) com matriz de verificação de paridade H . Para cada vetor código de peso l , existem l colunas de H tal que o vetor soma dessas l colunas é igual ao vetor zero. Inversamente, se existem l colunas de H cujo vetor soma é o vetor zero, existe um vetor código de peso l em C [Lin,1983].

Do teorema 3.1, tem-se os dois corolários a seguir.

Corolário 3.1.1:

Seja C um código linear com matriz de verificação de paridade H . Se nenhuma $d-1$ ou menos colunas de H somam zero, o código tem peso mínimo de pelo menos d .

Corolário 3.1.2:

Seja C um código de bloco linear com matriz de verificação de paridade H . O peso mínimo (ou a distância mínima) de C é igual ao menor número de colunas de H que somam zero.

Exemplo: Considere o código linear $(7,4)$ dado na tabela 3.1. A matriz de verificação de paridade desse código é:

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Vê-se que todas as colunas de H são diferentes de zero e nenhum par delas são iguais. Portanto, a soma entre duas colunas não resulta em zero; conseqüentemente, o peso mínimo deste código é no mínimo 3. As colunas zero, 2^a e 6^a, somam zero. Assim, o peso mínimo do código é 3. Como a distância mínima de um código de bloco linear é igual a peso mínimo, então a distância mínima também é 3.

Corolários 3.1.1 e 3.1.2 são geralmente usados para determinar a distância mínima de um código linear.

3.4.6 CAPACIDADE DE DETECÇÃO E CORREÇÃO DE ERROS ALEATÓRIOS

Quando um vetor código v é transmitido em um canal com ruído, resultará um erro padrão de l erros em um vetor recebido r , que difere do vetor transmitido v em l lugares, isto é, $d(v,r) = l$. Se a distância mínima de um código de bloco C é d_{\min} , qualquer dois vetores códigos distintos de C difere em pelo menos d_{\min} lugares. Para este código C , nenhum erro padrão de $(d_{\min}-1)$ ou menos erros pode trocar um vetor código em outro. Portanto, qualquer erro padrão de $d_{\min}-1$ ou menos erros resultará em um vetor recebido r que não é uma palavra código em C . Quando o receiver detecta que o vetor recebido não é uma palavra código de C , diz-se que erros são detectados. Conseqüentemente, um código linear com distância mínima d_{\min} é capaz de detectar todos os erros padrões de $d_{\min}-1$ ou menos erros. Mas, o código não pode detectar todos os erros padrões de d_{\min} erros, porque existe pelo menos um par de vetores códigos que difere em d_{\min} lugares e existe um erro padrão de d_{\min} erros que produz um outro código. O mesmo argumento aplica para erros padrões com mais de d_{\min} erros. Por esta razão, diz-se que a capacidade de detecção de erros aleatórios de um código de bloco com distância mínima d_{\min} é no máximo $d_{\min}-1$.

Embora um código de bloco com distância mínima d_{\min} garanta detectar todos os erros padrões de $d_{\min}-1$ ou menos erros, ele também é capaz de detectar uma grande fração de erros padrões com d_{\min} ou mais erros. De fato, um código linear (n,k) é capaz de detectar $2^n - 2^k$ erros padrões de tamanho n . Isto pode ser mostrado da seguinte maneira. Dentre os $2^n - 1$ erros padrões diferentes de zero possíveis, existem $2^k - 1$ erros padrões que são idênticos às $2^k - 1$ palavras códigos diferentes de zero. Se qualquer um desses $2^k - 1$ erros padrões ocorrem, a palavra código transmitida v é alterada em outra palavra código w . Desse modo, w será aceito pelo decodificador, pois sua síndrome é zero. Portanto, o decodificador faz uma decodificação errada. Logo, existem $2^k - 1$ erros padrões não detectáveis. Se um erro padrão não é idêntico a uma palavra código diferente de zero, o vetor recebido r não será uma palavra código e sua síndrome não será zero. Neste caso, o erro será detectado. Existem exatamente $2^n - 2^k$ erros padrões que não são idênticos às palavras códigos de um código linear (n,k) . Estes $2^n - 2^k$ erros padrões são detectáveis. Para n grande, $2^k - 1$ é, em geral, muito menor que 2^n . Portanto, somente uma pequena fração de erros padrões passam pelo decodificador sem ser detectado.

Uma questão central na teoria dos códigos é determinar quantos erros o código é capaz de corrigir. A distância mínima d_{\min} de um código de bloco C é par ou ímpar. Seja t um inteiro positivo tal que

$$2t + 1 \leq d_{\min} \leq 2t + 2. \quad (3.14)$$

A seguir, mostra-se que o código linear C é capaz de corrigir todos os erros padrões e tais que $w(e) \leq t$. Seja v e r o vetor código transmitido e o vetor recebido, respectivamente. Seja w qualquer outro vetor código em C . A distância entre v , r e w satisfaz a desigualdade triangular:

$$d(v,r) + d(w,r) \geq d(v,w) \quad (3.15)$$

Supondo que um erro padrão de t erros ocorre durante a transmissão de v . Então o vetor recebido r difere de v em t lugares e portanto $d(v,r) = t$. Desde que v e w sejam vetores códigos em C , tem-se:

$$d(v,w) \geq d_{\min} \geq 2t + 1 \quad (3.16)$$

Combinando (3.15) e (3.16) e usando o fato que $d(v,r) = t$, tem-se a seguinte desigualdade:

$$d(w,r) \geq 2t + 1 - t.$$

Se $t \leq t$, então $d(w,r) > t$.

A desigualdade acima diz que se um erro padrão de t ou menos erros ocorrem, o vetor recebido r está mais próximo (em distância) do vetor código transmitido v em relação a qualquer outro vetor código w em C . Baseado em um esquema de decodificação que pega a palavra código que está mais próxima, r é decodificado em v , que é o atual vetor código transmitido, e resulta em uma decodificação correta e assim erros são corrigidos.

Por outro lado, o código não é capaz de corrigir todos os erros padrões de t erros, com $t > t$, por existir pelo menos um caso onde um erro padrão de t erros resulta em um vetor recebido que está mais próximo de um vetor código incorreto de que do atual vetor transmitido. Para mostrar isto, seja v e w dois vetores códigos em C , tais que

$$d(v,w) = d_{\min}.$$

Seja e_1 e e_2 dois erros padrões que satisfaz as seguintes condições:

$$(i) \quad e_1 + e_2 = v + w$$

(ii) e_1 e e_2 não tem componentes diferentes de zero em lugares (posições) comuns. Obviamente, tem-se:

$$w(e_1) + w(e_2) = w(v + w) = d(v, w) = d_{\min} \quad (3.17)$$

Supondo que v seja transmitido e seja corrompido por um erro padrão e_1 . Então o vetor recebido é:

$$r = v + e_1.$$

A distância entre v e r é:

$$d(v, r) = w(v + r) = w(e_1). \quad (3.18)$$

A distância entre w e r é:

$$d(w, r) = w(w + r) = w(w + v + e_1) = w(e_2) \quad (3.19)$$

Supondo que o erro padrão e_1 contém mais do que t erros, isto é, $w(e_1) > t$. Desde que $2t + 1 \leq d_{\min} \leq 2t + 2$, de (3.17) $w(e_2) \leq t + 1$.

Combinando (3.18) e (3.19) e usando o fato que $w(e_1) > t$ e $w(e_2) \leq t + 1$, obtém-se a seguinte desigualdade:

$$d(v, r) \geq d(w, r).$$

Esta desigualdade diz que existe um erro padrão de l ($l > t$) erros que resulta num vetor recebido que está mais próximo de um vetor código incorreto do que do vetor código transmitido. Baseado em um esquema de decodificação que considera a palavra código mais próxima, uma decodificação incorreta será cometida.

Resumindo os resultados acima, um código de bloco com distância mínima d_{\min} garante corrigir todos os erros padrões de $t = \lfloor (d_{\min} - 1) / 2 \rfloor$ ou menos erros, onde $\lfloor (d_{\min} -$

$\lfloor (d_{\min}-1)/2 \rfloor$ representa o maior inteiro não maior de que $(d_{\min}-1)/2$. O parâmetro $t = \lfloor (d_{\min}-1)/2 \rfloor$ é chamado de capacidade de correção de erros aleatórios de um código de bloco. O código é referenciado como um código que corrige t -erros.

Exemplo: O código linear (7,4) dado na tabela 3.1 tem distância mínima 3 e portanto $t=1$. Este código é capaz de corrigir qualquer erro padrão simples sobre um bloco de sete dígitos.

Um código de bloco com capacidade de correção de erros aleatórios t é usualmente capaz de corrigir muitos erros padrões de $t+1$ ou mais erros. Para um código linear (n,k) que corrige t -erros, ele é capaz de corrigir um total de 2^{n-k} erros padrões, incluindo os erros padrões com t ou menos erros (isto será visto na próxima seção).

Do exposto acima, vê-se que a capacidade de detecção e correção de erros aleatórios de um código de bloco são determinados pela distância mínima de suas palavras códigos.

3.4.7 CONJUNTO PADRÃO E DECODIFICAÇÃO SÍNDROME

Nesta seção, é apresentado um esquema para decodificação de códigos de bloco lineares. Seja C um código bloco linear (n,k) . Seja $v_1 \ v_2 \ \dots \ v_{2^k}$ os vetores códigos de C . Não importa qual vetor código é transmitido em um canal com ruído, o vetor recebido r pode ser qualquer das 2^k n -tuplas sobre $GF(2)$. Qualquer esquema de decodificação usado pelo receptor é uma regra para partir os 2^k vetores recebidos em 2^k subconjunto disjuntos $D_1 \ D_2 \ \dots \ D_{2^k}$ tal que o vetor código v_i esteja contido no subconjunto D_i para $1 \leq i \leq 2^k$. Desse modo, cada subconjunto D_i é uma correspondência 1-para-1 para um vetor código v_i . Se o vetor recebido r pertence ao subconjunto D_i , r é

decodificado em v_i . Decodificação correta é feita, se e somente se, o vetor recebido r está no subconjunto D_i que corresponde ao atual vetor código transmitido.

É descrito a seguir um método para partir os 2^n vetores recebidos em 2^k subconjuntos disjuntos tal que cada subconjunto contém, um e somente um, vetor código. A partição se baseia na estrutura linear do código. Primeiro, os 2^k vetores código de C são colocados em uma linha com o vetor código zero $v_1 = (0, 0, \dots, 0)$ como primeiro elemento (o mais da esquerda). Das $2^n - 2^k$ n -tuplas restantes, uma n -tupla e_2 é escolhida e é colocada a baixo do vetor v_1 (primeiro elemento da segunda linha). Agora completamos a segunda linha adicionando e_2 a cada vetor código v_i da primeira linha e colocando a soma $e_2 + v_i$ a baixo de v_i . Tendo completado a segunda linha, uma nova n -tupla e_3 (n -tupla que ainda não foi usada) é escolhida entre as restantes e é colocada de baixo de v_1 na terceira linha (primeiro elemento da terceira linha). Então a terceira linha é formada adicionando e_3 a cada vetor código v_i na primeira linha e colocando $e_3 + v_i$ de baixo de v_i na terceira linha. Continua-se este processo até todas as n -tuplas sejam usadas. Então, tem-se um conjunto de linhas e colunas chamado de conjunto padrão do código linear C – ver figura 3.4.

$$\begin{array}{cccccc}
 v_1 = 0 & v_2 & \cdots & v_i & \cdots & v_{2^k} \\
 e_2 & e_2 + v_2 & \cdots & e_2 + v_i & \cdots & e_2 + v_{2^k} \\
 e_3 & e_3 + v_2 & \cdots & e_3 + v_i & \cdots & e_3 + v_{2^k} \\
 \vdots & \vdots & & \vdots & & \vdots \\
 e_l & e_l + v_2 & \cdots & e_l + v_i & \cdots & e_l + v_{2^k} \\
 \vdots & \vdots & & \vdots & & \vdots \\
 e_{2^{n-k}} & e_{2^{n-k}} + v_2 & \cdots & e_{2^{n-k}} + v_i & \cdots & e_{2^{n-k}} + v_{2^k}
 \end{array}$$

Figura 3.4 – Conjunto padrão para um código linear (n, k)

Da regra de construção de um conjunto padrão, a soma de quaisquer dois vetores na mesma linha é um vetor código em C . Mostra-se, a seguir, algumas importantes propriedades de um conjunto padrão.

Teorema 3.2:

Nenhuma duas n -tuplas na mesma linha de um conjunto padrão são idênticas. Cada n -tupla aparece em uma e somente uma linha [Lin,1983].

Do teorema 3.2, vê-se que existem $2^n/2^k=2^{n-k}$ linhas disjuntas no conjunto padrão e que cada linha consiste de 2^k elementos distintos. As 2^{n-k} linhas são chamadas os cosets do código C e a primeira n -tupla e_j de cada coset é chamada um coset leader (leader = cabeça, guia). Qualquer elemento em um coset pode ser usado como seu coset leader. Estes não trocam os elementos do coset; simplesmente permuta-os.

Um conjunto padrão de um código linear (n,k) C consiste de 2^k colunas disjuntas. Cada coluna consiste de 2^{n-k} n -tuplas, com o elemento do topo sendo um vetor código em C . D_j denota a j -ésima coluna do conjunto padrão. Então

$$D_j = \{v_j, e_2 + v_j, e_3 + v_j, \dots, e_{2^{n-k}} + v_j\}, \quad (3.20)$$

onde v_j é um vetor código de C e $e_2, e_3, \dots, e_{2^{n-k}}$ são os coset leaders. As 2^k colunas disjuntas $D_1, D_2, D_3, \dots, D_{2^k}$ podem ser usadas para decodificar o código C como descrito anteriormente nesta seção. Supondo que o vetor v_j é transmitido em um canal com ruído. De (3.20), vê-se que o vetor recebido r está em D_j se o erro padrão causado pelo canal é um coset leader. Neste caso, o vetor r será decodificado corretamente no vetor código transmitido v_j . Por outro lado, se o erro padrão causado pelo canal não é um coset leader, resultará em uma decodificação errada. Isto pode ser visto da seguinte maneira: o erro padrão x causado pelo canal deve estar em algum coset e debaixo de algum vetor código diferente de zero, diz-se no l -ésimo coset e debaixo do vetor código $v_l \neq 0$. Então $x = e_l + v_l$ e o vetor recebido é

$$r = v_j + x = e_l + (v_i + v_j) = e_l + v_s.$$

O vetor recebido r está, desse modo, em D_s e é decodificado em v_s , que não é o vetor código transmitido. Isto resulta em uma decodificação errada. Portanto, a decodificação é correta, se e somente se, o erro padrão causado pelo canal for um coset leader. Por esta razão os 2^{n-k} coset leaders (incluindo o vetor zero) são chamados de erros padrões corrigíveis. Dos resultados acima, tem-se o seguinte teorema.

Teorema 3.3:

Todo código linear (n,k) é capaz de corrigir 2^{n-k} erros padrões [Lin,1983].

Exemplo: Seja C um código linear binário $(7,3)$. Uma possível matriz geradora G deste

código é:
$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Um conjunto padrão de C é mostrado na tabela 3.2.

Tabela 3.2 – Um conjunto padrão para o código linear $(7,3)$.

0000000	1000111	0101011	0011101	1101100	1011010	0110110	1110001
0000001	1000110	0101010	0011100	1101101	1011011	0110111	1110000
0000010	1000101	0101001	0011111	1101110	1011000	0110100	1110011
00000100	1000011	0101111	0011001	1101000	1011110	0110010	1110101
0001000	1001011	0100011	0010101	1100100	1010010	0111110	1111001
0010000	1010101	0111011	0001101	1111100	1001010	0100110	1100001
0100000	1100101	0001011	0111101	1001100	1111010	0010110	1010001
1000000	0000111	1101011	1011101	0101100	0011010	1110110	0110001
0000011	1000100	0101000	0011110	1101111	1011001	0110101	1110010
0000110	1000001	0101101	0011011	1101010	1011100	0110000	1110111
0001100	1001011	0100111	0010001	1100000	1010110	0111010	1111101
0011000	1011111	0110011	0000101	1110100	1000010	0101110	1101001
0001010	1001101	0100001	0010111	1100110	1010000	0111100	1111011
0010100	1010011	0111111	0001001	1111000	1001110	0100010	1100101
0010010	1010101	0111001	0001111	1111110	1001000	0100100	1100011
0111000	1111111	0010011	0100101	1010100	1100010	0001110	1001001

A construção do conjunto padrão da tabela anterior envolveu um certo número de escolhas arbitrárias. Todos os erros padrões de peso 1 foram usados como primeiro elemento das linhas, mas não ocorreu o mesmo caso com os erros padrões de peso 2. Portanto, o código (7,3) não é capaz de corrigir todos os erros padrões de peso 2, pois é possível criar outro conjunto, utilizando outros erros padrões de peso 2 como primeiro da linha; o mesmo não ocorre com códigos perfeitos, os quais são estruturados de forma que os primeiros elementos das linhas consistem de todos os erros padrões de peso t ou menos e nenhum outro [Wic,1995].

Para diminuir a probabilidade de erro de decodificação, os erros padrões que mais ocorrem para um dado canal devem ser escolhidos como os “coset leaders” (primeiros elementos da linha). Portanto, para formar um conjunto, cada “coset leader” deve ser escolhido entre os vetores de menor peso entre todos os vetores restantes. Escolhendo os “coset leader” dessa maneira, cada “coset leader” tem peso mínimo em seu coset. Como um resultado, a decodificação baseada no conjunto padrão é a decodificação de distância mínima.

Um código linear (n,k) é capaz de detectar $2^n - 2^k$ erros padrões; mas, o código é capaz de corrigir somente 2^{n-k} erros padrões. Para n grande, 2^{n-k} é uma fração pequena de $2^n - 2^k$. Portanto, a probabilidade de um erro de decodificação é muito maior que a probabilidade de detectar o erro.

Teorema 3.4:

Para um código linear $C(n,k)$ com distância mínima d_{\min} , todas as n -tuplas de peso $t = \lceil (d_{\min}-1)/2 \rceil$ ou menos podem ser usadas como “coset leaders” de um conjunto padrão de C . Se todas as n -tuplas de peso t ou menos são usadas como coset leaders, existe

pelo menos uma n -tupla de peso $t + 1$ que não pode ser usada como um coset leader [Lin,1983].

O teorema 3.4 reafirma o fato que um código linear (n,k) com distância mínima d_{\min} é capaz de corrigir todos os erros padrões de $\lfloor (d_{\min}-1)/2 \rfloor$ ou menos erros, mas o código não é capaz de corrigir todos os erros padrões de peso $t + 1$.

Um conjunto padrão tem uma importante propriedade que pode ser usada para simplificar o processo de decodificação.

Teorema 3.5:

Seja H a matriz de verificação de paridade do código linear $C(n,k)$. Todas as 2^k n -tuplas de um coset tem a mesma síndrome. As síndromes para diferentes cosets são diferentes [Lin,1983].

Recorda-se que a síndrome de uma n -tupla é uma $(n-k)$ -tupla e existem 2^{n-k} $(n-k)$ -tuplas distintas. Do teorema 3.5 existe uma correspondência 1-para-1 entre um coset leader (erro padrão corrigível) e uma síndrome. Usando esta relação de correspondência 1-para-1, pode-se formar uma tabela de decodificação, que é muito mais simples de usar do que um conjunto padrão. A tabela consiste de 2^{n-k} "coset leaders" (os erros padrões corrigíveis) e suas síndromes correspondentes. Esta tabela é armazenada no "receiver". A decodificação de um vetor recebido consiste dos seguintes passos:

Passo 1: Calcula a síndrome de r , $r.H^T$;

Passo 2: Localiza o "coset leader" e_l cuja síndrome é igual a $r.H^T$. Então e_l é assumido como o erro padrão causado pelo canal.

Passo 3: Decodifica o vetor recebido r no vetor código $v = r + e_l$.

O esquema de decodificação descrito acima é chamado de decodificação síndrome ou “table-lookup decoding”. Em princípio, a esta tabela pode ser aplicada para qualquer código linear (n,k) . Isto resulta em uma decodificação de espera mínima e de probabilidade mínima de erro. Mas, para $n-k$ grande, a implementação deste esquema de decodificação torna-se impraticável e um armazenamento grande é necessário. Existem esquemas de decodificação práticos, que são variações da table-lookup decoding [Lin,1983]. Cada um desses esquemas de decodificação requer propriedades adicionais àquelas da estrutura linear.

Exemplo: Considere o código linear $(7,4)$ da tabela 3.1, que tem como matriz de verificação de paridade, a seguinte matriz:

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

O código tem $2^3 = 8$ (2^{n-k}) “cosets” e, portanto, existem oito erros padrões corrigíveis (incluindo o vetor zero). Como a distância mínima do código é três, este é capaz de corrigir todos os erros padrões de peso um ou zero. Os erros padrões corrigíveis e suas respectivas síndromes são mostrados na tabela 3.2.

Tabela 3.3-Tabela de decodificação para o código linear $(7,4)$ dado na tabela 3.1.

Síndromes	Coset leaders
(1 0 0)	(1 0 0 0 0 0 0)
(0 1 0)	(0 1 0 0 0 0 0)
(0 0 1)	(0 0 1 0 0 0 0)
(1 1 0)	(0 0 0 1 0 0 0)
(0 1 1)	(0 0 0 0 1 0 0)
(1 1 1)	(0 0 0 0 0 1 0)
(1 0 1)	(0 0 0 0 0 0 1)

Para decodificação de um vetor recebido r , basta seguir os três passos de decodificação descritos anteriormente.

3.4.8 CÓDIGOS DE HAMMING

Códigos de Hamming foram a primeira classe de códigos lineares projetadas para corrigir erros [Lin,1983]. Estes códigos e suas variações são usados para controle de erros em sistemas de comunicação digital e armazenamento de dados.

Para qualquer $m \geq 3$, existe um código de Hamming com os seguintes parâmetros:

Comprimento do código: $n = 2^m - 1$

Número de símbolos de informação: $k = 2^m - m - 1$

Número de símbolos de verificação de paridade: $n - k = m$

Capacidade de correção de erros: $t = 1$ ($d_{\min} = 3$)

A matriz de verificação de paridade H deste código consiste de todas as m -tuplas distintas e diferentes de zero em forma de colunas. Na forma sistemática, as colunas de H são assim organizadas:

$$H = [I_m \quad Q],$$

onde I_m é a matriz identidade $m \times m$ e a submatriz Q consiste de $2^m - m - 1$ colunas que são as m -tuplas de peso 2 ou mais. Por exemplo, seja $m=3$, a matriz de verificação de paridade de um código de Hamming de comprimento 7 pode ser colocada na forma:

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

As colunas de Q podem ser organizadas em qualquer ordem sem afetar a propriedade distância do código. Na forma sistemática, a matriz geradora do código é dada por:

$$G = [Q^T \quad I_{2^m - m - 1}], \text{ onde } Q^T \text{ é a transposta de } Q \text{ e } I_{2^m - m - 1} \text{ é a matriz identidade.}$$

Como as colunas de H são distintas e diferentes de zero, a soma de quaisquer duas colunas será diferente de zero. Conforme corolário 3.1.1, a distância mínima de um código de Hamming será no mínimo 3. Como H consiste de todas as m -tuplas distintas e diferentes de zero em forma de colunas, o vetor soma de quaisquer duas colunas tem que ser uma coluna de H . Portanto, $h_i + h_j = h_l$ e $h_i + h_j + h_l = 0$. Conforme corolário 3.1.2, a distância mínima de um código de Hamming será exatamente 3. Por exemplo, o código (7,4) dado na tabela 3.1 é um código de Hamming.

3.5 CONCLUSÃO

Sistemas de transmissão (ou armazenamento) ocorrem apenas em uma direção - da origem para o destino. Controles de erros para sistema unidirecional devem ser realizados através de correção radical, isto é, empregando código de correção de erros corrigindo os detectados no destino, automaticamente.

A matriz geradora de um código linear é $G = [P \quad I_k]$, onde $P_{ij} = 0$ ou 1; a submatriz P consiste de $k(n-k)$ entradas. Como cada entrada P_{ij} pode ser 0 ou 1, então existem $2^{k(n-k)}$ matrizes G 's distintas. Mas nem todas geram códigos com capacidade de correção de erros.

Os códigos de Hamming são eficientes para controle e correção de erros.

Em princípio, a “table-lookup decoding” pode ser aplicada para qualquer código linear (n,k) . Isto resulta em uma decodificação de espera mínima e de probabilidade mínima de erro. Mas, para $n-k$ grande, a implementação deste esquema de decodificação torna-se impraticável, pois um armazenamento grande é necessário.

O capítulo seguinte, estuda os códigos cíclicos, os quais possibilitam conhecer a matriz geradora a partir do polinômio gerador.

CAPÍTULO IV

CÓDIGOS CÍCLICOS

4.1 INTRODUÇÃO

Códigos cíclicos formam uma importante subclasse dos códigos lineares, pois eles têm estrutura algébrica inerente considerável, tornando possível encontrar vários métodos práticos para decodificá-los.

Códigos cíclicos foram primeiro estudados por Prange, que os identificou como ideais, em 1957 [Pra,1957]. A partir destes códigos, desenvolveu-se os códigos BCH para correção de múltiplos erros, para os quais há métodos práticos de decodificação, assunto abordado por Lin e Costello [Lin,1983].

Nesta seção, estuda-se a teoria geral de códigos cíclicos, envolvendo suas propriedades básicas, estrutura sistemática, obtenção das matrizes geradora e de verificação de paridade, cálculo da síndrome e correção de erros e um esquema de decodificação.

4.2 DEFINIÇÃO DE CÓDIGOS CÍCLICOS

Definição: Um código linear C é denominado cíclico, se para toda palavra código $v=(v_0, v_1, \dots, v_{n-2}, v_{n-1}) \in C$, existir também uma palavra código $v^{(1)}=(v_{n-1}, v_0, v_1, \dots, v_{n-2}) \in C$ [Wic,1995] e [Lin,1983].

Exemplo: Código cíclico – tabela 4.1.

Tabela 4.1 Código cíclico (7,4) gerado pelo polinômio $g(x) = 1 + x + x^3$.

Mensagem	Vetores código	Polinômios códigos
(0 0 0 0)	(0 0 0 0 0 0 0)	$0 = 0 \cdot g(x)$
(1 0 0 0)	(1 1 0 1 0 0 0)	$1 + x + x^3 = 1 \cdot g(x)$
(0 1 0 0)	(0 1 1 0 1 0 0)	$x + x^2 + x^4 = x \cdot g(x)$
(1 1 0 0)	(1 0 1 1 1 0 0)	$1 + x^2 + x^3 + x^4 = (1 + x) \cdot g(x)$
(0 0 1 0)	(0 0 1 1 0 1 0)	$x^2 + x^3 + x^5 = x^2 \cdot g(x)$
(1 0 1 0)	(1 1 1 0 0 1 0)	$1 + x + x^2 + x^5 = (1 + x^2) \cdot g(x)$
(0 1 1 0)	(0 1 0 1 1 1 0)	$x + x^3 + x^4 + x^5 = (x + x^2) \cdot g(x)$
(1 1 1 0)	(1 0 0 0 1 1 0)	$1 + x^4 + x^5 = (1 + x + x^2) \cdot g(x)$
(0 0 0 1)	(0 0 0 1 1 0 1)	$x^3 + x^4 + x^6 = x^3 \cdot g(x)$
(1 0 0 1)	(1 1 0 0 1 0 1)	$1 + x + x^4 + x^6 = (1 + x^3) \cdot g(x)$
(0 1 0 1)	(0 1 1 1 0 0 1)	$x + x^2 + x^3 + x^6 = (x + x^3) \cdot g(x)$
(1 1 0 1)	(1 0 1 0 0 0 1)	$1 + x^2 + x^6 = (1 + x + x^3) \cdot g(x)$
(0 0 1 1)	(0 0 1 0 1 1 1)	$x^2 + x^4 + x^5 + x^6 = (x^2 + x^3) \cdot g(x)$
(1 0 1 1)	(1 1 1 1 1 1 1)	$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 = (1 + x^2 + x^3) \cdot g(x)$
(0 1 1 1)	(0 1 0 0 0 1 1)	$x + x^5 + x^6 = (x + x^2 + x^3) \cdot g(x)$
(1 1 1 1)	(1 0 0 1 0 1 1)	$1 + x^3 + x^5 + x^6 = (1 + x + x^2 + x^3) \cdot g(x)$

4.3 PROPRIEDADES BÁSICAS DE CÓDIGOS CÍCLICOS

Códigos cíclicos possuem muitas propriedades algébricas que simplificam as implementações de codificação de decodificação. Para tais propriedades, associa-se os componentes de um vetor código $v = (v_0, v_1, \dots, v_{n-1})$ o polinômio

$$v(X) = v_0 + v_1X + v_2X^2 + \dots + v_{n-1}X^{n-1}.$$

Desse modo, cada vetor código corresponde a um polinômio de grau $n-1$ ou menor. A partir daqui serão usados os termos “vetor código” e “polinômio código” recíprocos. O polinômio código correspondente ao vetor código $v^{(1)}$ é

$$v^{(1)}(X) = v_{n-1} + v_0X + v_1X^2 + \dots + v_{n-2}X^{n-1}, \text{ que é equivalente ao deslocamento}$$

de um lugar para a direita e

$v^{(i)}(X) = v_{n-i} + v_{n-i+1}X + \dots + v_{n-1}X^{i-1} + v_0X^i + v_1X^{i+1} + \dots + v_{n-i-1}X^{n-1}$, que é equivalente ao deslocamento de i lugares para a direita.

Existe uma relação algébrica entre $v(X)$ e $v^{(i)}(X)$. Multiplicando $v(X)$ por X^i , obtém-se

$$X^i v(X) = v_0 X^i + v_1 X^{i+1} + \dots + v_{n-i-1} X^{n-1} + \dots + v_{n-1} X^{n+i-1}.$$

A equação acima pode ser manipulada na forma a seguir:

$$\begin{aligned} X^i v(X) &= v_{n-i} + v_{n-i+1}X + \dots + v_{n-1}X^{i-1} + v_0X^i + \dots + v_{n-i-1}X^{n-1} \\ &\quad + v_{n-i}(X^n - 1) + v_{n-i+1}X(X^n - 1) + \dots + v_{n-1}X^{i-1}(X^n - 1) \\ &= q(X)(X^n - 1) + v^{(i)}(X), \end{aligned} \quad (4.1)$$

onde $q(X) = v_{n-i} + v_{n-i+1}X + \dots + v_{n-1}X^{i-1}$. Da equação anterior, vê-se que o polinômio código $v^{(i)}(X)$ é simplesmente o resto da divisão do polinômio $X^i v(X)$ por $X^n - 1$.

Teorema 4.1:

O polinômio código diferente de zero de grau mínimo em um código cíclico C é único [Lin,1983].

Teorema 4.2:

Seja $g(X) = g_0 + g_1X + \dots + g_{r-1}X^{r-1} + X^r$ o polinômio código diferente de zero de grau mínimo em um código cíclico $C(n,k)$. Logo, o termo constante g_0 tem que ser igual a 1 [Lin,1983].

O teorema acima, diz que o polinômio código diferente de zero de grau mínimo em um código cíclico $C(n,k)$ é da forma a seguir

$$g(X) = 1 + g_1X + \dots + g_{r-1}X^{r-1} + X^r \quad (4.2)$$

Exemplo: O código cíclico da tabela 4.1 tem $g(X) = 1 + X + X^3$ como polinômio código diferente de zero de grau mínimo.

Considere os polinômios $Xg(X)$, $X^2g(X)$, ..., $X^{n-r-1}g(X)$, que tem graus $r+1$, $r+2$, ..., $n-1$, respectivamente. De (3.21) $Xg(X)=g^{(1)}(X)$, $X^2g(X)=g^{(2)}(X)$, ..., $X^{n-r-1}g(X)=g^{(n-r-1)}(X)$; isto é, eles são deslocamentos cíclicos do polinômio código $g(X)$. Portanto, eles são polinômios códigos em C . Como C é linear, uma combinação linear de $g(X)$, $Xg(X)$, ..., $X^{n-r-1}g(X)$,

$$\begin{aligned} v(X) &= u_0g(X) + u_1Xg(X) + \dots + u_{n-r-1}X^{n-r-1}g(X) \\ &= (u_0 + u_1X + \dots + u_{n-r-1}X^{n-r-1}).g(X), \end{aligned} \quad (4.3)$$

é também um polinômio código, onde $u_i = 0$ ou 1 .

Teorema 4.3:

Seja $g(X) = 1 + g_1X + \dots + g_{r-1}X^{r-1} + X^r$ o polinômio código diferente de zero de grau mínimo em código cíclico (n,k) C . Um polinômio binário de grau $n-1$ ou menor é um polinômio código, se e somente se, for múltiplo de $g(X)$ [Lin,1983].

O número de polinômios binários de grau $n-1$ ou menor que são múltiplos de $g(X)$ é 2^{n-r} . De acordo com teorema 4.3, estes polinômios formam todos os polinômios códigos do código cíclico (n,k) C . Como existem 2^k polinômios códigos em C , então 2^{n-r} tem que ser igual a 2^k . Portanto, $r=n-k$ e o grau de $g(x)$ é $n-k$. Consequentemente, o polinômio código diferente de zero de grau mínimo em um código cíclico (n,k) C é da forma a seguir

$$g(X) = 1 + g_1X + \dots + g_{n-k-1}X^{n-k-1} + X^{n-k}.$$

Teorema 4.4:

Em um código cíclico (n,k) C existe um, e somente um, polinômio código de grau $n-k$, $g(X) = 1 + g_1X + \dots + g_{n-k-1}X^{n-k-1} + X^{n-k}$. (4.4)

Todo polinômio código é um múltiplo de $g(X)$ e todo polinômio binário de grau $n-1$ ou menor que é múltiplo de $g(X)$ é um polinômio código [Lin,1983].

Do teorema 4.4, tem-se que todo polinômio código $v(X)$ em um código cíclico (n,k) pode ser da forma a seguir:

$$\begin{aligned} v(X) &= u(X)g(X) \\ &= (u_0 + u_1X + \dots + u_{k-1}X^{k-1}).g(X). \end{aligned} \quad (4.5)$$

Se os coeficientes de $u(X)$, u_0, u_1, \dots, u_{k-1} , são os k dígitos informação a ser codificados, $v(X)$ é o polinômio código correspondente. Conseqüentemente, a codificação pode ser realizada multiplicando a mensagem $u(X)$ por $g(x)$. Portanto, um código cíclico (n,k) é definido por seu polinômio código diferente de zero de grau mínimo, $g(X)$, dado em (4.4) que é denominado polinômio gerador do código.

Exemplo: O polinômio gerador do código cíclico $(7,4)$ da tabela 4.1 é $g(X) = 1 + X + X^3$. Cada polinômio código é múltiplo de $g(X)$.

Teorema 4.5:

O polinômio gerador $g(X)$ de um código cíclico (n,k) é um fator de $X^n - 1$ [Lin,1983][Wic,1995].

Para geração de códigos cíclicos, o teorema a seguir é de fundamental importância.

Teorema 4.6:

Se $g(X)$ é um polinômio de grau $n-k$ e é um fator de $X^n - 1$, então $g(X)$ gera um código cíclico [Lin,1983].

O teorema 4.6 garante que qualquer fator de $X^n - 1$ com grau $n-k$ gera um código cíclico (n,k) . Veja fatores de $X^n - 1$ em Wicker [wic,1995].

Exemplo: O polinômio $X^7 + 1$ pode ser fatorado da forma a seguir:

$$X^7 + 1 = (1 + X)(1 + X + X^3)(1 + X^2 + X^3).$$

Existem dois fatores de grau 3 e cada um gera um código cíclico (7,4). O código cíclico (7,4) da tabela 4.1 é gerado pelo polinômio $g(X) = 1 + X + X^3$. Este código tem distância mínima 3 e é um código de correção de erros simples.

4.3.1 ESQUEMA SINTÉTICO DAS PROPRIEDADES BÁSICAS

Dentro do conjunto dos polinômios código em C , existe um único polinômio "monic" $g(X)$ com grau mínimo r , $r < n$. $g(X)$ é denominado o polinômio gerador de C .

- 1- Todo polinômio código $v(X)$ em C pode ser expressado unicamente como $v(X) = u(X)g(X)$, onde $g(X)$ é o polinômio gerador de C e $u(X)$ é um polinômio de grau menor que $(n-r)$ em $GF(q)[X]$ ($GF(2)[X]$ para códigos binários).
- 2- O polinômio gerador $g(X)$ de C é um fator de $X^n - 1$ em $GF(2)[X]$.

4.4 ESTRUTURA SISTEMÁTICA DE CÓDIGOS CÍCLICOS

Observando a tabela 4.1, percebe-se que o código cíclico (7,4) não está na forma sistemática (os k dígitos mais à direita de cada vetor código são os dígitos informação inalterados e os $n-k$ dígitos mais à esquerda são os dígitos de verificação de paridade). A seguir, será mostrado o processo de obtenção do código cíclico na forma sistemática.

Dado o polinômio gerador $g(X)$ de um código cíclico (n,k) , este pode ser colocado na forma sistemática. Seja $u = (u_0, u_1, \dots, u_{k-1})$ a mensagem a ser codificada. O polinômio mensagem correspondente é

$$u(X) = u_0 + u_1X + \dots + u_{k-1}X^{k-1}.$$

Multiplicando $u(X)$ por X^{n-k} , obtém-se um polinômio de grau $n-1$ ou menor,

$$X^{n-k}u(X) = u_0X^{n-k} + u_1X^{n-k+1} + \dots + u_{k-1}X^{n-1}.$$

Dividindo $X^{n-k}u(X)$ pelo polinômio gerador $g(X)$, tem-se:

$$X^{n-k}u(X) = a(X)g(X) + b(X), \quad (4.6)$$

onde $a(X)$ e $b(X)$ são o quociente e o resto, respectivamente. Como o grau de $g(X)$ é $n-k$, o grau de $b(X)$ tem que ser $n-k-1$ ou menor, isto é,

$$b(X) = b_0 + b_1X + \dots + b_{n-k-1}X^{n-k-1}. \text{ Reorganizando (4.6), obtém-se o seguinte}$$

polinômio de grau $n-1$ ou menor:

$$X^{n-k}u(X) + b(X) = a(X)g(X). \quad (4.7)$$

Este polinômio é um múltiplo do polinômio gerador $g(X)$ e, portanto, é um polinômio código do código cíclico gerado por $g(X)$. Explicitando $b(X) + X^{n-k}u(X)$, tem-se

$$b(X) + X^{n-k}u(X) = b_0 + b_1X + \dots + b_{n-k-1}X^{n-k-1} + u_0X^{n-k} + u_1X^{n-k+1} + \dots + u_{k-1}X^{n-1}, \quad (4.8)$$

que corresponde ao vetor código

$$(b_0, b_1, \dots, b_{n-k-1}, u_0, u_1, \dots, u_{k-1}).$$

Vê-se que o vetor código consiste de k dígitos informação inalterados (u_0, u_1, \dots, u_{k-1}) seguidos por $n-k$ dígitos de verificação de paridade. Os $n-k$ dígitos de verificação de paridade são simplesmente os coeficientes do resto da divisão do polinômio mensagem $X^{n-k}u(X)$ pelo polinômio gerador $g(X)$.

Em resumo, a codificação de códigos cíclicos na forma sistemática consiste dos seguintes passos:

Passo 1: Multiplica o polinômio mensagem $u(X)$ por X^{n-k} .

Passo 2: Divide $X^{n-k}u(X)$ pelo polinômio gerador $g(X)$. Seja $b(x)$ (os dígitos de verificação de paridade) o resto.

Passo 3: Estabelece $v(X) = X^{n-k}u(X) + b(X)$.

Exemplo: Considere o código cíclico (7,4) gerado por $g(X) = 1 + X + X^3$. Seja $u(X) = 1 + X^3$ a mensagem a ser codificada.

Passo 1: $X^{n-k}u(X) = X^3u(X) = X^3 + X^6$.

Passo 2: Dividindo $X^3u(X) = X^3 + X^6$ por $g(X) = 1 + X + X^3$, obtém-se quociente $a(X) = X^3 + X$ e resto $b(X) = X^2 + X$.

Passo 3: $v(X) = X^3u(X) + b(X) = X^3 + X^6 + X^2 + X = X + X^2 + X^3 + X^6$ e corresponde ao vetor código $v(X) = (0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1)$, onde os quatro dígitos mais à direita são os dígitos informação.

4.5 MATRIZES GERADORA E DE VERIFICAÇÃO DE PARIDADE

a seguir, será mostrado um método conveniente para obtenção de códigos cíclicos, que é descrito através das matrizes geradora e de verificação de paridade.

4.5.1 MATRIZ GERADORA (G)

Considere um código cíclico $C(n, k)$ com polinômio gerador $g(X) = g_0 + g_1X + \dots + g_{n-k}X^r$ ($r=n-k$). Um bloco de mensagem $(u_0, u_1, \dots, u_{n-r-1})$ pode ser associado com um polinômio mensagem $u_0 + u_1X + \dots + u_{n-r-1}X^{n-r-1}$ e codificado através da multiplicação pelo polinômio gerador, como mostrado abaixo. O código C tem dimensão $k=(n-r)$ e contém q^{n-r} (2^{n-r} para códigos binários) palavras códigos.

$$u = (u_0, u_1, \dots, u_{n-r-1}) \leftrightarrow u(X) = u_0 + u_1X + \dots + u_{n-r-1}X^{n-r-1}$$

$$v_m = (v_0, v_1, \dots, v_{n-1}) \leftrightarrow v_m(X) = u(X)g(X) = v_0 + v_1X + \dots + v_{n-1}X^{n-1} \quad (4.9)$$

Pode-se reescrever a equação anterior usando multiplicação de matriz, como será mostrado a seguir

$$v_m(X) = u(X)g(X) = (u_0 + u_1X + \dots + u_{n-r-1}X^{n-r-1})g(X)$$

$$\begin{aligned}
&= u_0 g(X) + u_1 X g(X) + \dots + u_{n-r-1} X^{n-r-1} g(X) \\
&= [u_0 \ u_1 \ \dots \ u_{n-r-1}] \begin{bmatrix} g(X) \\ X \cdot g(X) \\ \vdots \\ X^{n-r-1} \cdot g(X) \end{bmatrix} \quad (4.10)
\end{aligned}$$

A equação anterior fornece uma forma geral conveniente para matrizes geradoras de códigos cíclicos.

$$v_m = u \begin{bmatrix} g_0 & g_1 & \dots & g_r & 0 & 0 & 0 & 0 \\ 0 & g_0 & g_1 & \dots & g_r & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & g_0 & g_1 & \dots & g_r & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & \dots & g_r \end{bmatrix} = u \cdot G \quad (4.11)$$

4.5.2 MATRIZ DE VERIFICAÇÃO DE PARIDADE (H)

O polinômio gerador $g(X)$ é um fator de $X^n - 1$,

$$X^n - 1 = g(X)h(X), \quad (4.12)$$

onde o polinômio $h(X)$ tem grau k e é da forma a seguir:

$$h(X) = h_0 + h_1 X + \dots + h_k X^k$$

com $h_0 = h_k = 1$. Precisa-se mostrar que uma matriz de verificação de paridade de C pode ser obtida de $h(X)$. Seja $v = (v_0, v_1, \dots, v_{n-1})$ um vetor código em C . Então $v(X) = a(X) \cdot g(X)$.

Multiplicando $v(X)$ por $h(X)$, obtém-se

$$\begin{aligned}
v(X) \cdot h(X) &= a(X) \cdot g(X) \cdot h(X) \\
&= a(X) \cdot (X^n - 1) \\
&= a(X) + X^n a(X).
\end{aligned} \quad (4.13)$$

Como o grau de $a(X)$ é $k-1$ ou menor, as potências $X^k, X^{k+1}, \dots, X^{n-1}$ não aparecem em $a(X) + X^n a(X)$. Expandindo o produto $v(X) \cdot h(X)$ no lado esquerdo de (4.13),

os coeficientes de $X^k, X^{k+1}, \dots, X^{n-1}$ têm que ser igual a zero. Portanto, obtém-se as seguintes $n-k$ igualdades:

$$\sum_{i=0}^k h_i v_{n-i-j} = 0, \text{ para } 1 \leq j \leq n-k. \quad (4.14)$$

A seguir, pega-se o recíproco de $h(X)$ que é definido por:

$$X^k \cdot h(X^{-1}) = h_k + h_{k-1}X + h_{k-2}X^2 + \dots + h_0X^k. \quad (4.15)$$

Pode-se ver que $X^k \cdot h(X^{-1})$ é também um fator de $X^n - 1$. O polinômio $X^k \cdot h(X^{-1})$

gera um código cíclico $(n, n-k)$ com a seguinte matriz geradora $(n-k) \times n$:

$$H = \begin{bmatrix} h_k & h_{k-1} & \dots & h_0 & 0 & 0 & 0 & 0 \\ 0 & h_k & h_{k-1} & \dots & h_0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & h_k & h_{k-1} & \dots & h_0 & 0 \\ 0 & 0 & 0 & 0 & h_k & h_{k-1} & \dots & h_0 \end{bmatrix}. \quad (4.16)$$

Das $n-k$ igualdades de (4.14), qualquer vetor código v em C é ortogonal a todas as linhas de H . Portanto, H é a matriz de verificação de paridade do código cíclico C e o espaço linha de H é o código dual de C . Como a matriz de verificação de paridade é obtida do polinômio $h(X)$, chama-se $h(X)$ o polinômio paridade de C . Conseqüentemente, um código cíclico é também unicamente definido por seu polinômio paridade.

Além disso, deduzindo uma matriz de verificação de paridade para um código cíclico, tem-se provado outra importante propriedade, que é declarada no teorema a seguir.

Teorema 4.7:

Seja $C(n, k)$ um código cíclico com polinômio gerador $g(X)$. O código dual de C é cíclico e é gerado pelo polinômio $X^k \cdot h(X^{-1})$, onde $h(X) = (X^n + 1) / g(X)$ [Lin, 1983] e [Wic, 1995].

4.5.3 MATRIZES: GERADORA E DE VERIFICAÇÃO DE PARIDADE NA FORMA SISTEMÁTICA

Pode-se obter a matriz geradora de um código cíclico na forma sistemática, dividindo X^{n-k+i} pelo polinômio gerador $g(X)$ para $i = 0, 1, \dots, k-1$, obtendo

$$X^{n-k+i} = a_i(X)g(X) + b_i(X), \quad (4.17)$$

onde $b_i(X)$ é o resto com a seguinte forma:

$$b_i(X) = b_{i0} + b_{i1}X + \dots + b_{i,n-k-1}X^{n-k-1}.$$

Como $b_i(X) + X^{n-k+i}$ para $i=0, 1, \dots, k-1$ são múltiplos de $g(X)$, eles são polinômios códigos. Arranjando estes k polinômios códigos como linhas de uma matriz $k \times n$, obtém-se

$$G = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \dots & b_{0,n-k-1} & 1 & 0 & 0 & \dots & 0 \\ b_{10} & b_{11} & b_{12} & \dots & b_{1,n-k-1} & 0 & 1 & 0 & \dots & 0 \\ b_{20} & b_{21} & b_{22} & \dots & b_{2,n-k-1} & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots & & & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} & \dots & b_{k-1,n-k-1} & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad (4.18)$$

que é a matriz geradora de C na forma sistemática. A correspondente matriz de verificação de paridade para C é:

$$H = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & b_{00} & b_{10} & b_{20} & \dots & b_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & b_{01} & b_{11} & b_{21} & \dots & b_{k-1,1} \\ 0 & 0 & 1 & \dots & 0 & b_{02} & b_{12} & b_{22} & \dots & b_{k-1,2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & b_{0,n-k-1} & b_{1,n-k-1} & b_{2,n-k-1} & \dots & b_{k-1,n-k-1} \end{bmatrix}$$

4.6 CÁLCULO DA SÍNDROME E DETECÇÃO DE ERROS

Para códigos lineares, a síndrome $s = r.H^T$. Para $s = 0$, r é um vetor código e para $s \neq 0$, r não é um vetor código e a presença de erros é detectada.

Para códigos cíclicos na forma sistemática, a síndrome s pode ser calculada facilmente. O vetor recebido r é considerado como um polinômio de grau $n-1$ ou menor, $r(X) = r_0 + r_1X + r_2X^2 + \dots + r_{n-1}X^{n-1}$. Dividindo $r(X)$ pelo polinômio gerador $g(X)$, obtém-se

$$r(X) = a(X)g(X) + s(X). \quad (4.19)$$

O resto $s(X)$ é um polinômio de grau $n-k-1$ ou menor. Os $n-k$ coeficientes de $s(X)$ formam a síndrome s . Do teorema 4.4 $s(X)$ será zero se, e somente se, o polinômio recebido $r(X)$ for um polinômio código.

Por causa da estrutura cíclica do código, a síndrome $s(X)$ tem a seguinte propriedade (teorema 4.8), que nos permite diminuir o tamanho da tabela síndrome para $(1/n)$ do tamanho original.

Teorema 4.8:

Seja $s(X)$ a síndrome de um polinômio recebido $r(X) = r_0 + r_1X + r_2X^2 + \dots + r_{n-1}X^{n-1}$. Então o resto $s^{(1)}(X)$ resultante da divisão $Xs(X)$ pelo polinômio gerador $g(x)$ é a síndrome de $r^{(1)}(X)$, que é um deslocamento cíclico de $r(X)$. [Wic, 1995]

Do teorema anterior o resto $s^{(i)}(X)$ resultante da divisão $X^i s(X)$ pelo polinômio gerador $g(X)$ é a síndrome de $r^{(i)}(X)$, que é o i -ésimo deslocamento cíclico de $r(X)$. Esta propriedade é útil para a decodificação de códigos cíclicos.

Seja $v(X)$ a palavra código transmitida e $e(X) = e_0 + e_1X + e_2X^2 + \dots + e_{n-1}X^{n-1}$ o erro padrão. Então o polinômio recebido é

$$r(X) = v(X) + e(X) \quad (4.20)$$

Como $v(X)$ é múltiplo do polinômio gerador $g(X)$, combinando (4.19) e (4.20), tem-se a seguinte relação entre o erro padrão e a síndrome:

$$e(X) = [a(X) + b(X)]g(X) + s(X), \quad (4.21)$$

onde $b(X)g(X) = v(X)$. Isto mostra que a síndrome é realmente igual ao resto da divisão do erro padrão pelo polinômio gerador, ela pode ser calculada a partir do vetor recebido; porém o erro padrão $e(X)$ é desconhecido pelo decoder. Portanto o decoder tem como estimar $e(X)$ baseado na síndrome $s(X)$. Se $e(X)$ for um coset leader no conjunto padrão e se for usado a table-lookup de decodificação, $e(X)$ pode ser corretamente determinado da síndrome.

De (4.21), vê-se que $s(X)$ será zero se, e somente se, o erro padrão $e(X)=0$ ou idêntico a um vetor código. Se $e(X)$ for idêntico a um vetor código, $e(X)$ será um erro padrão não detectável. Códigos cíclicos são mais eficientes para detectar erros aleatórios. Para maiores detalhes veja Lin e Costello [Lin,1983].

4.7 DECODIFICAÇÃO DE CÓDIGOS CÍCLICOS

Decodificação de códigos cíclicos consiste dos mesmos passos para decodificação de códigos lineares, descritos no item 3.4.7 (cálculo da síndrome, associação da síndrome a um erro padrão e correção do erro).

Exemplo: Considere o código cíclico (7,4) gerado por $g(X) = 1 + X + X^3$. Este código tem distância mínima 3 e é capaz de corrigir qualquer erro simples em um bloco de sete dígitos. Estes sete erros padrões e o vetor zero formam os coset leaders da tabela de decodificação.

Considere o método para obter a matriz geradora do código na forma sistemática: Dividindo X^3 , X^4 , X^5 e X^6 por $g(X)$, tem-se:

$$X^3 = g(X) + (1 + X),$$

$$X^4 = Xg(X) + (X + X^2)$$

$$X^5 = (X^2 + 1)g(X) + (1 + X + X^2)$$

$$X^6 = (X^3 + X + 1)g(X) + (1 + X^2).$$

Das equações acima, tem-se os seguintes polinômios códigos:

$$v_0(X) = 1 + X + X^3,$$

$$v_1(X) = X + X^2 + X^4,$$

$$v_2(X) = 1 + X + X^2 + X^5,$$

$$v_3(X) = 1 + X^2 + X^6.$$

Assumindo estes polinômios códigos como linhas da matriz 4×7 , tem-se a seguinte matriz geradora do código cíclico (7,4) na forma sistemática:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

A matriz de verificação de paridade é:

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

Os erros padrões corrigíveis e suas correspondentes síndromes são colocados na tabela a seguir, tal tabela poderá ser reduzida de $1/n$ do tamanho original, conforme teorema 4.8:

Tabela 4.2- Decodificação do código cíclico (7,4), gerado por $g(X)=1+X+X^3$.

Síndrome	Polinômio síndrome	Polinômio erro padrão	Coset leaders
(1 0 0)	1	1	(1 0 0 0 0 0)
(0 1 0)	X	X	(0 1 0 0 0 0)
(0 0 1)	X^2	X^2	(0 0 1 0 0 0)
(1 1 0)	$1+X$	X^3	(0 0 0 1 0 0)
(0 1 1)	$X+X^2$	X^4	(0 0 0 0 1 0)
(1 1 1)	$1+X+X^2$	X^5	(0 0 0 0 0 1)
(1 0 1)	$1+X^2$	X^6	(0 0 0 0 0 1)

Como os erros padrões consiste dos deslocamentos cíclicos de (0 0 0 0 0 0 1), o decoder necessita apenas de ser capaz de reconhecer uma das sete síndromes diferente de zero para corrigir todos os erros padrões (veja teorema 4.8).

4.8 CONCLUSÃO

Códigos cíclicos são obtidos a partir de um polinômio $g(X)$ - denominado polinômio gerador do código, o qual é um fator de $X^n - 1$ em $GF(2)[X]$.

Em princípio, a "table-lookup decoding" pode ser aplicada a qualquer código cíclico (n,k) e conforme teorema 4.8, esta tabela pode ser reduzida de $1/n$ do tamanho original. Isto resulta em uma decodificação de espera mínima e de probabilidade mínima de erro. Para $n-k$ grande, faz-se necessário um armazenamento grande.

No próximo capítulo, serão propostos dois sistemas criptográficos usando códigos lineares, tendo como comparação o DES para verificar sua eficiência em um, e utilizando o DES em outro.

CAPÍTULO V

SISTEMAS CRIPTÓGRAFICOS CLÁSSICOS USANDO TEORIA ALGÉBRICA DOS CÓDIGOS LINEARES.

5.1 - INTRODUÇÃO

São propostos dois sistemas criptográficos utilizando conceitos de códigos lineares. Um será produzido independente do DES, mas comparado a ele; o outro será composto com o DES.

Para tanto, apresenta-se os elementos básicos dos sistemas criptográficos utilizando os códigos lineares. Neste caso, a cada mensagem a ser cifrada e enviada, será adicionado um erro padrão corrigível. Isto cria dificuldades para o criptoanalista obter a mensagem original a partir da cifrada, eliminando assim a linearidade das substituições.

Será feita uma descrição detalhada dos sistemas criptográficos, analisando algumas questões de segurança e de velocidade em relação ao processo de cifragem e decifragem dos mesmos, encerrando com uma breve conclusão do capítulo.

5.2 – SISTEMA CRIPTÓGRAFICO CLÁSSICO USANDO CÓDIGOS LINEARES

Baseado na teoria algébrica de códigos lineares e na teoria de criptografia de chave secreta, é proposto o sistema abaixo.

O sistema criptográfico será baseado em um alfabeto de 26 letras e mais o caracter especial de espaço. Cada letra tem um correspondente numérico em binário, considerando os elementos de $GF(2)$. Na tabela 5.1, tem-se as letras com seus respectivos valores numéricos.

A cifragem do texto principal pode ser: de letra em letra, de duas em duas letras, assim sucessivamente - será cifrado pelo remetente e enviado ao destinatário bloco de texto principal.

Remetente e destinatário combinam a chave secreta, ou seja, combinam os valores de n , k e G , onde:

k = Quantidade de dígitos informação;

G = Matriz geradora do código linear;

n = Quantidade de dígitos de informação mais os dígitos de verificação de paridade.

Tabela 5.1 - Representação binária das letras do alfabeto.

Letra	Número	Notação binária
Espaço	0	00000
A	1	00001
B	2	00010
C	3	00011
D	4	00100
E	5	00101
F	6	00110
G	7	00111
H	8	01000
I	9	01001
J	10	01010
K	11	01011
L	12	01100
M	13	01101
N	14	01110
O	15	01111
P	16	10000
Q	17	10001
R	18	10010
S	19	10011

T	20	10100
U	21	10101
V	22	10110
W	23	10111
X	24	11000
Y	25	11001
Z	26	11010

Método para cifragem: Suponha que o remetente A necessite enviar uma mensagem para o destinatário B. Neste caso, o remetente A representa sua mensagem como uma “string” binária (texto em binário) de tamanho k (blocos de k bits informação) e envia para o destinatário B

$$C = u \cdot G + e,$$

onde:

u = vetor mensagem (ou informação) de k bits correspondendo a um bloco de texto principal;

e = vetor erro aleatório (erro padrão corrigível) de comprimento n e peso t ,
onde $t = \lfloor (d_{\min} - 1) / 2 \rfloor$;

C = criptotexto – texto cifrado em binário;

Método para decifragem: O destinatário B recebe o texto cifrado C. A decifragem do vetor recebido, denotado por r consiste dos seguintes passos:

1- Calcula a síndrome do vetor recebido r , $s = r \cdot H^T$ (como o destinatário conhece G , ele gera $H = [I_{n-k} P^T]$ e H^T);

2- Localiza o “coset leader” e_1 cuja a síndrome é igual a s . Então e_1 é assumido com um erro padrão adicionado ao código (o destinatário gera a tabela de decodificação síndrome, conforme descrita na seção 3.4.7);

3- Decodifica o vetor recebido r no vetor mensagem $u = r + e_1$.

A tabela que se segue sintetiza este sistema.

Tabela 5.2 – O sistema clássico usando códigos lineares

Secreto:	G, n e k
Mensagens de <u>A</u> par <u>B</u> :	Vetor binário u de comprimento k bits
Cifragem por <u>A</u> :	$C = u \cdot G + e$, onde o peso(e) $\leq t$
Decifragem por <u>B</u> :	<ol style="list-style-type: none"> 1- Calcula $s = r \cdot H^T$, onde: $r = C$. 2- Localiza e_1 cuja a síndrome seja igual a s 3- Calcula $u = r + e_1$

Exemplo: Considere que se queira cifrar e enviar o seguinte “texto”:

JUNIOR

Logo, $u_1 = 01010$, $u_2 = 10101$, $u_3 = 01110$, $u_4 = \dots$

Para cifrar uma mensagem remetente e destinatário combinam os valores de n e k e, definem a matriz geradora G , ou seja, definem as chaves de cifragem. Neste caso, cada bloco de texto principal corresponde a uma letra e o código terá quantidade de dígitos de informação $k=5$ (mensagem). Considerando $n=9$, tem-se o código linear $(9,5)$. Assim existe 4 $(n-k)$ dígitos de verificação de paridade do código. Portanto, a matriz geradora G será da ordem 5×9 ($k \times n$). A seguir será considerada uma matriz geradora G , dentre as possíveis G 's com capacidade de correção de erros, como exemplo.

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

Seja $u = (0 \ 1 \ 0 \ 1 \ 0)$ a mensagem a ser codificada. A palavra código

correspondente $v = u \cdot G$, é:

$$v = 0 \cdot g_0 + 1 \cdot g_1 + 0 \cdot g_2 + 1 \cdot g_3 + 0 \cdot g_4$$

$$= (0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0)$$

Tabela 5.3—Código linear (n,k) , com $k=5$ e $n=9$

$u =$ Mensagem	$v =$ Palavra código
(00000)	(000000000)
(00001)	(011100001)
(00010)	(111100010)
(00011)	(100000011)
(00100)	(101100100)
(00101)	(110000101)
(00110)	(010000110)
(00111)	(001100111)
(01000)	(110101000)
(01001)	(101001001)
(01010)	(001001010)
(01011)	(010101011)
(01100)	(011001100)
(01101)	(000101101)
(01110)	(100101110)
(01111)	(111001111)
(10000)	(111010000)
(10001)	(100110001)
(10010)	(000110010)
(10011)	(011010011)
(10100)	(010110100)
(10101)	(001010101)
(10110)	(101010110)
(10111)	(110110111)
(11000)	(001111000)
(11001)	(010011001)
(11010)	(110011010)
(11011)	(101111011)
(11100)	(100011100)
(11101)	(111111101)
(11110)	(011111110)
(11111)	(000011111)

Para cifragem da palavra código v ($u.G$), será adicionado um erro padrão e corrigível de forma aleatória. Como a matriz de verificação de paridade H tem distância mínima igual a 3 ($d_{min}=3$, veja corolários 3.1.1 e 3.1.2) e um código linear garante detectar e corrigir todos os erros padrões de $t = \lfloor (d_{min}-1)/2 \rfloor$ ou menos erros, onde $\lfloor (d_{min}-1)/2 \rfloor$ é o maior inteiro não maior que $\lfloor (d_{min}-1)/2 \rfloor$. O parâmetro t é chamado a capacidade de correção de erros aleatórios do código. Portanto, $t = \lfloor (3-1)/2 \rfloor = 1$, será adicionado à palavra código para cifragem um erro padrão e simples. Logo: $C = v + e$. Supondo que $e = (0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0)$ seja adicionado à palavra código v . Logo, $C = (0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0)$. Portanto C é enviado ao destinatário.

O destinatário recebe o "texto cifrado" C , denotado por r , e calcula a síndrome $s = r.H^T$.

Como o destinatário conhece G , então ele cria a matriz $H = [I_{n-k} \ . \ P^T]$. Logo:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} e$$

$$H^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} e \text{ calcula a síndrome } s. \text{ Logo } s = (1 \ 1 \ 0 \ 1). \text{ Como } s \neq 0,$$

a presença de erros é detectada.

A seguir será criada a tabela de decifragem para o código linear (9,5). A tabela consistirá de todos coset leaders (os erros padrões corrigíveis) e suas correspondentes síndromes. Esta tabela é armazenada pelo destinatário.

Tabela 5.4 – Tabela de decifragem para o código linear (9,5) (table lookup decoding)

Síndromes	Coset leaders
(1 0 0 0)	(1 0 0 0 0 0 0 0 0)
(0 1 0 0)	(0 1 0 0 0 0 0 0 0)
(0 0 1 0)	(0 0 1 0 0 0 0 0 0)
(0 0 0 1)	(0 0 0 1 0 0 0 0 0)
(1 1 1 0)	(0 0 0 0 1 0 0 0 0)
(1 1 0 1)	(0 0 0 0 0 1 0 0 0)
(1 0 1 1)	(0 0 0 0 0 0 1 0 0)
(1 1 1 1)	(0 0 0 0 0 0 0 1 0)
(0 1 1 1)	(0 0 0 0 0 0 0 0 1)

Logo, o destinatário localiza o coset leader e cuja a síndrome é igual a s . Então e é assumido como um erro padrão colocado na palavra código pelo emitente. Portanto o vetor recebido r é decodificado da seguinte maneira:

$v^* = r + e$. da tabela anterior, (1 1 0 1) é a síndrome do coset leader $e = (0 0 0 0 0 1 0 0 0)$. Portanto, (0 0 0 0 0 1 0 0 0) é assumido como o erro padrão colocado pelo emitente, e r é decifrado em

$$(0 0 1 0 0 0 0 1 0) + (0 0 0 0 0 1 0 0 0) = (0 0 1 0 0 1 0 1 0),$$

que é o verdadeiro vetor código enviado. Como o código está na forma sistemática, os k dígitos mais à direita correspondem à mensagem u . Logo $u = (0 1 0 1 0)$, que é a verdadeira mensagem transmitida.

5.2.1 - CRIPTOANÁLISE DO SISTEMA

Adiciona-se um erro padrão à palavra código para criar dificuldades para o criptoanalista obter v de C , pois o mesmo elimina a linearidade das substituições. Verifica-se a segurança deste sistema, considerando que o criptoanalista conhece o sistema usado e está interceptando a mensagem cifrada, através de alguns possíveis ataques, dentre eles:

Ataque 1: O criptoanalista conhece apenas n . o trabalho dele é criar, por tentativa, as possíveis matrizes G 's que geram códigos com capacidade de correção de erros. Exemplo: Com $n=63$, obtém-se códigos lineares (n,k) com valores de k no intervalo fechado $[7,57]$. Se o código tem capacidade de correção de t -erros, então t está no intervalo fechado $[1,15]$, sendo inversamente proporcional ao valor de k , conforme figura 5.1.

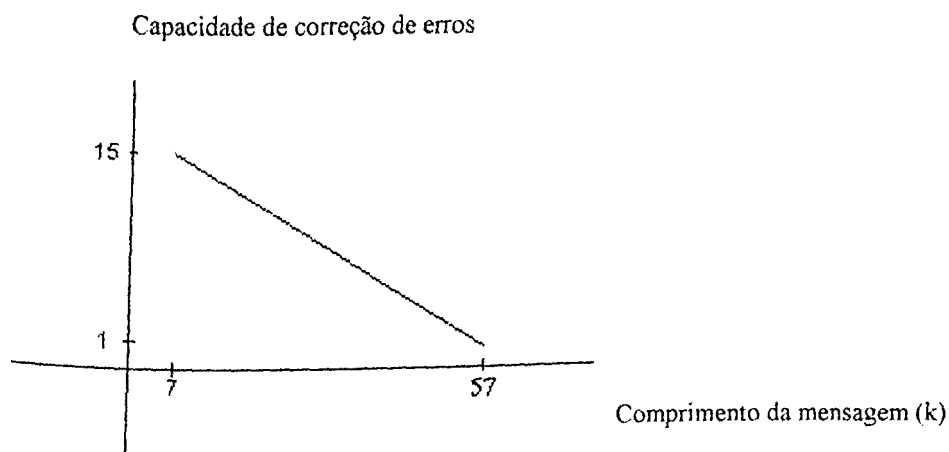


Figura 5.1 Capacidade de correção de erros (t) \times comprimento da mensagem (k) de um código linear de tamanho n .

Ataque 2: O criptoanalista conhece n e k . Considerando o código linear na forma sistemática, conforme item 3.4.2, a carga de trabalho dele é menor que no caso anterior. Neste caso, o criptoanalista conhece as dimensões da matriz G .

Ataque 3: O criptoanalista conhece n , desconhece k e utiliza o método de análise de frequência das mensagens. Este processo é complicado, pois a cada mensagem enviada é colocado um erro padrão de forma aleatória, que conforme ataque 1 ($n=63$), pode-se colocar em cada mensagem a ser cifrada um erro padrão capaz de alterar até 15 dígitos da mensagem, onde o vetor erro tem comprimento n .

5.2.2 CONSIDERAÇÕES SOBRE ESTE SISTEMA.

O sistema proposto é melhor que os sistemas criptográficos lineares clássicos, pois introduz um erro, o qual introduz uma não linearidade ao sistema. A adição de um vetor erro e altera alguns dígitos da mensagem original, tornando o sistema um pouco melhor em relação ao linear clássico.

Observe que deve-se utilizar apenas códigos que possuem capacidade de correção de erros.

O ideal é a utilização de um código linear com quantidade de dígitos suficientes para representar o maior número possível de letras (palavras) e com maior capacidade de correção de erros, para que o sistema seja mais seguro. Porém isso não é possível, pois ao aumentar a capacidade de correção de erros do código diminui a quantidade de dígitos mensagens do mesmo, conforme figura 5.1. A capacidade máxima de correção de erros de um código linear (n,k) é aproximadamente $\frac{1}{4}(n-1)$. Existem, portanto, códigos lineares com valores de n grande e com capacidade de correção de erros muito menor do que a "fração" acima.

Para construir um código linear, tem-se teoricamente, $2^{k(n-k)}$ possibilidades para matrizes geradoras G 's do código e correspondentes matrizes H . Mas existem G 's que geram códigos sem capacidade de correção, pois um código linear que corrige pelo menos

um erro ($t=1$) tem que ter pelos menos $d_{\min} = 3$ ($t = \lfloor (d_{\min} - 1)/2 \rfloor$). Portanto, tem-se as seguintes considerações para as matrizes H , para se ter códigos lineares capazes de corrigir pelos menos um erro (veja corolários 3.1.1 e 3.1.2):

- 1 - H não pode ter nenhuma coluna igual a zero;
- 2 - H não pode ter duas colunas iguais - todas as colunas de H devem ser diferentes.

A figura 5.2 indica a classe das matrizes úteis para criação de códigos lineares que podem ser utilizadas no sistema criptográfico proposto.

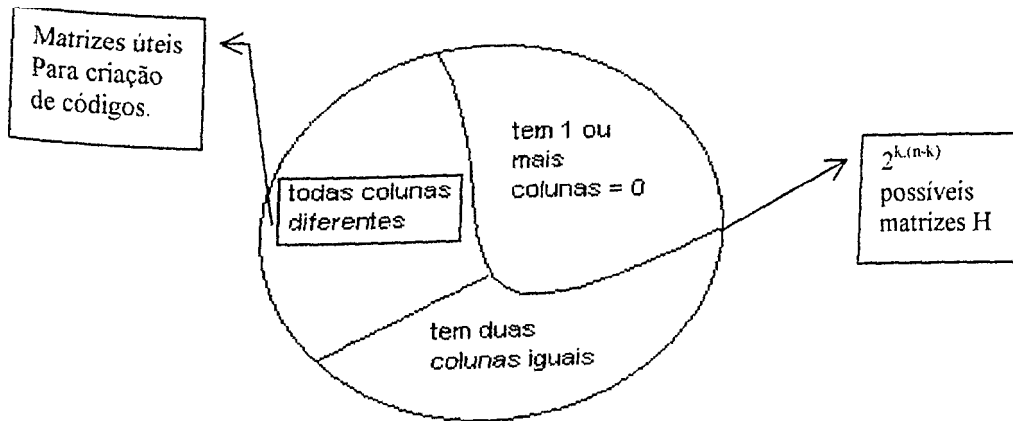


Figura 5.2 Possibilidades de matrizes de verificação de paridade H

O número de matrizes H com todas as colunas diferentes estabelece a complexidade da criptoanálise. Um problema relevante é determinar tal número, o que não foi feito no presente trabalho.

O sistema proposto é um sistema de chave privada e a chave a ser usada na “comunicação” tem que ser combinada previamente entre remetente e destinatário. Para isso, remetente e destinatário precisam se encontrar ou usar um meio de comunicação seguro. Considerando o código na forma sistemática, a chave secreta será uma parte da matriz $G = [P_{k,(n-k)} \ I_k]$, ou seja, apenas a submatriz P .

Neste sistema, para evitar a forma sistemática do texto cifrado, pode-se utilizar uma matriz para permutação P . Neste caso, a cifragem é dada por:

$$C = (u \cdot G + e) \cdot P$$

e a decifragem por:

- 1 - Cálculo de $C \cdot P^T$ obtendo $u \cdot G + e$;
- 2 - Repete os procedimentos considerados anteriormente para obter u a partir de $u \cdot G + e$.

5.3 SISTEMA CRIPTOGRÁFICO CLÁSSICO COMPOSTO POR CÓDIGOS LINEARES E DES

Propõe-se a seguir um sistema que utiliza códigos lineares e o DES. Para cifragem de uma mensagem u neste sistema, é usado o seguinte procedimento:


$$DESC = J * DES_{k_1},$$

onde: a operação $*$ significa que a seqüência originada de J é transferida para o DES_{k_1} ;

DES_{k_1} - é da forma dada em (2.2), e k é a chave secreta de 56 bits;

$J = [(u \cdot G) + e] \cdot P$, assim:

u - é um vetor mensagem (ou informação) de comprimento k bits;

G - é a matriz geradora do código linear;  Chaves secretas

P - é uma matriz para permutação de ordem $n \times n$;

e - é um vetor erro aleatório (erro padrão corrigível) de comprimento n e peso t ,

onde $t = \lfloor (d_{min}-1)/2 \rfloor$.

$DESC$ - é a seqüência de bits binários cifrada;

Para recuperar a mensagem u , é usado o seguinte procedimento:

$u = DES_{k_1}^{-1} * J^1$. A operação $*$ significa que a seqüência originada de $DES_{k_1}^{-1}$ é transferida para o J^1 .

onde: $DES_{k_1}^{-1}$ é da forma de (2.3) – página 13 e k_1 é a chave secreta usada na cifragem;

J^1 - é a operação de decodificação de J . Para decodificar J e conseqüente recuperação da mensagem u , faz-se necessário os seguintes passos:

- 1- Calcular $C' = J.P^T$.
- 2- Decodificar C' e encontrar a mensagem u . Para decodificar C' , basta usar a tabela de decodificação síndrome.

Tanto o DES quanto o sistema composto proposto anteriormente, podem ser implementados em hardware e software. Códigos lineares são implementados em hardwares, tendo como inconveniente o tamanho da tabela síndrome (ou circuito síndrome), para $n-k$ grande [Lin, 1983].

Não foi medido o tempo de execução deste sistema, porém acredita-se que em um hardware apropriado poderá ser executado de forma rápida.

5.3.1 CRIPTOANÁLISE DESTE SISTEMA

Foi visto no item 3.1.4 que a alternativa para se quebrar o DES é a força bruta. Considerando chaves de 56 bits, existem 2^{56} chaves diferentes e, portanto, são necessários $2^{56} = 10^{17}$ ciframentos para determinar a chave que está sendo utilizada.

No sistema J é utilizado uma matriz de permutação para evitar a forma sistemática da mensagem cifrada. Considerando a força bruta para quebra deste sistema, deve-se computar além das 2^{56} possibilidades de chaves do DES, as diferentes possibilidades para as matrizes P e G .

Portanto este sistema é mais difícil de ser quebrado do que o DES, utilizado individualmente.

A figura a seguir ilustra o exposto acima.

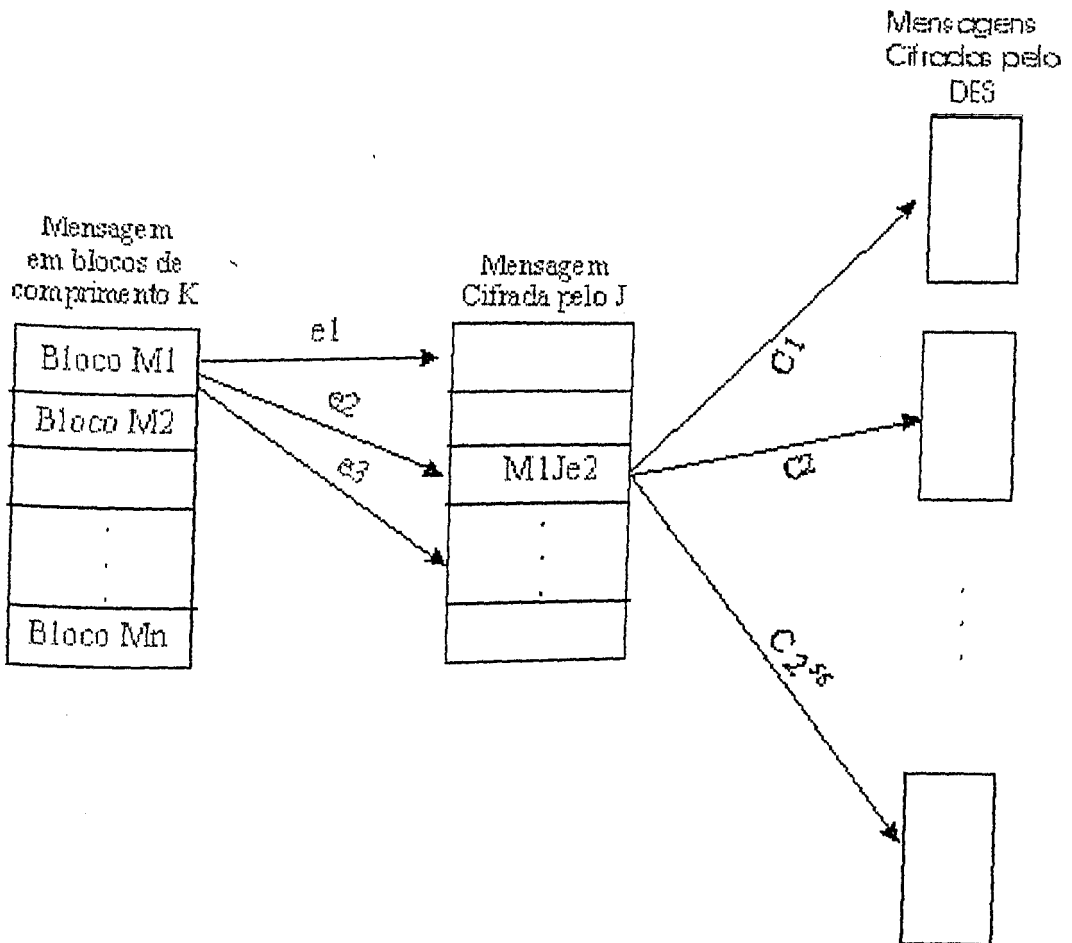


Figura 5.3 Bloco de mensagem de comprimento k e possíveis blocos de k cifrados.

Na figura 5.3, o bloco de mensagem M_1 , por exemplo, é cifrado em uma mensagem, que depende do erro utilizado.

Considerando o erro e_2 , M_1 é codificado em $M_1J_{e_2}$. Em seguida, dependendo da chave a ser utilizada pelo DES, tem-se a cifragem de $M_1J_{e_2}$. Portanto, em uma análise

“força bruta” deve-se considerar todas as possíveis chaves $C_1, C_2, \dots, C_{2^{56}}$ e em seguida todos os possíveis códigos lineares determinados por matrizes G e as permutações P .

Finalmente, deve ser salientado que a simples composição de dois sistemas DES não altera a complexidade final da criptoanálise. Tal fato não ocorre com o sistema proposto.

5.5 – CONCLUSÃO

Propõe-se neste capítulo dois sistemas criptográficos clássicos.

O primeiro utiliza princípios de códigos lineares, que introduzem uma não linearidade na cifragem do texto principal. Comparando tal sistema com sistemas criptográficos clássicos de deslocamento, há uma melhoria devido à presença do erro e aleatório, que é somado ao produto $u.G$. Entretanto, tal sistema é ainda inferior ao sistema do tipo DES, devido principalmente à simplicidade de decifragem e segurança.

O segundo sistema proposto é uma composição dos DES com o sistema criptográfico baseado em códigos; o qual é uma melhoria do DES, no que se refere à segurança. Entretanto, a decifragem neste sistema é mais complexa que no DES, devido à utilização de códigos. Mas, mesmo sendo mais complexo, ele ainda pode ser implementado por circuitos lógicos, o que ocorre com o DES e com códigos.

Finalmente, os dois sistemas propostos, sendo sistemas de chave privada, têm todas as limitações dos sistemas criptográficos clássicos.

CAPÍTULO VI

SISTEMA CRIPTÓGRAFICO DE CHAVE PÚBLICA USANDO TEORIA ALGÉBRICA DOS CÓDIGOS CÍCLICOS.

6.1 - INTRODUÇÃO

Atualmente, com a evolução tecnológica e a conseqüente diminuição dos custos dos computadores, tornou-se cada vez mais atraente e acessível a sua existência em diferentes pontos físicos, interconectados uns aos outros de forma a permitir o compartilhamento de recursos e a troca de informação de forma segura.

Existem sistemas criptográficos de chave pública eficientes, mas que são computacionalmente lentos, dentre eles, o RSA, apresentado no capítulo II. Refletindo sobre a “lentidão” do RSA, em especial, surge a questão: “ existe sistema criptográfico de chave pública tão bom quanto o RSA, mas computacionalmente mais rápido?”.

Na tentativa de resolver tal problema, utiliza-se teoria algébrica dos códigos, no projeto de um sistema criptográfico de chave pública.

Apresenta-se a seguir os elementos básicos desse sistema criptográfico e seus respectivos conceitos utilizando os códigos de blocos cíclicos. Neste sistema, e a cada mensagem a ser enviada, será adicionado um erro padrão corrigível, o qual cria dificuldades para o criptoanalista obter a mensagem original a partir da cifrada, eliminando assim, a linearidade das substituições.

Para se ter mais compreensão, serão explicitados alguns conceitos fundamentais de **matrizes de permutação e não singular**, as quais são fundamentais para o sistema em questão; logo após, será feita uma descrição do sistema criptográfico,

elucidando algumas questões no tocante à segurança do mesmo, encerrando com uma breve conclusão do capítulo.

6.2 CONCEITOS FUNDAMENTAIS SOBRE MATRIZES

6.2.1 MATRIZ NÃO SINGULAR

Definição:

Uma matriz $A_{n \times n}$ é denominada não singular se, e somente se, existir uma matriz B , tal que $AxB = I$, caso contrário, A é denominada singular. A matriz B é a inversa de A e é denotada por A^{-1} .

Matriz não singular possui uma inversa multiplicativa, conseqüentemente, seu determinante¹ é diferente de zero. Se uma matriz A for não singular, seu determinante, denotado por $\det(A)$, é diferente de zero e $\det(A^{-1}) = (\det(A))^{-1} \neq 0$.

6.2.2 MATRIZ DE PERMUTAÇÃO

Definição:

A matriz de permutação é obtida da matriz identidade, permutando suas colunas.

O efeito da matriz de permutação é a mudança na ordem das colunas na matriz original.

¹ Propriedades e Cálculo de Determinantes de Matrizes $n \times n$ e Cálculo de Matrizes Inversas em [Nob, 1986].

Exemplo: Considere as matrizes $A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ e $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. Observe que a

matriz P é uma permutação da 1ª linha com a 2ª da matriz identidade. O produto de A por

P , $AxP = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$. Comparando-se a matriz A com a matriz AxP , percebe-se que

houve apenas uma alteração na ordem das colunas da matriz A em relação a AxP , ou seja, a 1ª coluna da matriz A corresponde a 2ª coluna da matriz AxP e a 2ª coluna da matriz A corresponde a 1ª coluna da matriz AxP .

Para reaver a matriz A , a partir de AxP , basta multiplicá-la pela transposta da matriz de permutação P (ou simplesmente P).

6.3 – SISTEMA CRIPTOGRÁFICO DE CHAVE PÚBLICA

Baseado na teoria algébrica de códigos cíclicos binários e na teoria de criptografia de chave pública, propõe-se o seguinte sistema, no qual cada usuário escolhe:

Um número n que é igual ao tamanho do bloco de mensagem. O número n tem que dividir $2^m - 1$, para $m = 1, 2, \dots$. Em seguida, determina-se códigos cíclicos possíveis para n . Exemplo: para $n = 7$, fatorando $x^7 - 1$, obtém-se: $(1+x)(1+x^2+x^3)(1+x+x^3)$ que corresponde a um polinômio de grau 1 e dois polinômios de grau 3 (fatoração sobre $GF(2^m)$). Logo, os códigos cíclicos possíveis para $n = 7$ são: $(7,1)$, $(7,3)$, $(7,4)$, $(7,6)$ e $(7,7)$;

O próximo passo é determinar um polinômio $p(X)$ de grau $n-k$, que seja fator de $x^n - 1$ e criar a matriz geradora $G_{k,n}$ do código cíclico correspondente. Seja t a capacidade de correção de erros do código, portanto $t = \lfloor (d_{\min} - 1) / 2 \rfloor$;

Finalmente define-se uma matriz não singular, densa, $S_{k,k}$ e uma matriz de permutação $P_{n,n}$ de forma aleatória e calcula $G^* = S_{k,k} \cdot G_{k,n} \cdot P_{n,n}$. O usuário publica G^* , k e n , mas mantém G , S , P e t secretas.

O método de cifragem é determinado como segue:

Supõe-se que o usuário A necessita enviar uma mensagem para o usuário B. Ele representa sua mensagem como uma string binária u , de tamanho k_B . A mensagem cifrada que A envia para B é obtida da seguinte forma:

$$\begin{aligned} C &= u \cdot G_B^* + e, \\ &= u \cdot (S_{k,k} \cdot G_{k,n} \cdot P_{n,n}) + e. \end{aligned}$$

onde e é um vetor erro aleatório (erro padrão corrigível) de comprimento n_B e peso $\leq t_B$.

Para decifrar, o usuário B recebe a mensagem cifrada C e calcula com sua matriz de permutação P secreta:

$$C \cdot P_B^T = u \cdot S_B \cdot G_B \cdot P_B \cdot P_B^T + e \cdot P_B^T = (u \cdot S_B) G_B + e', \text{ onde } e' = e \cdot P_B^T \text{ tem peso } \leq t_B.$$

Com o “algoritmo/esquema” de decodificação de códigos cíclicos, o receptor (usuário B), pode, neste momento, recuperar $u \cdot S_B$. Multiplicando $u \cdot S_B$ à direita por S_B^{-1} (conhecido apenas por B), a mensagem u original é obtida.

A tabela que se segue sintetiza este sistema.

Tabela 6.1 O sistema de chave pública usando códigos cíclicos

Público:	G^* , t , k e n para todos os usuários
Secreto:	$p(X)$, S e P .
Propriedades:	G é matriz geradora do código cíclico gerado por $p(X)$. $G^* = S_{k,k} \cdot G_{k,n} \cdot P_{n,n}$.
Mensagens de <u>A</u> para <u>B</u> :	Vetor binário u de comprimento k_B .
Cifragem por <u>A</u> :	$C = u \cdot G^* + e$, onde o peso(e) $\leq t_B$.
Decifragem por <u>B</u> :	1- Calcula $C \cdot P^T = C'$. 2- Decodifica C' e encontra $u' = u \cdot S_B$. 3- Calcula $u = u' \cdot S_B^{-1}$.

Exemplo: Cifragem e decifragem.

Considerando o comprimento do bloco de mensagem codificada igual a 7 ($n=7$), fatorando $x^7 - 1$, obtém-se:

$$(1+x)(1+x^2+x^3)(1+x+x^3).$$

Dentre os possíveis códigos cíclicos, é escolhido o (7,4), portanto a quantidade de bits informação será 4 ($k=4$). O polinômio gerador para este código, tem grau $n-k$, ou seja, um polinômio fator de $x^7 - 1$ de grau 3. Então é escolhido $g(X) = 1 + X + X^3$.

Usando a forma sistemática para a obtenção do código cíclico, tem-se as matrizes:

- Geradora

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

- Verificação de Paridade $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

Regra de cifragem: $C = u \cdot G^* + e$. Os dados necessários para cifrar são:

- Obtenção de G^* . Neste caso, considera-se a matriz não singular densa $S_{4,4}$ e de permutação $P_{7,7}$ a seguir:

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad e \quad P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Portanto, $G^* = S \cdot G \cdot P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$

- u Mensagem original a ser enviada - $u = (1 \ 0 \ 1 \ 1)$

- e Vetor erro padrão de comprimento n e peso $\leq t$ ($t = \lfloor (3-1)/2 \rfloor = 1$).

$$e = (1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0).$$

Cifrando: $C = u \cdot G^* + e$.

$$C = (1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1).$$

Regras de decifragem:

1 - Calcula $C \cdot P^T = C' = (1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1)$

2 - Decodifica C' e encontra u' . Para decodificar C' , faz-se necessário um

método de decodificação de códigos cíclicos. Será usado o método de decodificação síndrome, conforme tabela a seguir:

Tabela 6.2 Tabela de decodificação síndrome para o código cíclico (7,4)

Síndrome	Erro padrão
(0 0 0)	(0 0 0 0 0 0)
(1 0 0)	(1 0 0 0 0 0)
(0 1 0)	(0 1 0 0 0 0)
(0 0 1)	(0 0 1 0 0 0)
(1 1 0)	(0 0 0 1 0 0)
(0 1 1)	(0 0 0 0 1 0)
(1 1 1)	(0 0 0 0 0 1)
(1 0 1)	(0 0 0 0 0 1)

Logo, o destinatário calcula a síndrome s de C' ($s = C' \cdot H^1$) e encontra $s = (0 1 0)$ que corresponde ao erro padrão $(0 1 0 0 0 0)$. Portanto, $u' = (1 1 0 1 0 1 1) + (0 1 0 0 0 0) = (1 0 1 1)$, pois o código está na forma sistemática.

3 - Calcula $u = u' \cdot S^{-1}$. Para se ter S^{-1} calcula-se a inversa da matriz S , obtendo:

$$S^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \text{ Portanto, } u = u' \cdot S^{-1} = (1 0 1 1).$$

6.4 - CRIPTOANÁLISE DO SISTEMA

Discute-se a segurança deste sistema, considerando que o criptoanalista conhece o sistema usado e está interceptando a mensagem cifrada, analisando alguns possíveis ataques [Van,1988], tendo como referência o código cíclico (n,k) , com $n=1023$, $k=523$ e $t=55$.

Ataque 1: O criptoanalista estima S_B e P_B para calcular G_B a partir de G_B^* . Considerando S_B e P_B corretas, pode-se seguir o esquema de decifragem do usuário B para encontrar a mensagem u . Entretanto, o número de matrizes diferentes S_B e P_B é grande e a probabilidade de sucesso desse ataque é muito menor do que a probabilidade de estimar o verdadeiro vetor u .

Ataque 2: Compara o vetor recebido C , com todas as palavras códigos no código gerado por G_B^* , a fim de encontrar as palavras códigos mais próximas. Desta forma, conhecerá: $u \cdot G_B^*$. Para descobrir a mensagem u , poderá usar o processo de eliminação de Gauss². Esta abordagem envolve $2^k \cong 2^{523} \cong 10^{158}$ comparações.

Ataque 3: Seleciona k posições, supondo a inexistência de erros nas mesmas, isto é, espera que e tenha zero coordenadas nestas k posições. Se estas k posições forem independentes, u poderá ser encontrado pelo processo de eliminação de Gauss. Caso contrário, há grande possibilidade que as k colunas em G_B^* correspondentes a estas k posições, tenham Rank² igual a k . Assim o processo de eliminação de Gauss terá mínimas possibilidades para u . Somente uma dessas possibilidades corresponderá a uma palavra código com distância mínima $\leq t$ de C . A probabilidade de que as k posições estejam corretas é aproximadamente $(1 - t/n)^k$. O processo de eliminação de Gauss envolve k^3 passos. Assim a carga de trabalho esperada é $k^3(1-t/n)^k \cong 2^{65} \cong 10^{19}$.

Considerando os ataques expostos e/ou o método de decodificação utilizado:

² Processo de Eliminação de Gauss e definição de Rank em [Ant, 1994]

- Para valores de n e k pequenos, percebe-se que a mensagem é cifrada/enviada e recebida de forma instantânea, porém o sigilo não é assegurado.
- Para valores de n e k grandes, é requerido um espaço de memória para armazenar a tabela de decodificação síndrome: $2^{n-k} / n$ síndromes e respectivos erros padrões. Considerando o código cíclico (127,57) como exemplo, tem-se uma tabela síndrome com $2^{70}/127$ “elementos”. Esta caracterização implica que não se conhece algoritmo capaz de resolver este problema em tempo de execução, dependendo exponencialmente do tamanho dos parâmetros de entrada. O tempo a ser gasto na decifragem será mínimo, mas o espaço para armazenamento da tabela será grande.
- Um dos meios com grande capacidade de armazenamento de dados, por baixo custo, é o Disco Versátil Digital (DVD). Sua razão de transferência de dados limita-se em 10.08 Mbps. DVD – RAM : Capacidade máxima de armazenamento é de 5.16 Gb / DVD –ROM : Capacidade máxima de armazenamento é de 17 Gb.
- Independente do tamanho de n e k , o receptor tendo armazenado a tabela síndrome ordenada, terá que fazer uma busca binária para decifrar a mensagem recebida, para a qual o tempo mínimo será $-\log_2^{2^{n-k}}$, porém tem-se como inconveniente o armazenamento da tabela.
- O produto matricial é polinomial e é executado de forma rápida.

O capítulo II expõe uma forma de gerar assinatura digital através de um criptosistema de chave pública. Embora o sistema produzido seja de chave pública, não

suporta a assinatura digital, pois o método de cifragem faz uma correspondência de k -tuplas em n -tuplas binárias. Esta função não é uma bijeção, ou seja, não há uma correspondência biunívoca entre texto principal e texto cifrado. No caso do sistema criptográfico anterior, tem-se duas situações possíveis:

1ª – Suponha uma mensagem u de comprimento k bits, conforme figura 6.1, que representa a assinatura de um indivíduo A que deseja enviá-la para o indivíduo B . A assinatura u é cifrada por E_B , obtendo $E_B u$, que é uma palavra de comprimento n .

Em seguida o indivíduo A , calcula $D_A E_B u$ e envia para B . Observe que a palavra enviada por A tem k bits.

Ao receber a palavra $D_A E_B u$ o indivíduo B , utiliza a chave pública, E_A para calcular $E_A D_A E_B u$. Entretanto, observa-se que $E_A D_A E_B u$ não necessariamente é igual a $E_B u$, pois o erro introduzido a cada aplicação de E_A pode ser diferente. Desta forma, ao aplicar E_A à palavra $D_A E_B u$ tem-se um conjunto de possibilidades para o resultado, sabendo-se apenas que este conjunto contém $E_B u$.

Finalmente, ao se calcular $D_B E_A D_A E_B u$ pode-se obter $u' \neq u$.

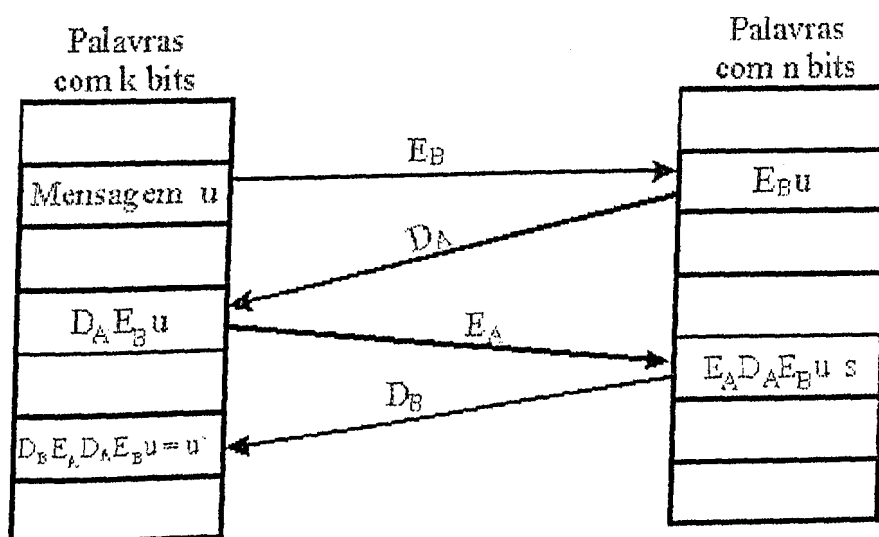


Figura 6.1 – Tentativa de assinatura digital utilizando palavras com k bits.

2^a – Suponha que a assinatura u do indivíduo \underline{A} contém n bits, conforme figura 6.2. Inicialmente, calcula-se $D_A u$, que é uma palavra de comprimento k . Em seguida repete-se o processo da 1^a situação, ou seja, calcula-se $E_A D_A u$ e envia tal palavra para o indivíduo \underline{B} . Ao recebe-la, \underline{B} calcula: $E_A D_B E_B D_A u$.

Neste ponto, como na 1^a situação, tem-se que $E_A D_B E_B D_A u$ não necessariamente é igual a u , pois o erro introduzido por E_A é aleatório.

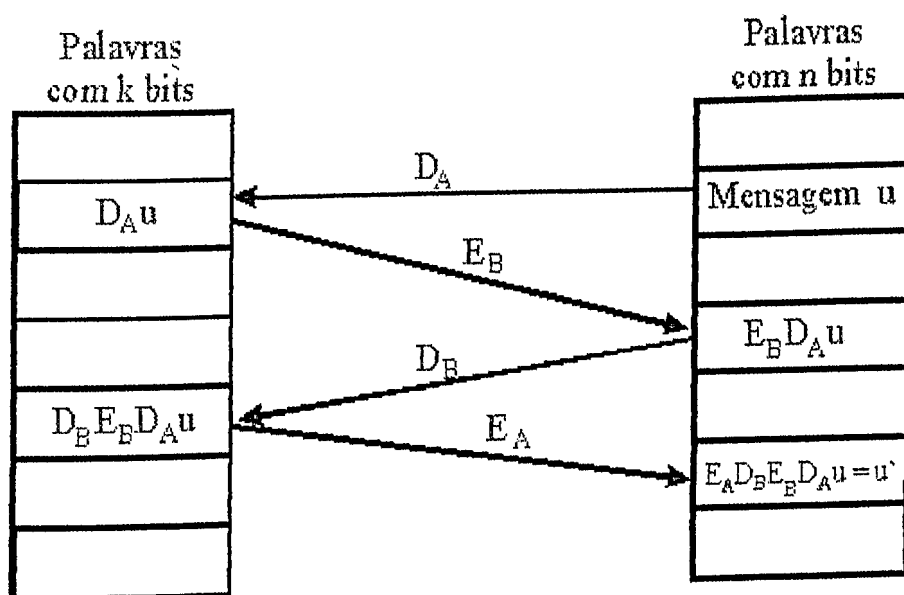


Figura 6.2 – Tentativa de assinatura digital utilizando palavras com n bits.

6.5 – CONCLUSÃO

Para propor o sistema criptográfico foram necessárias considerações gerais de chave pública aliadas ao conceito de códigos cíclicos.

Observa-se que no ataque 1, que o problema intratável neste sistema é a obtenção das matrizes S e P a partir de G^* , considerando n e k grandes.

O sistema produzido, com valores de n e k pequenos, apresenta-se computacionalmente rápidos, no entanto, sua segurança torna-se frágil. À medida em que

os valores de n e k crescem, requer por parte do receptor o armazenamento da tabela de decodificação cada vez maior, enquanto n e k crescem linearmente, a tabela de decodificação cresce exponencialmente.

Tal sistema apresentou um outro ponto fraco: a autenticação de usuários, não suporta a assinatura digital.

Surgem questões essenciais relacionadas ao método de decodificação: Para uma tabela com 2^{800} posições, que tipo de dispositivo é capaz de armazená-la? Para consultar um "elemento" na tabela, qual é o tempo necessário? Este tipo de implementação não é considerada neste trabalho, sendo tema de pesquisa futura.

CAPÍTULO VII

CONCLUSÕES

Foram apresentados dois sistemas criptográficos clássicos e um de chave pública usando códigos lineares e cíclicos, respectivamente.

Considerando os sistemas criptográficos DES e o RSA, verifica-se que:

- O DES evita a linearidade das substituições do algoritmo - ciframentos lineares são extremamente frágeis e sucumbem rapidamente.
- A alternativa para a criptoanálise do DES é a força bruta. Se a chave possui 56 bits, existem 2^{56} chaves diferentes e, portanto, são necessários $2^{56} = 10^{17}$ ciframentos para determinar a chave.
- O sistema clássico produzido somente com códigos lineares possui uma fragilidade maior em relação ao DES.
- Na criptoanálise do sistema clássico composto por códigos lineares e o DES, a criptoanálise força bruta é ainda mais difícil do que testar as 2^{56} chaves diferentes. Portanto este sistema é mais seguro em relação ao DES simplesmente, mas requer um número maior de chaves secretas.
- Na criptografia de chave pública, a cifragem é feita através de uma função unidirecional com segredo, mediante um algoritmo de cifragem público e uma chave de cifragem pública. O segredo, conhecido somente pelo destinatário, é a

chave de decifragem. Para se conseguir a unidirecionalidade desejada com segredo, lança-se mão de um problema computacionalmente intratável, que geralmente o destinatário sabe resolver. No caso do sistema RSA, o problema intratável é a fatoração de inteiros, o destinatário calcula o produto de dois primos e publica-o. Para decifrar mensagens, a fatoração desse produto é necessária, e somente o destinatário a conhece.

- No sistema RSA a criptografia de chave pública não funciona de forma adequada quando os espaços para os textos principais forem pequenos. Um criptoanalista pode construir, para o estágio de pré-processamento, uma tabela de decifragem completa simplesmente pela cifragem de todos os possíveis textos principais, alterando o criptotexto resultante em uma ordem alfabética conveniente.
- No sistema proposto de chave pública, o problema intratável é obter S e P a partir de G^* . Para valores de n e k grandes, existe uma quantidade astronômica de diferentes matrizes S e P . Este sistema, com valores de n e k pequenos, apresenta-se computacionalmente rápidos, no entanto sua segurança é frágil.
- Para valores de n e k grandes é requerido por parte do receptor um armazenamento grande da tabela de decodificação síndrome.
- Um outro problema apresentado por este sistema é que o mesmo não suporta a assinatura digital, devido o método de cifragem fazer um mapa de k -tuplas em n -tuplas binárias. Este mapa não é uma bijeção, ou seja, não há uma correspondência biunívoca entre texto principal e texto cifrado.

As “propostas”, embora tenham revelado pequenos ganhos, trouxe uma enorme bagagem a nível teórico e embasamento para análise crítica e reflexiva dos assuntos abordados, bem como despertou interesse para os inventos atuais e possíveis nesta área do conhecimento.

O presente trabalho possibilita o desenvolvimento de alguns trabalhos futuros, tais como:

- Análise quantitativa da criptoanálise dos sistemas propostos;
- Implementação de sistemas criptográficos mais rápidos que possam ser utilizados em telefonia – criptografia em tempo real; redes de computadores.

REFERÊNCIAS BIBLIOGRAFIA

- [Ant,1994]. ANTON, HOWARD, Elementary Linear Algebra. John Wiley & Sons, New York, 7th Ed., 1994.
- [Bek,1982]. BEKER, H.& PIPER, F., Cipher Systems. Northwood Books, London 1982
- [Bia,1983]. BIALUT, RICHARD E., Theory and Practice of Error Control Codes. Addison-Wesley, 1983.
- [Boy,1995].BOYD, MANCHESTER C., Cryptography and Coding. Fifth IMA Conference, LNCS, vol. 1025, December 1995
- [Cun,1996] CUNHA, MARIA ISABEL DE, O Bom Professor e Sua Prática. 6^a ed.-Campinas, SP: Papyrus, 1996.
- [Den,1982].DENNING, DOROTHY E., (DOROTHY ELIZABETH), Cryptography and Dara Security. Addison-Wesley, 1982.
- [Dif,1976]. DIFFIE, W. & HELLMAN M., New Direections in Cryptography. IEEE Transactions on Information Theory IT-22 (1976) 644-654

[Hol,1992]. HOLLAND, NORTH, The Theory of Error-Correcting Codes. Macwillians, Florence Jessie, 1992.

[Kob,1987]. KOBLITZ, N., A Course in Number Theory and Cryptography. Springer, Berlin Heidelberg New York, 1987

[Kon,1981]. KONHEIM, A. G., Cryptography: A Primer, Wiley, 1981.

[Lin,1983]. LIN, S. & COSTELLO, D., Error Control Coding Fundamentals and Applications. Prentice-Hall, Inc. Englewood Cliffs, New Jersey 1983.

[Luc,1986]. LUCCHESI, CLÁUDIO LEONARDO, Introdução à Criptografia Computacional, Campinas: Editora da UNICAMP/Editora Papiros, 1986.

[Mer,1981]. MERKLE, R. C. AND HELLMAN, M. E., "On the Security of Multiple Encryption," Comm. ACM Vol. 27 pp. 465 – 467 (July 1981).

[Mic,1985]. MICHELSON, ARNOLD M., Error-Control Techniques for Digital Communication. GTE Government Systems Corporation Needham Heights, Massachusetts, 1985.

[Nob,1986]. NOBLE, BEM & DANIEL, JAMES W., Applied Linear Algebra. Prentice-Hall, New Jersey, 1986.

- [Pra,1957]. PRANGE, E., "Cyclic Error-Correcting Codes in Two Symbols," AFCRC-TN-57, 103, Air Force Cambridge Research Center, Cambridge, Mass., Sptember 1957.
- [Sal,1990]. SALOMAA, ARTO, Public-Key Cryptography (EATCS monographs on theoretical computer science; v.23), 1990
- [Sch,1994]. SCHNEIER, BRUCE., Applied Cryptography: Protocols, algorithms and source code in C, New York: John Wiley, 1994.
- [Sha,1949]. SHANNON, C. E., Comunication Theory of Secrecy Systems. Bell System Technical Journal 28 (1949) 656-715
- [Van,1988]. VAN TILBORG, HENK C. A., An Introduction to Cryptology. Kluwer International Series in Engineering and computer science, Hardcover, 1988.
- [Wic,1995]. WICKER, STEPHEN B., Error Control Systems for Digital Commnication and Storage. Prentice-Hall, New Jersey 1995.