UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE COMUNICAÇÕES

UMA PROPOSIÇÃO E ANÁLISE DE COMPLEXIDADE DE UM CRIPTOSISTEMA DE CHAVE PÚBLICA UTILIZANDO CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA

Este exemplar corresponde à redação final da tese defendida por Dalila Haickel
e aprovada pela Comissão Julgadora em 11/04/91.
Reginaldo Palazzo Jr.
Orientador

AUTORA: Dalila Haickel
ORIENTADOR: Reginaldo Palazzo Jr.

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

Campinas - SP
abril 1991

in memorian de Antônio Haickel Neto

Para a minha mãe, Muriel,
para meu irmão, Nando e
para Gorgonio.

AGRADECIMENTOS

Ao prof. Dr. Reginaldo Palazzo Jr., pela sua orientação, confiança e apoio para a realização deste tabalho.

Ao prof. Dr. Paulo Cesar Bezerra, pelo apoio, amizade e discussões técnicas, que deram suporte ao crescimento deste trabalho.

A banca examinadora, composta pelos membros: prof. Dr. Normonds Allens da Escola Politécnica, prof. Dr. Paulo Cesar Bezerra da UNICAMP e prof. Dr. Celso Almeida da UNICAMP.

A CAPES pelo suporte financeiro dado durante o mestrado.

A minha mãe e meu irmão, por terem sido pacientes, compreensivos, amigos e por terem dedicado-me tanto amor e apoio, mas principalmente por sempre estarem ao meu lado quando deles precisei.

Ao Gorgonio pelo seu amor , companheirismo e pela ajuda na confecção final desta dissertação.

A amiga Kenia pelas suas inúmeras demontrações de amizade.

Ao Dinho pela amizade e contribuições dadas a este trabalho.

Ao Aguilera pela sua ajuda para a obtenção dos resultados finais.

A Lúcia pelos desenhos desta dissertação.

A Ademilde pela ajuda na digitação do trabalho.

E a todos que direta ou indiretamente contribuiram de alguma forma para a realização deste trabalho.

RESUMO

A Proposição de um Criptosistema de Chave Pública utilizando Códigos de Memória Unitária neste trabalho, tem por objetivo básico a análise de complexidade e verificar o comportamento das funções armadilhas utilizadas no modelo, para que seja alcançado um alto grau de privacidade na informação a ser transmitida.

Neste estudo são considerados como fatores do processo para a obtenção dos dados de análise: 1) complexidade para encontrar os códigos ótimos de memória unitária; 2) desempenho das funções armadilhas e 3) eficiência de processamento do algoritmo.

No decorrer da apresentação do modelo são feitas as considerações necessárias para que se chegue ao resultado de complexidade do sistema e sua eficiência de operação, computando-se a relação custo/benefício.

ABSTRACT

In this work we present a scheme based on a public-key cryptosystem employing unit-memory codes. The main goal is the analysis of a model of the trapdoor function being proposed in order to guarantee a desired privacy during information transmission.

We consider in this work three basic points: 1) the complexity of finding good unit-memory codes; 2) the performance achieved by use of the trapdoor functions; and 3) the complexity and efficiency of the algortihm.

Finally in this work present the necessary considerations to obtain the system complexity result and its operation efficiency, and also the trade-off between rate and costs.

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO

A necessidade de serem enviadas informações entre dois ou mais pontos, sem que as mesmas sejam interceptadas ou alteradas entre os pontos de envio e recepção, deu origem à Criptografia. Fundamentalmente, a criptografia consiste da proposição dos métodos que tornam o conteúdo das mensagens ininteligível à pessoas não autorizadas ao mesmo tempo permitindo que os destinatários recuperem a mensagem original.

A Criptografia era conhecida a quatro mil anos pelos antigos Egípcios. A palavra Criptografia de origem grega (kriptos = secreta e grafos = escrita) significa escrita secreta. Júlio César, fazia uso da Criptografia para troca de mensagens com as tropas, nas campanhas do exército Romano. Durante a Idade Média, o interesse pela Criptografia diminuiu, bem como em muitas outras áreas de atividades intelectuais.

Com a vinda do Renascimento Italiano a arte da Criptografia novamente floresceu. Na época de Luis XIV da França,um código baseado em 587 chaves aleatoriamente selecionadas era usado em mensagens governamentais. Por volta de 1800, dois fatores ajudaram no desenvolvimento da Criptografia. Primeiro foram as estórias de Edgar Allan Poe, tais como "The Gold Bug", que apresentava mensagens em código, excitando assim a imaginação dos leitores. O segundo foi a invenção do telegráfo e do código Morse.

O código Morse foi a primeira representação binária (traço e ponto) do alfabeto que teve grande aplicação.

Ao chegar a I Guerra Mundial, várias nações construiram ''Máquinas de Codificação'' mecânicas que permitiam facilmente codificar e decodificar textos usando cifragens sofisticadas e complexas. Aqui, surge a decifragem de códigos advinda da

Criptografia Eletrônica. Antes da utilização de dispositivos mecânicos, codificar e decodificar mensagens, cifragens complexas eram raramente usadas devido ao tempo e o trabalho necessários para o processo de codificação e decodificação. Assim, a maioria dos códigos eram decifrados num período relativamente curto de tempo.

Durante a II Guerra Mundial uma máquina para cifrar (máquina de Hagelin/M-209) foi extensivamente usada pelo exército dos Estados Unidos. O livro ''The Codebreaks, The Story of Secret Writing'',[3] é o estudo mais completo sobre a Criptografia praticada desde os primórdios até a II Guerra Mundial.

Até recentemente a Criptografia era utilizada exclusivamente pelos serviços militares e comunicações diplomáticas. Entretanto, com o grande desenvolvimento ocorrido em setores comerciais, industriais, negociações bancárias, etc., o interesse pela Criptografia foi se tornando cada vez maior, com o objetivo de proteger os interesses de cada parte envolvida na comunicação.

Com o advento dos computadores - especialmente os multiusuários - códigos inquebráveis e seguros têm se tornado ainda mais necessários. Não só arquivos necessariamente precisam ser mantidos em sigilo, mas o acesso ao computador em si deve ser também gerenciado e regulado.

O potencial de ataque dos computadores aos criptosistemas tem estimulado o aparecimento de novas técnicas de cifragem e decifragem, bem como uma nova conceituação de chave.

Na Criptografia, existem dois métodos pelos quais podem ser feita a cifragem e a decifragem através dos criptosistemas, são eles: o método convencional e o método por chaves públicas.

No sistema convencional, dois usuários que desejam se comunicar devem pré-estabelecer uma chave comum através de um canal seguro, entendendo-se por canal o meio físico que interliga os dois pontos, de envio e recepção da mensagem.

Em um sistema com "N" usuários este processo de cifragem torna-se restritivo diante desta segurança. Esta restrição deve-se ao estabelecimento de conexões seguras, aonde o número de chaves

criptográficas necessárias é de no máximo N(N-1)/2.

O estabelecimento desta quantidade de chaves, pode ser inviável para um sistema com 100 mil usuários, pois, para o caso da existência de ligações entre todos os usuários o número total de chaves seria de 5 bilhões.

A idéia básica da utilização de chave pública é a eliminação da necessidade de utilização de um canal eficiente e de alta segurança para a distribuição de chaves.

Do ponto de vista de implementação, os sistemas que usam chaves públicas se fundamentam na existência de métodos diferentes para a cifragem e decifragem. O primeiro modelo de chaves públicas na Criptografia foi apresentado por Diffie e Hellman em 1976 [1].

Existem duas formas de implementar um sistema do tipo proposto por Diffie e Hellman. Na primeira as duas partes envolvidas na comunicação trocam chaves entre si, através do canal, que permitem aos dois e apenas aos dois, estabelecerem uma chave comum que será usada para cifrar e decifrar.

Na segunda forma duas chaves são usadas, uma para cifrar e outra para decifrar. A chave de cifragem é enviada publicamente ao usuário que deseja transmitir uma mensagem cifrada. A chave de decifragem mantida secreta, permite ao usuário receptor da mensagem cifrada e apenas a ele, decifrar a mensagem.

Os sistemas que utilizam a primeira técnica são denominados Sistemas com Distribuição Pública de Chaves. Conforme já mencionado, a chave única para cifrar e decifrar é conhecida apenas pelos dois usuários, ainda que as informações para o estabelecimento desta chave tenham sido trocadas publicamente através de um canal inseguro. Os sistemas que utilizam a segunda técnica, são denominados Sistemas de Chaves Públicas. Nestes sistemas, duas chaves são utilizadas para cada usuário, uma pública - de cifragem, outra secreta - de decifragem.

Basicamente os dois sistemas citados acima diferem muito pouco em fundamentos matemáticos que garantem a sua segurança.

Os sistemas criptográficos de chaves públicas utilizam

métodos diferentes de implementação, são eles: o sistema RSA e o sistema Knapsack.

Neste trabalho é feita uma proposta de um criptosistema de chaves públicas (CSCP), utilizando o método do knapsack.

Este sistema é baseado em códigos corretores de erros, mais especificamente os códigos convolucionais de memória unitária, aonde busca-se explorar todo o seu grau de complexidade inerente ao processo de buscas de códigos ótimos e ao processo de decodificação.

A análise de complexidade, é feita mediante os resultados de tempos obtidos da implementação, que foi realizada em máquinas diferentes, mostrando a dependência entre o processo e o equipamento utilizado.

A análise de complexidade também considera o grau de dificuldade inserido pela função armadilha e portanto busca um fundamento consistente para a proposta de trabalho aqui apresentada.

São utilizados cinco códigos convolucionais de memória unitária, com taxas: R=2/3, R=2/4, R=2/5, R=7/14 e R=8/24, que são os códigos ótimos.

A organização básica dos Capítulos que compõem o escopo deste trabalho está dividida como segue:

No Capítulo 2, é feita uma revisão sobre códigos corretores de erros dos tipos de bloco e convolucional. Também é feita uma revisão sobre os métodos criptográficos existentes desde os convencionais até os de chaves públicas.

No Capítulo 3, é descrita a proposta do sistema "Knapsack" CSCP, analisando-se no procedimento o grau de dificuldade das variáveis envolvidas no processo de implementação. Também é feita uma análise detalhada das funções armadilhas, o grau de dificuldade para quebra do sistema e a complexidade computacional envolvida.

No Capítulo 4, serão apresentados os resultados encontrados. Apresentamos também para cada um dos códigos de memória unitária aqui utilizado a análise de dificuldade de quebra, bem

como a finalidade a que devem ser usados cada um destes códigos.

Finalmente no Capítulo 5, são feitas as considerações finais, e sugestões de sistemas baseados na proposta aqui apresentada.

CAPÍTULO 2

REVISÃO DE CÓDIGO CORRETOR DE ERROS E SISTEMAS CRIPTOGRÁFICOS

2.1 - INTRODUÇÃO

Neste capítulo será feita uma revisão sobre códigos corretores de erros, dando-se ênfase aos códigos convolucionais de memória unitária. Desde que este assunto é bastante abrangente, os conceitos serão apresentados de forma sucinta e objetiva à finalidade deste trabalho.

Nesta mesma linha é feita uma revisão dos sistemas criptográficos, desde os sistemas convencionais até os sistemas de chaves públicas, onde procuramos mostrar as vantagens deste último em relação ao primeiro, através de uma análise comparativa, enfatizando-se o porque da sua preferência na criptografia.

2.2 - CÓDIGO CORRETOR DE ERROS

O problema da confiabilidade na transmissão de informações, vem representando um desafio constante para profissionais que atuam nesta área. Esta confiabilidade é referente à integridade da informação através do meio de transmissão à ação do ruído.

Na Figura 2.2.1 é apresentado um esquema genérico de um sistema de comunicações, cuja descrição de cada um dos blocos é feita a seguir:

Fonte - Gerador da informação a ser transmitida.

Transmissor - Transforma o sinal na saída da fonte na forma adequada para a transmissão através do canal. A função do transmissor é dividida em três etapas distintas, apresentadas a seguir:

1 - Codificador de Fonte - Este bloco consiste basicamente de um conversor analógico/digital e de um codificador, podendo este ser de bloco ou de treliça. Em aplicações mais complicadas realiza a função de remover dados desnecessários à informação, como exemplo, no caso de tratamento de imagem.

2 - Codificador de canal - É neste bloco que são adicionados dígitos de redundância ao sinal, com a finalidade de reduzir os efeitos do ruído sobre o sinal original.

3 - Modulador - Translada o sinal de saída do codificador da fonte para uma forma de onda adequada para a transmissão através do canal.

Canal -É o meio físico pelo qual a informação passa do transmissor para o receptor. Ao passar pelo canal a informação está sujeita à ações de pertubações indesejáveis e diversificadas, denominadas de ruído. A ação desta pertubação sobre o sinal que contém a informação ocasiona uma degeneração no conteúdo da mensagem, antes que esta chegue ao seu destino.

Receptor - Recebe do canal a forma de onda transmitida, possivelmente degenerada pela ação do ruído. A função do receptor consiste então em recuperar da forma mais fiel possível o sinal original, através das seguintes etapas:

1 - Demodulador - A partir da forma de onda recebida do canal o demodulador estima a forma de onda que foi enviada pelo transmissor e entrega na saída a versão digital correspondente. Esta função consiste em fazer a operação inversa do modulador,

sendo que neste caso o sinal que passa ao estágio seguinte possivelmente estará contaminado com ruído.

2 - Decodificador de Canal - Neste estágio o decodificador de canal tenta corrigir possíveis erros e então produz uma estimativa dos dígitos de saída passando esta informação ao decodificador de fonte.

3 - Decodificador de Fonte - Nesta fase é recolocada a redundância do sinal, removida no transmissor, antes que este seja entregue ao destinatário.

4 - Destinatário - É o ponto final onde deve chegar a informação transmitida.

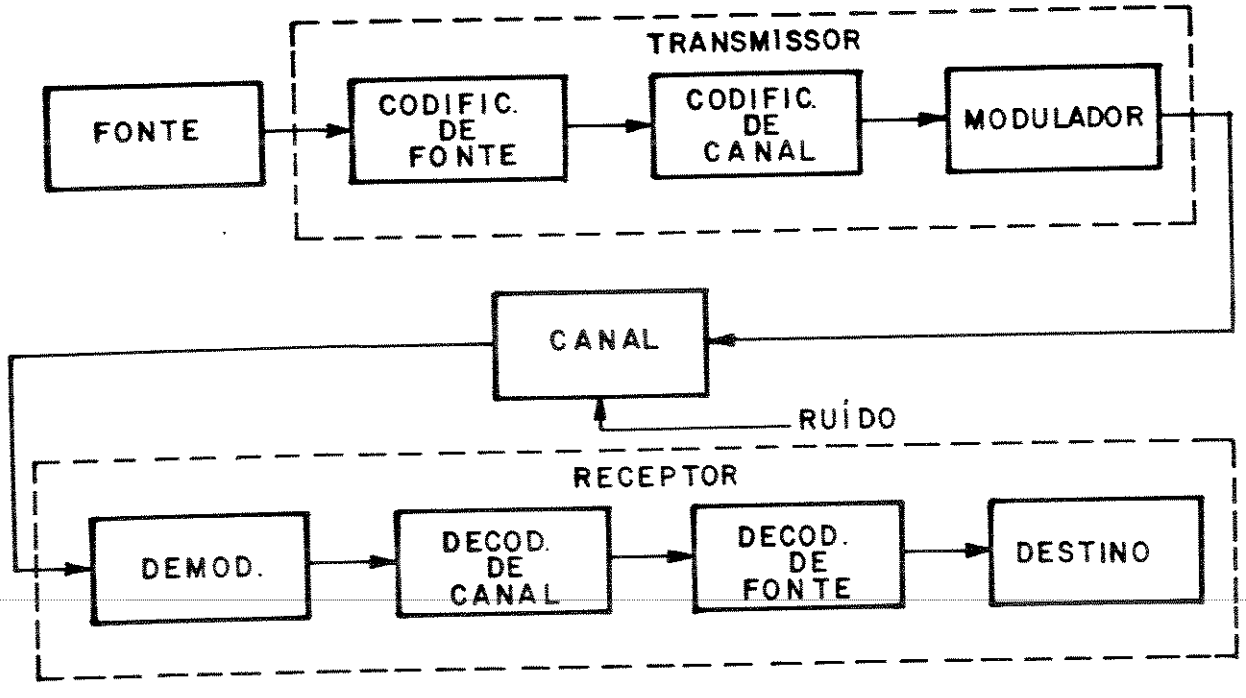

Fig. 2.2.1 - Sistema Genérico de Comunicação

Nesta breve exposição de um sistema genérico de comunicação, observa-se que o problema central está na confiabilidade do sinal recebido em relação ao sinal transmitido.

O sinal quando passa pelo meio de transmissão pode ser pertubado por dois tipos de ruído: os esporádicos e independentes, gerando os chamados erros aleatórios ou em surtos de vários erros de cada vez neste caso diz-se que o canal tem memória.

Com a finalidade de diminuir ou até mesmo eliminar os erros ocorridos atráves do canal é que são utilizados códigos corretores de erros. A potencialidade máxima dos códigos corretores de erros, foi estabelecida por Shannon em 1948.

Neste teorema Shannon estabelece que todo canal tem uma capacidade C, e para qualquer taxa de transmissão R < C, existem códigos de taxa R, que têm uma probabilidade de decodificação errônea arbitrariamente pequena. Isto significa que para qualquer R <= C porém fixo e o comprimento n tão grande quanto se possa desejar, existe um código tal que a probabilidade de decodificação errônea P_c é dada por:

$$P_c <= \exp(-nE(R)); \quad E(R) > 0 \quad \text{para } R < C \qquad (2.2.1)$$

onde, E(R) é a função confiabilidade, especificada pelas probabilidades apriori e de transição do canal [4].

Este teorema mostra que é possível diminuir a probabilidade de decodificação errônea através do aumento de n para uma taxa R constante. Então, com o objetivo de diminuir os efeitos do ruído na mensagem através do canal a teoria de codificação tem duas finalidades básicas:

1 - Encontrar códigos longos e eficientes;

2 - Encontrar métodos práticos e eficientes de codificação e decodificação.

A teoria de codificação está dividida em duas classes a saber:a classe dos códigos de bloco e a classe dos códigos de árvore. Cada uma destas classes está sudividida em subclasses de códigos lineares e não lineares.

A classe que abordar-se -á neste trabalho é essencialmente a de códigos corretor de erros pertencentes a classe dos códigos de árvore lineares tendo os códigos convolucionais como a classe de maior importância. Aos códigos pertencentes à esta classe dar-se-á ênfase à estrutura dos códigos convolucionais de memória unitária.

2.2.1 - CÓDIGOS DE ÁRVORE E DE TRELIÇA

Um código convolucional, como citado anteriormente, é uma sub-classe dos códigos de árvore. Na intenção de formalização de conceitos e fundamentos desta classe de códigos, inicialmente será feita uma abordagem dos códigos de árvore e de sua sub-classe que são os códigos de treliça.

Uma árvore é um grafo bastante importante para a descrição dos códigos que se seguem. Em teoria dos grafos, uma árvore pode ser descrita através dos seguintes parâmetros:

$$A = (D,M) \qquad (2.2.1.1)$$

onde, D especifica a profundidade com μ ramos divergindo de cada vértice e M é a profundidade onde somente um ramo sai de cada vértice.

Uma árvore pode ser descrita como sendo um grafo (D,M) μ - ário em que:

1 - μ ramos divergem de cada vértice até uma profundidade D do vértice inicial;

2 - somente um ramo diverge de cada vértice com profundidade maior ou igual a D, porém menor ou igual a D+M, onde D, M, e μ são inteiros tal que $D \geq 1$, $M \geq 0$ e $\mu \geq 2$.

Os códigos de árvore podem ser descritos mais genericamente através de uma quártupla dada por (n,R,D,M), onde R representa a taxa do código e n é o comprimento da palavra.

Tomando $\mu = 2^{nR}$ que é o número que diverge de cada vértice até a profundidade D, e $\gamma = (M+1).n$ que representa o número total de dígitos desde o vértice D - 1 até D+M, sendo denominado de comprimento de restrição da árvore.

Sendo assim, podemos obter a taxa deste código como segue:

$$\tilde{R} = \frac{\log_2(\mu^D)}{(D+M).n} = \frac{D \log_2 \mu}{(D+M).n} \qquad (2.2.1.2)$$

onde substituindo o valor de μ em (2.2.1.2) temos que a taxa $\tilde{R}$ é dada por :

$$\tilde{R} = \left[\frac{D}{D+M} \right] . R \qquad (2.2.1.3)$$

Sabendo-se que para a maioria dos casos D>>M então $\tilde{R} = R$.

Dentre as classes de códigos de árvore encontrar-se-á a dos códigos de treliça.

Esta classe de códigos pertence à classe dos códigos de árvores pela introdução da dependência entre os símbolos da fonte no processo de codificação.

Esta introdução de dependência gera a necessidade de se estabelecer um procedimento de codificação para o código de árvore.

Desta forma, seja $\{u_i\}_{i=0}^{D-1}$ uma sequência de dígitos de informação. À cada dígito de informação u_0, u_1,..., associamos um

ramo na árvore que diverge de cada vértice até a profundidade D. Desta associação tem-se que u_o, u_1, u_2, ..., u_{D-1}, formam um caminho na árvore

Se a cada vértice desta árvore for associado um inteiro m, entre M e D+M inclusive, rotulemos cada vértice da árvore (D,M) μ-ária com m dígitos de informação anteriores, também suponha que somente o dígito "0" será associado a cada vértice entre D e D+M. Desta forma, fica estabelecido o esquema de codificação de um código de árvore (D,M,m) μ-ária onde os vértices são enumerados.

Se esta árvore tem memória m, de modo que para quaisquer dois vértices na mesma profundidade resultem na mesma sequência codificada restante quando a mesma sequência de dígitos de informação é aplicada à entrada do codificador a partir de quaisquer um desses vértices, então precisaremos manter somente uma dessas sequências. Quando isto ocorre, temos o que é chamado de uma treliça (D,n,m) μ-ária.

Como (n,R,D,M,m) é uma classe especial de (n,R,D,M) o comprimento de restrição γ continua sendo válido, o mesmo acontecendo com a taxa do código.

Podemos então formalizar a representação de códigos de treliça, através de uma quíntupla (n,R,D,M,m), para um canal discreto sem memória.

2.2.2 - CÓDIGOS CONVOLUCIONAIS

Seja $\{u_i\}_{i=0}^{D-1}$ a sequência dos dígitos de informação a ser codificada. Seja $\{v_i\}_{i=0}^{D+M-1}$ a sequência codificada correspondente à sequência de informação $\{u_i\}_{i=0}^{D+M-1}$.

Assim a representação matemática do codificador, para o modelo considerado é dada por:

$$v_i = \Gamma(u_i, u_{i-1}, \ldots, u_{i-m}, i) \qquad (2.2.2.1)$$

onde $\Gamma(.)$ é uma função arbitrária dos valores de u_i. Sendo uma função arbitrária $\Gamma(.)$ pode ser uma função linear ou não linear.

Os códigos convolucionais formam uma classe especial da classe dos códigos de treliça, quando $\Gamma(.)$ é uma função linear.

Para que seja definida uma estrutura matricial para o código convolucional, suponha um código sobre o corpo de Galois de q elementos GF(q). A sequência de informação u_i forma uma D-upla sobre GF(q).

O codificador convolucional (n,R,D,m) cuja sequência codificada v_i é dada por: $v_i = \Gamma(u_i, u_{i-1}, \ldots, u_{i-m}, i)$ é tal que :

$$u_i \in GF(q)^L$$

e

$$v_i \in GF(q)^n$$

e $\Gamma(.,i)$ são funções lineares que podem ser representadas por:

$$v_i = u_i G_0(i) + u_{i-1} G_1(i) + \ldots + u_{i-m} G_m(i) \qquad (2.2.2.2)$$

onde $G_j(i)$ são matrizes L x n sobre GF(q). Para o caso do codificador ser invariante no tempo temos que: $\Gamma(.,i) = \Gamma(.)$, e portanto a equação (2.2.2.2) torna-se:

$$v_i = u_i G_0 + u_{i-1} G_1 + \ldots + u_{i-m} G_m \qquad (2.2.2.3)$$

com G_j matrizes L x n sobre GF(q).

Observando as equações (2.2.2.2) e (2.2.2.3) notamos que a sequência de entrada faz uma convolução com as matrizes G_j, e portanto justificando assim o nome convolucional.

Para que possamos conseguir uma estrutura matricial para os códigos convolucionais, para o caso particular de invariante no tempo, basta que a sequência de informação e a sequência codificada sejam similares, e portanto a matriz geradora G do código convolucional invariante no tempo é dada por:

$$v_{[0,D+M]} = u_{[0,D]} \cdot \begin{bmatrix} G_0 & G_1 & \cdots & G_m & & \\ & G_0 & \cdots & G_{m-1} & G_m & \\ & & \ddots & & & \\ & & & G_0 & G_1 & \cdots \\ & & & & G_0 & \cdots & G_M \end{bmatrix}$$

(2.2.2.4)

2.2.2.1 - MÉTODO DE OBTENÇÃO DOS CÓDIGOS CONVOLUCIONAIS

Seja um código convolucional caracterizado pelos seguintes parâmentros: C = (n,k,m); onde:

n é o comprimento da palavra-código ramo na saída do codificador, na unidade de tempo T.

k é o comprimento da entrada na unidade de tempo T, isto é, corresponde ao comprimento dos bits da sequência de informação.

m número de memória do codificador convolucional.

Para a geração e codificação dos códigos convolucionais, considere como exemplo aplicativo o codificador como mostrado na Figura 2.2.2.1.1:

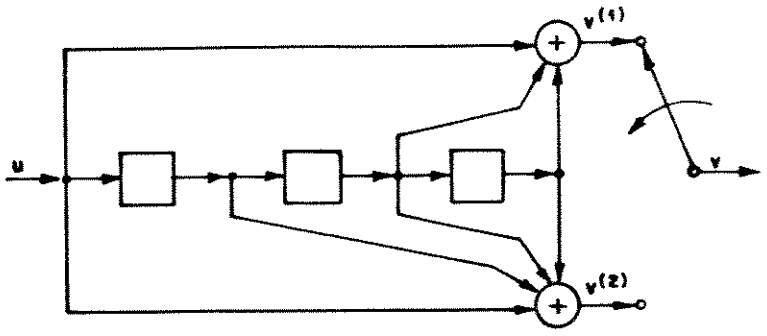

Figura 2.2.2.1.1 - Codificador Convolucional Binário (2,1,3).

Este codificador opera sobre o corpo de Galois de dois elementos, GF(2).

Seja a sequência de informação dada por $u = (u_0, u_1, u_2, \ldots)$, na saída do codificador linear temos duas sequências codificadas, a saber:

$$v^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \ldots)$$

e

$$v^{(2)} = (v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, \ldots)$$

Estas sequências são obtidas através do processo de

convolução entre a entrada u e a resposta impulsiva do sistema. Através da resposta ao impulso obtemos a função de transferência do sistema. Isto é equivalente a obtermos as sequências geradoras deste sistema, que no caso do exemplo apresentado são dadas por:

$$g = (g_o^{(1)}, g_1^{(1)}, \ldots, g_m^{(1)})$$

e

$$g = (g_o^{(2)}, g_1^{(2)}, \ldots, g_m^{(2)})$$

Sabemos que as sequências codificadas de saída representam a convolução entre os vetores u e g, temos então que v é dada por:

$$v^{(1)} = u \cdot g^{(1)} \qquad (2.2.2.1.1)$$

e

$$v^{(2)} = u \cdot g^{(2)} \qquad (2.2.2.1.2)$$

De uma forma geral podemos escrever que:

$$v_i^{(j)} = \sum u_{l-i}\, g_i^{(j)} \qquad (2.2.2.1.3)$$

sendo o sistema não antecipativo os valores de u_i são nulos quando $l < i$.

Depois da codificação as duas sequências de saída são multiplexadas em uma única sequência, denominada de palavra código, que será a sequência transmitida através do canal. A palavra código na saída do codificador da Figura 2.2.2.1.1 é então dada por:

$$v = (v_o^{(1)}\ v_o^{(2)}, v_1^{(1)}\ v_1^{(2)}, v_2^{(1)}\ v_2^{(2)} \ldots)$$

e a matriz geradora é dada pelo entrelaçamento das sequências $g^{(1)}$ e $g^{(2)}$ arranjadas da forma que segue:

$$G = \begin{bmatrix} g_o^{(1)} g_o^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} \cdots g_m^{(1)} & g_m^{(2)} & & \\ & g_o^{(1)} g_o^{(2)} & g_1^{(1)} g_1^{(2)} \cdots g_{m-1}^{(1)} & g_{m-1}^{(2)} & g_m^{(1)} & g_m^{(2)} \\ & & g_o^{(1)} g_o^{(2)} \cdots g_{m-2}^{(1)} & g_{m-2}^{(2)} & g_{m-1}^{(1)} g_{m-1}^{(2)} & g_m^{(1)} g_m^{(2)} \end{bmatrix}$$

(2.2.2.1.4)

onde os espaços em brancos da matriz G são zeros. A partir da matriz acima podemos então obter uma equação de codificação que é dada por:

$$v = u \cdot G \qquad (2.2.2.1.5)$$

De uma forma geral, a matriz geradora de um código convolucional é dada pela representação matricial de (2.2.2.1.4). As sub-matrizes G_i têm dimensão k x n, e são compostas pelos vetores geradores do codificador, isto é, a resposta do sistema ao impulso.

Exemplo 2.2.2.1.1

Para o codificador da Figura 2.2.2.1.1 seja a sequência u a ser codificada dada por u = (1 0 1 1 1). As sequências de saídas são dadas por:

$$v = u \cdot g^{(1)} = (1\ 0\ 1\ 1\ 1) \cdot (1\ 0\ 1\ 1) = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 1)$$

e

$$v = u \cdot g^{(2)} = (1\ 0\ 1\ 1\ 1) \cdot (1\ 1\ 1\ 1) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 1)$$

respectivamente.

Note que as sequências geradoras do codificador são encontradas a partir da entrada u = (1 0 0 0 0...) e todos os deslocamentos do bit "1". A palavra código então é dada por:

$$v = (1\ 1,\ 0\ 1,\ 0\ 0,\ 0\ 1,\ 0\ 1,\ 0\ 1,\ 0\ 0,\ 1\ 1)$$

As sub-matrizes geradoras são dadas por:

$$G_0 = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \text{e} \qquad G_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \text{e} \qquad G_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

2.2.2.2 - PROPRIEDADES ESTRUTURAIS DOS CÓDIGOS CONVOLUCIONAIS

Desde que o código convolucional foi proposto por Elias em 1955, como uma alternativa para o códigos de blocos, várias foram as contribuições apresentadas. Entre estas contribuições tem-se: 1) a proposta de um esquema eficiente de decodificação sequencial para os códigos convolucionais feita por Wozencraft; e 2) em 1966 Viterbi propôs um algoritmo de decodificação, extremamente eficiente, porém, de complexidade dependente do número de memórias envolvidas, mais tarde reconhecido como sendo de máxima verossimilhança. Além destes muitos outros algoritmos e estruturas algébricas foram estabelecidas para que o método de codificação se tornasse sistemático.

Com a intenção de tornar sistemático e bem estruturado o processo de codificação, foi estabelecido que os códigos convolucionais podem ser vistos como uma máquina de estados finitos do tipo Mealey, onde a entrada e o estado definem a saída,

e portanto podendo ser descrito através de um diagrama de estados.

Assim, seja S_k a representação do estado k, e a saída do codificador v_k dada como uma função da sequência de entrada u_k, onde a função do estado é dada por: $S_k = (u_{k-1}, u_{k-2}, \ldots, u_{k-m})$ e a função de saída representada por $v_k = \Gamma(u_k, S_k)$. Assim a cada novo u_k resulta na transição do estado S_k para um novo estado com sua correspondente saída v_k.

Para o exemplo da Figura 2.2.2.1.1 de um codificador convolucional caracterizado por (2,1,3) o diagrama de estado é mostrado na Figura 2.2.2.2.1.

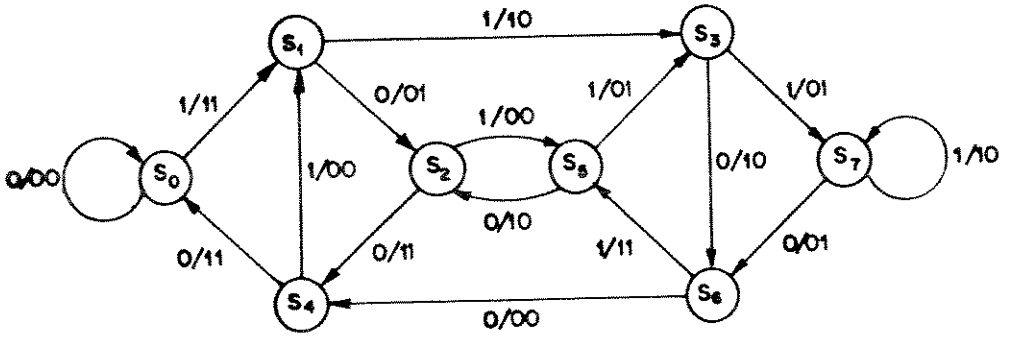

Figura 2.2.2.2.1 - Diagrama de Estado do Codificador Convolucional (2,1,3).

O diagrama de estado pode ser modificado para que possa fornecer toda a descrição dos pesos de Hamming das palavras códigos não nulas. Deste diagrama é possível obtermos a função enumeradora ou a função de transferência de um codificador convolucional. Esta função pode ser determinada através da fórmula de Mason, ou através da resolução do sistema referente ao diagrama.

Na Figura 2.2.2.2.2 é mostrado o diagrama de estado modificado para o código convolucional (2,1,3), onde Z representa o ramo, isto é, a ligação entre dois estados, X representa o discriminante com os pesos das palavras código ramo e Y representa o discriminante que contém o peso da informação de uma transição de um estado para outro qualquer. O sistema genérico referente à solução deste problema é então descrito por:

$$E(i+1) = A(i) \cdot E(i) + B(i) \qquad (2.2.2.2.1)$$

$$T(i) = H(i) \cdot E(i) \qquad (2.2.2.2.2)$$

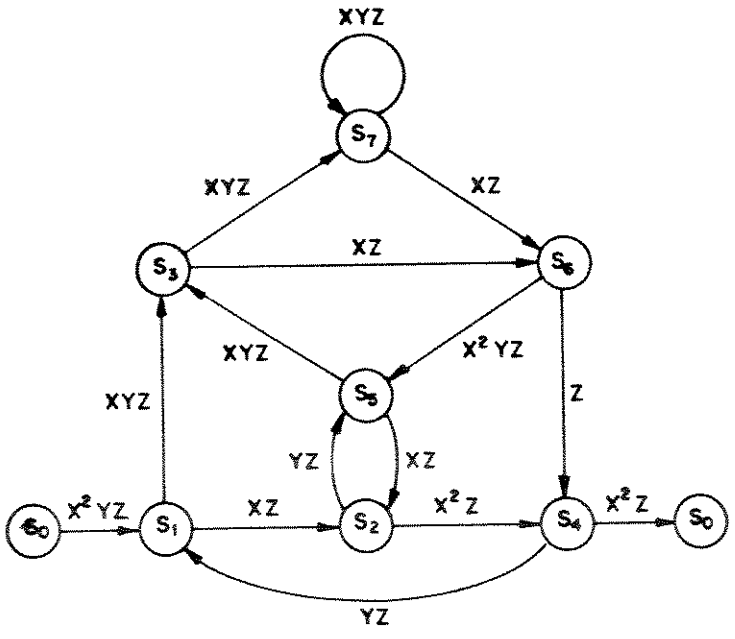

Figura 2.2.2.2.2 - Diagrama de Estado Modificado para o Código (2,1,3).

Onde E(i) é a matriz de estados que especifica os valores das transições do estado inicial para os estados intermediários, A(i)

é uma matriz de transição entre os estados intermediários, H(i) é a matriz de saída que especifica os valores das transições dos estados intermediários para o estado final e B(i) é a matriz que especifica os valores das transições do estado inicial para os estados intermediários.

2.2.2.3 - PROPRIEDADES DE DISTÂNCIA DOS CÓDIGOS CONVOLUCIONAIS

O desempenho dos códigos convolucionais está diretamente relacionado com o algoritmo de decodificação utilizado e com as propriedades de distância do código.

Os códigos convolucionais possuem diferentes medidas de distâcia, porém as mais importantes são: a distância livre, d_{free}, função distância de coluna d_i e distância mínima, d_{min}.

Dentre todas essas medidas a mais importante é a distância livre, que é definida como o peso mínimo da sequência codificada entre os caminhos que saem do estado zero e retornam posteriormente a este estado. Isto corresponde a tomar o caminho de menor peso na treliça.

Formalmente a distância livre é definida como sendo:

$$d_{free} = \min \{ d(v', v'') : u' \neq u'' \}$$

onde v' e v'' são sequências codificadas correspondentes respectivamente às sequências de informação u' e u''.Como os códigos convolucionais são lineares, temos que :

$$\begin{aligned} d_{free} &= \min \{ \omega(v' + v'') : u' \neq u'' \} \\ &= \min \{ \omega(v) : u \neq 0 \} \\ &= \min \{ \omega(u \cdot G) : u \neq 0 \} \end{aligned} \qquad (2.2.2.3.1)$$

onde $\omega(.)$ é o peso de Hamming.

A distância de coluna de ordem i, d_i, é definida como sendo:

$$d_i = \min \{ d(v'[0,i], v''[0,i]) : u'[0] \neq u''[0] \}$$
$$= \min \{ \omega(v[0,i]) : u[0] \neq 0 \} \qquad (2.2.2.3.2)$$

onde tanto a sequência codificada como a correspondente sequência de informação são truncadas. Assim, a distância de coluna é o peso mínimo da sequência codificada de comprimento (i+1) com $u[0] \neq 0$.

Sob o ponto de vista matricial temos que:

$$v[0,i] = u[0,i] \cdot G[0,i] \qquad (2.2.2.3.3)$$

portanto:

$$d_i = \min \{\omega (u[0,i] \cdot G[0,i]) : u[0] \neq 0\} \qquad (2.2.2.3.4)$$

depende somente das n(i+1) colunas de G. Note que d_i não decresce para valores crescentes de i.

Para este tipo de distância existem dois casos particulares de bastante interesse, um é para quando i = m, então d_i é dita ser a distância mínima do código convolucional. Esta distância representa o peso mínimo da sequência codificada para a primeira restrição de memória.

O segundo caso é quando $i \to \infty$, e o $\lim_{i\to\infty} d_i$, é o peso mínimo da da sequência codificada de comprimento qualquer e a sequência de informação correspondente é não nula. Das definições de $\lim_{i\to\infty} d_i$ e d_{free} temos que $d_{free} = \lim_{i\to\infty} d_i$. Assim d_i alcançará o valor de d_{free} e a partir daí não crescerá mais.

O poder de correção dos códigos, denominado t é dado em função da distância mínima ou em função da distância livre.

Assim o poder de correção dos códigos convolucionais t é:

$$t = \left[\frac{d_{free} - 1}{2} \right]$$

onde $[\alpha]$ é o maior inteiro menor ou igual a α.

2.2.3 - CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA

Os códigos convolucionais de memória unitária constituem uma classe especial dos códigos convolucionais. Estes códigos são equivalentes aos convolucionais, com a vantagem de terem uma distância livre maior ou igual à distância dos códigos convolucionais de mesma complexidade [8].

Se G_0 e G_1 são as submatrizes geradoras do código convolucional de memória unitária invariante no tempo, e $\underline{x}_t$ é um vetor de dados de entrada k dimensional e $\underline{y}_t$ um vetor de dados codificados n dimensional, para um código convolucional de memória unitária de taxa R = k/n, o processo de codificação é dado por:

$$\underline{y}_t = \underline{x}_t \cdot G_0 + \underline{x}_{t-1} \cdot G_1 \qquad (2.2.3.1)$$

para t>0 com sistema não causal.

Para um melhor entendimento da abordagem dada a este trabalho, é necessário que sejam apresentados e definidos alguns conceitos, a saber:

Código Ótimo - é aquele cuja função enumeradora possui o menor número de palavras-código com peso igual à d_{free}.

Máximo Fluxo - ϕ , é a soma dos pesos de Hamming das palavras códigos ramo que divergem e convergem para o mesmo estado, e satisfaz a equação:

$$\phi = n \cdot (q-1) \cdot q^{(k-1)} \qquad (2.2.3.2)$$

Conservação de Fluxo - que o fluxo que entra e/ou sai de cada estado, com exceção do estado zero, é constante.

Através destas definições em conjunto com os conceitos de fluxos de redes foi proposto um algoritmo para encontrar códigos ótimos de memória unitária [8].

Para um dado código de taxa R=k/n com m=1, sobre GF(2), os passos a serem seguidos para a determinação de um código ótimo de memória unitária são apresentados a seguir:

P1 - Determinar a distância mínima que deve ser menor ou igual a d_0 (distância de projeto). Determinar o fluxo ϕ dado pela equação 2.2.3.2.

P2 - Resolver o problema da mochila através do algoritmo do "branch-and-bound".

P3 - Para cada possível solução do passo dois determinar através do uso da representação modular de código de bloco [9], as submatrizes G_i.

P4 - Enumerar as palavras códigos resultantes do procedimento anterior através do algoritmo de Viterbi.

P5 - Se o valor resultante para o d_{min} proveniente d_0 ítem anterior não é o valor de d_0, então realize troca de colunas nas matrizes G_i e retorne ao passo quatro; caso

todas as possíveis trocas de colunas tenham sido realizadas e o valor de do não foi alcançado, retorne ao passo três, caso contrário vá para o procedimento seis.

P6 - Diminua de uma unidade o valor de d_{min} e retorne ao procedimento dois.

Para que se possa entender melhor o método de obtenção dos códigos convolucionais ótimos de memória unitária, será apresentado um exemplo. Contudo, se faz necessário apresentar primeiramente o algoritmo de decodificação de Viterbi.

2.2.4. ALGORITMO DE VITERBI

Um decodificador de máxima verossimilhança para um determinado tipo de canal sem memória, escolhe como palavra código aquela que maximiza o logaritmo da função probabilidade condicional conjunta de uma sequência na saída dado uma sequência de informação na entrada do canal. Tal função recebe o nome de métrica associada ao caminho.

Sendo o algoritmo de Viterbi um decodificador de máxima verossimilhança, temos como consequência que este algoritmo fornecerá a melhor sequência estimada dentre todas as possíveis sequências na treliça via a métrica acumulada.

Seja r uma sequência de informação do canal,e v uma sequência de informação na entrada do canal, isto é, $r = (r_1, r_2, \ldots, r_N)$ e $v = (v_1, v_2, \ldots, v_N)$ onde r_i e v_i tem comprimento n.

Para o caso especial do canal binário simétrico (BSC) com com probabilidade de transição $p < 1/2$, a sequência recebida é binária, e a função log-probabilidade é dada por:

$$\log P(r/v) = d(r,v)\log\frac{p}{1-p} + N\log(1-p) \qquad (2.2.4.1)$$

onde d(r,v) é a distância de Hamming entre r e v, e p é a probabilidade de transição do canal.

Definiremos com janela de tempo ao conjunto das possíveis transições entre estados na treliça entre os instantes de tempo i e i+1.

O algoritmo de Viterbi consiste assim dos seguintes passos:

P1 - Inicie o procedimento de comparação entre r e v através da comparação de r_1 com todas as transições entre estados da janela de tempo da treliça. Para i=1, compute a métrica parcial e armazene a maior métrica que chega a cada um dos estados;

P2 - Passe para a próxima janela de tempo, incrementando i de uma unidade. Compute a nova métrica para todas as transições de estados de janela de tempo i+1, e acumule à métrica obtida na janela de tempo anterior. Novamente armazene a maior métrica obtida em cada um dos estados no instante de tempo i+2;

P3 - Caso i seja menor que o tamanho da mensagem a ser decodificada, repita o passo dois. Caso contrário, pare.

Seja o diagrama de treliça como mostrado na Figura 2.2.4.1 representando um código convolucional (3,1,2)

Seja r=(110,110,110,111,010,101,101) a palavra recebida, através de um canal BSC, para este tipo de canal computamos como a métrica utilizada pelo algoritmo de Viterbi resulta na distância de Hamming [12].

A palavra decodificada está mostrada em negrito na Figura 2.2.4.1. A correspondente sequência de informação estimada é $\hat{u}$ = (1 1 0 0 1).

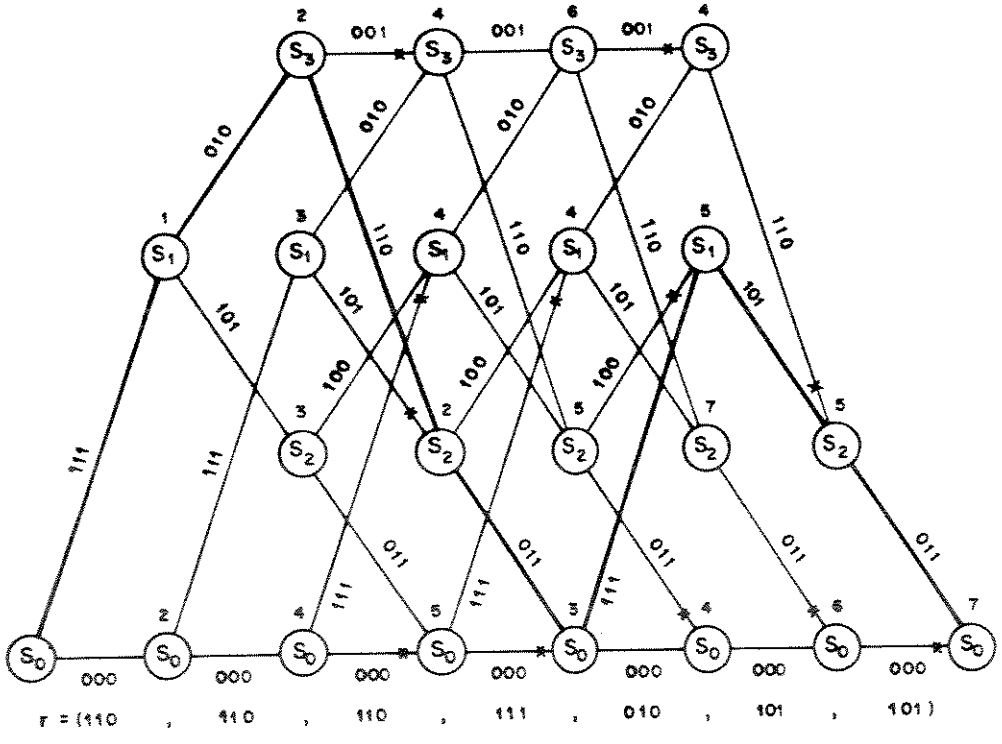

Figura 2.2.4.1 - Algoritmo de Viterbi para o BSC

Após a apresentação do algoritmo de Viterbi é interessante que se faça um exemplo para encontrar um código convolucional ótimo de memória unitária seguindo o algoritmo apresentado no ítem anterior.

Considere o código de taxa r = 2/3, e m = 1 sobre o corpo de dois elementos. A determinação do código ótimo então segue os seguintes passos:

passo 1 - O valor de d_{min} (um limitante superior baseado no limitante de Plotkin) e é dado por [8]:

$$d_{min} <= \min_{p>=1} \{ \frac{2^{p-1}}{2^p-1} \} . n/k (p + km)$$

onde m=1 para códigos de memória unitária.

O valor do fluxo é $\phi = n . 2^{k-1} = 3 . 2 = 6$

A minimização do limitante superior pode ser realizada trivialmente via substituição dos valores inteiros que pode assumir. Isto é mostrado na tabela abaixo.

p	d_{min}
1	4,5
2	4,0
3	4,3
4	4,8

passo 2 - Resolução do problema da Mochila (Knapsack)

$2\phi = a_0 d_0 + a_1 d_1 + a_2 d_2$: (determinação do número de caminhos em duas janelas de tempo, com distâncias d_0, d_1, d_2)

$d_i = d_0 + i$: (condição inicial de distância)

$a_0 + a_1 + a_2 = 3$: (restrição ao número de caminhos)

a partir da equação de d_i temos que:

$$4a_0 + 5a_1 + 6a_2 = 12$$

aplicando o algoritmo do "branch- and- bound", as soluções encontradas são:

(3,0,0) e (0,0,2)

pela restrição de que a soma dos a_i tem que ser três é válida somente a solução (3,0,0).

passo 3 - Aplicando a representação modular à solução do passo 2, temos que o vetor das distâncias é dado por:

$$\underline{d} = (4,4,4)$$

Note que $\underline{d}$, o vetor de distâncias, explicita os pesos das palavras código que o código de bloco deve ter. O que se pretende agora é, de posse destes pesos, determinar qual deve ser a matriz geradora G do código de bloco que forneça palavras código com a distribuição de pesos dada por $\underline{d}$.

Esta matriz geradora pode ser deteminada via representação é, os elementos c_i são representação inteiras das colunas binárias da matriz geradora do código de bloco. De [9], podemos explicitar a relação entre o vetor $\underline{d}$ e o vetor C através de:

$$C = \underline{d} \cdot H_k$$

onde H_k são matrizes definidas como:

$$H_1 = [\ 1\ ]; \qquad H_3 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{bmatrix}$$

Assim o valor de C é dado por:

$$C = (2,2,2)$$

então a matriz G associada a este código de bloco possui dois elementos do tipo um, dois elementos do tipo dois, e dois elementos do tipo três. Consequentemente,

$$G = \begin{array}{c} \begin{array}{cccccc} & & & G_0 & & G_1 \end{array} \\ \left[\begin{array}{ccc|ccc} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{array}\right] \end{array}$$

Portanto, este método propicia a obtenção do código convolucional ótimo. Note que a complexidade deste algoritmo está relacionada com a solução do problema da mochila. Este problema pertence à classe de problemas não polinomiais completos, que será visto com mais atenção na seção seguinte.

2.3 - SISTEMAS CRIPTOGRÁFICOS

Como mencionado no Capítulo 1 os sistemas criptográficos podem ser classificados como sendo os do tipo convencional e os do tipo chave pública.

Nesta seção procuraremos apresentar os conceitos básicos dos sistemas criptográficos convencionais seguido dos conceitos de chave pública, uma vez que o núcleo da proposta do sistema de chave pública é mais facilmente introduzido via os sistemas convencionais.

2.3.1 - SISTEMAS CRIPTOGRÁFICOS CONVENCIONAIS

Em qualquer um dos sistemas criptográficos utilizados, isto é, o convencional ou de chaves públicas, existem dois tipos de problemas a serem resolvidos, a saber:

1 - A privacidade, tem por objetivo evitar que a informação seja interceptada no canal por pessoas não autorizadas. Estas pessoas não autorizadas são denominadas de criptoanalistas. Na Figura 2.3.1.1 é mostrado um sistema criptográfico com privacidade.

2 - A autenticidade , busca evitar que a informação seja alterada pelo criptoanalista. Na Figura 2.3.1.2 temos um sistema criptográfico que ilustra este tipo de problema.

Estes dois tipos de problema estão ligados intrinsicamente e para resolvê-los uma única técnica é aplicada.

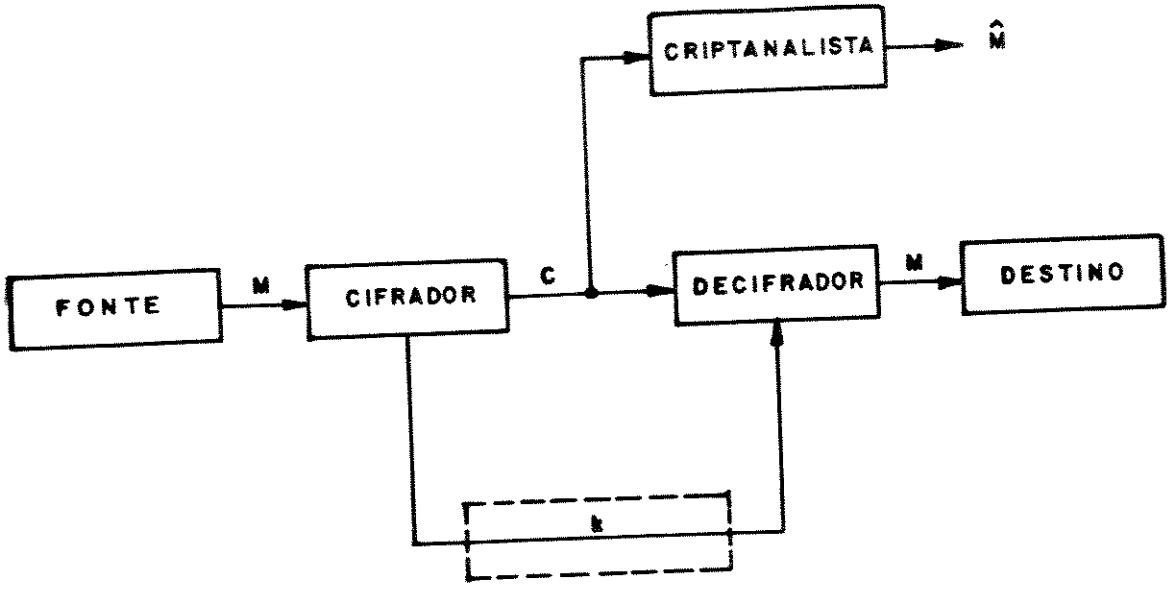

Figura 2.3.1.1 - Criptosistema com Privacidade

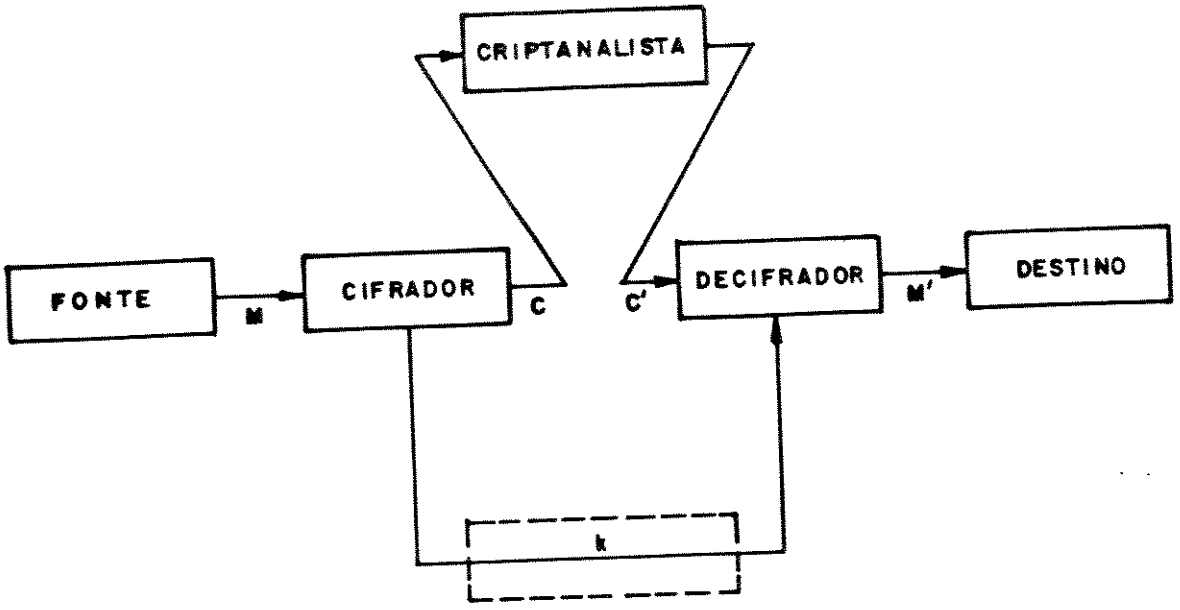

Figura 2.3.1.2 - Criptosistema com Autenticidade

Na Figura 2.3.1.1 é ilustrado um sistema criptográfico envolvendo o problema da privacidade. O transmissor gera uma mensagem M, esta mensagem é enviada através de um canal inseguro, monitorado por um criptoanalista. Afim de evitar que esta mensagem seja interceptada por pessoas não autorizadas, o transmissor cifra o texto M com uma transformação inversível T_k, através da transformação $C = T_k(M)$.

Assim a tarefa do receptor autorizado ao receber a mensagem $T_k(M)$ será a de aplicar uma transformação inversa T_k^{-1} em $T_k(M)$, isto é,

$$T_k^{-1}(T_k(M)) = M,$$

recuperando dessa forma a mensagem original M.

A transformação T_k, aplicada a M, é escolhida aleatoriamente de um conjunto de transformações. O parâmetro que seleciona a transformação individual é dita ser uma chave específica ou simplesmente chave.

Podemos definir um sistema criptográfico mais formalmente como sendo uma classe de transformações inversíveis, isto é,

$$k \in \bar{K} \qquad (2.3.1.1)$$

onde;

$$T_k : \bar{M} \rightarrow \bar{C} \qquad (2.3.1.2)$$

onde $\bar{M}$ é o espaço que contém as mensagens originais , $\bar{C}$ é o espaço que contém as_mensagens cifradas, e a chave k é selecionada de um espaço finito K, denominado de espaço de chaves.

Em um sistema criptográfico toda a segurança reside na chave utilizada para cifrar. Podemos até supor que o criptoanalista conheça o espaço de chaves que está sendo usado, porém, a probabilidade do mesmo descobrir a transformação utilizada é mínima. Portanto, em um sistema criptográfico convencional a chave deve ser enviada do transmissor para o receptor através de um canal seguro, residindo aí, uma das desvantagens dos sistemas convencionais.

Na Figura 2.3.1.2 observamos porque um sistema criptográfico pode ser usado também para resolver o problema da autenticidade. Para este caso o criptoanalista não somente intercepta a mensagem, como também altera o seu conteúdo. O receptor autorizado protege-se deste tipo de problema aceitando somente as mensagens cifradas que chegam através do canal, correspondendo à chave correta.

Como um exemplo de um criptosistema convencional, podemos ilustrar como cifrar um texto em claro representado por um vetor w de dimensão n. Para tal, basta multiplicar o vetor ω por uma matriz inversível E de dimensão 1 x 1. A mensagem cifrada é obtida

através do produto Ew. Se $D = E^{-1}$, temos que w = DEw. Então os processos de cifragem e decifragem requerem l^2 operações. Para o cálculo de D e E, portanto, envolve um problema de inversão de matriz que no pior caso apresenta complexidade de l^3 operações. Portanto, não sendo um problema de fácil solução.

Existem dois tipos de ataques que podem ser utilizados pelo criptoanalista; a saber:

Ataque ao Texto Cifrado - É o ataque em que o criptoanalista está de posse somente do texto cifrado, e não possui nenhuma informação sobre o texto original.

Ataque ao Texto Pleno - É aquele em que o criptoanalista possui uma quantidade substancial de informação correnpondente ao texto original (em claro), e além disso está de posse do texto cifrado.

Normalmente, na prática os ataques são desenvolvidos aos textos cifrados.

Existem fundamentalmente dois tipos diferentes de segurança exigidas para que se possa considerar um criptosistema seguro. Em alguns sistemas, a quantidade de informação útil que o criptoanalista tem posse, é insuficiente para que ele determine o par de tranformação correspondente à cifragem e decifragem, mesmo que este possua grandes recursos computacionais. A este tipo de sistema diz-se que se tem uma **segurança incondicional**.

Quando a mensagem interceptada contém informação suficiente que permite uma solução única, para o problema da criptoanálise, não existe uma garantia de que mesmo com recursos computacionais limitados, o criptoanalista não encontre a solução.

Acontece que por vezes mesmo de posse de dados que possam levar o criptoanalista à quebra da mensagem é necessário que ele realize um trabalho computacional exaustivo. E a este tipo de segurança, denomina-se **segurança computacional**.

A segurança incondicional foi analisada com mais detalhes por Shannon em 1949, onde foi estabelecido que se o criptoanalista

possuir tempo de computação ilimitado, então o mesmo não necessita de um processo computacional eficiente.

Assim, sabendo-se que em muitos textos interceptados uma única solução é possível, Shannon denominou a isto de distância de unicidade No, que é o número de caracteres para se ter uma única solução.

Shannon através desta distância estabeleceu um modelo. De acordo com este modelo de cifragem aleatória tem-se que:

$$No = H(k)/D \qquad (2.3.1.3)$$

onde H(k) é a entropia da chave, ou seja , o comprimento da chave e D é a redundância de linguagem medida em bits/caracter. A partir da equação 2.3.1.3 pode-se obter que:

$$H(k) <= N_0 D \qquad (2.3.1.4)$$

onde $N_0 D$ representa o número de equações úteis para a solução do problema da chave [3].

Quando o número de equações é maior que a entropia da chave H(k), como dado pela equação 2.3.1.4, uma única solução é possível e neste caso o sistema não é incondicionalmente seguro.

No caso em que $H(k) = \infty$, então pela equação 2.3.1.3 o valor de No é infinito. Shannon mostrou então que para este último caso o valor de No = 28 [2] é excelente para que se possa implementar um sistema, porém considera que o criptoanalista tenha capacidade computacional limitada.

Dentre os algoritmos utilizados em criptografia convencional podemos citar: os de substituição e os de transposição. Para que possamos definir estes dois sistemas é necessário introduzirmos os conceitos básicos, a saber:

Alfabeto - constitui um conjunto de símbolos ou letras para representar a mensagem.

Mapa - um mapa σ de um conjunto A em um conjunto B é uma relação que associa a cada elemento $a_i \in A$ um elemento $b_i \in B$.

Substituição - é aplicada em um alfabeto através de um mapa biunívoco σ. A este alfabeto associa-se uma possível substituição de letras mapeada por um dos possíveis conjuntos de σ_i, i variando de 0 a N - 1, obedecendo a propriedades convenientes [3].

Podemos agora definir os sistemas criptográficos convencionais pelos métodos das transformações de:

1 - Substituição, este cifrado é caracterizado por uma transfomação de substituição aplicada ao alfabeto. A decifragem é obtida através do mapeamento de cada letra do criptograma aplicada a transfomação de substituição inversa.

2 - Transposição, esta técnica emprega a transposição das posições das letras da mensagem, ou similarmente pela permutação de forma predeterminada das letras da mensagem dentro de um bloco. A mensagem M é subdividida em blocos de tamanho N, podendo ser representada através de uma sequência caracterizada por:

$$"M = (m_{00}, m_{01}, \ldots, m_{0(N-1)}) \ldots (m_{i0}, m_{i1}, \ldots, m_{i(N-1)}) \ldots"$$

onde o primeiro índice em m_{ij} caracteriza o bloco da mensagem e o segundo caracteriza a posição de uma letra no bloco [3].

Um outro sistema de cifragem bastante utilizado é o DES (Data Encryption Standard), que se propõe a proteger a comunicação de dados entre computadores. Este sistema opera sobre blocos de 64 bits do texto em claro para produzir o texto cifrado. Um cifrador de blocos divide a mensagem em claro em blocos de tamanho fixo, e opera em cada bloco separadamente para produzir o texto cifrado [3].

2.3.2 - SISTEMAS CRIPTOGRÁFICOS DE CHAVE PÚBLICA

Como dito anteriormente, uma das grandes limitações dos sistemas convencionais, reside no fato de ter que proporcionar um canal seguro para o envio da chave. Por outro lado, como o processo de cifragem e decifragem são inseparáveis nestes sistemas estas chaves devem ser enviadas de forma bastante protegida.

Com a intenção de resolver este problema Diffie e Hellman em 1976, lançaram um novo conceito em criptografia, que é a implementação de sistemas criptográficos que utilizam a técnica de chaves públicas.

Este tipo de sistema é mostrado na Figura 2.3.2.1, e descrito com detalhes no ítem 2.3.2.1.

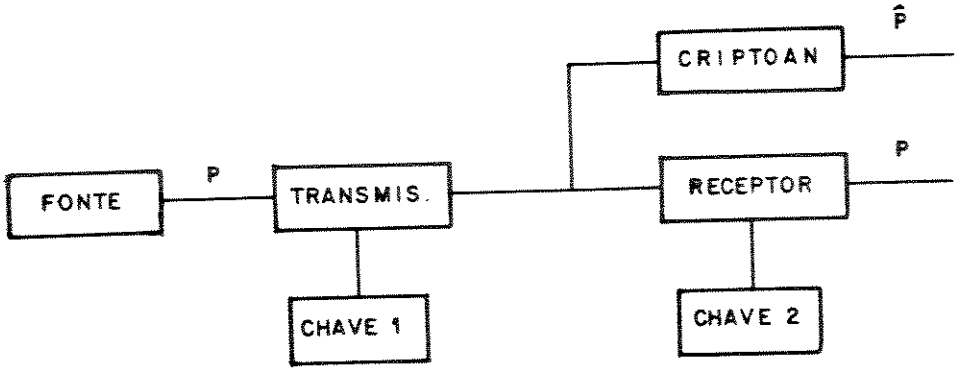

Figura 2.3.2.1- Fluxo de Informação em um Sistema de Chaves Públicas.

Como mencionado no Capítulo 1, existem dois tipos de sistemas de chave pública, a saber;

1 - os criptosistemas de chave pública e;

2 - os sistemas com distribuição pública de chaves.

Estes dois sistemas diferem muito pouco no que se refere aos conceitos matemáticos. Neste trabalho será abordado os sistemas de chave pública, que é descrito no ítem 2.3.2.1.

2.3.2.1 - CRIPTOSISTEMAS DE CHAVE PÚBLICA

A idéia básica dos criptosistemas de chave pública é a utilização de uma família de pares de transformação: (E_k, D_k), onde $k \in \{k\}$ e E_k e D_k são mapas definidos por:

$$E_k : \{M\} \rightarrow \{C\} = E_k(M) \qquad (2.3.2.1)$$

$$D_k : \{C\} \rightarrow \{M\} = D_k(C) \qquad (2.3.2.2)$$

sobre um espaço de mensagem finito {M}, tal que:

1 - Para cada $k \in \{K\}$, D_k é uma transformação inversa de E_k;

2 - Para cada $k \in \{K\}$ e $m \in \{M\}$, as operações E_k e D_k são simples do ponto de vista computacional;

3 - Para cada $k \in \{K\}$, é computacionalmente complexo descobrir a transformação D_k a partir de E_k;

4 - É computacionalmente simples a obtenção do par de transformações inversas E_k e D_k.

Dessa forma, o sistema criptográfico em questão consiste essencialmente de duas partes, a saber: uma que compreende a transformação para cifrar e outra para decifrar, e mais, dado como conhecida uma das partes é um problema difícil determinar a outra parte.

A quarta propriedade garante que existe uma solução para computar os correspondentes pares de transformações inversas.

Como exemplo de um criptosistema de chave pública temos o algoritmo do RSA descrito a seguir.

ALGORITMO DO RSA

Este algoritmo foi proposto por Rivest, Shamir e Adleman e sua segurança esta baseada em um problema reconhecidamente difícil em Teoria dos Números.

Para descrever o procedimento considere que o texto em claro que o usuário B deseja cifrar e transmitir para o usuário A é um número $M \in [0, N-1]$. O usuário A possui uma chave de cifragem pública, dada pelo par de números inteiros positivos (e, N) e uma chave de decifragem, secreta, dada pelo par de números (d, N), também inteiros positivos.

Para enviar a mensagem M cifrada com a chave pública (e, N) o usuário B utiliza o seguinte algoritmo de cifragem:

1 - Algoritmo de Cifragem

$$C = E_{(e,\ N)}(M) = M^e \bmod(N)$$

Para o usuário A decifrar a mensagem ele utiliza o algoritmo de decifragem abaixo:

2 - Algoritmo de Decifragem

$$M = D_{(d,\ N)}(C) = C^d \bmod(N)$$

2.3.2.2 - COMPLEXIDADE COMPUTACIONAL

Do ponto de vista computacional existe uma preocupação inerente para com os processos desenvolvidos. Esta preocupação reflete o objetivo imediato de se encontrar algoritmos eficientes.

A avaliação da eficiência depende do tempo de processamento de um algoritmo em uma determinada máquina, este tempo está diretamente ligado à quantidade de operações a serem realizadas no no processo. Como consequência, determina a complexidade do algoritmo.

A complexidade de um algoritmo pertence basicamente a duas classes, a saber: a classe polinomial P e a classe não polinomial NP.

A complexidade de um algoritmo é dita ser **polinomial** quando esta for uma função polinomial nos tamanhos dos dados de entrada. Como exemplo podemos citar o algoritmo que verifica se um grafo é ou não biconexo. A complexidade deste algoritmo é linear com relação ao tamanho do grafo. Logo, o problema de biconetividade pertence à classe P.

Um algoritmo é dito pertencer a classe não polinomial NP, quando os problemas de decisão não admitem algoritmo polinomial. Neste caso a complexidade é exponencial. Como exemplo podemos mencionar o problema do caixeiro viajante.

Os problemas NP são classificados ainda como aqueles que correspondem ao de menor dificuldade de solução, e os problemas NP que correspondem a uma maior dificuldade de solução dentre todos

os NP. Esta classe de maior dificuldade de solução dentre os problemas NP são denominados NP completos. Karp lista 21 problemas que podem ser NP - completo [5], dentre eles encontra-se o problema da Mochila.

Podemos definir o problema da mochila da seguinte forma: considere um conjunto de objetos cuja a solução de um determinado problema, está contido em um subconjunto deste conjunto de objetos, a este conjunto total denominamos de mochila. Como exemplo podemos ilustrar um problema numérico, suponha que queiramos saber se é possível escrever 31 como a soma de um subconjunto dos números abaixo:

[10, 17, 9, 12, 40, 60]

a resposta é sim: 31 = 10 + 9 + 12. Onde os números [10, 9, 12], compõem um subconjunto do conjunto dado.

A classe de problemas NP - completo é vista com bastante interesse para uso em sistemas criptográficos, pelo fato da dificuldade de solução, exigindo um considerado tempo computacional para que sejam resolvidos. Desta forma, a de um bom sistema criptográfico é equivalente a resolver um problema NP - completo.

Considere o seguinte exemplo: Seja $Y = f(x) = ax$, onde a é um vetor conhecido de n inteiros $(a_1, a_2, \ldots, a_n)$ e x é um vetor binário n-dimensional. O cálculo de Y é simples, envolvendo somente a soma de n números inteiros. Entretanto,o problema da inversão de f é conhecido como o problema da Mochila e requer como solução encontrar um subconjunto de $\{a_i\}$, cuja a soma seja Y.

O caminho inverso, isto é, a partir de f, obter um subconjunto de elementos de $\{a_i\}$ cuja a soma seja Y, é conhecido como o problema da mochila.

A busca exaustiva de todos os 2^n subconjuntos de $\{a_i\}$, cresce exponencialmente com n, sendo esta busca computacionalmente impraticável para n muito grande.

CAPÍTULO 3

MÉTODO DO KNAPSACK BINÁRIO PARA CRIPTOSISTEMAS DE CHAVE PÚBLICA

3.1 - INTRODUÇÃO

No Capítulo 2 foi mostrado que para a determinação de códigos ótimos de memória unitária, temos que resolver o problema da mochila, como passo fundamental. Sabemos que este problema no pior caso pertence à classe dos problemas não polinomias completos (NP - completo), e desde que estes tipos de problemas não são de fácil solução, justificando assim a dificuldade na determinação de bons códigos convolucionais, é desejável o seu uso em sistemas criptográficos convencionais e mesmo nos sistemas de chaves públicas.

A utilização de códigos corretores de erros em criptografia surgiu inicialmente com o uso de códigos de bloco, mais especificamente com os códigos de Goppa. Esta proposta está baseada no fato de que esses códigos (códigos de Goppa), são dificeís de serem quebrados, pois sua alta eficiência de correção é destruida se as palavras do código tiverem algum embaralhamento dos bits que a compõem.

Por outro lado, vários criptosistemas propostos estão baseados em problemas bastante complexos em Teoria dos Números e Teoria Combinatorial, com reconhecida dificuldade de solução.

A segurança do CSCP baseado em códigos convolucionais aqui proposto leva em consideração os seguites fatos:

1 - O embaralhamento das colunas das submatrizes geradoras do código convolucional faz com que o peso de Hamming da palavra código ramo diminua ,

causando a queda no poder de correção dos erros;

2 - O fato adicional de que o problema da mochila do tipo binário, tem que ser resolvido.

Para que possamos obter os resultados esperados no desempenho de um sistema criptográfico utilizando códigos convolucionais, será necessário que uma função armadilha seja utilizada.

Como consequência dos fatos acima citados, na Seção 3.2 será apresentada uma proposta de um esquema criptográfico utilizando códigos convolucionais de memória unitária.

Na Seção 3.3, serão abordadas as propriedades das funções armadilhas, para a implementação desse criptosistema.

Na Seção 3.4, será realizada a análise de complexidade computacional deste criptosistema, onde é apresentamos o método utilizado para encontar as matrizes A e B.

Finalmente na Seção 3.5 é feita uma análise sobre o comportamento do vetor erro quando se utiliza transformações isométricas e transformações não isométricas nas submatrizes geradoras.

3.2 - DESCRIÇÃO DO MÉTODO DO SISTEMA CRIPTOGRÁFICO

Para a estrutura de códigos de memória unitária descrita no Capítulo 2 temos k entradas paralelas nos registradores de deslocamento. O codificador código de memória unitária é mostrado na Figura 3.2.1. Este codificador tem taxa R = 2/4 com as seguintes submatrizes geradoras.

$$G_o = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad G_1 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

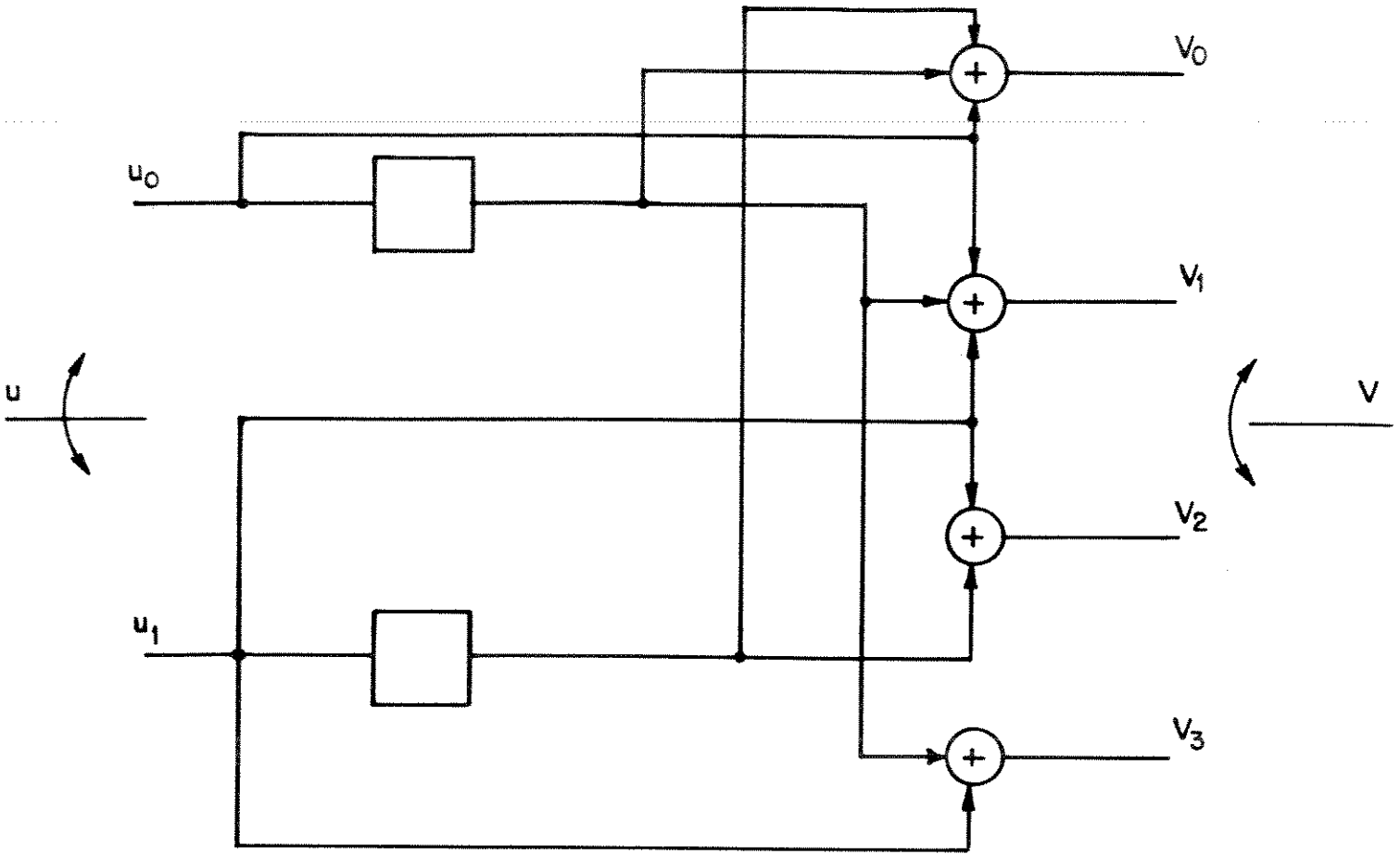

Fig. 3.2.1 - Codificadores de um Código (4, 2, 1)

Este código é o código ótimo de taxa R = 2/4. Para se enfatizar-mos algumas das propriedades dos códigos de memória unitária é também apresentado o diagrama de estado do referido código.

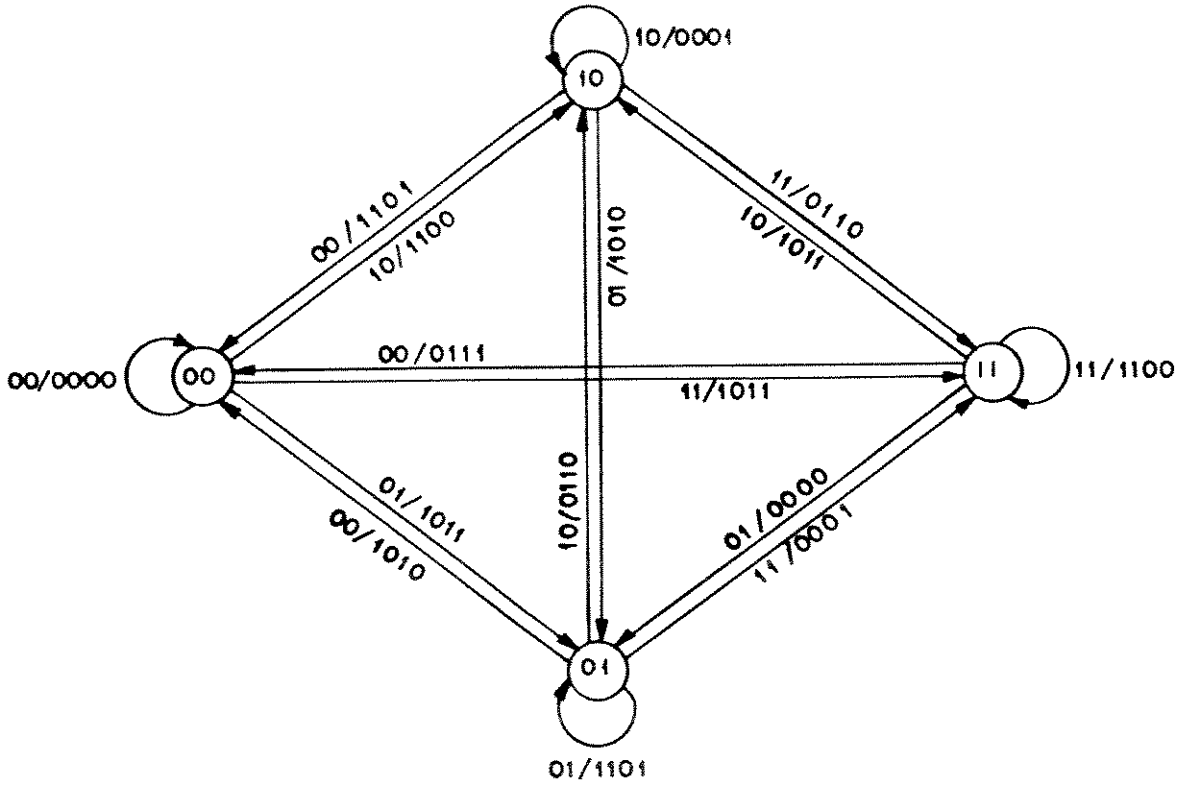

Fig. 3.2.2 - Diagrama de Estados do Código com R = 2/4

Neste diagrama observamos que em cada estado temos $2^{k.m}$ saídas e $2^{k.m}$ entradas em cada estado. Esta propriedade será vista posteriormente no Capítulo 4, como parte da análise de complexidade dos sistemas criptográficos. Se observarmos o diagrama de estado deste código, constatamos que o número de ramos depende exponencialmente das k entradas.

Esta rápida revisão é para que possamos introduzir os conceitos e procedimentos para a obtenção das funções armadilhas

do "knapsack" binário.

As funções armadilhas são transformações aplicadas às submatrizes geradoras do código $[G_o, G_1]$, com a finalidade de reduzir o poder de correção deste código à um valor desejável ou mesmo fixado pela aplicação.

Estas transformações aplicadas às submatrizes geradoras, são feitas através de um par de matrizes A e B. As dimensões destas matrizes A e B são respectivamente k x k e n x n.

Esta função armadilha uma vez aplicada, gera novas submatrizes geradoras denominadas de G'_o e G'_1, dadas por:

$$G'_o = AG_oB \ , \qquad (3.2.1)$$

e

$$G'_1 = AG_1B \qquad (3.2.2)$$

Além deste procedimento de multiplicação pelo par de matrizes A e B, cuja finalidade é reduzir o poder de correção do código, através da alteração da distância livre, pode-se ainda adicionar um vetor erro à mensagem cifrada. Este vetor erro denominado y_e adiciona na mensagem cifrada uma quantidade t de erros correspondendo ao poder de correção do código original.

O processo de codificação para um código de memória unitária é definido por:

$$\tilde{y}_t = \tilde{x}_t \, G_o \oplus \tilde{x}_{t-1} \, G_1 \quad \text{onde } \tilde{x}_t = 0 \text{ para } t < 0 \qquad (3.2.3)$$

onde $\tilde{y}_t$ éa seqüência codificada na entrada do canal e $\tilde{x}_t$ a seqüênciade informação.

Substituindo então as equações (3.2.1) e (3.2.2) em (3.2.3) temos que:

$$y_t = x_t G'_o \oplus x_{t-1} G'_1 \qquad \text{com } t \geq 0 \quad \text{e } x_{-1} = 0 \tag{3.2.4}$$

A equação (3.2.4) contém conjuntamente o processo de codificação e cifragem .

Agora ao vetor y_t adicionamos um vetor erro que corresponderá à quantidade de erro permissível ao código original, assim:

$$y_{te} = y_t + y_i \tag{3.2.5}$$

onde:

y_{te} → é o vetor correspondente à mensagem codificada/cifrada com a inserção do erro.

y_i → é a quantidade de erros inseridos na mensagem no seu ponto de envio.

y_t → mensagem cifrada/codificada.

O vetor y_i dá origem a um vetor erro y_e, na decodificação, cujo peso de Hamming deste vetor é menor ou igual a capacidade de correção do código. O vetor erro na decodificação é obtido através da multiplicação do vetor erro de transmissão pela transformação inversa de B.

Para facilidade de compreensão, a seguir apresentamos em forma de diagrama de blocos o sistema de chave pública sendo considerado, Fig. 3.2.3.

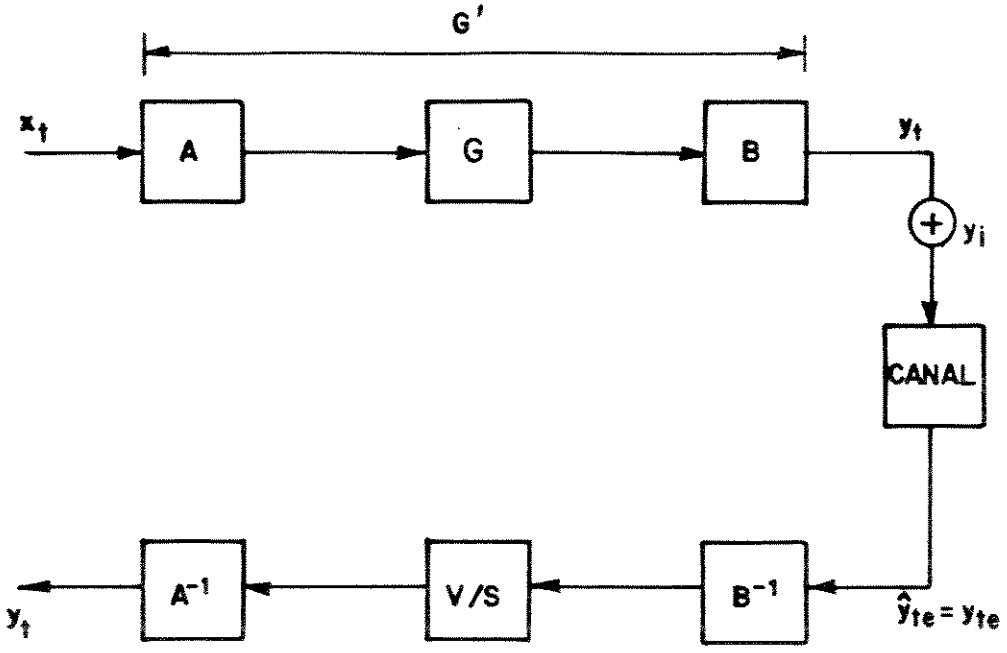

Fig. 3.2.3 - Diagrama em Bloco do Criptosistema de Chave Pública

Como o sistema criptográfico é de chave pública as matrizes

$$G' = [G'_o \quad G'_1] \tag{3.2.6}$$

são colocadas em uma lista pública, podendo também ser divulgado o

número de erros que será adicionado à mensagem.

A soma do vetor erro de transmissão ao vetor que contém a informação y_t é que segue através do canal. Deste modo o processo de decodificação é aplicado ao vetor y_{te}.

No processo de decodificação empregado à sequência de saída do canal, $\hat{y}_{te}$, considerando que o canal é livre de ruído temos que $\hat{y}_{te}$ (valor estimado de y_{te}) é igual ao valor de y_{te} na entrada do canal. Utilizamos então a transformação inversa de B, como explicitado na equação 3.2.7.

$$y_{te} \cdot B^{-1} = (x_t\ A)\ G_o \oplus (x_{t-1}.A)\ G_1 \qquad (3.2.7)$$

Consequentemente x_t é obtido aplicando o algoritmo de Viterbi. Para este criptosistema a matriz B^{-1} é a transformação aplicada às submatrizes geradoras originais do código cujo objetivo é reduzir a mínima distância livre, enquanto que a matriz "A" realiza o embaralhamento dos bits, nas palavras-código ramo.

Note que para que este criptosistema seja quebrado, é necessário que o criptoanalista primeiramente resolva o problema da Mochila relacionado a encontrar as submatrizes geradoras do código ótimo, G_o e G_1, e depois encontrar as transformações A^{-1} e B^{-1}. Estes dois fatos não apresentam soluções triviais.

Para ilustrar o procedimento de cifragem, tomemos como exemplo um código ótimo com taxa R = 2/4 e cujas submatrizes geradoras são dadas por:

$$G_o = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad G_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

onde a distância livre deste código é dada por:
$d_{free} = \{G_o\ G_1\} = 5$. Aplicando as transformações A e B dadas por:

$$A^{-1} = A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$B^{-1} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

em G_o e G_1 obtemos G'_0 e G'_1, como segue:

$$G'_o = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad G'_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a mínima distância livre deste novo código é: $d_{min}\{G'_o, G'_1\} = 3$. Assim o poder de correção foi reduzido, ou seja, enquanto o código original corrige até 2 erros o novo código corrige apenas 1 erro.

Suponha agora que se queira transmitir a mensagem u = (10, 01, 11, 00), considere o codificador inicialmente resetado. Logo a palavra a ser transmitida será: y_t = (0111, 1101, 1001, 0011).

Adicionando o vetor erro de transmissão y_i = (1000, 0000, 0000, 0000) a palavra que irá pelo canal é: y_{te} = (1111, 1101, 1001, 0011). Agora aplicamos ao vetor y_{te} a transformação B^{-1} como mostra a equação (3.2.7) resultando em y_{te}(1110, 0011, 1111, 0111). Em seguida aplicamos o processo de decodificação através do

algoritmo de Viterbi obtendo a sequência de informação u = (10, 01, 11, 00).

3.3 - PROPRIEDADES DAS FUNÇÕES ARMADILHAS

Nesta seção iremos estabelecer as propriedades das funções armadilhas. Sejam então, as submatrizes geradoras G_i com $0 \leq i \leq m$ de um código convolucional sobre GF(2). Seja $\omega(G_i)$ o peso total de Hamming de cada submatriz. O fluxo das submatrizes é dado por $\phi(G_i)$, isto é,

$$\phi(G_i) = n.2^{k-1} \quad , \quad 0 \leq i \leq m \qquad (3.3.1)$$

como mostrado no Capítulo 2.

Com isso temos que:

Definição : [6]

Uma matriz G de dimensão k x n sobre GF(2) é dita ser igualmente distribuída com relação aos pesos de Hamming se e somente se os $2^k - 1$ elementos não nulos gerados por G têm pesos de Hamming dados por:

$$\tilde{\omega} = \frac{\phi(G)}{2^k - 1} \qquad (3.3.2)$$

Se $2^k - 1$ dividir $\phi(G)$ então $[\tilde{\omega}] = \tilde{\omega}$; caso contrário, $[\omega] = (|\tilde{\omega}| : |\omega| \pm 1 \geq 0)$, onde $|.|$ é a função piso .

Considere a matriz geradora G com dimensão 2k x n, de um código convolucional de memória unitária, a busca de um código ótimo pertencente a esta classe pode ser restringida pelo Lema 1, mostrado a seguir.

Lema 1 : [7]

Se o rank de G é igual a 2k e $\boxed{2k \leq n}$ então G tem 2k vetores linha linearmente independentes com pesos de Hamming igualmente distribuídos e geradores de todas as 2^{2k} palavras código também com pesos de Hamming igualmente distribuídos.

Demonstração

Um código de memória unitária com taxa R = k/n tem sua representação em treliça com 2^k estados e 2^k transições de cada estado, como mostra a Figura 3.2.2. Assim o número total de transições em uma janela de tempo é 2^{2k}.

Seja $[\tilde{\omega}]$ a parte inteira de $\tilde{\omega}$. Existem então:

$$\frac{n!}{(n-[\tilde{\omega}])!\,.\,[\tilde{\omega}]!}\,.\,[\tilde{\omega}]! \qquad (3.3.3)$$

vetores tendo peso de Hamming igual a $[\tilde{\omega}]$. Podemos mostrar facilmente que existe um n_o tal que para:

$$n \geq n_o \;;\; n!/(n-[\tilde{\omega}])!\,.\,[\tilde{\omega}]! \geq 2k \qquad (3.3.4)$$

logo, pelo menos 2k vetores tem peso de Hamming $[\tilde{\omega}]$.

Como cada transição na treliça tem associada uma palavra - código ramo diferente, é possível encontrar um código de bloco (k,2n) com o tamanho do bloco igual a 2n e o tamanho da informação k tal que a matriz gerada seja constituída de vetores

tendo pesos de Hamming igualmente distribuídos onde a distância mínima iguala aquelado código de bloco (k,2n). Portanto, obtemos os 2k vetores linha (linearmente independentes) de G de tal forma que G gera um espaço vetorial com 2^{2k} vetores com pesos de Hamming igualmente distribuídos.

CQD

A conseqüência imediata do Lema 1 é que o código é do tipo não catastrófico. Por outro lado, se 2k > n então a base do espaço vetorial gerado por G tem dimensão menor ou igual a n, então pelo menos uma das linhas de G é combinação linear das demais. Isto implica que nem todas as transições na treliça terão associadas palavras código diferentes. Assim, a busca pelo código ótimo deve ser bastante criteriosa, uma vez que o mesmo pode ser catastrófico.

Estabeleceremos agora a condição de existência da função armadilha composta pelo par de transformação (A,B), sob a condição do Lema 1 e que o código que esteja sendo utilizado seja ótimo.

Existem necessariamente duas condições, a primeira delas está relacionada com a aplicação da função armadilha a G_i significando multiplicar G pelo par (A,B), resultando em um novo código com a mesma taxa e número de memórias. A partir daí temos a seguinte proposição:

Proposição 1 : [7]

Sejam G_i $0 \leq i \leq m$ submatrizes de um código convolucional ótimo não catastrófico sobre GF(2). Sejam A e B matrizes inversíveis com dimensões k x k e n x n, respectivamente, então:

$$d_{min}\{A.[G_o, G_1, \ldots, G_m].B\} \leq d_{min}\{[G_o, G_1, \ldots, G_m]\}$$

onde A e B não são necessariamente unitárias.

Demonstração :

Partimos da hipótese que $G = [G_o, G_1, \ldots, G_m]$ é a matriz geradora de um código ótimo. Seja $d_{min}.(G) = d_{min}$. Agora seja $G' = AGB$ com $d_{min}(G') = d'_{min}$. Com isso temos duas condições, a saber:

1) $d'_{min} > d_{min}$

ou

2) $d'_{min} \leq d_{min}$

Definimos S como sendo o conjunto de todas as possíveis transformações (A,B) sobre GF(2). Note que S pode ser particionado em três subconjuntos:

S_1 - é o conjunto de transformaçõs (A,B) que conduzem a códigos com d_{min} maiores.

S_2 - é o conjunto de transformações (A,B) que conduzem aos códigos equivalentes, isto é, com o mesmo d_{min}.

S_3 - é o conjunto de todas as outras transformações (A,B) que conduzem aos códigos com menores d_{min}.

Sabemos que se o código que está sendo utilizado é ótimo então a condição (1) não é satisfeita, logo para, uma determinada taxa se existir um código com d_{min} maior para a mesma taxa, este código pode ser catastrófico ou não-linear.

Agora G' é equivalente a G se G' = AGB, onde A e B são matrizes inversíveis com dimensão k x k e n x n com

determinantes igual a 1, então (A,B) pertence a S_2 pois (A,B) é uma transformação isométrica ou uma equivalência, logo a igualdade é válida em 2. Obviamente, se A e B pertencem a S_3, então a transformação resultante conduzirá a um código com capacidade corretiva de erro menor.

CQD

3.4 - MÉTODO PARA ENCONTRAR AS MATRIZES DE TRANSFORMAÇÃO

Sabemos que os códigos convolucionais podem ser analisados como um problema de fluxo em rede [8].

Consideraremos a distribuição do fluxo em rede de um código convolucional ótimo de memória unitária, com taxa R = 2/3, mostrado na Fig. 3.4.1.

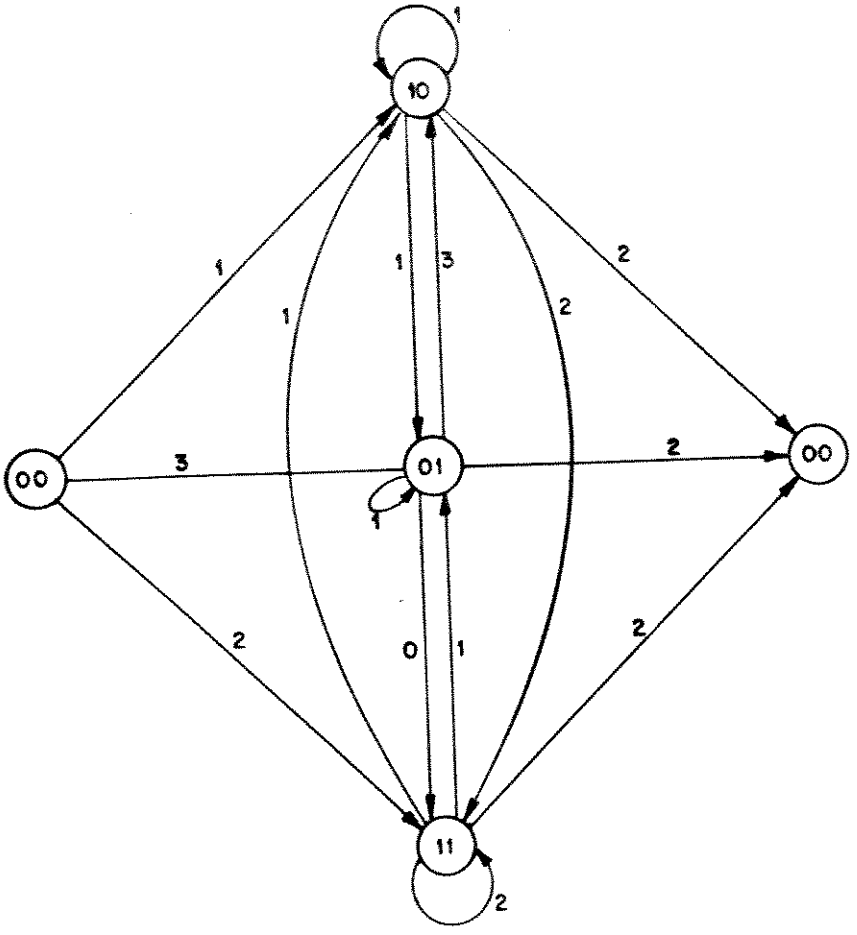

Fig. 3.4.1 - Diagrama de Fluxo em Rede de um Código Convolucional com R = 2/3.

A finalidade é encontrar uma matriz B cuja multiplicação $G_i.B$, onde i = 0, 1, faça com que a distância livredo código diminua a um valor desejado para a aplicação, e consequentemente o poder de correção do código também diminua. O procedimento será feito somente para a matriz B, já que foi mencionado que a transformação A apenas ocasiona o embaralhamento dos bits, e não a queda da distância livre.

Seja então o código de memória unitária apresentado na Fig. 3.4.1. As matrizes geradoras deste código são:

$$G_o = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

cuja distância livre é três.

O procedimento que será utilizado para se encontrar a matriz "B" é o do diagrama de fluxo em rede.

Sabemos que o fluxo máximo é dado por: $\phi = n2^{k-1} = 3.2 = 6$. Considere a seguinte distribuição do fluxo em rede como mostrado na Fig. 3.4.2. Esta distribuição do fluxo permite encontrar os elementos das submatrizes geradoras.

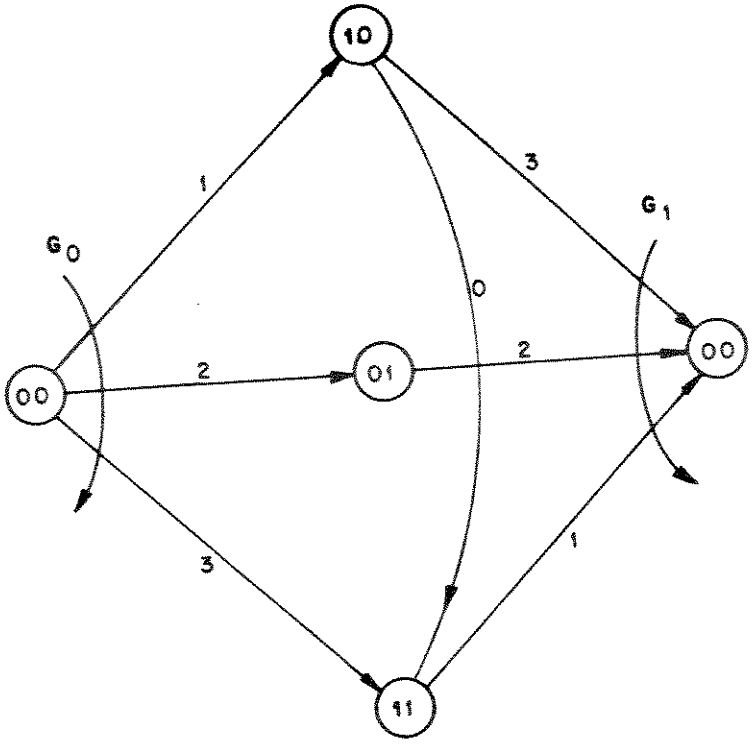

Fig. 3.4.2 - Diagrama Para as Matrizes Geradoras

Note que não necessariamente o fluxo tem que ser menor do que o fluxo máximo para que se consiga uma redução no valor da mínima distância livre do código. Para o caso em consideração as matrizes geradoras são dadas por:

$$G'_o = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad G'_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

tendo distância livre igual a 2. Esta distância corresponde à 2/3 da distância original.

A matriz B que leva a transformação do par $[G_o, G_1]$ para $[G'_o, G'_1]$ é dada por:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Observe também que a primeira linha de G'_1 é uma combinação linear das duas linhas de G'_o.

Podemos também obter a matriz "B", considerando as linhas das submatrizes G'_o e G'_1 como sendo a escolha de três vetores linearmente independentes e um vetor linearmente dependente, isto é, um quarto vetor que seja a combinação linear de dois outros vetores dentro deste espaço.

Logo, para este caso, teríamos que escolher estes vetores dentro de um espaço formado por:

001
010
011
100
101
110
111

Portanto, a partir deste processo observamos que quanto maior for o valor de n, maior será o grau de liberdade para se escolher os vetores e fazer as combinações lineares entre eles, encontrando assim as submatrizes geradoras que poderão ser usadas no processo de cifragem.

As transformações através das matrizes "A" e "B" podem ser de dois tipos:

As Transformações Isométricas:

São feitas com o par de matrizes (A,B) obtidas de permutações de colunas da matriz identidade.

As Transformações Não Isométricas:

São aquelas em que as matrizes A e B são obtidas não somente de permutações de colunas, mas também da combinação linear das linhas da matriz identidade.

Para estas transformações, o valor do determinante não é a unidade e apresenta características que devem ser tratadas diferentemente caso a caso.

Essas transformações, na sua maioria, conservam as características de distâncias do código somente em casos de termos o valor de "n" relativamente grande dando origem a um conjunto maior de vetores para que se possa escolher de quanto o poder de correção do código deverá ser reduzido.

As transformações não isométricas são capazes de causar maior redução ao poder de correção dos códigos, podendo inclusive diminuir o espaço vetorial em que operam as submatrizes geradoras, como é o caso do exemplo da Seção 2 deste Capítulo, onde a submatriz G'_1 possui duas colunas nulas, diminuindo, portanto, o espaço de 4 para 2 dimensões.

Para os sistemas criptográficos devemos nos ater ao cuidado de evitar uma redução do espaço vetorial em que estamos trabalhando. Já que quando ocorre uma queda na dimensão do espaço vetorial de um código, não se tem garantia que no processo de decodificação a mensagem possa ser recuperada, pois se a distância livre do código for um valor par, então neste caso se ocorrer uma coluna nula em uma das submatrizes geradoras, o poder e correção diminui de uma unidade. Já para para o caso da distância livre ser ímpar, se ocorrer uma coluna nula em uma das submatrizes geradoras, o código não perde no poder de correção, mas apenas deixa de detectar algum possível erro.

3.5 - VETOR ERRO

O vetor erro na decodificação y_e citado na Seção 3.2 representa mais uma função armadilha.

Este vetor erro apresenta características não lineares, entendendo-se por não linearidade a não equivalência na disposição dos un's entre os vetores erros de transmissão e decodificação, dependendo unicamente da matriz de transformação B utilizada no processo.

Se a transformação for isométrica temos então que a matriz inversa de "B" apresentará determinante também igual à unidade. Assim, o vetor erro de transmissão, quando multiplicado pela inversa de B apresentará equivalência na disposição dos un's no vetor.

Tomemos como exemplo a inversa de matriz B de dimensão 3 x 3 para o código de taxa R = 2/3. A matriz B e sua inversa são dadas
por:

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

e

$$B^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Seja por exemplo y_i = [001]. Então $y_i B^{-1}$ = [111] que é maior que a capacidade de correção do código.

Fazendo uso deste fato na Figura 3.5.1, podemos observar a "não linearidade" entre os vetores y_i e y_e. Mais exemplos serão apresentados no Capítulo 4 deste trabalho.

y_i	y_e
000	000
001	111
010	010
011	101
100	110
101	001
110	100
111	011

Fig. 3.5.1 - Disposição do Vetor Erro

Assim, como o código ótimo com taxa R=2/3 possui distância livre igual a 3, então somente um erro poderá ser adicionado. Logo y_i = [001, 010, 110] consequentemente $y_i.B$ = [101, 010, 110].

CAPÍTULO 4

ANÁLISE E AVALIAÇÃO DO MODELO DO CSCP

4.1 - INTRODUÇÃO

Neste Capítulo serão apresentados os resultados obtidos através da implementação do CSCP. Esta implementação foi dividida em duas etapas, a saber: uma de cifragem/codificação e uma segunda de decifragem/decodificação. Estas duas etapas serão vistas separadamente no decorrer deste Capítulo.

Os códigos escolhidos são todos convolucionais ótimos de memória unitária. Com a intenção de observarmos o comportamento do CSCP quanto à sua vulnerabilidade é que utilizaremos inicialmente códigos com pequena complexidade, para depois então, utilizarmos códigos com uma maior complexidade, consequentemente aumentando o grau de dificuldade de implementação e de quebra do criptosistema.

Serão considerados cinco códigos para análise e avaliação do modelo CSCP. Os três primeiros, apenas para observar o comportamento do sistema, que são os códigos com taxa: R = 2/3, R = 2/4 e R = 2/5. Em seguida foram considerados os códigos de taxa R = 7/14 e R = 8/24, este último código em particular é o código de Golay e portanto classificado como perfeito.

A análise do comportamento do código mediante o efeito das tranformações impostas pelas matrizes "A" e "B", como visto no Capítulo 3, é mostrado de forma qualitativa. Onde buscamos enfatizar os critérios adotados para as conclusões aqui apresentadas.

Em seguida são apresentados os algoritmos usados para a montagem do sistema, bem como são feitas as suas descrições em

detalhes.

Por fim mostraremos a análise de complexidade do modelo baseado no tempo de processamento para a quebra do sistema, e a avaliação dos diferentes graus de dificuldades impostas ao criptoanalista.

4.2 - AVALIAÇÃO DAS MATRIZES DE TRANSFORMAÇÃO

Como visto no Capítulo 3, as transformações "A" e "B" quando aplicadas às submatrizes G_0 e G_1 do código de memória unitária dão origem à dois tipos de resultados. Um deles caracterizados pelas transformações isométricas, que mantem inalteradas as caracteristicas de distribuição dos pesos de Hamming das palavras código, enquanto que o segundo resultado é caracterizado pelas transformações não isométricas reponsáveis pela alteração substancial dos pesos de Hamming das palavras-código.

Uma característica inerente à transformação advem do fato de que esta transformação é obtida via permutação ou das linhas ou das colunas da matriz identidade, enquanto que a característica da transformação não-isométrica engloba não apenas permutações de colunas ou linhas mas considera também combinações lineares entre elas.

Como visto anteriormente, a transformação "B" é sem dúvida a transformação responsável pela redução da distância livre do código a um valor desejável ou fixado pela aplicação, enquanto que à matriz "A" cabe influir no processo de embaralhamento dos bits que compõem a palavra código. Sendo assim, quase todo o esforço de determinação das transformações A e B ficará concentrado basicamente na determinação da transformação "B".

Para os códigos considerados neste trabalho, a Tabela 4.2.1 apresenta as submatrizes geradoras em notação hexadecimal, e enquanto que na Tabela 4.2.2 são apresentadas as distâncias livres dos códigos e o valor mínimo a que deve ser reduzida, para

garantir a confiabilidade do processo.

Para as transformações isométricas efetuadas sobre todas as submatrizes dos códigos considerados o que observou-se foi que: 1) não houve alteração do valor da distância livre dos códigos; 2) houve uma preservação das características destes códigos; 3) observamos somente uma troca na posição das palavras código ramo do código gerado para o código original.

Concluimos, portanto, que estas transformações não satisfazem os critérios para que possamos utilizá-las no modelo criptográfico aqui apresentado.

Para as tranformações não isométricas observamos que para determinadas transformações "B" é possível reduzirmos o valor da distância livre do código até o valor desejado ou fixado pelo problema.

k	n	G_0 G_1							
2	3							4 5	7 3
2	4							C D	7 A
2	5							19 15	16 1B
7	14		06FC 1727	0AD7 1E95	089D 3DE6	321A 14B6	0277 290E	1A64 0DF7	19E3 0F4D
8	24	5038CB F91370	DEFF9A CA97EE	B665F4 6D3CF9	A2A6D8 066764	713837 564213	1E5BF7 BD8835	20E8F7 193E04	C821AB C76829

Tabela 4.2.1 - Linhas das Submatrizes Geradoras (em Hexadecimal).

k	n	dfree	2/3 dfree
2	3	3	2
2	4	5	3
2	5	6	4
7	14	12	8
8	24	20	13

Tabela 4.2.2 - Distância Livre dos Códigos Convolucionais.

Para as transformações não isométricas algumas considerações devem ser feitas:

1 - Não existe uma lei de formação que caracterize a matriz de transformação para cada código isoladamente. Assim sendo, essas matrizes são de certa forma aleatórias, e encontradas como descritas no Capítulo 3, a partir das submatrizes G'_0 e G'_1.

2 - Para os códigos considerados podemos afirmar que não existe uma transformação que eleve o valor da distância livre, referente a classe dos códigos lineares. Podemos então concluir que a relação $R.d_{free}$ x R do código convolucional é convexa como mostrado gráfico 1 da Figura 4.2.1. Entretanto é desconhecido se entre os códigos não lineares é possível encontrarmos um código cuja distância livre, possa ser maior que a do código linear.

Caso isto aconteça, a relação R. d_{free} x R para o código não linear é mostrado no gráfico 2 da Figura 4.2.1. Note que o código ótimo linear é representado pelo ponto de inflexão A, ou o máximo da função convexa.

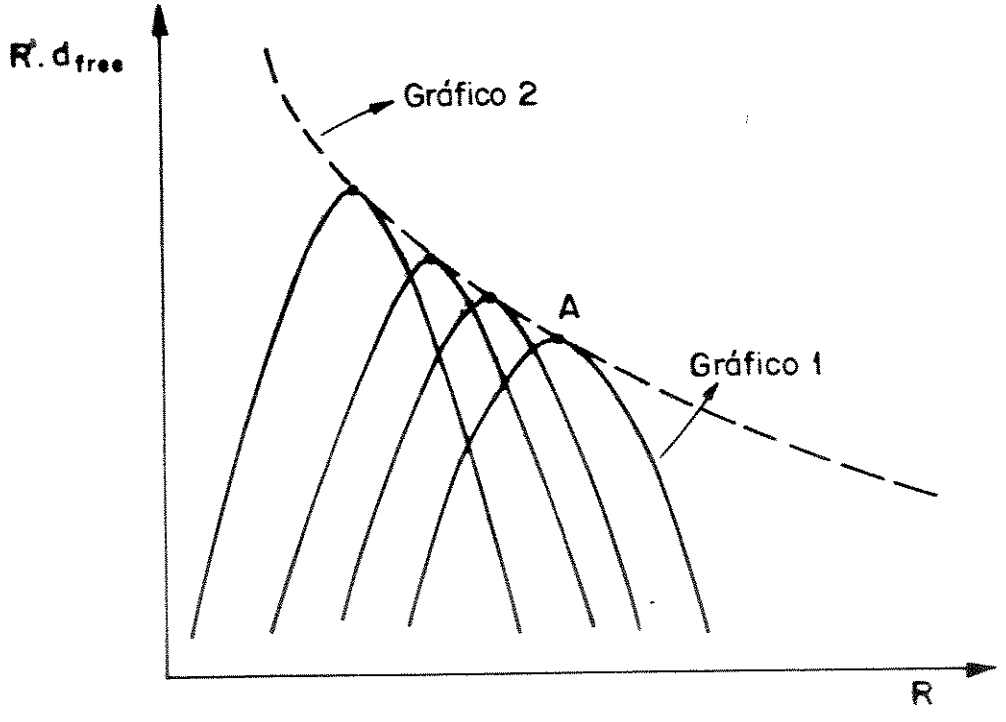

Fig. 4.2.1 - Curvas que Definem o Comportamento do Código Convolucional Ótimo.

3 - Para os códigos de taxas 2/3, 2/4 e 2/5, observamos que a maioria das transformações efetuadas pela matriz B que conduziram a uma redução no valor da distância livre, sem que ocorressse uma queda na dimensão do espaço de operação tem determinante igual a 1. Por outro lado, para qualquer valor de determinante, $d \neq 1$, da matriz B, dificilmente foi alcançado o objetivo sem que houvesse uma redução da dimensão do espaço de operação. Esta redução na dimensão do espaço, acontece quando uma ou mais colunas das submatrizes resultantes G'_0 e G'_1 são nulas. Este fato pode ser observado no caso considerado no Exemplo 4.2.1.

Exemplo 4.2.1 - Código de taxa R = 2/4.

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

Determinante de B = 2.

$$G'_0 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G'_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

No exemplo 4.2.1, observamos que o espaço vetorial tem 4 dimensões, representado por 4 colunas não nula nas duas submatrizes geradoras do código ótimo, diminuindo para um espaço vetorial de uma dimensão, pelo fato da matriz G'_1 possuir três colunas nulas.

4 - Para o código com taxa R = 8/24, observamos que apesar das inúmeras tentativas não foi conseguida uma transformação que reduzisse o valor da distância livre até 2/3 do valor original. Este fato, era de certa forma esperado, uma vez que este código é um

código perfeito , e portanto, com uma estrutura geométrica (reticular) muito forte.

5 - O vetor erro como definido no Capítulo 3, quando sob ação de transformações não isométricas não apresenta uma correpondência 1 a 1 com o vetor erro transformado. Este fato é analisado com maiores detalhes no sub-ítem seguinte.

4.2.1 - ANÁLISE DO VETOR ERRO

Como visto no modelo do CSCP apresentado no Capítulo 3, o vetor erro representa um grau de dificuldade a mais para o criptoanalista, porém a quantidade de erro a ser inserida depende única e exclusivamente do poder de correção do código, t, dado por:

$$t = \left[\frac{d^{*} - 1}{2} \right] \qquad (4.2.1.1)$$

O valor de d^{*} corresponde à distância mínima do código de memória unitária para uma sequência finita de informação ou equivalentemente ao primeiro comprimento de restrição.

Para o caso específico deste trabalho, foi considerado somente quatro transições na treliça. Assim sendo, a distância mínima do código deve aumentar uma vez que estaremos considerando como evento de erro aquele que divergirá do estado zero e retornará pela primeira vez ao estado zero após a quantidade fixada de janelas de tempo ou de transições na treliça. Como consequência a quantidade de erros que poderá ser inserido na mensagem aumenta.

Na Tabela 4.2.1.1 são mostradas as distâncias mínimas dos códigos considerados para quatro transições na treliça e seus

respectivos poder de correção de erros.

k	n	dmin	t	dfree
2	3	4	1	3
2	4	6	2	5
2	5	9	4	6
7	14	13	6	12
8	24	26	13	20

Tabela 4.2.1.1 - Distância Mínima dos Códigos Convolucionais em Quatro Janelas de Tempo.

Como mencionado no Capítulo 3, a disposição dos un's no vetor erro na decodificação (y_e) depende única e exclusivamente da matriz de transformação "B".

Como exemplo, seja o código de taxa R = 2/3, onde as submatrizes geradoras são:

$$G_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Seja "B" a matriz de transformação e sua inversa dadas por:

$$B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Se o vetor **x** correspondente à sequência de informação for dada por: **x** = (10, 01, 11, 00), então o vetor **y** que corresponde à sequência cifrada/codificada será: (111, 001, 000, 110). Por outro

lado, sabemos que em quatro janelas de tempo o código convolucional com taxa R = 2/3 é capaz de corrigir 1 erro. Pela Tabela 4.2.1.2 são apresentadas as possibilidades de inserção de erro no ponto de transmissão através do vetor y_i, com seu vetor erro correspondente na decodificação após a aplicação da transformação inversa de B.

i	y_i	y_e
1	001	111
2	010	010
3	011	101
4	100	110
5	101	001
6	110	100
7	111	010

Tabela 4.2.1.2 - Vetores Erros para a Matriz "B" do Código com R = 2/3.

Assim a palavra **y** poderá ser transmitida com os vetores y_i dados por:

y_i	y_e
010	010
101	001
110	100

Tabela 4.2.1.3 - Vetores Erros que Podem Ser Inseridos na Palavra Código do Exemplo.

e então, as suas respectivas palavras são dadas pela equação:

$$\mathbf{y}_{te} = \mathbf{y}_i + \mathbf{y} \qquad (4.2.1.2)$$

Note que pela Tabela 4.2.1.3 os vetores erros considerados são aqueles que têm peso de Hamming menor ou igual a capacidade de correção do código original. Este fato necessariamente deve ocorrer, pois se na transmissão inserirmos um erro através do vetor y_i, que dará origem a um vetor erro na decodificação e este vetor possuir peso de Hamming maior que o poder de correção do código original, este não será capaz de corrigir.

Além de tornar público G'_0 e G'_1, podemos também publicar a quantidade de erros que poderão ser inseridos na palavra a ser transmitida através do canal. Para o caso específico do exemplo acima somente 1 erro na decodificação poderá ser gerado pelo vetor de transmissão.

A seguir apresentaremos algumas transformações "B" e suas respectivas inversas, bem como os vetores $\mathbf{y}_i$ com seus correspondentes vetores erros, para que possamos exemplificar melhor as afirmações feitas anteriormente.

Suporemos que a matriz "A" será obtida através de uma tranformação isométrica, e portanto sem causar qualquer influência nos pesos de Hamming das palavras código ramo.

Código de Taxa R = 2/3 com t = 1.

$$B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

y_i	y_e
100	010
011	001
001	100
000	000

CÓDIGO DE TAXA R = 2/4 COM t = 2

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

y_i	y_e
1111	0011
1110	0110
1100	1000
1011	0100
1010	0001
1001	1010
0111	1100
0110	1001
0101	0010
0001	0101
0000	0000

Código com Taxa R = 2/5 com t = 4.

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

y_i	y_e
11111	00011
11110	00101
11101	11000
11100	11110
11011	10001
11010	10111
11001	01010
11000	01100
10110	11001
10101	00100
10100	00010

continuação da Tabela do exemplo anterior.

y_i	y_e
10011	01101
10010	01011
10001	10110
10000	10000
01111	10011
01110	10101
01101	01000
01100	01110
01011	00001
01010	00111
01001	11010
01000	11100
00111	01111
00110	01001
00101	10100

continuação da Tabela do exemplo anterior.

y_i	y_e
00100	10010
00011	11101
00010	11011
00001	00110
00000	00000

4.3 - ALGORITMOS DO MODELO DO SISTEMA SENDO PROPOSTO

Para a implementação deste modelo de criptosistema de chave pública, foram implementados não somente os algoritmos referentes ao codificador/cifrador e decodificador/decifrador, mas também algumas simulações tiveram que ser feitas; a saber: simulação para encontrar os vetores erros e simulação para encontrar a distância livre do código.

A simulação referente ao codificador\cifrador, a simulação para o cálculo da distância livre e a simulação para encontrar o vetor erro, foram feitas para análise do CSCP a ser implementado, enquanto que a simulação do decodificador\decifrador foi feita para a avaliação da complexidade de quebra do sistema pelo criptoanalista.

Nos ítens subsequentes analisaremos separadamente cada uma das etapas envolvidas no processo de implementação, onde serão realizados os comentários e análise de cada um dos algoritmos

apresentados.

4.3.1 - ALGORITMO DO CODIFICADOR/CIFRADOR

Neste algoritmo o objetivo é enviar através do canal a mensagem cifrada, pelo processo já mencionado no Capítulo 3. Neste estágio o emissor apenas utiliza as matrizes G'_0 e G'_1, postas em uma lista pública para enviar a informação para o receptor autorizado.

ALGORITMO

1 - Faça i = 0, correspondendo à primeira posição do vetor que contém a informação.

2 - Faça $\mathbf{y}_p = [\ x_{(k+1)} \ .\ G'_0 + x_k \ .\ G'_1]$; operação feita em módulo dois.

onde:

$\mathbf{y}_p$ → vetor que contém a palavra cifrada/codificada.

$\mathbf{x}_k$ → vetor $\mathbf{x}_{(k+1)}$ com uma unidade de tempo de atraso.

3 - Adicione o vetor erro à palavra cifrada.

4 - Verifique se i atingiu o valor máximo de janelas de tempo. Se sim, pare. Caso contrário, execute 5 e 6.

5 - Atualize a posição do vetor de informação.

6 - Retorne ao passo três.

4.3.2 - ALGORITMO PARA ENONTRAR A DISTÂNCIA LIVRE

O algoritmo para o cálculo da distância livre dos códigos convolucionais de memória unitária foi necessário no processo do CSCP, para que pudessemos encontrar as submatrizes geradoras que satisfizessem às exigências para serem utilizadas na aplicação, isto é, que as transformações efetuadas pela matriz B às submatrizes geradoras do código original, reduzisse o valor da distância livre ao limite estabelecido.

Na primeira etapa deste algoritmo é gerada a matriz de peso de transição de estados, para que possamos a partir desta matriz encontrar a distância livre do código.

Para garantir a confiabilidade desejada é necessário que sejam consideradas pelo menos cinco vezes o comprimento de restrição do código no processo de decodificação de Viterbi ou equivalentemente, cinco transições entre os estados na treliça [14].

ALGORITMO DE GERAÇÃO DE TRELIÇA

1 - Gerar o vetores $\mathbf{x}_i$ e x_j com comprimento k, correspondente ao número de entradas do codificador;

2 - Se i = j, então retorne para 1. Caso contrário, vá para 3;

3 - Faça $\mathbf{p}_0 = (\mathbf{x}_j . G'_0) + (\mathbf{x}_i . G'_1)$, operação feita em módulo dois que calcula a palavra de transição do estado $\mathbf{x}_i$ para o estado $\mathbf{x}_j$;

4 - Calcule o peso de Hamming de $\mathbf{p}_0$, e coloque este valor na posição $G_{[i][j]}$, que é a matriz de peso de transição de estado, e retorne para 1. Caso contrário, pare.

ALGORITMO PARA O CÁLCULO DO dfree

1 - Calcule a distância livre em duas janelas de tempo, inicializando a variável i = 2, e armazene o menor valor encontrado;

2 - Faça i = i + 1 e calcule o valor de d_{free} em i + 1 janelas de tempo. Verifique se existe algum valor menor que o armazenado. Caso exista, atualize esse valor e retorne para o passo 2;

3 - Se i = 6 pare o processo e armazene o menor valor de d_{free}, como resultado final.

4.3.3 - ALGORITMO PARA O CÁLCULO DO VETOR ERRO

Seja t o poder de correção do código. O que se pretende com este algoritmo é a geração de todas as possíveis entradas do vetor $\mathbf{y}_i$ com comprimento k, que multiplicados pela inversa da matriz de transformação B, gere todos os vetores com peso de Hamming menor ou igual ao poder de correção t.

Este procedimento se faz necessário, para que possamos introduzir o vetor erro na transmissão de forma a não comprometer o processo de decodificação, isto é para que não sejam utilizados vetores erro com peso de Hamming maior que o poder de correção do

código.

Para os códigos de taxas R = 2/3, R = 2/4 e R = 2/5, foram consideradas todas as possibilidades do vetor $\mathbf{y}_i$. Enquanto que para os códigos de taxas R = 7/14 e R = 8/24,foram considerados somente os 500 resultados de vetores que safizessem a condição de possuir o peso de Hamming menor ou igual ao poder de correção t.

ALGORITMO

1 - Gere o vetor $\mathbf{y}_i$.

2 - Multiplique cada um dos vetores $\mathbf{y}_i$ pela transformação inversa da matriz "B".

3 - Calcule o peso de Hamming do resultado da multiplicação do passo 2 e armazene os vetores que tiverem peso menor ou igual ao poder de correção do código original. Pare quando $\mathbf{y}_i$ for igual a 2^n.

4.3.4 - ALGORITMO DO DECODIFICADOR E DECIFRADOR

Neste ítem, o que se pretende é a realização da decifragem e a decodificação da mensagem. Para tal assumiremos o conhecimento das inversas das matrizes de tranformação A e B; com o objetivo de avaliar o tempo de processamento que o criptoanalista deverá dispor para decifrar a mensagem.

A estratégia considerada neste algoritmo foi a de otimizar o tempo de processamento através do uso de funções lógicas.

Nesta etapa é também gerada uma matriz de transição; que contém as palavras-código ramo (transição entre os estados). Note que o criptoanalista detém o conhecimento a priori deste fato, uma vez que G'_0 e G'_1 são conhecidas.

Na segunda fase então é feita a decodificação propriamente dita.

Na Seção 4.4 são apresentados os resultados dos tempos observados em três "máquinas" diferentes para que possamos então fazer uma avaliação do sistema em diferentes equipamentos e para diferentes finalidades, levando em consideração a relação custo/benefício, entre a eficiência da privacidade do sistema em relação aos ataques do criptoanalista.

ALGORITMO

1 - Faça $\mathbf{y}_{d_{[L]}} = 0$; onde L é o comprimento da mensagem;

2 - Multiplique a bloco da saída do canal pela inversa da matriz de transformação B;

3 - Decodifique a sequência proveniente de 2 através do algoritmo de Viterbi.

4 - Aplique a inversa da matriz de transfomação A a cada uma das possíveis soluções do passo três.

4.4 - RESULTADOS OBTIDOS

Nesta seção apresentaremos os resultados de tempo obtidos em três máquinas diferentes, bem como uma análise da quantidade de caracteres que podem ser enviados através de um canal com capacidade de 20 Kbits/seg.

No subítem seguinte será também apresentada a complexidade do algoritmo em três fases diferentes, para depois então ser apresentada a complexidade total envolvida. A análise é feita

considerando-se a condição de pior caso, isto é, quando o problema é classificado como um problema NP-completo.

4.4.1 - AVALIAÇÃO DO TEMPO DE PROCESSAMENTO

Nas tabelas que se seguem são apresentados os diferentes tempos de processamentos obtidos quando a utilização de três "máquinas".

As Tabelas 4.4.1.1, 4.4.1.2, 4.4.1.3 respectivamente apresentam tempos das máquinas AT - 286 com 16 MHz, AT - 386 com 20 MHz e uma estação de trabalho Sparc com 20 Mips.Todos os tempos são dados em segundos. Como visto na Seção 4.1, os códigos com taxasR = 2/3, R = 2/4 e R = 2/5, foram utilizados somente para uma avaliação primária do CSCP.

Máquina Código	AT - 286 com 16 MHz
8/24	304,34
7/14	20,00
2/5	0
2/4	0
2/3	0

Tabela 4.4.1.1 - Tempo de Processamento em Segundos no AT - 286.

Máquina Código	AT - 386 com 20 MHz
8/24	243,90
7/14	15,60
2/5	0
2/4	0
2/3	0

Tabela 4.4.1.2 - Tempo de Processamento em Segundos no AT - 386.

Máquina Código	Estação de Trabalho
8/24	9,72
7/14	1,65
2/5	0
2/4	0
2/3	0

Tabela 4.4.1.3 - Tempo de Processamento dado em Segundos para a Estação de Trabalho.

Observamos que com o aumento da complexidade relacionada com o número de entradas do código, conseguimos um aumento no tempo de processamento. Concluimos então, que para os códigos utilizados o que tem boa aplicação é o código de taxa R = 8/24.

Para as três máquinas temos quantidades de caracteres diferentes que podem ser transmitidos através do canal, sem que o criptoanalista consiga interceptar a mensagem. Note que para a análise apenas se está referindo aos códigos com taxas R = 8/24 e R = 7/14, já que para as outras taxas de códigos o tempo medido foi zero. O que de certa forma era esperado desde que a complexidade é pequena.

Na Tabela 4.4.1.4 são mostradas as quantidades de caracteres possíveis de serem transmitidos pelos equipamentos através de uma canal com capacidade de 20 Kbits/seg. Os valores são dados em

Kbits.

	AT - 286	AT - 386	Estação de Trabalho
R=8/24	6086,93	4878,02	194,33
R=7/14	400	312,09	32,99

Tabela 4.4.1.4 - Quantidade de Caracteres em Kbits através de um canal de 20 Kbits/seg.

Portanto, dependendo da aplicação desejada, devemos considerar a quantidade de caracteres através do canal e a relação custo benefício para a escolha de um equipamento para o sistema, a ser implementado.

Gostariamos de lembrar que os tempos aqui registrados referem-se à hipótese de que o criptoanalista detem o conhecimento das transformações aplicadas.

4.4.2 - COMPLEXIDADE DO MODELO

A complexidade total deste sistema pode ser obtida pela complexidade referente às 3 etapas principais descritas a seguir.

A análise de complexidade parte da hipótese de pior caso, isto é, estamos imaginando que o criptoanalista terá que realizar todos os passos necessários para chegar a uma possível solução.

Assim, podemos dividir o problema em três etapas de complexidades, a saber:

1 - A complexidade para resolver o problema

NP-completo, encontrando o código ótimo.

2 - A complexidade para encontrar a inversa das matrizes de transformação.

3 - A complexidade referente ao tempo de processamento, para decodificação da mensagem tendo o criptoanalista posse das inversas das matrizes A e B.

Gostariamos de ressaltar, que a complexidade deste sistema criptográfico, não cai, quando tentamos encontrar as inversas das matrizes G'_0 e G'_1(submatrizes geradoras do código convolucional).

Note que ao tentarmos encontrar a inversa da matriz geradora do código convolucional, temos que procurar a inversa de matriz semi-infinita da forma apresentada no Capítulo 2, equação 2.2.2.4.

Pelo fato desta matriz não ser quadrada, esta não obedece à lei comutativa, e assim, não se tem uma única matriz inversa de G'. E a complexidade para resolver este problema torna-se equivalente a resolver os três passos acima, quando utilizamos k(número de entradas do sistema) muito grande, que é o caso de interesse.

A seguir apresentamos a análise das complexidades parciais referentes às etapas 1,2 e 3.

COMPLEXIDADE DO NP-COMPLETO

Considerando que o fluxo de um código convolucional de memória unitária e a distância projetada são dados respectivamente por:

$$\phi = n \cdot q^{(k-1)} \cdot (q-1) \qquad (4.4.2.1)$$

e

$$d = \min \{ [2^{(p-1)} / (2^{p} - 1)] \cdot n/k \cdot (p + k) \} \qquad (4.4.2.2)$$

onde p é o número inteiro que produz o menor valor de d, então a complexidade relativa à solução do problema da Mochila é dado por:

$$2.\phi = a_0.d_0 + a_1.d_1 + \ldots + a_{(2^k-1)} .d_{(2^k-1)} \qquad (4.4.2.3)$$

$$\prod_{i=0}^{2^k-1} [2\phi/d_i] \cong [2\phi/d_0]^{2^k} \qquad (4.4.2.4)$$

COMPLEXIDADE PARA INVERSÃO DAS MATRIZES

Pelo processo de triangularização de matrizes, temos pelo menos:

$$2^{n(n-1)/2} \qquad (4.4.2.5)$$

matrizes inversíveis.

E se o algoritmo mais eficiente para o cálculo da matriz inversa tem ordem de complexidade nlogn [10], então a complexidade envolvida nesta etapa será:

1 - Caso A e B sejam da mesma ordem:

$$2^{k(k-1)/2}\ k\log k$$

2 - Caso a ordem de B seja muito maior que a ordem de A:

$$2^{n(n-1)/2}\ n\log n \qquad (4.4.2.6)$$

COMPLEXIDADE DO ALGORITMO DE DECODIFICAÇÃO

No caso do algoritmo de decodificação ser o Viterbi, temos que em cada estado são feitas 2^k comparações. Como no nosso caso foram consideradas 4 transições na treliça e que por hipótese o tempo gasto para cada comparação é Δ, e ainda que o número total de comparações por transição de estado na treliça é de 2^{2k}. A complexidade do algoritmo de decodificação é então:

$$4.\Delta.2^{2k} \qquad (4.4.2.7)$$

COMPLEXIDADE TOTAL

A complexidade total C do sistema é então dada pela composição das equações 4.4.2.3, 4.4.2.6 e 4.4.2.7. Logo C é dado por:

$$C = [2\phi/d]^{2^k-1} \cdot [2.2^{k(k-1)/2}.k\log k] \cdot 4\Delta 2^{2k} \cong$$

$$\cong \Delta\ 2^{(2k + k2^k + k^2)}\ k\log k \qquad (4.4.2.8)$$

quando as transformações A e B apresentam a mesma ordem de grandeza, isto é, k x .n

Note que $[2\phi/d]$ é da ordem de 2^k, pois,

$$2\phi = n\, 2^{(k-1)}$$

$$d \cong [2^{(p-1)}/2^p - 1]\ (n/k)\ (p + k)$$

consequentemente,

$$[2\phi/d] \cong 2^k$$

$2.2^{k(k-1)/2}$ klogk é da ordem de:

$\cong 2^{k^2}$ klogk

e o termo $4\Delta 2^{2k}$ é da ordem de:

$\cong 2^{2k}\ \Delta$

Para o caso da ordem de B ser maior que a ordem de A, fazendo um raciocínio análogo ao da equação 4.4.2.8 temos que C é da ordem de:

$$C \cong \Delta\ 2^{(2k + k2^k + (k/R)^2}\ (k/R)\log(k/R)$$

(4.4.2.9)

Portanto qualquer que seja o caso considerado a complexidade total do criptosistema proposto varia exponencialmente com o comprimento da entrada do sistema.

CAPÍTULO 5

CONCLUSÕES

Neste trabalho foi apresentado uma proposta de um sistema criptográfico de chave pública usando códigos convolucionais. Neste estudo foram apresentados: 1) análise de complexidade; 2) análise de desempenho; e 3) condições pelas quais as transformações devem satisfazer para que a segurança e privacidade do esquema, bem como da informação possa ser atingida.

No Capítulo 2 procuramos fazer uma abordagem geral sobre as ferramentas utilizadas para o desenvolvimento deste trabalho. Foram apresentados os conceitos básicos de codificação e sistemas criptográficos. Ainda neste Capítulo tentamos explorar o grau de dificuldade existente na decodificação dos códigos convolucionais de memória unitária e a dificuldade em encontrar os códigos ótimos pertencentes a esta classe.

No Capítulo 3 foram estabelecidas as propriedades das funções armadilhas, bem como toda análise para a implementação do CSCP sendo proposto. Nesta análise estabelecemos os tipos de transformações que podiam ser efetuadas pela função armadilha no código original; a saber: 1) as transformações isométricas, que são aquelas decorrentes somente da permutação de colunas da matriz identidade; e 2) as tranformações não isométricas que são decorrentes não somente da permutação de colunas da matriz identidade mas também da combinação linear de suas linhas.

A finalidade da função armadilha representada por uma dessas transformações foi de reduzir a distância livre do código convolucional a um valor especificado ou desejado para a aplicação.

Além destas transformações estabelecemos também um grau de dificuldade a mais na função armadilha, que foi a inserção de

erros na mensagem no ponto de envio, gerando no ponto de recepção uma quantidade de erros na mensagem com peso de Hamming menor ou igual ao poder de correção do código original, para que este pudesse ser capaz de decodificar corretamente a informação.

No Capítulo 4 foram apresentados os resultados obtidos da implementação do CSCP, a saber:

.Com relação a busca das matrizes de transformação que levassem o código original a uma redução na sua distância livre foram conseguidos os reultados a seguir.

..Com as transformações isométricas não alcançamos o objetivo de redução da distância livre. Neste caso,apenas conseguimos embaralhar as palavras código, mantendo inalterada a característica de distância do código original.

Com as transformações não isométricas obtivemos em alguns casos a redução da distância livre do código original a um valor desejado para a aplicação. Apesar de termos conseguido alguns exemplos para cada um dos códigos, com exeção do código com taxa R = 8/24, onde não conseguimos reduzir o valor do d_{free} ao valor desejado, não obtivemos uma lei de formação para as matrizes de transformação que efetuavam a redução no valor do d_{free}. Isto não foi possível, nem a nível de matrizes, nem a nível de sistemas.

Para as transformações não isométricas observamos ainda que para os códigos de taxas R = 2/3, R = 2/4 e R = 2/5, as matrizes de transformação B que efetuavam a redução na distância livre do código sem ocasionar uma redução da dimensãodo espaço vetorial do mesmo, apresentam como valor absoluto do determinantea unidade. Sendo que para determinantes diferentes deste valor não foi possível atingir o objetivo sem que houvesse uma redução na dimensão do espaço vetorial de operação das submatrizes geradoras.

Ainda sobre as funções armadilhas A e B, observamos que tanto para as transformações isométricas como para as não isométricas não foi atingido um aumento da distância livre dos códigos ótimos. O que nos leva a concluir que dentre os códigos pertencentes à classe linear, o ótimo é o ponto máximo da curva convexa.

Para o vetor erro observamos que quando aplicada uma transformação isométrica, este apresentou uma equivalência na

disposição dos un's entre os vetores erros de transmissão e decodificação, fato que não ocorreu quando aplicada uma transformação não isométrica, dependendo neste caso da inversa da transformação B utilizada.

Na análise de desempenho foram obtidos tempos diferentesde decodificação da mensagem, dependendo essencialmentedo tipo de "máquina" utilizada para este processo.

Estes resultados de tempos diferentes de decodificação diferentes pelo criptoanalista levam consequentemente a uma menor ou maior quantidade de informação que possa ser enviada através do canal. Neste caso quanto maior for o tempo utilizado pelo criptoanalista para a decodificação, maior poderá ser a quantidade de informação enviada através do canal.

Para a análise de complexidade total do sistema, foram levados em consideração três fatores, a saber:

1 - Resolver o problema do Knapsack, para encontrar o código convolucional ótimo de memória unitária;

2 - o grau de dificuldade para encontrar as matrizes A e B;

3 - a complexidade do algoritmo de decodificação e decifragem.

Assim a complexidade do CSCP foi obtida através do computo de três fatores, verificamos que esta complexidade cresce exponecialmente com a entrada do sistema k, e portanto torna o sistema seguro para aplicações com k muito grande.

Como sugestão para trabalhos futuros seria a implementação de um CSCP para dois níveis de Knapsack como sugerido em [6], isto é, além das funções armadilhas usadas aqui, acrecentar mais um nível de dificuldade, que consistiria da escolha de dois números primos entre si, p e w, e utilizar como submatrizes geradoras, depois do processo de transformação das matrizes A e B, o produto destas submatrizes por p sobre GF(w). Para este sistema a complexidade é

analisada através da teoria dos números.

APÊNDICE A

Aqui apresentamos algumas matrizes de transformação B, que reduziram a distância livre do código convolucional de memória unitária para as taxas apresentadas no trabalho. Como visto no Capítulo 4, não foi possível encontrar matrizes de transformação que reduzissem a distância livre do código com taxa R=8/24, a um valor desejado.

Apresentaremos aqui somente a matriz de transformação B, já que a matriz de Transformação A é obtida através de permutação de colunas da matriz identidade.

A representação é dada através das linhas da matriz em notação hexadecimal.

CÓDIGO DE TAXA R = 2/3

1 - B = 6,2,3

2 - B = 1,4,3

CÓDIGO DE TAXA R = 2/4

1 - B = C,B,5,A

2 - B = 7,B,8,D

3 - B = 6,D,5,A

4 - B = C,A,D,6

5 - B = 5,E,B,6

6 - B = C,6,1,2

7 - B = C, A, D, 6

8 - B = 3, D, E, B

CÓDIGO DE TAXA R = 2/5

1 - B = 10, D, 15, 14, A

2 - B = 16, 8, 10, 14, B

3 - B = 19, C, 7, 16, 17

4 - B = 15, A, 1E, 3, 1A

5 - B = 13, 1C, 12, D, 7

6 - B = 5, C, 12, 9, 3

7 - B = 7, 16, 5, A, 13

CÓDIGO COM TAXA R=7/14

1 -B =	2 - B =
1ACC	1F2B
3A99	2279
1593	68
3191	2128
39CA	385
2817	1604
2C8	2835
3210	30
171C	290
30EE	8B9
39E8	3810
31D1	F88
3001	373B
1FF9	903

3 - B =	4 - B =
3555	2FD3
3C18	212A
2272	3E59
339D	3C36
34A9	25A1
794	1FA2
32BA	3B34
3ABA	3EB1
2DF5	20
3326	57D
33A2	2856
2A6D	2222
3FA2	3B06
22EE	2620

```
5 - B = 2FD0
        2345
        269C
        3ABB
        3BD
        1EE1
        121E
        1B4A
        3247
        135
        36E0
        D1C
        1971
        371C
```

BIBLIOGRAFIA

[1] Diffie,W. & Hellman, E., M., "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, No. 6, november 1976, pp - 644 - 654.

[2] Diffie,W. & Hellman,E.M., "Privacy and Authentication: An Introdution to Cryptography", Proceedings of the IEEE, vol. 67, No. 3, march 1979, pp - 397 - 427.

[3] Finamore, W., "Criptografia", Internatinal Symposium on Information and Coding Theory, july 27 to august 1, 1987, Campinas - SP - Brazil.

[4] Gallager, R. G., "Information theory and Reliable Communication", Jonh Wiley & Sons, U.S.A., 1968.

[5] Massey,L.J., "Cryptography - A Selective Survey, Digital Communicatons", Elsevier Science Publishers B.V.(North - Holland), 1986.

[6] Palazzo,R.Jr., "Cryptography Systems Based on Trellis Codes", III Simpósio Braileiro de Telecomunicações, S.J. dos Campos - SP, setembro 2 - 4, 1985.

[7] Palazzo, R.Jr., "Propriedades das Funções Armadilhas e Análise de Cripto-Sistemas de Chave Pública Usando Códigos Convolucionais", V Simpósio Brasileiro de Telecomunicações, Campinas - SP, setembro 8 - 10, 1987.

[8] Palazzo, R.Jr., "Códigos de Treliça Fixos e Variantes no Tempo", Dissertação de Livre Docência, Universidade Estadual de Campinas, Campinas, 1987.

[9] Peterson, W., & Weldon, E.J.Jr., "Error Correcting Code", Halliday Litograph Corporation, U.S.A., 1961.

[10] Ribeiro, C.C., "Parallel Computer Models Combinatorial Algorithms", School on Combinatorial Optmization, Rio de Janeiro - Brazil, july 8 - 9 1985.

[11] Rocha, V.Jr., & Palazzo,R.Jr., "Códigos Corretores de Erros", International Symposium on Information and Coding Theory july 27 to august 1, 1987, Campinas - SP - Brazil.

[12] Shu-Lin & Costello, D.Jr., "Error Control Coding: Fundamental and applications", Prentice-Hall, U.S.A., 1983.

[13] Sloane, N.J.A., "Error Correcting Codes and Cryptography - Part I", Cryptologia, vol. 6, No 2, april - 1982, pp 128 - 153.

[14] Sloane, N.J.A., "Error Correcting Codes and Cryptography - Part II", Cryptologia, vol. 6, No 3, july - 1982, pp 258 - 278.

[15] Viterbi, A. & Omura, J.K., "Principles of Digital Communications and Coding", McGraw-Hill, Inc., U.S.A., 1979.