UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
MESTRADO PROFISSIONAL EM ENGENHARIA DA COMPUTAÇÃO E SISTEMAS

EDSON VERDE DE SOUSA

APLICAÇÃO DA TRANSFORMAÇÃO ARMADILHA À MATRIZ DE HADAMARD EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA

São Luís
2017

EDSON VERDE DE SOUSA

APLICAÇÃO DA TRANSFORMAÇÃO ARMADILHA À MATRIZ DE HADAMARD EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA

Dissertação apresentada ao programa do Mestrado Profissional de Engenharia da Computação e Sistemas da Universidade Estadual do Maranhão como parte dos requisitos para a obtenção do título de Mestre em Engenharia da Computação.

Orientador: Prof. Dr. João Coelho Silva Filho

Universidade Estadual do Maranhão - UEMA

São Luís

2017

Sousa, Edson Verde de

Aplicação da transformação armadilha à matriz de Hadamard em códigos convolucionais de memória unitária / Edson Verde de Sousa - 2017.

79 f.

Orientador: Prof. Dr. João Coelho Silva Filho.

Dissertação (Mestrado)–Mestrado Profissional em Engenharia da Computação e Sistemas, Universidade Estadual do Maranhão, 2017..

1. Transformação armadilha 2. Códigos convolucionais 3. Matriz de Hadamard 4.Distância livre. I.Título.

CDU 004:621.382

EDSON VERDE DE SOUSA

APLICAÇÃO DA TRANSFORMAÇÃO ARMADILHA À MATRIZ DE HADAMARD EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA

Dissertação apresentada ao Programa do Mestrado Profissional de Engenharia da Computação e Sistemas da Universidade Estadual do Maranhão como requisito parcial para a obtenção do título de Mestre em Engenharia da Computação.

Aprovado em 17 de Março de 2017

BANCA EXAMINADORA

Prof. Dr. João Coelho Silva Filho

Universidade Estadual do Maranhão - UEMA

Prof. Dr. Rogerio Moreira Lima Silva

Universidade Estadual do Maranhão - UEMA

Prof. Dr. Ricardo Coelho Silva

Universidade Federal do Ceará - UFC

À minha família, em especial a minha esposa Ielda, e meus meus filhos: Grazielli e Enzo, por serem as pessoas que me motivaram, a alcançar, mas uma etapa da minha vida.

AGRADECIMENTOS

Quero comunicar, primeiramente meus sinceros agradecimentos ao Prof.Dr.João Coelho Silva Filho, meu orientador, por seu total apoio, confiança, opinião, otimismo e amizade com que conduziu nosso trabalho.

Aos meus amigos do curso, que me ajudaram, a esclarecer algumas duvidas:Charles e David, e também não poderia deixar de citar, a amiga de pesquisa: Dayane.

Aos meus professores que me enriqueceram com suas sabedorias durante todo o curso.

A PECS/UEMA, pela oportunidade de desenvolver e concluir com sucesso a dissertação.

À banca examinadora, composta pelos membros:Prof. Dr. Rogério Moreira Lima Silva da Universidade Estadual do Maranhão - UEMA e Prof. Dr. Ricardo Coelho Silva da Universidade Federal do Ceará - UFC, sugestão e leitura que levaram ao enriquecimento deste trabalho.

A todos que de uma forma ou de outra contribuíram para a conclusão deste trabalho.

"Se você se sente menos e menos satisfeito com suas respostas a perguntas que você mesmo elabora mais e mais perfeitamente, é sinal de que sua capacidade intelectual está aumentando"

Charles West Churchman(1913-2004), cientista e filósofo ameicano; citado por J.E.Littlewood no livro A Mathematician's Miscellany(1953)

RESUMO

Essa trabalho se propõe, fazer a aplicação, e a analise da Transformação Armadilha - Matriz de Hadamard em códigos convolucionais de memória unitária(CCMU), códigos do tipo $(n, k, 1)$, com taxas: $r = \frac{2}{8}$, $r = \frac{3}{8}$, $r = \frac{2}{4}$ e $r = \frac{3}{4}$, sendo que essas transformações, ao serem aplicada às submatrizes geradoras, desses códigos convolucionais de memória unitária, produzem outras submatrizes geradoras capazes de diminuir, a capacidade de correção de erros do código através da análise da distância livre, esse processo de redução da capacidade de correção de erro dos códigos ocasionada pelo embaralhamento das colunas das submatrizes geradoras, proporciona um aumento no grau de privacidade da informação a ser enviada

Palavras-chave: Transformação Armadilha. Códigos Convolucionais. Matriz de Hadamard. Distância Livre.

ABSTRACT

This work proposes to apply and analyze the Hadamard Array - Array Transformation in convolutional unit memory (CCMU) codes, codes of type $(n, k, 1)$, with rates: $r = \frac{2}{8}$, $r = \frac{3}{8}$, $r = \frac{2}{4}$ e $r = \frac{3}{4}$, Transformations, when applied to the generating submatrix, of these unitary memory convolutional codes, produce other generating submatrices capable of decreasing, the code error correction capability through free distance analysis, this process of reducing the error correction capacity of Caused by the shuffling of the columns of the generating submatrix, provides an increase in the degree of privacy of the information to be sent

Keywords: Trap Transformation. Convolutional Codes. Matrix of Hadamard. Free Distance.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CSL	Circuito Sequencial Linear
CSCP	Cripto-Sistema de Chave Pública
CC_S	Codigos Convolucionais
MATLAB	MATrix LABoratory
SCILAB	Scilab Enterprises
MU	Memória Unitária
MUP	Memória Unitária Parcial
NP	Não-Polinomial
P	Polinomial
RSA	Rivest, Shamir e Adleman
V/S	Algoritmo de Viterbi/sequencial

Lista de Símbolos

$GF(q)$	Corpo finito
D	Operador de Retardo
$c(n,k,m)$	Código convolucional com n saídas, k entradas e m memórias
R	Taxa de um código
F_2	Grupo de Galois aditivo-modulo-2
d_{free}	Distância livre
d_{min}	Distância minima
t_{free}	Máxima capacidade de correção de erros
mdc	Máximo divisor comum
mod	Resto da divisão(usada na divisão de inteiros)
ϕ	Fluxo máximo
T_k	Transformação
T_k^{-1}	Transformação inversa
w_H	Peso de Hamming da palavra-código
G	Matriz geradora
G_i	Submatrizes Geradoras
A^+	Matriz Inversa a direita
A^-	Matriz Inversa a Esquerda
H_4	Matriz de Hadamard(Tipo Sylvester) de ordem 4×4
H_8	Matriz de Hadamard (Tipo Sylvester) de ordem 8×8
A_i	Matriz de ordem $n \times n$
$\otimes$	Produto de Kronecker
$\oplus$	Somador módulo-2
H^T	Matriz Transposta da Matriz de Hadamard
I_h	Matriz Identidade de ordem h

SUMÁRIO

1 Introdução

Em processamento digital de sinais a codificação significa a modificação e adaptação de características de um sinal para torná-lo apropriado para uma aplicação especifica, como por exemplo transmissão ou armazenamento de dados, dentro desse contexto existem três tipos de codificação. A codificação de canal, nesse tipo estão os códigos detectores ou corretores de erros, a codificação de fonte, onde se trabalha a criptografia e compressão de dados e finalmente a codificação de linha, onde se especificam a forma do sinal elétrico que será usado para representar os símbolos de informação.

A teoria dos códigos corretores de erro ou códigos de canal, dizem respeito a erros nas linhas de transmissão devido a vários fatores, como ruído térmico, ruído impulsivo e de ruído de intermodulação. Dependendo do meio de transmissão e o tipo de encriptação utilizado, podem existir outros tipos de anomalias, tais como arredondamento e atenuação de ruído e de diafonia e eco durante a transmissão.

Há duas estratégias diferentes para o tratamento de erros, a primeira utiliza-se os códigos detectores de erro, essa estratégia, consiste em incluir na transmissão de dados um número de bits redundantes, de modo a permitir ao receptor detectar que ocorreu um erro, mas não identifique o tipo de erro ou de onde surgiu, então é feita uma nova retransmissão de dados. Na segunda estratégia, utiliza-se os códigos de correção de erro, este envolve a mesma filosofia da primeira, incluem informação redundante, mas neste caso, essa informação é suficiente para permitir que o receptor deduza o qual foi o caractere que é transmitido com erro, portanto, o receptor tem a capacidade para corrigir uma série de erro limitado.

Os códigos detectores de erro, estão se desenvolvendo bastante em diversas áreas do conhecimento movidos pela contribuição, na Matemática, na Computação, na Engenharia Elétrica e na Estatística. A essa Teoria dos Códigos Corretores de Erros se apresenta de forma bastante dinâmica, envolvendo conhecimentos de varias áreas, o que torna motivador seu estudo. A Álgebra é uma dessas áreas e ao estudarmos os códigos sob o ponto de vista algébrico estamos mesclando conceitos da Matemática Pura e da Aplicada.

A teoria surgiu em 1948, com um trabalho de Shanon do Laboratório Bell, uma empresa norte americana de telefonia, a partir principalmente de um questionamento do porque das maquinas não localizarem a posição de um erro e corrigi-lo numa transmissão de informações, já que essas podiam detectar um erro.

Um código corretor de erros é, em essência, um modo organizado de acrescentar algum dado adicional a cada informação que se queira transmitir ou armazenar, que permita, ao recuperar a informação, detectar e corrigir erros. Diversos pesquisadores apresentaram seus inventos em congressos defendendo o uso de códigos combinados, que aproximam o desempenho dos codificadores ao limite teórico de Shannon, como por exemplo, Golay e Huffmann, citados em [8].

Os primeiros códigos foram projetados por Hamming em 1950 e ficaram conhecidos como códigos de Hamming, esses utilizavam o processo de verificação de paridade bit a bit. Mais tarde, passou-se a introduzir bits de redundância por blocos de informação, em um processo de verificação de paridade par e ímpar. Com o crescente aumento da demanda de serviços, a modernização dos equipamentos e uso em alta escala de serviços digitais, tornou-se necessário o desenvolvimento de novos codificadores, com o objetivo de melhorar o aproveitamento da banda disponível para transmissão de informações. Como exemplo de meios de transmissão de banda limitada, pode-se citar satélites e rádios.

A pesquisa executará à aplicação e analise sobre os códigos convolucionais ótimos de memoria unitária, primeiramente, define-se as propriedades das transformação armadilha, preocupando no enfoque da matriz de Hadamard, que posteriormente aplica-se a transformação armadilha - matriz de Hadamard em códigos, finalizando com a comparação das aplicações.

Na proposta feita por [21], as transformações armadilha foram definidas, sendo estas de três tipos: Grupos de Permutação, Matriz de hadamard e Matriz triangular Superior, e posteriormente apresentados os resultados da aplicação destas transformações aos códigos convolucionais ótimos de memoria unitária.

Neste contexto, propõe-se fazer a aplicação e a analise da Transformação Armadilha - Matriz de Hadamad em códigos convolucionais de memória unitária(CCMU) com outras taxas $r = \frac{2}{8}$, $r = \frac{3}{8}$, $r = \frac{2}{4}$ e $r = \frac{3}{4}$, e seguida fazer analise dos resultados, e compara-los. Considerando o exposto, este Dissertação se justifica pela necessidade de se fazer a aplicação da Transformação Armadilha - Matriz de Hadamad em códigos de

memória unitária(MU) com taxa diferentes.

Este Trabalho encontra-se organizado da seguinte forma:

No Capitulo 2 , é mostrada uma abordagem básica sobre os sistema de comunicação, com a descrição de cada um dos blocos, e também a introdução dos conceitos e definições de classes de códigos, tais como: código de arvore, código de treliça e códigos convolucionais.

No Capitulo 3, estuda-se os códigos convolucionais , as representações dos códigos convolucionais, os códios convolucionais e equivalentes, propriedades das distancias dos códigos convolucionais, os tipos de codificadores convolucionais, decodificação dos códigos convolucionais e o algorítimo de viterbi.

No Capitulo 4, classifica-se os sistemas criptográficos em: sistemas criptográfico convencionais e sistemas criptográfico de chave publica , dentro do sistema de chave publica, analisa-se o Kanapsak binário e o criptosistema RSA convencional de chave publica.

No Capitulo 5, analisa-se o método knapsak binário para sistemas criptográfico e chave publica, é feita a descrição do método do sistema criptográfico, e foram estabelecidas as propriedades das funções armadilhas, a finalidade da função armadilha representada pela matriz de Hadamard, foi de reduzir a distancia livre do código convolucional a um valor especifico ou desejado para a aplicação.

No Capitulo 6, é apresentada a transformação armadilha utilizado no desenvolvimento do trabalho e a matriz de Hadamard, inicialmente é analisada a matriz de Hadamard, do tipo sylvestre da ordem de 4 e 8 , em seguida a aplicação da transformação armadilha matriz de Hadamard e analise do efeito da aplicação das matrizes.

2 Códigos Corretores de Erro

O problema da confiabilidade na transmissão de informações vem representando um desafio constante para profissionais que atuam nesta área. Esta confiabilidade é referente á integridade de informação enviada através do meio de transmissão sob ação do ruido. Na Figura 2.1 é apresentado um esquema genérico de um sistema de comunicações, mostrando a descrição de cada um dos blocos:

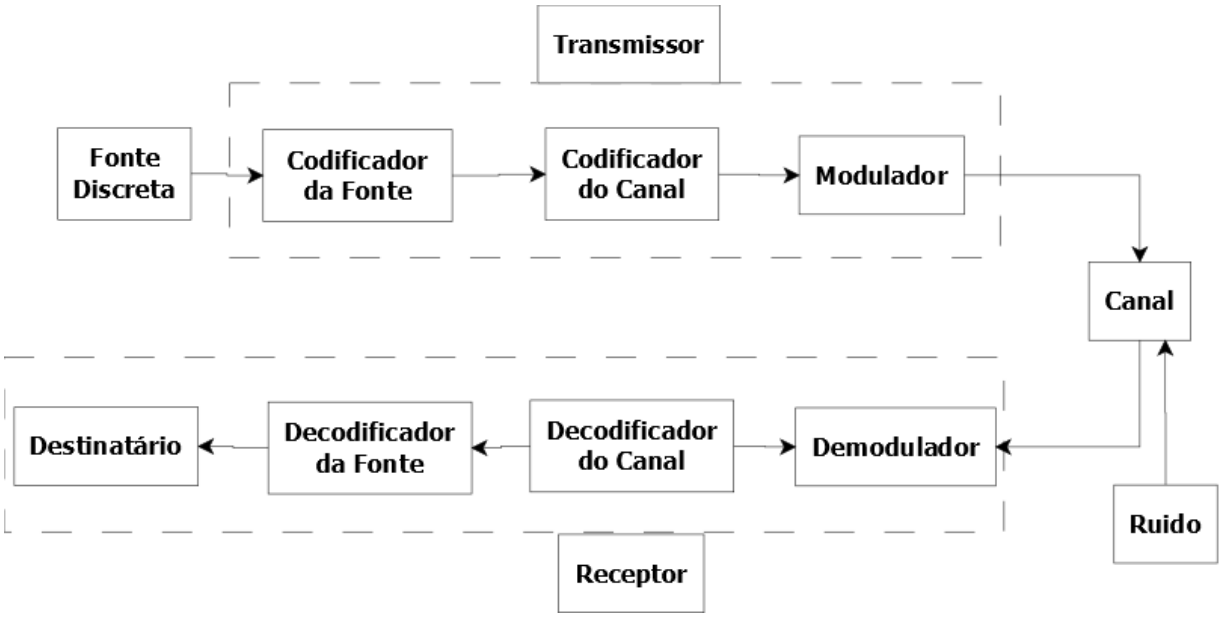

Figura 2.1: Sistema de Comunicações

Fonte - Gerador da informação a ser transmitida. Os dados ou informações originam a partir da fonte de informação. Esta informação materializa como um todo, sem perda de generalidade, símbolos ou mensagens discretas, N(representa o número de mensagens transmitidas) distintas e independentes que são transmitidas. A fonte de informação assim definido é chamado "fonte discreta sem memória, existem muitos tipos de fontes de informação, incluindo pessoas e máquinas, assim os símbolos ou mensagens podem tomar uma variedade de formas: uma sequência de símbolos letras discretas ou, magnitudes que variam no tempo, etc.; mas seja qual for a mensagem, a finalidade do sistema de comunicação é fornecer uma réplica mais ou menos exata de mesmo destino

Transmissor - Transforma o sinal na saída da fonte na forma adequada para a transmissão através do canal, sendo dividido em três etapas:

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 á informação, como exemplo no tratamento de imagens.

2. **Codificador do Canal** - Neste bloco que são adicionados dígitos de redundância ao sinal, com a finalidade de reduzir os efeitos do ruido sobre o sinal original.

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

Canal - O canal de transmissão, é a ligação eléctrica entre o transmissor e o receptor. Ele pode ser fios, cabos, fibra óptica ou simplesmente espaço livre em que sinal se propaga como uma onda eletromagnética.

Ruido- No processo de transmissão através de um canal de comunicação, que contêm sinais de informação cheguem ao seu destino distorcida que muitas vezes é impossível extrair as informações que possuem. Esta distorção é devido principalmente a dois fatores que estão sempre presentes nos canais físicos: a distorção produzida pelas características físicas do canal que produzem distorção de fase e de amplitude, e a distorção produzida por sinais aleatórios. Este tipo de sinais espúrios natureza aleatória é o que é conhecido pelo nome genérico de "ruído ". Na prática verifica-se que existem muitas potenciais fontes de ruído num sistema comunicação: fontes de ruído externas (natural e artificial) e de fontes ruído do sistema interno. Em particular, o ruído interno inclui uma importante classe de sinais perturbação gerado pelas flutuações espontâneas de corrente ou circuitos de tensão e elementos eléctricos, que representam uma limitação básica de transmissão ou detecção sinais

1. **Ruido Atmosférico** - Produzido por descargas elétricas associadas com trovoadas. Sabe-se geralmente como "estáticos". Abaixo de 100 MHz, a intensidade do campo é inversamente proporcional à frequência. No domínio do tempo que é caracterizada por picos de alta amplitude e curta duração; é um ruído impulsivo.

2. **Ruido Extraterrestre** - Devido ao sol e outros corpos quentes no céu. Devido à sua alta temperatura e proximidade com a terra, o sol é uma fonte intensa, mas a energia radiante, felizmente, localizada em uma ampla gama de frequências. As estrelas são

fontes de energia radiante de banda larga, embora mais distante e, portanto, menos intensa, sendo mais numerosos são coletivamente importantes como fontes de ruído.

3. **Ruído produzido pelo homem** -Ele inclui descarga de corona em linhas de alta tensão, produzidos pelos motores elétricos, sistemas de Diatermia, o ruído de comutação, etc. Mudando sistemas de ruído e de ignição é o tipo impulsivo, como o ruído atmosférico. Devido à iluminação fluorescente é um ruído muito freqüente em nosso ambiente

Receptor - O objetivo do receptor é extrair o sinal desejado a partir do sinal degradado transmitido pelo canal. À medida que os sinais recebidos são geralmente fracos os ruídos contaminados, uma primeira operação do receptor é a amplificação e filtragem de sinais de forma que possam ser processadas. Mas o funcionamento fundamental de que o receptor é "desmodulação"ou de "detecção", o qual é o inverso do processo de modulação no transmissor. Devido à degradação do sinal recebido, o receptor não pode reconstruir exatamente o sinal original, embora o tipo de degradação que resulta depende do sistema de modulação utilizada. A função do receptor consiste então em recuperar o sinal original, por meio das etapas:

1. **Demodulador** - A partir da forma de onda recebida do canal o demodulador aproxima a forma de onde que foi enviada pelo transmissor e entrega na saída a versão digital correspondente. Essa função consiste em fazer a operação inversa do modulador, sendo que neste caso o sinal que passa ao estagio seguinte possivelmente estará contaminado com ruido.

2. **Decodificador do Canal** - Nesta etapa 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 da Fonte** - Nesta etapa é devolvida a redundância do sinal, sendo ela removida no transmissor, antes que seja entregue ao destino.

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

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 sofre pertubações, gerando os chamados

erros aleatórios ou em surtos de vários erros de cada vez, nesse caso, diz-se que o canal tem memória. Com o objetivo de diminuir ou até eliminar os erros ocorridos através do canal, são utilizados códigos corretores de erros.

2.1 Código de Árvore

Dentre as classes de grafos, a classe das árvores é bastante importante para a conceituação dos códigos que serão descritos a seguir. Uma árvore $A = (D, M)$ é um grafo conectado sem ciclos 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. Mais formalmente, definimos uma arvore (D, M) μ-ária como sendo uma árvore tal que [19] :

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, Para
   (a) $D \in Z \;\; e \;\; D \geq 1$,
   (b) $M \in Z \;\; e \;\; M \geq 0$,
   (c) $\mu \in Z \;\; e \;\; \mu \geq 2$.

Uma árvore (D, M) μ-ária possuem μ^D vértices terminais com profundidade $D + M$. À cada ramo r desta arvores associaremos uma palavra de ramo com S símbolos do alfabeto de entrada de um canal discreto sem memoria. Seja R a taxa do código de arvore (D, M) μ-ária e N é comprimento da palavra-código. Então $\mu = 2^{NR}$ é o numero de ramos que divergem de cada vértice até uma profundidade D, λ o comprimento de restrição da árvore, isto é, o numero total de dígitos desde o vértice $D - 1$ até $D + M$, temos que λ é dado por: $\lambda = (M + 1) N$.

Defina como um código de árvore a quádrupla (N, R, D, M) , este código forma uma classe especial dos códigos de bloco. O código de arvore (N, R, D, M) considerado como um código de bloco consiste de μ^D palavras, onde cada uma tem comprimento $(D + M) N$. A taxa deste código é dada por:

$$\tilde{R} = \frac{\log_2 \left(\mu^D\right)}{(D + M) N} = \frac{D \log_2 \mu}{(D + M) N},$$

como $\mu = 2^{NR}$, assim

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

como em geral $D \gg M$, tem-se $\tilde{R} \cong R$.

2.2 Código de Treliça

Essa classe de código pertence a classe de código de arvore pela introdução da dependência entre os símbolos da fonte no processo de codificação. A introdução da dependência gera a necessidade de se estabelecer um procedimento de codificação para o código de arvore. A sequencia é determinada por $(\alpha_i)_{i=0}^{D-1}$, com, $i = 0$ sendo o limite Inferior e $i = D-1$ o limite Superior. Seja uma sequência de dígitos de informação n-ária, cada digito estará sendo associado um ramo que saí de cada vértice até a profundidade D. Para cada sequência $\alpha_0, \alpha_1, \alpha_2, ...\alpha_{D-1}$, estará sendo associado um caminho nesta arvore até a profundidade D.

Suponha que para algum $\gamma \in Z$, entre M e D + M, identificamos cada vértice da arvore (D, M) n-ária com os γ dígitos de informação anteriores, e somente o digito 0 será associado á cada vértice entre D e D + M. Dessa forma, fica estabelecido o esquema de codificação de um código de arvore (D, M, γ) n-ária, onde os vértices são enumerados.

Se esta arvore tem memória γ, no sentido de que para dois vértices na mesma profundidade, resultem em uma mesma sequencia codificada resultante, quando a mesma sequencia de dígitos de informação é aplicada à entrada do codificador a partir de quaisquer um desses vértices, então precisa manter somente uma dessas sequencias. Quando isso ocorre, é chamado de uma treliça (D, M, γ) n-ária.

A quíntupla (N, R, D, M, γ) define um código de treliça para um canal discreto sem memória, onde a cada ramo desta treliça associa-se uma palavra ramo (código) consistindo de N símbolos do alfabeto de entrada do canal. A quíntupla (N, R, D, M, γ) é uma classe especial do código de arvore definido pela quádrupla (N, R, D, M) a restrição de memória γ continua sendo valida, o mesmo acontecendo com a taxa R do código.

2.3 Código Convolucional

Por meio do conjunto de medias, utilizado por Shannon na demonstração do Teorema Fundamental da Teoria de Informação, que o conjunto dos códigos de arvore definido quadrupla (N, R, D, M) para um canal discreto sem memoria coincide com o conjunto dos códigos de treliça definido pela quíntupla $(N, R, D, M, \gamma = D + M)$. Assim é considerado somente o conjunto dos códigos de treliça.

Seja $\alpha_0, \alpha_1, \alpha_2, ...\alpha_{D-1} = (\alpha_i)_{i=0}^{D-1}$, a sequencia dos dígitos de informação a ser codificada, onde cada α_i está associado a um ramo da treliça a ser seguido pela sequencia até a profundidade D, e seja $\beta_0, \beta_1, \beta_2, ...\beta_{D+M-1} = (\beta_i)_{i=0}^{D+M-1}$ a sequencia codificada correspondente á sequencia de informação $\alpha_0, \alpha_1, \alpha_2, ...\alpha_{D+M-1} = (\alpha_i)_{i=0}^{D+M-1}$, assim a representação do codificador, segundo esse modelo é dado por:

$$\beta_i = \Gamma(\alpha_i, \alpha_{i-1}, \alpha_{i-2}, ...\alpha_{1-\gamma}, i)$$

Sendo que Γ é uma função arbitraria dos β_i, que pode ser linear ou não linear. Quando a função Γ é linear os códigos de treliça formam uma classe especial de códigos chamados de convolucionais.

De um modo formal, suponha que o alfabeto de entrada de um canal discreto sem memoria é formado pelo corpo finito $F_q\,(GF(q))$ com q elementos, onde q é um numero primo ou uma potencia de numero primo. Suponha que $\mu = q^L$, tal que α_i forma uma L-úpla sobre $F_q\,(GF(q))$. Assim a taxa desse código é dada por:

$$R = \frac{\log \mu}{N} = \frac{L}{N}.\log q$$

O codificador convolucional : (N, R, D, γ) com

$$\beta_i = \Gamma\left(\alpha_i, \alpha_{i-1}, \alpha_{i-2}, ...\alpha_{1-\gamma}, i\right),$$

é tal que :

1. $\alpha_i \in [GF(q)]^L$
2. $\beta_i \in [GF(q)]^N$ e $\Gamma(\ \ , i)$ são funções lineares que podem ser representadas por:

$$\beta_i = \alpha_i G_0(i) + \alpha_{i-1} G_1(i) + \alpha_{i-2} G_2(i) +, ... + \alpha_{i-\gamma} G_\gamma(i),$$

sendo $G_j(i)$ matrizes L x N sobre $F_q\,(GF(q))$, para o caso invariantes no tempo, $\Gamma(\ ,i) = \Gamma$, e assim obtem-se:

$$\beta_i = \alpha_i G_0 + \alpha_{i-1} G_1 + \alpha_{i-2} G_2 +, ... + \alpha_{i-\gamma} G_\gamma,$$

sendo G_j matrizes L x N sobre $F_q\,(GF(q))$. A representação matricial para os códigos convolucionais, como ocorre com os códigos de bloco. Nesta direção, para conseguir uma estrutura matricial para os códigos convolucionais, para o caso particular de invariante no tempo, basta que a sequencia de informação e a sequencia codificada sejam similares, e portanto a matriz geradora G do codigo convolucional invariante no tempo é dada por:

$$\beta_i = \alpha_i G_0 + \alpha_{i-1} G_1 + \alpha_{i-2} G_2 +, ... + \alpha_{i-\gamma} G_\gamma$$

e

$$\beta_{[\mathbf{0},\mathbf{D+M}]} = \alpha_{[\mathbf{0},\mathbf{D}]} = \begin{pmatrix} G_0 & G_1 & \cdots & \cdots & G_\gamma & 0 & 0 \\ 0 & G_0 & G_1 & \cdots & G_{\gamma-1} & G_\gamma & \cdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & G_0 & G_1 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & G_0 & \cdots & G_\gamma \end{pmatrix}$$

3 Códigos Convolucionais

Em codificador convolucional a mensagem m a ser codificada é representada pela sequência $m = m_1, m_2, ...m_i, ...$, onde cada m_i representa um bit e i é o índice correspondente ao tempo, e a sequência codificada $c = G(m)$, onde $c = c_1, c_2, ..., c_i, ...$, sendo então representado na Figura 3.1

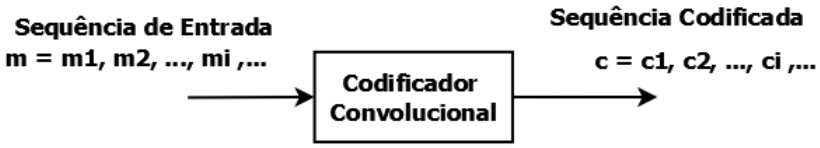

Figura 3.1: Codificador convolucional

Considerando que m_i assuma os valores 0 ou 1 de forma equiprovável, e que a presença de um ou outro seja estatisticamente independente, o bit presente não depende de seu predecessor nem influência seu sucessor. O codificador transforma cada sequencia m em uma única sequência código $c = G(m)$. A sequência codificada U pode ser segmentada em uma sequência de palavras ramos $c = c_1, c_2, ..., c_i, ...$, e cada palavra ramo c_i é um simbolo código binário, chamado frequentemente de simbolo do canal, bits do canal ou bits códigos.

Ao contrário dos bits da mensagem de entrada m, os bits dos símbolos códigos não são independentes. Embora uma sequência m defina uma única sequência c, uma característica dos códigos convolucionais é que um elemento m_i de uma sequência de entrada m não é suficiente para definir seu simbolo código associado c_i em U, uma vez que a codificação de cada elemento m_i não é apenas função do próprio m_i mas também do elemento m_{i-1} que o procede.

A Figura 3.2 representa um codificador convolucional genérico, sendo constituído de um registador de deslocamento de kK estágios e somadores módulo-2, onde temos que K determina a extensão de influência, ou seja a profundidade do código e k sendo os bits de informação. A profundidade de código convolucional é definida como sendo o máximo numero de bits codificados que podem ser afetados por um único bit de entrada.

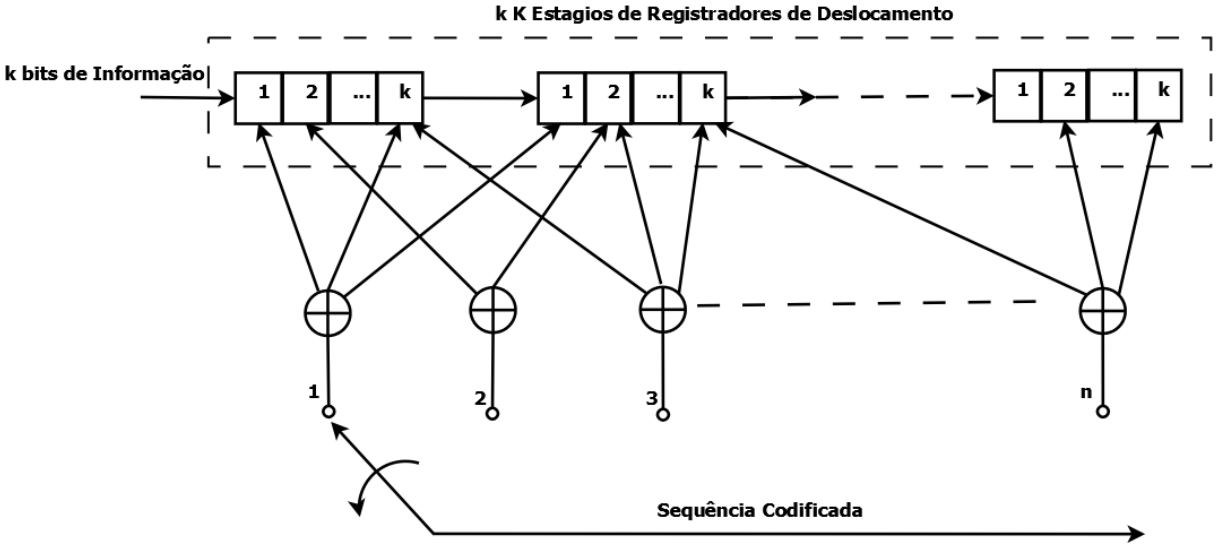

Figura 3.2: Diagrama de blocos de um codificador convolucional genérico

A cada unidade de tempo, k bits são deslocados para os primeiros k estágios do registrador, todos os bits no registrador são deslocados k bits para direita e as saídas dos n somadores são sequencialmente exibidas para a obtenção do simbolo códigos ou bits códigos. Portanto, n bits códigos são obtidos na saída para cada grupo de k bits de mensagem, a taxa do código é $\frac{k}{n}$ bits de mensagem por bits códigos, onde $k < n$.

3.1 Codificador Convolucional

A codificação convolucional envolve memória, portanto um codificador convolucional, pode ser implementado como um circuito lógico-sequencial. Em um CSL, os sinais escolhidos de um corpo finito $GF(q)$ são aplicados simultaneamente a todos os terminais de entrada em instantes discretos de tempo, este circuito consiste de uma rede de interconexões entre um numero finito de componentes primitivos.

Dois tipos de componentes primitivos são considerados em codificadores convolucionais: os somadores para implementar adição modulo-q e os elementos de memória para atrasar (ou armazenar) um sinal de entrada por uma unidade de tempo.

Os conceitos de codificador e de código convolucional estão intimamente relacionados, mas são distintos. Sendo que na literatura existem duas interpretações distintas, mas complementares. Uma destas interpretações define o código primeiro, como um subespaço C, k-dimensional de um espaço vetorial n-dimensional sobre um corpo apropriado, e define o codificador como uma matriz de dimensões $k \times n$ cujas linhas são uma base

para o código, esta interpretação está descrito em [9] e [14]. Outra interpretação define o codificador como um CSL com k entradas e n saídas, e então define o código como o conjunto de sequencias de saída geradas pelo codificador para todas as possíveis sequencias de entrada.

3.2 Representações de codificadores convolucionais

Para descrever um código convolucional é necessário caracterizar a função de codificação de tal forma que, dada uma sequência de entrada k, seja possível determinar a saída codificada n. Uma abordagem muito comum para apresentar o processo da codificação convolucional é por meio de exemplo, sendo assim as subsecções apresentadas a seguir mostram as formas de representação mais comuns bem como as características básicas de um codificador convolucional e seu princípio de operação

3.2.1 Representação Discreta

Um codificador convolucional de taxa $r = \frac{k}{n}$ é representado por nk sequencias de geradores: $g_i{}^j = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \cdots, g_{i,m}^{(j)})$ para $i = 1, \cdots, k$ e $j = 1, \cdots, n$, onde i representa a entrada, j a saída e m o número de memórias. Evidentemente, a operação do codificador convolucional é a convolução discreta da sequência de informação com as sequencias de geradores expressa como sendo

$$v^{(j)} = \sum_{i=1}^{k} (u^i * g_i^i)$$

para $j = 1, \cdots, n$, onde $*$ é a operação de convolução. Cada uma das n sequencias codificadas

$$v^{(j)} = (v_0^j, v_1^j, v_2^j, \cdots),$$

onde v_i^j tem comprimento n, para $j = 1, ..., n$ pode depender de cada uma das k sequencias de informação

$$u^{(i)} = (u_0^i, u_1^i, u_2^i, \cdots),$$

onde u_i^j tem comprimento k, para $i = 1, \cdots, k$. A sequência geradora composta para a i-ésima entrada do codificador é definida como:

$$g_i = g_{i,0}^1, g_{i,0}^2, ..., g_{i,0}^n, g_{i,1}^1, ..., g_{i,1}^n, ..., g_{i,m}^1, ..., g_{i,m}^n.$$

A convolução discreta pode ser expressa de uma forma mais compacta através de uma multiplicação de matrizes. As k sequências de informação podem ser escritas como sendo a sequência de informação:

$$u^{(i)} = (u_0, u_1, u_2, \cdots) = (u_0^1, u_0^2, ..., u_0^k, u_1^1, ..., u_1^k, \cdots),$$

onde $u_t = (u_t^{(1)}, ..., u_t^{(k)})$ é bloco de informação no instante de tempo t. Da mesma forma, a palavra-código obtida a partir das n sequencias codificadas é dada por:

$$v^{(i)} = (v_0, u_1, v_2, \cdots) = (v_0^1, v_0^2, ..., v_0^k, v_1^1, ..., v_1^k, \cdots),$$

onde $v_t = (v_t^{(1)}, ..., v_t^{(k)})$ é o bloco codificado no instante de tempo t. portanto, a codificação convolucional pode ser escrita como $v = uG$, onde G é uma matriz geradora semi-infinita definida como:

$$\mathbf{G} = \begin{pmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_m - 1 & G_m & \\ & & G_0 & \cdots & G_m - 2 & G_m - 1 & G_m \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

os espaços em branco indicam zeros e as matrizes G_l são da forma

$$\mathbf{G_l} = \begin{pmatrix} g_{1,l}^1 & g_{1,l}^2 & \cdots & g_{1,l}^n \\ g_{2,l}^1 & g_{2,l}^2 & \cdots & g_{2,l}^n \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,l}^1 & g_{k,l}^2 & \cdots & g_{k,l}^n \end{pmatrix}$$

onde $g_{i,l}^j$ é a sequencia geradora entre a i-ésima entrada e a j-ésima saída no l-ésimo instante de tempo.

Aplicação: [24]Considere a mensagem $M = 0110$ aplicada no codificador da Figura 3.3. Para este codificador pode-se escrever os vetores conexão g_1 e g_2 para os dois somadores como:

$$g_1 = 101,\ g_2 = 111 \ \text{ e } \ 110111.$$

A resposta ao impulso deslocada é representada na forma matricial, e a sequência código c é obtida por $c = m \cdot G$.

$$c = m \cdot G = (0110) \begin{pmatrix} 11 & 01 & 11 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 \end{pmatrix} = 001110101100.$$

3.2.2 Representação Polinomial

As sequências de geradores $g_i^{(j)}$ com $i = 1, \cdots, k$ e $j = 1, \cdots, n$ são finitas e podem ser representadas como polinômios de grau finito em um operador de retardo D. Os polinômios geradores para um codificador (n, k, m) são representados por

$$g_i^{(j)}(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)} D + \cdots + g_{i,m}^{(j)} D^m, \text{ onde } i = 1, \cdots, k \text{ e } j = 1, \cdots, n.$$

As sequências de entrada e saída do codificador podem ser escritas como polinômios utilizando o operador de retardo, como:

$$u^{(i)}(D) = u_0^i + u_1^i D + u_2^i D^2 + \cdots$$

e

$$v^{(i)}(D) = v_0^i + v_1^i D + v_2^i D^2 + \cdots,$$

respectivamente, sendo a convolução no domínio do tempo equivalente à multiplicação no domínio das transformadas, a operação de decodificação é escrita por:

$$v^{(j)}(D) = \sum_{1=1}^{k} u^{(i)}(D) g_i^{(j)}(D). \tag{3.1}$$

Um codificador (n, k, m) no domínio das transformadas, pode ser representado por uma matriz G de dimensão $k \times n$ denominada *matriz geradora polinomial* e é dada por

$$G_1 = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \cdots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \cdots & g_2^{(n)}(D) \\ \vdots & \vdots & \vdots & \vdots \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \cdots & g_k^{(n)}(D) \end{bmatrix},$$

onde $g_i^j(D)$ é um polinômio de tal forma que, para um dado i e para todo $j = 1, \cdots, n$, capturam a influência do i-ésimo registro de deslocamento sobre todas as n saídas.

A operação de codificação pode ser escrita em termos da matriz geradora na forma polinomial como:

$$v(D) = u(D)G(D). \tag{3.2}$$

O vetor $u(D)$ tem k componentes $u^{(i)}(D)$, para $i = 1, \cdots, k$. Analogamente, o vetor $u(D)$ tem n componentes $v^{(i)}(D)$, para $j = 1, \cdots, n$.

Aplicação: [24]Considere a mensagem $M(x) = x + x^2$ aplicada pelo codificador da Figura 3.3. Para este codificador pode-se escrever os vetores conexão $g_1(x)$ e $g_2(x)$ para os dois somadores como:

$$g_1(x) = 1 + x^2 \text{ e } g_2(x) = 1 + x + x^2,$$

os polinômios geradores, em termo de operador atraso D, são:

$$g_1(D) = 1 + D^2 \text{ e } g_2(D) = 1 + D + D^2,$$

onde o termo de mais baixa ordem corresponde ao estágio de entrada do codificador. A sequência de saída do codificador é obtida por $c(D) = m(D)g_1(D)$ intercalado com $m(D)g_2(D)$. assim, a mensagem em termos de operador atraso, tem-se:

$$m(x) = x + x^2 \text{ e } m(D) = D + D^2,$$

onde o primeiro bit a entrar no codificador é aquele que corresponde ao termo de menor ordem. Assim,

$$m(D)g_1(D) = (D + D^2)(1 + D^2) = D + D^2 + D^3 + D^4$$

$$m(D)g_2(D) = (D + D^2)(1 + D + D^2) = D + D^4$$

$$m(D)g_1(D) = D + 1D^2 + 1D^3 + D^4$$

$$m(D)g_2(D) = D + 0D^2 + 0D^3 + 1D^4$$

$$c = 11101011.$$

3.2.3 Representação Gráfica

As representações gráficas proporcionam um melhor entendimento da operação de codificação convolucional. No Diagrama de Estados o codificador convolucional é um circuito logico linear é implementado com o uso de registros de deslocamento, o conteúdo desses registros é chamado de estado do codificador. Todos os possíveis estados do codificador e as entradas que causam as transições entre estes estados podem ser compactamente representados pelo diagrama de estados. A Figura 3.4 mostra o diagrama de estados para o codificador (2, 1, 2) da Figura 3.3.

Consideremos em todos os diagramas de estados, que o bit mais recente e o menos significativo no estado e cada elipse no diagrama é um estado. Assim, o estado

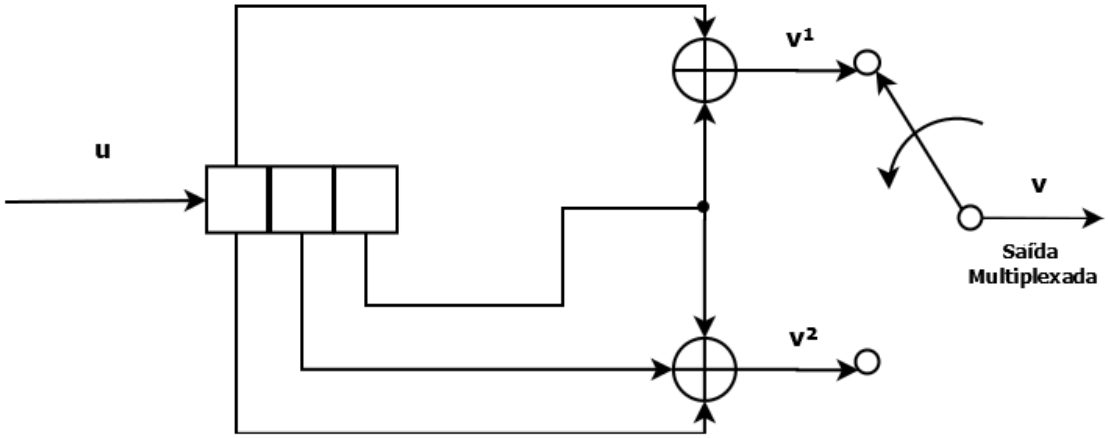

Figura 3.3: Codificador (2, 1, 2)

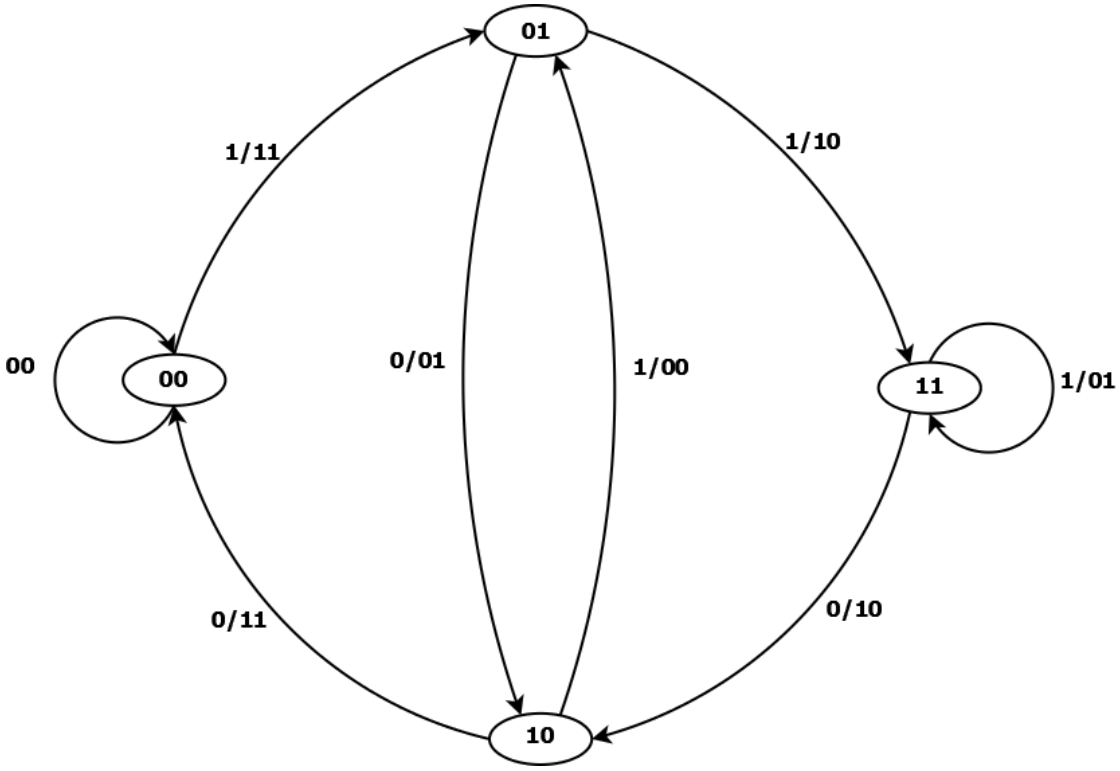

Figura 3.4: Diagrama de Estados-Codificador (2, 1, 2)

deste codificador no instante de tempo i é formado pelos bits contidos na segunda e nas primeiras células e o estado no instante de tempo $i-1$ é formado pelos bits contidos na terceira e nas segundas células, cada bit que entra no codificador produz uma saída (dois bits nesse caso) que depende do bit de entrada e dos bits armazenados na memória.

A operação de codificação corresponde a uma sequência de transições de estados, começando por um estado conhecido.Em geral, existem 2^k transições saindo de cada estado, que correspondem a sequência de informação u_i, de comprimento de k. Para uma dada sequência de informação, que correspondem a sequência codificada é obtida

percorrendo-se o diagrama de estados, a começar pelo estado todo nulo.

A escolha do estado todo nulo como estado inicial é equivalente a inicializar o codificador com todos os elementos de memória dos registros de deslocamento preenchidos por zeros. Na Figura 3.4, o diagrama de estados da sequência de saída gerada pelo codificador para a sequência de informação u = 1011001 é v = 11010010101111. O diagrama de treliça outra representação muito comum para o codificador convolucional é o diagrama de treliça. O diagrama de treliça para o codificador (2, 1, 2) é mostrado na Figura 3.5, onde existem quatro possíveis estados, assumindo que o estado inicial é o estado todo nulo 00.

Após m = 2 unidades de tempo, a treliça expande-se para incluir todos os quatro estados. De cada estado saem duas transições; as linhas tracejadas correspondem à entrada dos bits 0 e as linhas contínuas correspondem à entrada do bit 1, cada linha é acompanhada pelos correspondentes bits de saída, os últimos m estágios da treliça são utilizados para o retorno ao estado inicial 00, e correspondem à entrada de m blocos de bits zeros. Cada palavra código é um caminho que começa no estado 00, no instante de tempo t = 0, e termina no estado 00.

Diagrama de Estados

Um codificador convolucional é um circuito lógico linear, e é implementado com o uso de registro de deslocamento. O conteúdo desses registros é chamado de *estado do codificador*.

Todos os possíveis estados do codificador e as entradas que causam as transições entre estes estados podem ser compactamente representados pelo *diagrama de estados*. O diagrama de estados mostra a transição entre os estados, os k bits do bloco de informação que provocam cada transição e os n bits do bloco de saídas do codificador, [4] [15]. A Figura 3.4, mostra o diagrama de estados para o codificador (2, 1, 2). Consideremos que, em todos os diagramas de estados, que o bit mais recente é o menos significativo no estado. Cada elipse no diagrama é um estado. Assim, o estado deste codificador no instante de tempo i é formado pelos bits contidos na segunda e primeira células e o estado no instante de tempo $i - 1$ é formado pelos bits contidos na terceira e segunda células. Cada bit que entra no codificador produz uma saída (dois bits nesse caso) que depende do bit de entrada e dos bits armazenados na memória.

A operação de codificação corresponde a uma sequência de transições de es-

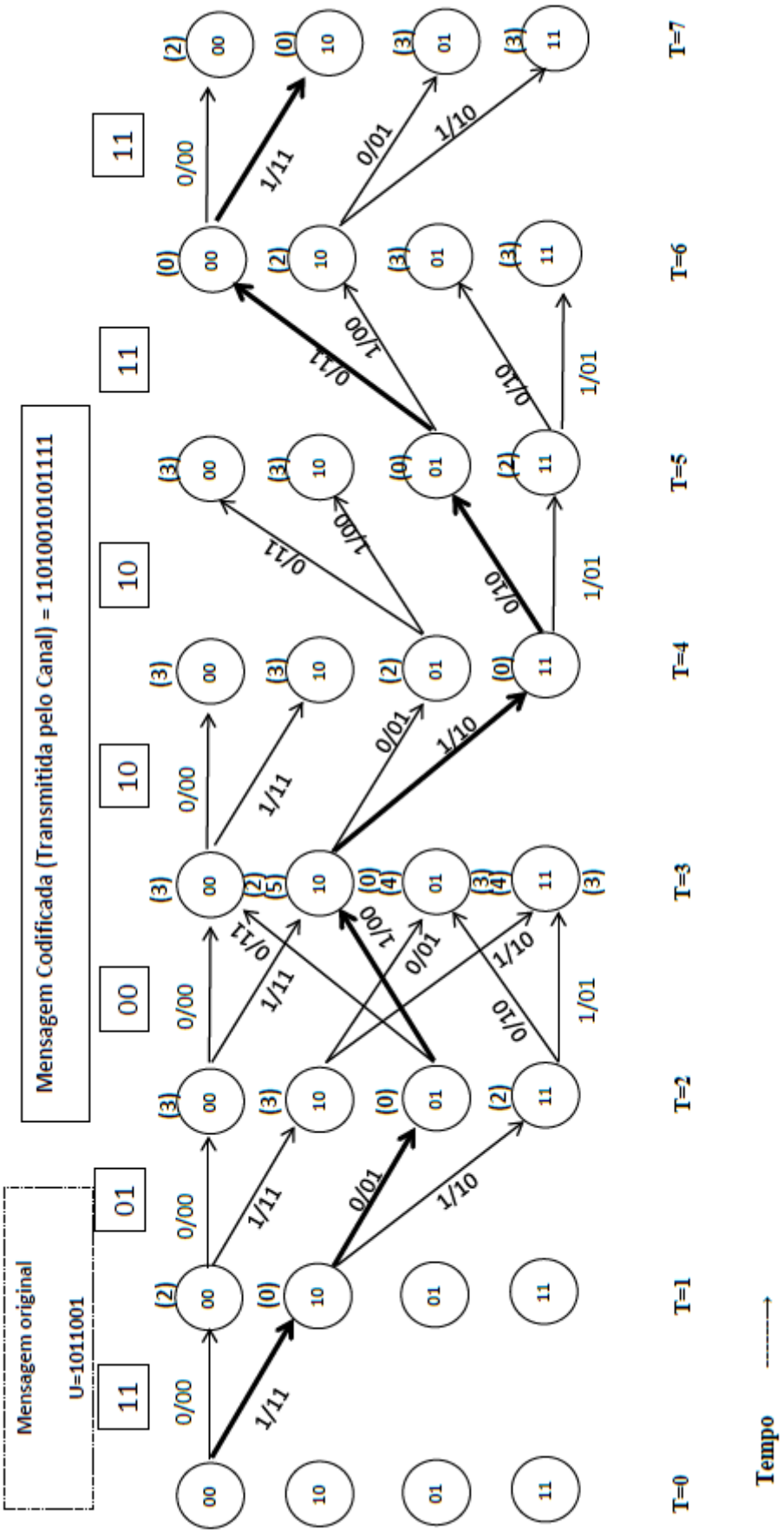

Figura 3.5: Diagrama de Treliça - Codificador (2, 1, 2)

tados, começando por um estado conhecido. Existem, em geral, 2 transições saindo de cada estado, que correspondem à sequência de informações u_i, de comprimento k. Para uma dada sequência de informação, correspondendo à sequência codificada é obtida percorrendo-se o diagrama de estados, a começar pelo estado todo nulo.

Diagrama de Treliça

Outra representação que é muito comum para o codificador convolucional é o *diagrama de treliça*. Na Figura 3.5 é mostrado o diagrama de treliça para o codificador (2, 1, 2), o qual existem quatro possíveis estados. Estado inicial é o estado todo nulo $a = 00$. Após $m = 2$ unidades de tempo, a treliça expande-se para incluir todos os quatro estados. De cada estado saem duas transições; as linhas tracejadas correspondem à entrada do bit 1. Cada linha é acompanhada pelos correspondentes bits de saídas. Os últimos m estágios da treliça são utilizados para retorno ao estado inicial $a = 00$, correspondem à entrada de m blocos de bits zeros. Cada palavra código é um caminho que começa no estado $a = 00$, no estante de tempo $t = 0$, e termina no estado $a = 00$.

3.3 CC_s Equivalentes

Um código convolucional é gerado por varios codificadores, e uma noção de equivalência de codificadores pode ser introduzida, temos que dois codificadores convolucionais são equivalentes se eles geram o mesmo código.

Construção de um codificador equivalente ao codificador da Figura 3.3 com taxa $\frac{2}{4}$ e com diferentes número de memorias. A solução trivial é utilizarmos a mesma matriz geradora discreta semi-infinita. Esta matriz, quando associada a um codificador $(4, 2, 1)$ mostrado na Figura 3.6 , pode ser escrita como:

Forma Matricial:

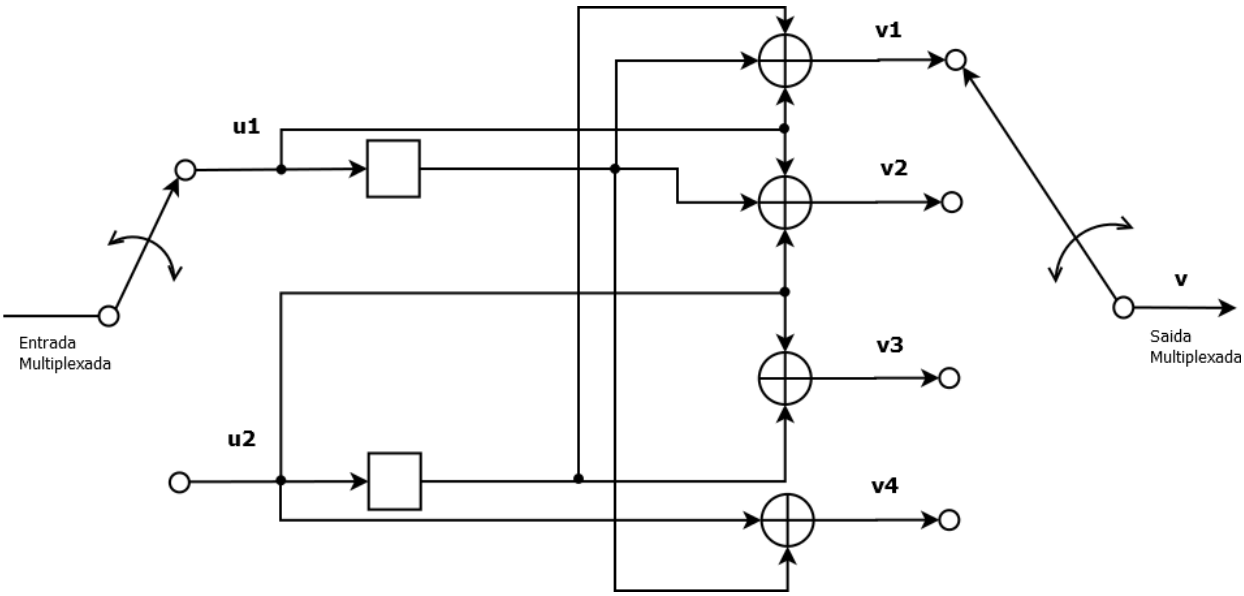

Figura 3.6: Codificador (4, 2, 1)

$$\mathbf{G} = \begin{pmatrix} 1101 & 1100 & & & \\ 0011 & 0111 & & & \\ & 1101 & 1100 & & \\ & 0011 & 0111 & & \\ & & 1101 & 1100 & \\ & & 0011 & 0111 & \\ & & & \ddots & \ddots \end{pmatrix}$$

Forma Polinomial:

$$\mathbf{G} = \begin{pmatrix} 1+D & 1+D & 0 & D \\ D & 1 & 1+D & D \end{pmatrix}$$

O CSL é constituído deste codificador a partir de qualquer uma destas duas matrizes. O codificador $(4, 2, 1)$ é apresentado na figura 3.6, e o diagrama de estados do codificador $(4, 2, 1)$ está mostrado na Figura 3.7.

3.4 Propriedade da Distância dos Códigos Convolucionais

O desempenho dos códigos convolucionais esta diretamente ligado ao algoritmo de decodificação e também as propriedades de distância do código, as distância mais

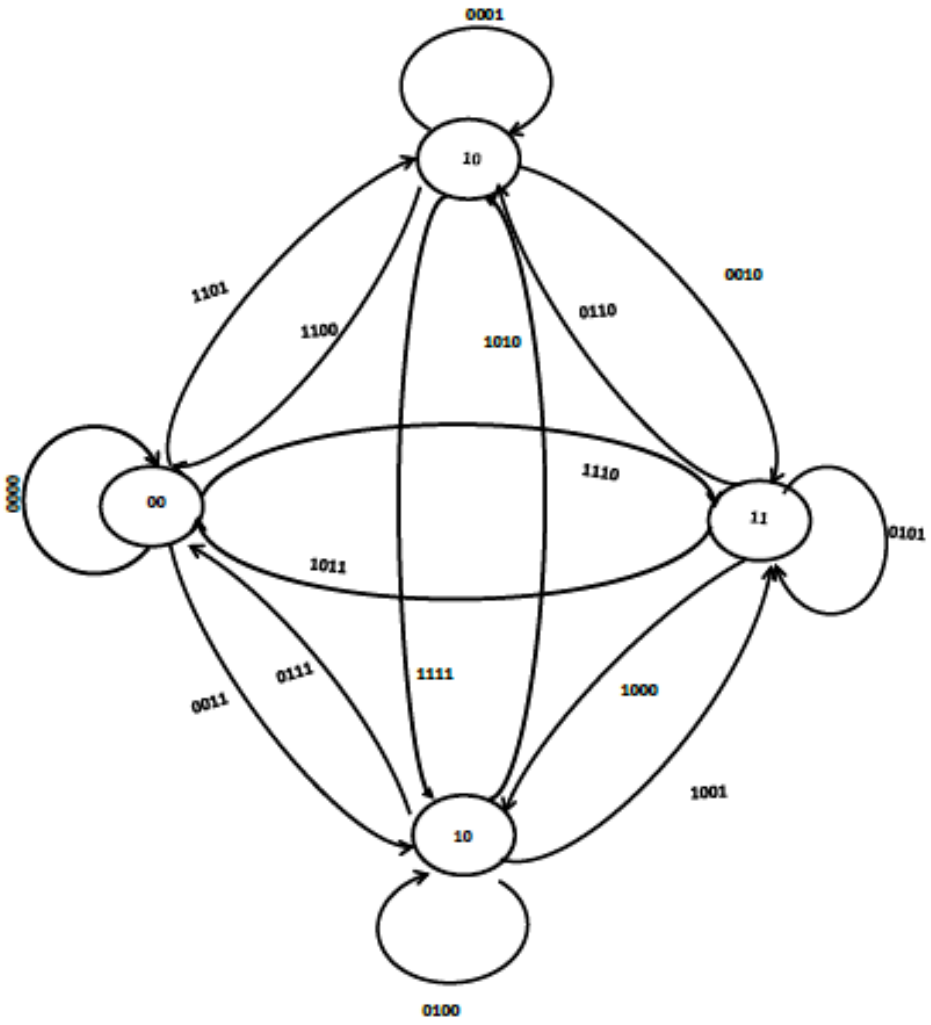

Figura 3.7: Diagrama de Estados do Codificador(4, 2, 1)

importantes que são utilizadas no estudo dos códigos convolucionais são:

- distância livre denotada d_{free} ou d_{∞} ;

- distância de coluna denotada d_i;

- distância mínima denotada d_{min};

A mais importante dentre tais distancias é a distância livre, definida por Costello em [3] e [4], pois é através desta que é estabelecida a capacidade de correção ou de detecção de erros associados aos códigos convolucionais.

3.4.1 Distância Livre de um Código Convolucional

A distância livre, d_{free}, de um código convolucional como sendo a menor distância de Hamming entre quaisquer duas sequencias codificada proveniente de duas sequências

de informação distintas.

Simbolicamente, temos que :

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

$$= min\{w_H(v) : u \neq 0\}$$

$$= min\{w_H(uG) : u \neq 0\}$$

, onde v, é são as sequencias codificadas correspondentes às sequencias de informação u e u', respectivamente. Portanto, a distância livre é a distância de Hamming mínima entre quaisquer duas palavras códigos distintas. Por causa da linearidade dos códigos convolucionais, a distância livre é também o menor peso da palavra código não nula. Os códigos gerados por codificadores equivalentes possuem o mesmo d_{free} e, consequentemente, a mesma capacidade de correção. Este parâmetro é usado para analisar o desempenho de algoritmos de decodificação baseados no princípio de máxima verossimilhança, compreendendo o algoritmo de Viterbi, que será a apresentado na próxima seção, e também o algoritmo de decodificação por síndromes

Exemplo: Através do diagrama de estados da Figura 3.4 ou do diagrama de treliça da Figura 3.8 , pode-se facilmente verificar que a distância livre do código gerado pelo codificador do código convolucional (2, 1, 2) é $d_{free} = 5$. O caminho de d_{free} é constituído de três transições de estados, a saber:

$$a = 00 \longrightarrow b = 01 \longrightarrow c = 10 \longrightarrow a = 00,$$

referentes à palavra código 110111 e à sequência de informação $u = 100$.

Um código convolucional tem distância livre ótima se sua distância livre é igual ou superior a qualquer outro código convolucional de mesma taxa e mesmo comprimento de restrição de saída. A partir da definição da distancia livre, ou seja d_{free} é possível definir que a *máxima capacidade de correção de erros* t_{free} de um código convolucional é dada por :

$$t_{free} = \left\lfloor \frac{d_{free} - 1}{2} \right\rfloor . \tag{3.3}$$

Os códigos gerados por codificadores equivalentes (dois codificadores que geram o mesmo código) possuem o mesmo d_{free} e, consequentemente, a mesma capacidade de correção.

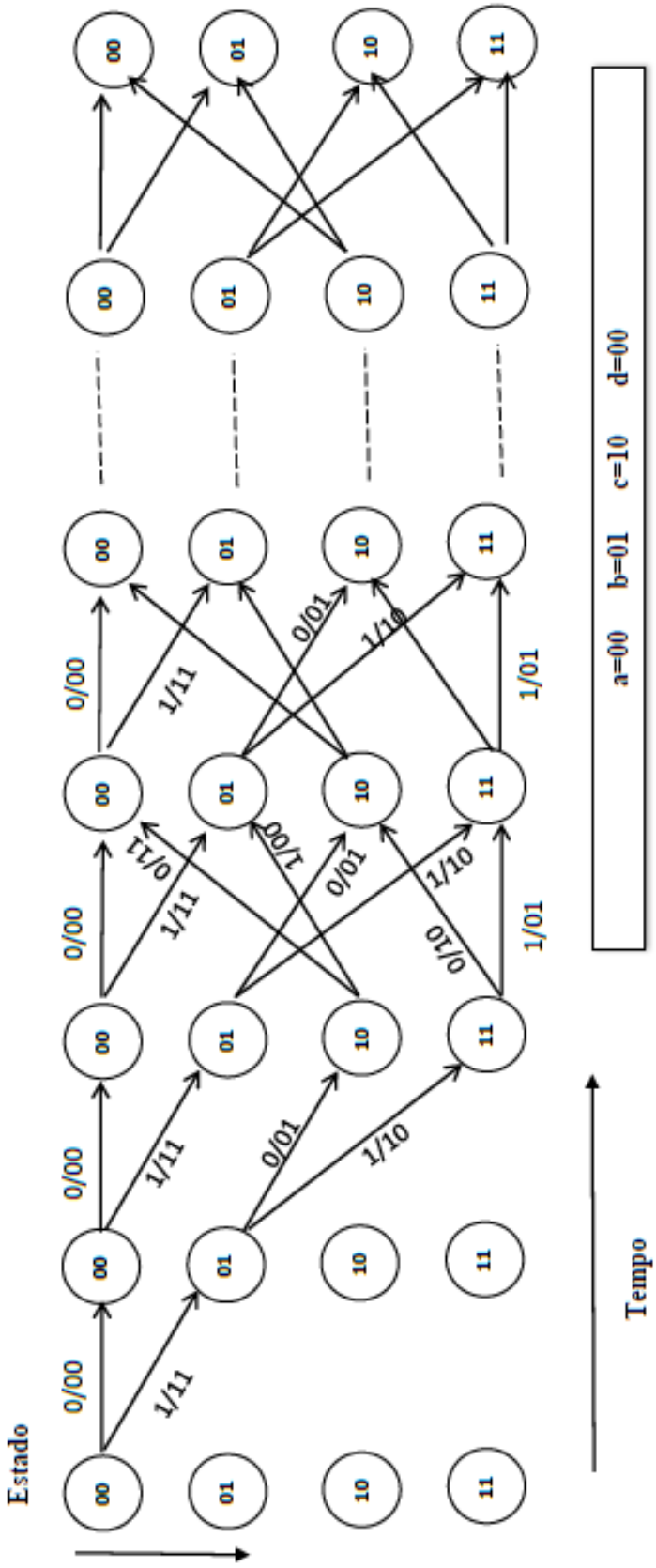

Figura 3.8: Diagrama de treliça

O parâmetro t_{free} é utilizado para analisar o desempenho de algoritmos de decodificação baseados no princípio de máxima verossimilhança, compreendendo o algoritmo

de Viterbi, que será apresentado na próxima seção.

3.5 Tipos de Codificadores Convolucionais

Neste Capitulo é mostrado alguns tipos de codificadores convolucionais, sendo os mesmos importantes para o desenvolvimento desse trabalho, apresentando um breve estudo sobre os seguintes codificadores: codificadores catastróficos, não catastróficos, codificadores de memória unitária e codificadores de memória parcialmente unitária.

3.5.1 Codificadores Catastróficos e Não Catastróficos

Os codificadores catastróficos são aqueles que geram uma palavra-código de peso finito para uma sequência de informação de peso infinito,[13]. Nestes casos, existe uma probabilidade não nula de que quando esta palavra-código é transmitida através de um canal ruidoso, um numero finito de erros introduzidos pelo canal, provoque um número infinito de erros de decodificação. Esta situação é conhecida como propagação catastrófica de erros. Massey e Sain em [14] propuseram um teorema para determinar se um codificador é catastrófico ou não. Segundo este teorema, para evitar a propagação catastrófica de erros, os codificadores de taxa $\frac{1}{n}$ devem satisfazer a condição:

$$mdc[g^{(j)}(D), j = 1, 2, \cdots, n] = 1,$$

onde mdc denota o máximo divisor comum. Através de um diagrama de estados, uma forma de saber se um código convolucional é catastrófico ou não é observar se existe uma malha fecha de peso nulo.

Exemplo. O codificador $(2, 1, 2)$ possui os seguintes polinômios geradores:

$$g^{(1)}(D) = 1 + D$$

e

$$g^{(2)}(D) = 1 + D^2,$$

temos

$$mdc[g^{(1)}(D), g^{(2)}(D)] = 1 + D,$$

e portanto o codificador é catastrófico.

Considere a sequência de informação $u(D) = \frac{1}{(1+D)}$, ou seja, $u = 1111....$ neste caso, as sequencias codificadas são:

$$v^{(1)} = u(D)g^{(1)}(D) = 1$$

e

$$v^{(2)} = u(D)g^{(2)}(D) = 1 + D,$$

o que corresponde à sequência codificada $v = 11010000\cdots$ se os três bits não nulos forem afetados por erros, a sequência recebida será $r = 000000\cdots$ neste caso, o codificador irá produzir a sequência toda nula como estimativa da sequência de informação original, resultando portanto em um número infinito de erros de decodificação.

3.5.2 Codificadores de Memória Unitária

Sejam u_t o vetor de entrada k-dimensional e v_t o vetor n-dimensional de dígitos codificados cujas componentes pertencem a F_q definidos por:

$$u_t = (u_{t1}, u_{t2}, \cdots, u_{tk})$$

e

$$v_t = (v_{t1}, v_{t2}, \cdots, v_{tn}),$$

respectivamente, com $t = 0, 1, \cdots$.

Sejam $G_0(t)$ e $G_1(t)$ matrizes $k_0 \times n_0$ variantes no tempo com elementos sobre F_q, geradoras por um código convolucional (n_0, k_0) de memória m definido pela seguinte regra de codificação

$$v_t = u_t G_0 + u_{t-1} G_0 + \cdots + u_{t-m} G_m, \tag{3.4}$$

onde $t > 0$ e u_{-1} =0, e 0 é definido como sendo a matriz linha onde todos seus elementos são iguais a zero. Note que as operações são as operações em F_q.

O codificador convolucional :

$$n' = mn_0, k' = mk_0, m' = 1$$

definido por,

$$\mathbf{G_0'} = \begin{pmatrix} G_0 & G_1 & \cdots & G_{m-1} \\ 0 & G_0 & \cdots & G_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G_0 \end{pmatrix} \quad e \quad \mathbf{G_1'} = \begin{pmatrix} G_m & 0 & \cdots & 0 \\ G_{m-1} & G_m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ G_1 & G_2 & \cdots & G_m \end{pmatrix}$$

é equivalente ao código em (3.4), no sentido que a mesma sequencia codificadora binaria semi-definida, está associada á mesma sequencia de entrada semi-definida,[12] e [16]

3.5.3 Códigos de Memória Unitária Parcial(MUP)

A classe de código convolucional denominado código convolucionais de memória unitária parcial(MUP), possui a importante característica de conter codificadores mínimos atingindo a mesma distância livre, que a dos códigos de memória unitária, proposto em [11]. A propriedade de minimalidade resulta da submatriz G_1' tendo posto menor que k, isso implica menor complexidade em termos do número de estados quando comparado ao código de memória unitária.

Uma condição suficiente, para um código de memória unitária parcial ser não catastrófico é que G_0 tenha posto completo e que as n linhas não nulas de G_1 sejam linearmente independente entre si, e linearmente independentes das linhas deG_0, temos que, a importância de n é que este, determina o número 2^n de estados no decodificador quando utilizado o algoritmo de Viterbi.

Definição 3.5.1. Um código convolucional é denominado ótimo se sua função enumeradora possui o menor número de palavras código com peso igual à d_{free}.

Entretanto, se pelo menos dois códigos apresentam o mesmo número de palavras-códigos com o valor d_{free}, então aquele que apresentar o menor número de palavra-código com distância $d_{free}+1$, será considerado ótimo. Se persistir o fato que ambos os código tenham o mesmo número de palavra-código com distância $d_{free}+1$, a politica adotada será a de continuar o processo até que um destes códigos, apresente um menor número de palavras-códigos com distância $d_{free}+j$, sendo $j \geq 1$ um inteiro positivo.

Definição 3.5.2. O fluxo máximo (ϕ) é a soma dos pesos de Hamming das palavras-código ramo que saem e entram no estado zero na representação do diagrama de estados, e satisfaz a equação:

$$\phi = n \cdot (q-1) \cdot q^{(k-1)}, \tag{3.5}$$

temos que a equação 3.5, representa o peso de Hamming total de um código de bloco sobre F_q contendo q^k palavras-código de comprimento n.

Definição 3.5.3. Conservação de Fluxo é o fluxo ϕ que entra e sai de cada estado, com exceção do estado zero no diagrama de estados, é constante. Consequentemente,

as propriedades de fluxo máximo e conservação de fluxo implicam no fato que nenhuma coluna G_0 e G_1 possam ser nula.

3.6 Decodificação dos Códigos Convolucionais

O Algoritmo de decodificação dos códigos convolucionais mais conhecido é 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 de saída, dado uma sequência de informação na entrada do canal. Esta função é chamada métrica associada ao caminho. Como o algoritmo de Viterbi é um decodificador de máxima verossimilhança, este algoritmo escolhe o caminho na treliça com a maior métrica e assim, fornece a melhor sequência estimada dentre todas as possíveis sequencias na treliça a partir da métrica acumulada. Seja r a sequência recebida e v a sequência na entrada do canal, isto é,

$$r = (r_1, r_2, ., r_n) \quad e \quad v = (v_1, v_2, ..., v_n),$$

onde r_i e v_i têm comprimento n. o algoritmo de Viterbi consiste dos seguintes passos:

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

2. Passar para a próxima janela de tempo, incrementando-se i de uma unidade. Computar nova métrica para todas as transições de estados na janela de tempo $i + 1$, e acumulara métrica obtida na janela de tempo anterior. Novamente, armazenar a maior métrica obtida em cada um dos estados no instante de tempo $i + 2$;

3. Caso i seja menor que o tamanho da mensagem a ser decodificada, repetir o passo dois. Caso contrário, parar.

3.6.1 O Algoritmo de Viterbi

O algoritmo de Viterbi é um decodificador de *máxima verossimilhança* com baixa carga computacional em função da utilização da estrutura dos diagramas de treliça dos códigos convolucional, [9] [8], isto é, escolhe o caminho na treliça com menor métrica

acumulada e assim, fornece a melhor sequência estimada dentre todas as possíveis sequências na treliça a partir da métrica acumulada. A distância de Hamming constitui uma métrica.

Definição 3.6.1. [26] e [15] Dado dois elementos u e v do código, a distância de Hamming é definida por

$$d(u,v) = |\{i|u_i \neq v_i, 1 \leq i \leq n\}|\,. \tag{3.6}$$

Considere a sequência de informação original $u = 1011001$ que após ser codificada pelo codificador da Figura 3.3, obteve a sequência $v = 11010010101111$ e esta foi enviada.Aplicando a decodificação abrupta pelo algoritmo de Viterbi [1] [20] , obtém-se um diagrama de treliça que ao aplicar um erro no quarto par de bits obtém-se a Figura 3.9.

A decodificação abrupta pelo algoritmo de Viterbi consiste dos seguintes passos:

1. Iniciando do estado 00, compara-se a distância de Hamming do primeiro par de bits da sequência código com as distâncias de Hamming das duas transições possíveis a partir do estado 00. As distâncias de Hamming obtidas são armazenadas.

2. A distâncias de Hamming do próximo par bits da sequência código comparada com distâncias de Hamming das transições possíveis a partir dos estados alcançados após o passo anterior. As distâncias de Hamming encontradas são somadas àquela obtidas nas transições anteriores.

3. O mesmo procedimento apresentado no passo 2 é repetido para o próximo par da sequência código. No passo atual alguns dos estados são alcançados por dois caminhos diferentes. O caminho sobrevivente (caminho a ser seguido no próximo passo) deve ser aquele que apresenta a menor distância de Hamming acumulada.

4. Repete-se o mesmo procedimento até o final da sequência.

Na análise do último par de bits recebidos, na Figura 3.9, o caminho sobrevivente aponta uma distância de Hamming acumulada igual a 1, identificando um erro no quarto par de bits. Este resultado mostra que a sequência decodificada com maior probabilidade ter sido a sequência transmitida é a sequência 11010010101111, correspondente a mensagem

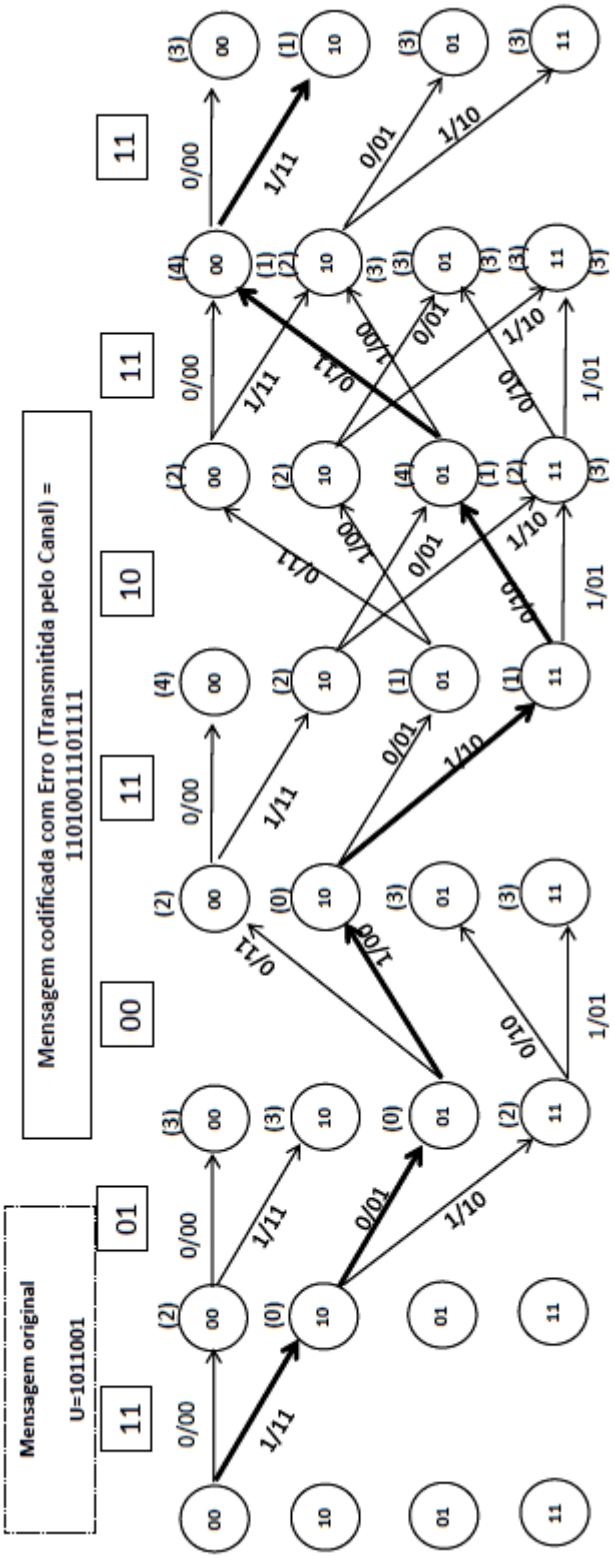

Figura 3.9: Treliça com um erro no quarto par de bits da mensagem codificada

$u = 1011001$. A sequência recebida apresenta um erro em relação à sequência decodificada, a distância de Hamming acumulada é 1.

4 Sistemas Criptográficos

Nos sistemas criptográficos, existem dois métodos, para cifragem e decifragem dos sistemas, a saber: o método convencional e método por chaves publicas. No sistemas convencionais, considerando dois usuários que desejam se comunicar, devem pré-estabelecer uma chave comum, por meio de um canal seguro de envio e recepção de mensagens. Observa-se que 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, pois o número de chaves criptográficas necessárias é no máximo, $\frac{N(N-1)}{2}$. Sendo que essas chaves pode ser inviável, para um sistema com 100.000 usuários.

A ideia básicas da utilização das chaves publicas é a eliminação da necessidade de utilização de um canal eficiente e de alta segurança para distribuição e chaves. Do ponto de vista da implementação os sistemas que usam chaves publicas se fundamentam na existência de métodos diferentes para cifragem e decifragem. Os primeiros métodos de chaves publicas á criptografia foi apresentado por Diffie e Hellman em [6].

4.1 Sistemas Criptográficos Convencionais

1. A privacidade:

   Tem por objetivo evitar que a informação seja interceptada no canal por pessoas não autorizadas. Essas pessoas são chamadas de criptoanalistas. Na Figura 4.1 é mostrado um sistema criptográfico com privacidade;

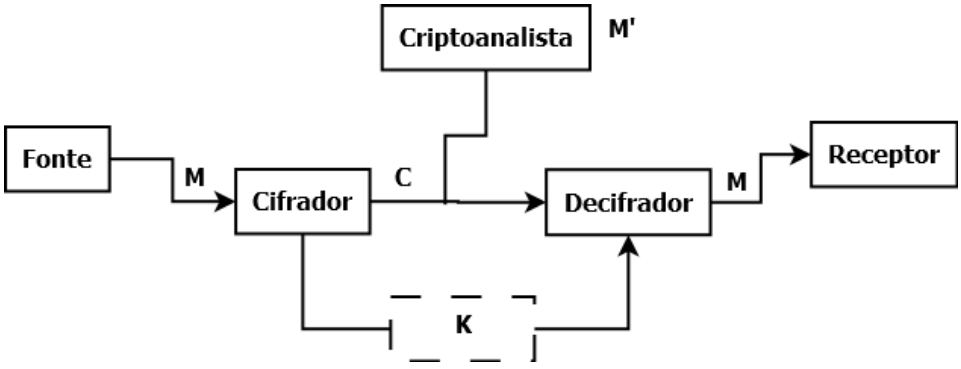

Figura 4.1: Sistema Criptográfico com Privacidade

2. A autenticidade:

   Busca evitar que a informação seja alterada pelo criptoanalista. Na Figura 4.2 temos uma sistema criptográfico que ilustra este tipo de problema. Este dois tipos de problemas estão ligados intrinsecamente e para resolvê-los uma única técnica é aplicada. Neste caso, o transmissor gera uma mensagem M, que é enviada por um canal inseguro monitorado por um criptoanalista. Com o objetivo de evitar que esta mensagem seja interceptada por pessoas não autorizadas, o transmissor cifra o texto M com uma transformação invertível 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 $T_k(M)$, ou seja,

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

   , de modo a recuperar a mensagem original M.

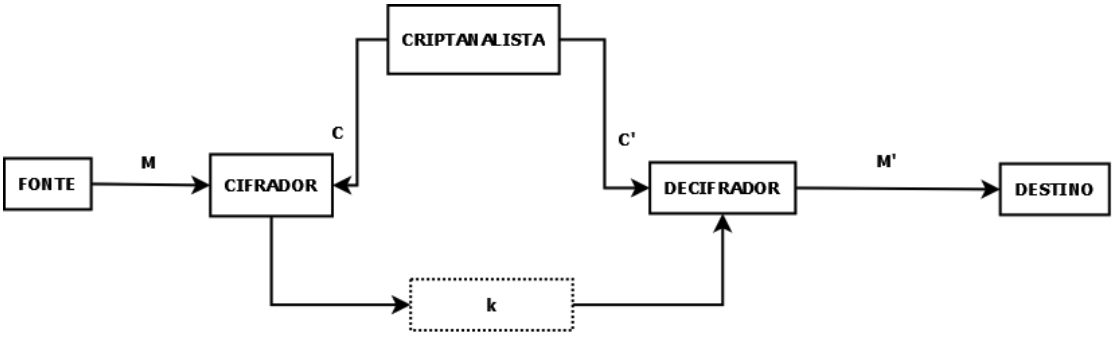

Figura 4.2: Sistema Criptográfico com Autenticidade

4.2 Sistemas Criptográficos de Chave Pública

Como mencionado anteriormente, uma das grandes limitações dos sistemas convencionais, reside no fato de que tais sistemas necessariamente gerem um canal seguro para o envio da chave. Por outro lado, como o processo de cifrarem e decifrarem são inseparáveis nestes sistemas estas chaves devem ser enviadas de forma bastante protegida.

Com o objetivo de solucionar este problema, em [6] lançaram um novo conceito em criptografia, que é a implementação que utiliza a técnica de chaves públicas. Este tipo de sistema é mostrado na Figura 4.3. Existem dois tipos de sistemas de chave pública, a saber:

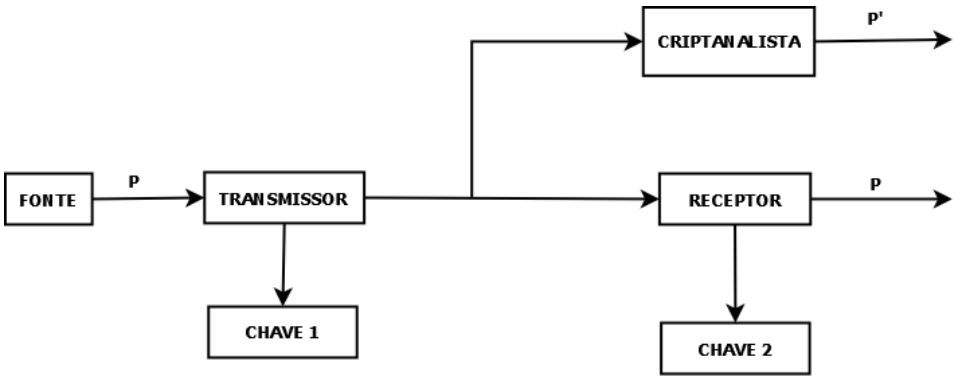

Figura 4.3: Sistemas de Chave Pública

1. Os criptosistemas de chave pública;

2. Os sistemas com distribuição pública de chaves.

4.2.1 Knapsak Binário

Em [19] o cripto sistema do tipo Knapsack tendo como base os códigos convolucionais pode ser descrito como mostrado na Figura 4.4.

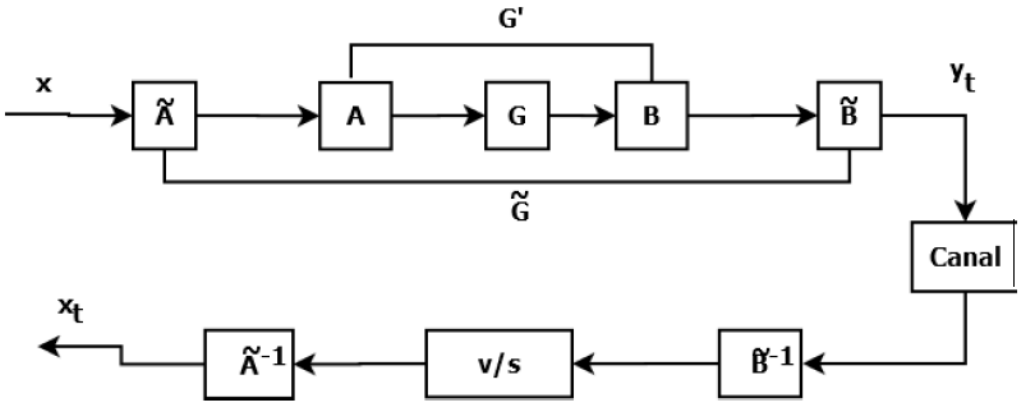

Figura 4.4: Modelo de Cripto sistema de Chave Pública-Knapsack Binário

Na figura 4.4, o bloco G representa na forma matricial a matriz geradora de um código convolucional, os blocos A e B representam transformações armadilhas apropriadas $p \times b$ e $n \times q$, respectivamente, dando origem a um novo código convolucional com taxa $r = \frac{p}{q}$ e com o mesmo número de registros que o código gerado por G. Conduzindo á código convolucional com capacidade de correção maior que o original, continua-se com a aplicação de uma função armadilha consistindo de transformações $\tilde{A}$ e $\tilde{B}$ que são matrizes

quadradas $p \times p$ e $q \times q$ respectivamente de modo a reduzir a capacidade de correção do novo código convolucional.

No receptor $\tilde{B}^{-1}$ é aplicado á sequência de saída do canal, antes de entrar na caixa rotulada V/S, que é decodificação sequencial, de modo a reduzir a complexidade computacional de decodificação. Em seguida a decodificação propriamente dita é realizada, e finalmente a transformação inversa de embaralhamento $\tilde{A}^{-1}$ é aplicada.

Seja $x = (\dots, x_{-1}, x_0, x_1, \dots)$ sequência de dados de entrada onde x_i possuem comprimento p, cada submatriz de G_i , tem dimensão $b \times n$, então a matriz A tem que ter dimensão $p \times b$ e a matriz B de dimensão $n \times q$. Com a determinação das submatrizes passa pela solução do problema da mochila e que complexidade deste problema é 2^b, então valores relativamente pequenos de b tem que ser usados. Suponha que esse problema tenha sido resolvido, seguindo o modelo da figura 4.4, a aplicação das transformações A e B ás submatrizes geradoras do código convolucional como sendo uma função armadilha apropriada.

Seja G_i com i entre 0 *e* m, as submatrizes geradoras do código convolucional, temos que a aplicação das transformações A e B, á G_i resulta em novas submatrizes geradoras G'_i :

$$G'_i = A \cdot G_i \cdot B, i = 0, 1, 2, \cdots, m$$

Note que a matriz A possuí inversa a esquerda A^-, se e somente se o $posto(A) = b$, e possuirá inversa á direita se e somente se o $posto(A) = p$, de forma análoga, B terá inversa a esquerda B^-, se e somente se o $posto(B) = q$ e inversa á direita se e somente se $posto(B) = n$.

Como o código gerado por G' tem capacidade de correção maior que G, dado que essas transformações assim o permitem, teremos necessariamente que aplicar uma transformação á G' de modo a reduzir essa capacidade de correção á níveis desejados. Seja a função armadilha $\tilde{A}$ e $\tilde{B}$ matrizes inversíveis p x p e q x q, respectivamente. Então a nova função geradora $\tilde{G}_i$ dada por:

$$\tilde{G}_i = \tilde{A} \cdot G'_i \cdot \tilde{B}, i = 0, 1, 2, \cdots, m$$

Como o sistema criptográfico é de chave publica, essas matrizes geradoras serão

colocadas numa lista como :

$$\tilde{G} = \left[\tilde{G}_0; \tilde{G}_1; \cdots; \tilde{G}_m\right]$$

O processo de decodificação de códigos convolucionais é definido por:

$y_t = x_t \cdot \tilde{G}_0 + x_{t-1} \cdot \tilde{G}_1 + \cdots + x_{t-m+1} \cdot \tilde{G}_{m-1} + x_{t-m} \cdot \tilde{G}_m$, temos que :

$$t \geq 0 \quad \text{e} \quad x_{-1} = 0$$

Adicionamos um padrão de erro de igual valor da capacidade de correção de erro do código em consideração á y_t. Como no caso mais geral $\tilde{A}$ e $\tilde{B}$ podem ser matrizes retangulares $p_1 \times p$ e $q \times q_1$, respectivamente, suas inversas à direita A^+, B^+ e à esquerda A^-, B^-, devem estar disponíveis no receptor. Dadas por:

$$C^+ = C^t.(C.C^t)^{-1}$$

e

$$C^- = (C^t.C)^{-1}.C^t$$

Então $C = A$ ou B, tal que $A^- \cdot \tilde{G}_i \cdot B^+ = G'_i$, onde C^t significa transposta.

A sequência y_t passa através do canal que poderá ou não introduzir erro(s), a saída do canal será a sequência de dígitos a ser decodificada. Antes da decodificação, aplicaremos $\tilde{B}^+$ à sequência de saída do canal de forma a obter:

$$y_t \cdot \tilde{B}^+ = \left(x_t \cdot \tilde{A}\right) \cdot G'_0 + \left(x_{t-1} \cdot \tilde{A}\right) \cdot G'_1 + \cdots + \left(x_{t-m} \cdot \tilde{A}\right) \cdot G'_m \ .$$

Após a aplicação dessa transformação á sequência de saída do canal, o processo de decodificação segue seu procedimento usual, na sequência decodificada aplicamos A^+ com o objetivo de desembaralhar e produzir a sequência origina.

4.2.2 Cripto-sistema RSA Convolucional de Chave Pública

Este algoritmo foi proposto por Rivest,Shamir e Adleman(RSA), e sua segurança esta baseada em um difícil problema de Teoria dos Números [5] [6]. No sistema RSA são selecionados dois números primos muito grandes $\tilde{p}$ e $\tilde{q}$ e definindo $\tilde{A}$, calcular $\tilde{A} = \tilde{p} \cdot \tilde{q}$, que será utilizado para codificar a mensagem; uma vez que conheçamos a Função Totiente de Euler Φ, isto é calcular a função $\Phi(\tilde{A}) = (\tilde{p} - 1) \, . \, (\tilde{q} - 1)$, selecionaremos um

número inteiro E entre 2 e $\Phi(\tilde{A})$, de forma que E e $\Phi(\tilde{A})$, sejam primos entre si, em seguida, determina-se um número inteiro D, que seja o inverso de $E\ mod(\Phi(\tilde{A}))$.

A cifragem de uma mensagem M, representada pelo correspondente texto C, a qual pode ser colocada como um numero inteiro entre 0 e $\tilde{A}-1$, segue a operação $C = M^E mod(\tilde{A})$. O processo de decifragem do texto cifrado C segue através do emprego do número de decifragem D através da operação:

$$M = C^D mod\ (\tilde{A}).$$

Tendo essa visão do sistema RSA, introduziremos uma variação do mesmo, sejam dados:

$$E, D, \tilde{A}\ \ e\ \ \left[\tilde{A}_0; \tilde{G}_1; \cdots; \tilde{G}_m\right],$$

o cifrador, decifrador, um inteiro e as submatrizes geradoras do código convolucional com $r = \frac{b}{n}$. O sistema RSA convolucional invariante no tempo consiste basicamente na colocação em uma lista dos elementos:

$$E,\ \tilde{A}\ e\ \left[\tilde{G}_0; \tilde{G}_1; \cdots; \tilde{G}_m\right].$$

Qualquer um que se deseja se comunicar com o usuário identificado por esses três elementos, será feita através da codificação de p dígitos da sequência de dados de entrada por meio da matriz geradora publicada na lista. A saída codificada será então transformada para inteiro correspondente. seja $M-i$ tal inteiro, esse numero será cifrado através do uso no sistema RSA, ou seja por:

$$C_i = (M_i)^E mod\ (\tilde{A}).$$

Para uma dada sequencia de dados de comprimento N.p estará sendo codificada convolucionalmente em uma sequencia de comprimento N e, a ela estará sendo associado um caminho na treliça. A decodificação dessa sequencia de comprimento N segue por meio do algoritmo de viterbi onde cada um desses N inteiros é processado através do sistema RSA:

$$M_i = (C_i)^D mod\ (\tilde{A}).$$

4.3 Complexidade Computacional

Existe uma preocupação inerente com os processos desenvolvidos, do ponto de vista computacional. Esta preocupação reflete o objetivo imediato de se encontrar algoritmos eficientes.

O tempo de processamento de um algoritmo em uma determinada máquina avalia a sua eficiência; este tempo está diretamente ligado à quantidade de operações a serem realizadas no processo, o que determina a complexidade do algoritmo.

A complexidade de um algoritmo pertence basicamente a duas classes: *polinomial P* e *não polinomial NP*. Ela é dita ser polinomial quando esta função for uma função polinomial nos tamanhos dos dados de entrada.

Caso contrário, quando os problemas de decisão não admitem algoritmo polinomial, o algoritmo pertence a classe não polinomial NP. Neste caso a complexidade é exponencial.

O problema da Mochila é denominado como NP completo, que são os problemas NP com maior dificuldade de solução dentre todos os NP.

Como exemplo tem-se o algoritmo que verifica se o grafo é ou não biconexo. A complexidade deste algoritmo é linear com relação ao tamanho do grafo. Logo, o problema de biconectividade pertence a classe P.

Como exemplo para um algoritmo que pertence a classe não polinomial NP pode-se mencionar o problema do caixeiro viajante.

O problema da Mochila é denominado como NP completo, que são os problemas NP com maior dificuldade de solução dentre todos os NP. Pode-se definir o problema da mochila como: considere um conjunto de objetos cuja solução de um determinado problema está contido em um subconjunto deste conjunto de objetos, a este conjunto total denominamos de mochila.

Como exemplo pode-se ilustrar um problema numérico:

É possível escrever 32 como a soma de um subconjunto dos números, $[10, 9, 17, 13, 40, 60]$?

A resposta é sim, essa soma pode ser dada por:

$$32 = 10 + 9 + 13,$$

onde os números [10, 9, 13], compõem um subconjunto do conjunto dado.

Assim, 32 pode ser representado como a soma do primeiro, segundo e quarto números da lista.

Esse resultado também pode ser escrito como,

$$32 \rightarrow (1, 1, 0, 1, 0, 0).$$

E reciprocamente, dado (1, 1, 0, 1, 0, 0) recupera-se 32.

A classe de problemas NP completo é vista com bastante interesse para uso em sistemas criptográficos, pelo fato da dificuldade da solução, exigindo um enorme tempo computacional para resolvê-los. Desta forma, a quebra 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 $(a_1, a_2, \cdots, a_n)$ e x é um vetor binário n-dimensional. O cálculo de Y é simples, envolvendo somente a soma de n inteiros.

Entretanto, o caminho inverso, isto é, a partir de f obter um subconjunto de elementos de $\{a_1\}$ cuja soma seja Y, é conhecido como o problema da mochila. A busca exaustiva dos 2^n subconjuntos de $\{a_1\}$, cresce exponencialmente com n, sendo esta busca computacionalmente impraticável para n muito grande.

5 Método Knapsack Binário Para Sistema Criptográfico e Chave Pública

Para que possamos encontrar códigos ótimos de memória unitária, deve-se resolver o problema do knapsack, como primeiro passo. Esse problema pertence a classe dos problemas não polinomiais completos, sendo que esse problemas não são fáceis de solução, justificando dessa forma a grande dificuldade na determinação de bons códigos convolucionais, sendo desejável o seu uso nos sistemas criptográficos convencionais e também nos sistemas de chaves públicas.

A segurança do cripto-sistemas de chave publica é baseada em códigos convolucionais aqui proposto leva em consideração os seguintes fatos:

- O embaralhamento das colunas das submatrizes geradoras do código convolucional faz com que o pesos de hamming das palavras-código ramo diminua, causando a queda no poder de correção dos erros;

- O fato adicional de que o problema knapsack do tipo binário, tem que ser resolvido; para que se possa obter os resultados esperados, no desempenho de um sistema criptográfico utilizando códigos convolucionais, será necessário utilizarmos uma função armadilha

5.1 Descrição do Método do Sistema Criptográfico

A estrutura de códigos de memória unitária tem k entradas paralelas nos registradores de deslocamento, o codificador do código de memória unitária é mostrado na 3.6. Este codificador tem taxa $r = \frac{2}{4}$ e $m = 1$ com as seguintes submatrizes geradoras:

$$\mathbf{G_0} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{G_1} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Este código é o código ótimo da taxa $r = \frac{2}{4}$. Para enfatizar algumas propriedades dos códigos de memória unitária é também apresentado o diagrama de estados do referido código, este diagrama mostra que em cada estado temos: $2^{k.m}$ saídas e $2^{k.m}$ entradas em cada estados, essa propriedade será vista posteriormente ,como parte da analise de complexidade dos sistemas criptográficos, se observamos o diagrama de estados deste código, constatamos que o número de ramos depende exponencialmente das k entradas.

As funções armadilhas são transformações aplicadas as submatrizes geradoras do código G_0 e G_1 com a finalidade de reduzir o poder de correção deste código, essas transformações aplicadas ás submatrizes geradoras, são feitas através de um par de matrizes A e B, sendo que as dimensões destas matrizes são respectivamente $k \times k$ e $n \times n$. Essa matrizes uma vez aplicadas, gera novas submatrizes geradoras denominadas de G'_0 e G'_1:

$$G'_0 = A.G_0.B \tag{5.1}$$

$$G'_1 = A.G_1.B \tag{5.2}$$

Além desse procedimento de multiplicação pelo par de matrizes A e B, cuja objetivo é reduzir o poder de correção do código através da alteração da distância livre, pode-se ainda adicionar um vetor erro á mensagem cifrada. Esse vetor erro denominado α_i 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:

$$\alpha_t = \beta_t.G_0 \oplus \beta_{t-1}.G_1, \tag{5.3}$$

onde $\beta_t = 0$ para $t < 0$. onde α_t é a sequencia codificada na entrada do canal e β_t a sequência de informação. Substituindo então as equações 5.1 e 5.2 em 5.3 temos que:

$$\alpha_t = \beta_t.G'_0 \oplus \beta_{t-1}.G'_1, \tag{5.4}$$

com $t \leq 0$ e $\beta_{-1} = 0$, a equação 5.4 contém os processos de cifragem e codificação, ja o vetor α_t adicionamos um vetor erro α_e que corresponderá á quantidade de erro permissível ao código original, temos que: $\alpha_{te} = \alpha_t + \alpha_i$, onde α_{te} é o vetor correspondente á mensagem codificada/cifrada com erro, α_i é a quantidade de erros inseridos na mensagem no seu ponto de envio e α_t é mensagem cifrada/codificada.

O vetor α_i da origem a um vetor erro α_e, na decodificação, cujo o peso de Hamming desse vetor menor ou igual a capacidade de correção do código, o vetor erro

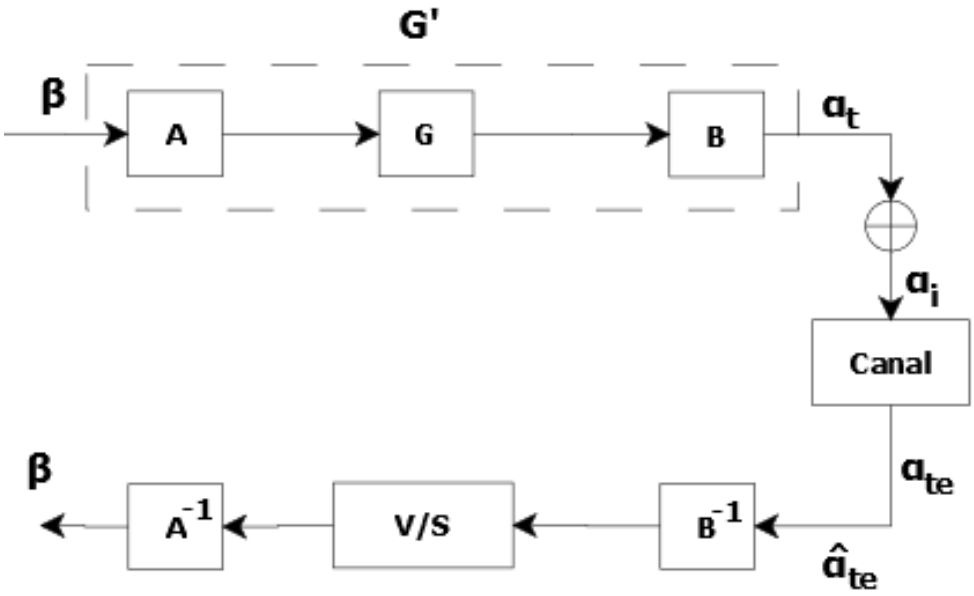

Figura 5.1: Diagrama em Bloco do Criptosistemas de chave Pública

na decodificação é obtido por meio da multiplicação do vetor erro de transmissão pela transformação inversa de B (B^{-1}), ou seja $\alpha_e = \alpha_i B^{-1}$ na figura 5.1

A soma do vetor erro de transmissão ao vetor que contem a informação α_t é o que prossegue pelo canal, desta forma o processo de decodificação é aplicado ao vetor α_{te}. O processo de decodificação empregado á sequência de saída do canal $\hat{\alpha}_{te}$, considerando que o canal está livre de ruido temos que $\hat{\alpha}_{te}$ que é o valor estimado de α_{te}, seja igual ao valor de α_{te} na entrada do canal. Utilizamos então a transformação inversa de B (B^{-1}), logo:

$$\alpha_{te} = \hat{\alpha}_{te}$$

$$\alpha_{te}.B^{-1} = (\beta_t.A).G_0 \oplus (\beta_{t-1}.A).G_1$$

β_t é obtido aplicando o algoritmo de Viterbi. Para esse criptosistema a matriz B^{-1} é a transformação aplicada as submatrizes geradoras originais do código cujo função é reduzir a minima distancia livre, enquanto que a matriz A realiza o embaralhamento dos bits,nas palavras código ramo, cuja o objetivo principal a obtenção da equiprobabilidade dos símbolos binários um e zero, antes de serem enviados ao bloco de transmissão.

Para que esse criptosistema seja quebrado, é necessário que o criptoanalista primeiramente resolva o problema knapsak relacionado a encontrar as submatrizes geradoras do código ótimo G_0 e G_1, e depois encontrar as transformações: A^{-1} e B^{-1}, sendo que esses dois problemas não são trívias.

Aplicação: [21] Um código ótimo com taxa $r = \frac{2}{4}$ e m= 1 para ilustrar o procedimento de cifragem, cujas submatrizes geradoras são:

$$\mathbf{G_0} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

e

$$\mathbf{G_1} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix},$$

onde a distância livre deste código é $d_{free} = \{G_0 \ G_1\} = 5$. Aplicando as transformações armadilhas A e B dadas por:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad e \quad \mathbf{B} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

e suas matrizes inversas, respectivamente são:

$$\mathbf{A^{-1}} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad e \quad \mathbf{B^{-1}} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Aplicando A e B em G_0 e G_1 em 5.1 e 5.2, obtemos G'_0 e G'_1:

$$\mathbf{G'_0} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad e \quad \mathbf{G'_1} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

A distância livre deste novo código é $d_{free} = \{G_0 \ G_1\} = 4$. Assim o poder de correção foi reduzido, ou seja enquanto o código original corrige até 2 erros o novo código corrige apenas 1 erro.

Aplicação: Considerando um código convolucional, com taxa $r = \frac{2}{4}$, com as seguintes submatrizes geradoras do código:

$$\mathbf{G_0} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad e \quad \mathbf{G_1} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Representadas e referentes ao codificador figura 3.6, onde temos que o $d_{free} = \{G_0 \ G_1\} = 5$. Aplicando as transformações A e B (matriz de Hadamard - H_4) dadas por:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

e

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

suas inversas são:

$$\mathbf{A^{-1}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

e

$$\mathbf{B^{-1}} = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & -0.25 & 0.25 & -0.25 \\ 0.25 & 0.25 & -0.25 & -0.25 \\ 0.25 & -0.25 & -0.25 & 0.25 \end{pmatrix}.$$

Para possibilitar o cálculo em F_2, trocamos os -0.25 por $1's$ e os 0.25 por $0'$s, tornando-a matriz de Hadamard binária. obtemos por meio de G_0 e G_1, as novas submatrizes geradoras:

$$\mathbf{G'_0} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} e\ \mathbf{G'_1} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

onde passa a ser $d_{free} = \{G'_0\ \ G'_1\} = 2$.

Com isso o poder de correção reduziu-se menos de 1 erro, quando o código original era capaz de corrigir até 2 erros. suponha que queiramos transmitir a mensagem $\beta = (11, 01, 10, 00)$, considere o codificador inicialmente resetado, assim a palavra a ser transmitida será: $\alpha_t = \beta.G$, Fazendo a multiplicação no MATLAB e no SCILAB, temos com resposta o seguinte:

$$\beta = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & & & & & & & & & & & \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & & & & & & \\
 & & & & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & & & & & & & \\
 & & & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & & \\
 & & & & & & & & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & & & \\
 & & & & & & & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & & & \\
 & & & & & & & & & & & & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
 & & & & & & & & & & & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & & & 0 & 0 & 1 & 1 \\
 & & & & & & & & & & & & & & & & 0 & 1 & 0 & 1
\end{pmatrix}$$

A palavra a ser transmitida é:

$$\alpha_t = (0110, 0000, 0011, 0101, 0000, 0000, 0000, 0000, 0000),$$

adiciona-se o vetor erro de transmissão:

$$\alpha_i = (1000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000)$$

A palavra que percorrerá o canal será:

$$\alpha_{te} = (1110, 0000, 0011, 0101, 0000, 0000, 0000, 0000, 0000)$$

ou seja :

$$\alpha_{te} = (1110, 0000, 0011, 0101)$$

Aplicando ao vetor α_{te} a transformação $B^{-1}(H_4)$, obtemos:

$$\hat{\alpha}_{te} = \alpha_{te}.B^{-1},$$

logo

$$\hat{\alpha}_{te} = (0110, 0000, 0101, 0011),$$

Em seguida, aplicamos o processo de decodificação utilizando-se o Algoritmo de Viterbi, obtemos a sequência de informação:

$$\beta = (11, 01, 10, 00).$$

Na figura 5.2 o fluxograma da aplicação transformação armadilha as submatrizes geradoras do código

5.2 Propriedades das Funções Armadilhas

Sejam as submatrizes geradoras G_i com dimensão $k \times n$ e onde $0 \leq i \leq m$ de um código de memória unitária sobre o corpo de dois elementos, GF(2). Seja $\phi(G_i)$, o peso total de Hamming de cada G_i. $\phi(G_i)$ é dado explicitamente por:

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

Definição:[17] Uma matriz G com dimensão $k \times n$ 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 tem pesos de Hamming:

$$\tilde{w}_H = \frac{(G)}{(2^k - 1)}$$

Se $2^k - 1$ dividirmos $\phi(G)$ então $\lfloor \tilde{w} \rfloor = \tilde{w}$ ($\tilde{w}$ é um numero inteiro), caso contrário, $\lfloor \tilde{w} \rfloor = \{\tilde{w},\ |\tilde{w}| \pm j\}$, onde $j \geq 0$, onde $\lfloor \tilde{w} \rfloor$ é a função piso. Como os códigos de memória unitária são equivalentes aos códigos convolucionais, iremos considerar somente os de memoria unitária onde G, a matriz geradora tem dimensão $2k \times n$. Desta forma a temos que:

lema 1: [18]Se o $posto(G) = 2k$ e $2k \leq n$, então G tem $2k$ vetores linha que são linearmente independentes com pesos de Hemming igualmente distribuídos e são capazes de gerar todas as 2^{2k} palavras do código também com pesos de Hamming igualmente distribuídos

Demonstração: Um código de memória unitária com taxa $r = \frac{k}{n}$ tem sua em treliça com 2^k estados e 2^k transições de cada estado.

Deste modo o número total de transições: 2^{2k}.

Neste momento, como 2^k vetores linha de G são linearmente independentes, sendo que G gera um espaço vetorial com 2^{2k} vetores.

Seja $\lfloor \tilde{w} \rfloor$ a parte inteira de $\tilde{w}$, existem:

$$\frac{n!}{(n - \lfloor \tilde{w} \rfloor)!}.\lfloor \tilde{w} \rfloor!$$

Vetores tendo peso de Hemming igual a $\lfloor \tilde{w} \rfloor$.

Como n $\geq$ 2k, pode-se mostrar que:

$$\frac{n!}{(n - \lfloor \tilde{w} \rfloor)!}.\lfloor \tilde{w} \rfloor! \geq 2k$$

.

Deste modo temos que pelo menos 2k vetores tem peso de Hemming $\lfloor \tilde{w} \rfloor$. Como cada transição tem associado um vetor diferente, é possível encontrar um código de bloco $(k, 2n)$ com tamanho do bloco igual a $2n$ e o tamanho de informação k tal que a matriz geradora seja constituída de vetores, tendo pesos de Hamming igualmente distribuídos com distância mínima igual á do código de bloco $(k, 2n)$. Esta é a menor das distancias, uma vez que outras transições poderão, ser maiores do que, pelo menos uma unidade.

A consequência precedente do **lema 1**, é que o código é do tipo não catastrófico, por outro lado se tivermos $2k > n$, 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 uma combinação linear das demais.

Isto implica que nem todas as transações na treliça terão associadas palavras códigos diferentes. Portanto, a busca pelo código ótimo deve ser bastante criteriosa, uma vez que o mesmo pode ser catastrófico.

Estabelecemos agora a condição de existência da função armadilha composta pelo par de matrizes de transformação (A, B) sob o **lema 1** admitindo que o código utilizado seja ótimo.

Existem fundamentalmente duas condições, a primeira delas está relacionado com a aplicação da função armadilha G_i significando multiplicar G_i pelo par de matrizes de transformação (A, B), resultando em um novo código com a mesma taxa e número de memórias. como consequência disso, temos a seguinte proposição.

Proposição:[18] Sejam G_i onde $0 \leq i \leq m$, submatrizes geradoras de um código convolucional ótimo não catastrofístico sobre GF(2), sejam A e B matrizes inversíveis com dimensão $k \times k$ e $n \times n$, respectivamente,

$$d_{min}\left\{A.[G_1, G_2, ..., G_m].B\right\} \leq d_{min}\left\{[G_1, G_2, ..., G_m]\right\},$$

onde A e B não são necessariamente unitárias.

Demonstração:

Partimos da Hipótese que $G = [G_1, G_2, ..., G_m]$ é a matriz geradora de um código ótimo.

Seja

$$d_{min}(G) \neq d_{min} \quad e \quad G^{'} = AGB,$$

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 dividido em três subconjuntos:

1. S_1 - é o conjunto de transformações (A, B) que conduzem a códigos com d_{min} maiores.
2. S_2 - é o conjunto de transformações (A, B) que conduzem aos códigos equivalentes, isto é, com mesmo d_{min}.
3. S_3 - é o conjunto de todas as outras de transformações (A, B) que conduzem a códigos com d_{min} menores.

Sabemos que se o código que está sendo utilizado é ótimo, então a condição S_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 catastrofístico ou não linear.

Agora G' é equivalente a G, se $G' = AGB$, onde A e B são matrizes inversíveis com dimensões $k \times k$ e $n \times n$, respectivamente com determinantes iguais a 1, então pertence a S_2, e assim a igualdade é valida em para segunda condição do d_{min}.

Obviamente, se A e B pertencem a S_3, então a transformação resultante conduzirá a um código com capacidade corretiva de erro menor.

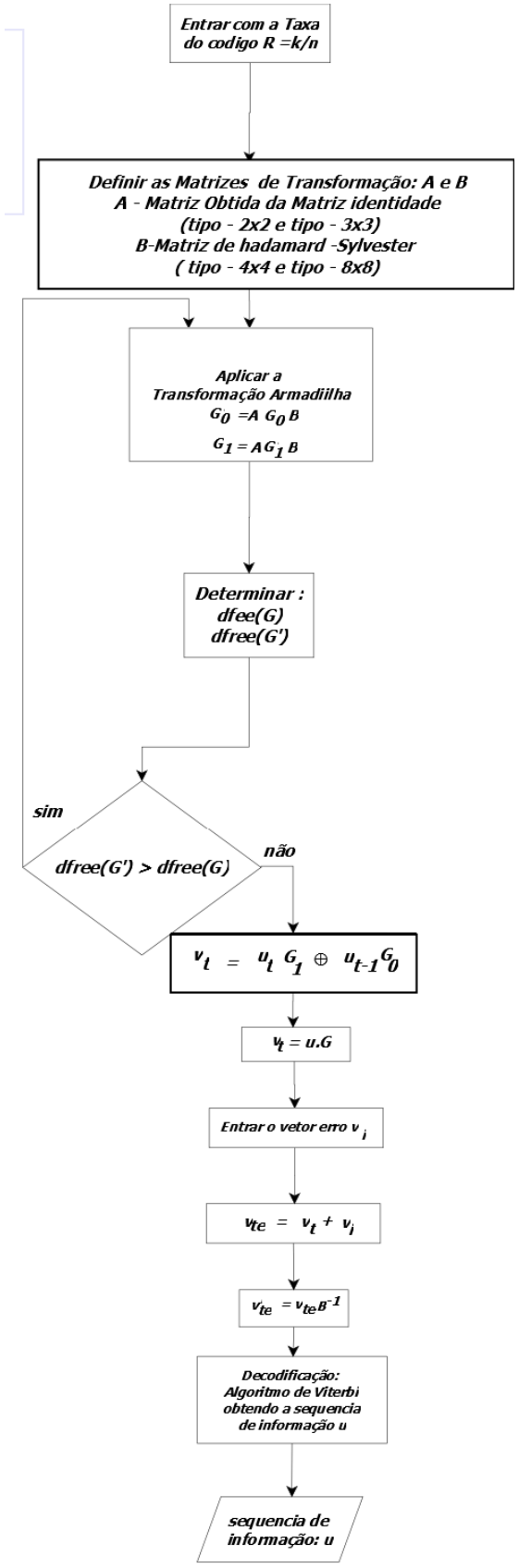

Figura 5.2: Fluxograma da aplicação Armadilha Matriz de Hadamard do código com taxa $R = k/n$

6 Transformação Armadilha

Definiremos as transformações armadilha utilizadas no desenvolvimento do trabalho, sendo feita uma comparação entre as Transformações Armadilha - Matriz de Hadamard, onde analisamos o seu desempenho e observamos qual função armadilha e a que conduz ao melhor resultado. Dessa forma, aplicamos a função à armadilha - Matriz de Hadamard em quatro código $(n, k, 1)$ de memória unitária, de inicio aplicaremos as transformações armadilhas aos códigos com taxas:

1. $R = \frac{2}{8}$ e $R = \frac{3}{8}$, códigos convolucionais de memoria unitária(CCMU);

Em seguida aplicando as transformações armadilhas aos códigos com taxas:

2. $R = \frac{2}{4}$ e $R = \frac{3}{4}$, também códigos convolucionais de memoria unitária(CCMU);

Sendo aqueles utilizados com uma pequena complexidade para que pudêssemos observar o comportamento do criptossistema quanto a sua vulnerabilidade. As matrizes apresentadas, tanto as matrizes geradoras do código como também as matrizes de transformação armadilha, estão na forma octal. Como exemplo, considere as matrizes

$$\mathbf{G_0} = \begin{pmatrix} 07 & : 17 \end{pmatrix} \quad e \quad \mathbf{G_1} = \begin{pmatrix} 02 & : 01 \end{pmatrix}$$

é a representação octal das matrizes,

$$\mathbf{G_0} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad e \quad \mathbf{G_1} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

6.1 Matriz de Hadamard

Uma matriz quadrada com elementos 1 e tamanho h, cujos vetores de linha distintos são ortogonais é uma matriz de Hadamard de ordem h. Exemplo de matrizes de

Hadamard normalizadas

$$[1], \quad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} e$$

Estas foram primeiro estudados em [10], que observou que, se H é uma matriz Hadamard, então

$$\begin{pmatrix} H & H \\ H & -H \end{pmatrix}$$

,É também uma matriz de Hadamard.

Lema 1 (Sylvester [10]): Existe uma matriz de Hadamard de ordem 2^t para todos os inteiros t.

A matrizes de ordem 2^t construídas através das matrizes de Hadamard-Sylvester, eles são naturalmente associados a funções ortogonais discretas chamadas funções de Walsh. Usando o método de Sylvester, as primeiras matrizes de Hadamard obtidas são:

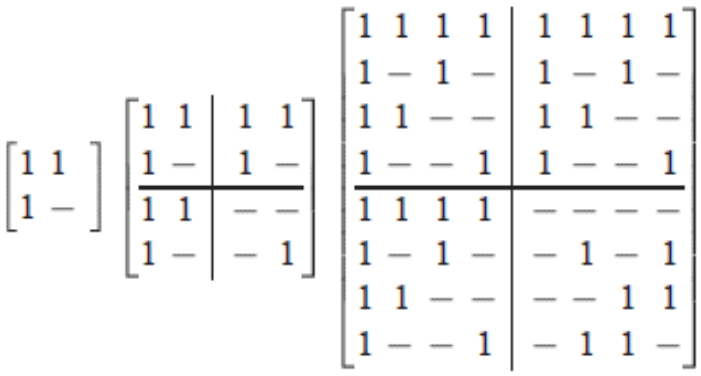

Figura 6.1: Matrizes de Hadamard-Ordem: 2, 4 e 8

Para essas matrizes é contado, linha a linha, o número de vezes que o sinal muda, de modo que 1 -1 -1 1 mudanças de sinal duas vezes. Isto dá:

- Para a matriz de ordem 2: $0, 1$
- Para a matriz de ordem 4: $0, 3, 1, 2,$
- Para a matriz de ordem 8: $0, 7, 3, 4, 1, 6, 2, 5$

Com efeito, veremos que o conjunto do número de mudanças de sinal em uma matriz de Sylvester-Hadamard de ordem n é $\{0, 1, \ldots, n-1\}$, correspondendo ao número de zero cruzados nas funções de Walsh. Vejamos algumas propriedades básicas de matrizes de Hadamard:

Lema2: Seja H uma matriz de Hadamard de orden h. Então, deverá satisfazer as seguintes propriedades:

1. $HH^T = hI_h$;
2. $|detH| = h^{\frac{1}{2}h}$;
3. $HH^T = H^T H$;
4. As matrizes de Hadamard podem ser transformadas em outras matrizes de Hadamard, permutando linhas e colunas e multiplicando-se linhas e colunas por -1. Chamamos matrizes que podem ser obtidas umas das outras por esses métodos Equivalente-H (nem todas as matrizes de Hadamard do mesmo Ordem são equivalentes a H);
5. Toda matriz de Hadamard é H-equivalente a uma matriz de Hadamard que tem todos os elementos de sua primeira linha e coluna +1 e -1, matrizes desta última forma são chamadas normalizadas;
6. A ordem de uma matriz de Hadamard é 1,2, ou $4n$, onde n é um inteiro positivo.

Definição : Se $M = (m_{ij})$ é uma matriz $m \times p$ e $N = (n_{ij})$ é uma matriz $n \times q$, então o produto Kronecker $M \otimes N$ é a matriz $mn \times pq$ dada por:

$$\mathbf{M} \otimes \mathbf{N} = \begin{bmatrix} m_{11}N & m_{12}N & \ldots & m_{1p}N \\ m_{21}N & m_{22}N & \ldots & m_{2p}N \\ \vdots & \vdots & & \vdots \\ m_{m1}N & m_{m2}N & \ldots & m_{mp}N \end{bmatrix}$$

Exemplo:

Se

$$\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad e \quad \mathbf{N} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

então,

$$\mathbf{M} \otimes \mathbf{N} = \begin{pmatrix} N & N \\ N & -N \end{pmatrix}.$$

Portanto

$$\mathbf{M} \otimes \mathbf{N} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

Lema 3: (Sylvester-Hadamard): Seja H_1 e H_2 matrizes de Hadamard das ordens h_1 e h_2. Em seguida, pelas propriedades dos produtos Kronecker H = $H_1 \otimes H_2$ é uma matriz de Hadamard de ordem h_1 x h_2.

6.2 Aplicação da Transformação Armadilhas-Matriz de Hadamard

As representações matriciais dos elementos da matriz de Hadamard de ordem 4 e 8 foram aplicadas como transformações armadilha B, temos que :

$$\mathbf{H_4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} e \ \mathbf{H_8} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

Para códigos ótimos de memória unitária com taxas: $R = 2/4$, $R = 3/4$, $R = 2/8$ e $R = 3/8$, para possibilitar o cálculos em GF(2), trocamos $-1's$ por $1's$ e os $1's$ por $0's$, tornando-as matrizes de Hadamard binárias.

Como função armadilha a responsável pelo embaralhamento das linhas, são as matrizes denotadas por A_i, onde $i = \{1, 2, 3, 4, 5\}$, constituindo matrizes do tipo (2×2), sendo que, as matrizes $A_{i's}$ de ordem 2, são obtidas de permutações das colunas, mas também da combinação linear das linhas da matriz identidades de ordem 2:

$$\mathbf{A_1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \mathbf{A_2} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \ \mathbf{A_3} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \ \mathbf{A_4} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} e$$

$$\mathbf{A_5} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

para os códigos com taxa :$R = 2/4$ e $R = 2/8$.

E como função armadilha a responsável pelo embaralhamento das linhas, onde as matrizes componentes são denotadas por A_i, utilizamos as matrizes (3×3), as matrizes $A_{i's}$ de ordem 3, são obtidas de permutações das colunas, mas também da combinação linear das linhas da matriz identidades de ordem 3:

$$\mathbf{A_1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{A_2} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{A_3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{A_4} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} e$$

$$\mathbf{A_5} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Para os codigos com Taxas: $R = 3/4$ e $R = 3/8$, utilizando o MATLAB e SCILAB, obtemos os seguintes resultados.

1. **Código com Taxa $R = \frac{2}{4}$:**

Para o código convolucional (4, 2, 1) de matrizes geradoras:

$$\mathbf{G_0} = \begin{pmatrix} 15 & : 03 \end{pmatrix} e \ \mathbf{G_1} = \begin{pmatrix} 14 & : 07 \end{pmatrix}$$

e $d_{free} = \{G_0 \ G_1\} = 5$, foram utilizadas as transformações armadilhas A_1, A_2, A_3, A_4 e A_5 ja apresentadas, e as transformações armadilhas B que são as representações matriciais dos elementos da matriz de Hadamard H_4.

Na Tabela 6.1, é apresentada a transformações armadilhas A_i onde i varia 1 ate 5, sendo elas do tipo 2 x 2, As submatrizes geradoras do código convolucional G_0 e

Tabela 6.1: **Transformação de Hadamard(H_4),** $R = \frac{2}{4}$

Código	A	$B(H_4)$	G'_0	G'_1	d_{free}	d'_{free}
(4, 2, 1)	[02 : 01]	[00 : 05 : 03 : 06]	[03 : 05]	[05 : 00]	5	2
	[03 : 01]	[00 : 05 : 03 : 06]	[06 : 05]	[05 : 00]	5	2
	[02 : 03]	[00 : 05 : 03 : 06]	[03 : 06]	[05 : 06]	5	4
	[01 : 02]	[00 : 05 : 03 : 06]	[05 : 03]	[00 : 05]	5	2
	[03 : 02]	[00 : 05 : 03 : 06]	[06 : 03]	[05 : 05]	5	4

G_1 e a distancia livre tanto do original como do novo código, a também as submatrizes geradoras do código convolucional G'_0 e G'_1.

O $d_{free} = \{G_0\ G_1\}$ do código original que era 5, ao fazer aplicação de transformação armadilha a matriz de Hadamard o novo código, passa a ter um novo $d'_{free} = \{G'_0\ G'_1\}$, o seu valor, varia em 2 e 4, ou seja, antes da transformação o código o original podia corrigir 2 erros, quando aplicada a transformação o poder de correção é 0 ou 1 erros, algumas transformações reduziram para 2, a distancia livre, com as transformações A_1, A_2 e A_4, enquanto que as transformações A_3 e A_5 reduziram para 4 a distancia livre.

Apesar de ter alcançado o objetivo de redução da capacidade de correção, a maioria dos códigos obtidos tiveram uma redução na dimensão do espaço de operação pois um coluna das submatrizes G'_0 e G'_1 são nulas.

Outra consequência, é que três dos cinco submatrizes,resultante da transformação armadilha, origina códigos de memoria unitária parcial

1. **Código com Taxa** $R = \frac{2}{8}$**:**

   Para o código convolucional $(8, 2, 1)$ de matrizes geradoras:

$$\mathbf{G_0} = \begin{pmatrix} 370 & : 037 \end{pmatrix} e\ \mathbf{G_1} = \begin{pmatrix} 174 & : 237 \end{pmatrix}$$

   Utilizadas as transformações armadilhas A_1, A_2, A_3, A_4 e A_5, onde o $d_{free} = \{G_0\ G_1\} = 10$, todas as matrizes, as transformações armadilhas e as submatrizes, estão representadas na forma octal, onde cada linha da matriz é separada por dois pontos (:).

   Na tabela 6.2, é apresentada a transformações armadilhas A_i onde i varia 1

até 5, sendo elas do tipo 2×2, As submatrizes geradoras do código convolucional G_0 e G_1 e a distancia livre tanto do original como do novo código, a também as submatrizes geradoras do código convolucional G'_0 e G'_1.

Tabela 6.2: **Transformação de Hadamard(H_8),** $R = \frac{2}{8}$

Código	A	$B(H_8)$	G'_0	G'_1	d_{free}	d'_{free}
(8, 2, 1)	[02 : 01]	[00:125:63:146:17:132:74:151]	[17 : 146]	[125 : 146]	10	8
	[03 : 01]	[00:125:63:146:17:132:74:151]	[151 : 146]	[63 : 146]	10	8
	[02 : 03]	[00:125:63:146:17:132:74:151]	[17 : 151]	[125 : 63]	10	8
	[01 : 02]	[00:125:63:146:17:132:74:151]	[146 : 17]	[146 : 125]	10	8
	[03 : 02]	[00:125:63:146:17:132:74:151]	[151 : 17]	[63 : 125]	10	8

Quando aplicado a transformação ao código $(8, 2, 1)$, notamos uma redução no $d_{free} = \{G_0\ G_1\} = 10$ para $d'_{free} = \{G'_0\ G'_1\} = 8$, independente da transformação, o poder de correção do código passou de 4 para 3 erros

1. **Código com Taxa $R = \frac{3}{4}$:**

Para o código convolucional de memoria unitária $(4, 3, 1)$, as matrizes geradoras são:

$$\mathbf{G_0} = \begin{pmatrix} 03 & :04 & 16 \end{pmatrix} e\ \mathbf{G_1} = \begin{pmatrix} 11 & :12 & 14 \end{pmatrix}$$

O código original apresenta $d_{free} = \{G_0\ G_1\} = 3$, nesse código, foram utilizadas as transformações armadilhas A_1, A_2, A_3, A_4 e A_5, ja apresentadas, e as transformações armadilhas B que são as representações matriciais dos elementos da matriz de Hadamard H_4.

Na Tabela 6.3, é apresentada as transformações armadilhas A_i onde i varia 1 ate 5, sendo elas as matrizes quadradas de ordem 3×3, também são mostradas as submatrizes geradoras do código convolucional original G_0 e G_1, as submatrizes geradoras do código convolucional novo G'_0 e G'_1 e as a distancias livres tanto do código convolucional original, como do novo código.

Tabela 6.3: **Transformação de Hadamard(H_4),** $R = \frac{3}{4}$

Código	A	$B(H_4)$	G'_0	G'_1	d_{free}	d'_{free}
(4, 3, 1)	[04 : 03: 01]	[00 : 05 : 03 : 06]	[05 : 03: 06]	[06 : 06: 05]	3	4
	[06 : 03: 01]	[00 : 05 : 03 : 06]	[00 : 03: 06]	[05 : 06: 05]	3	2
	[04 : 02: 01]	[00 : 05 : 03 : 06]	[05 : 05: 06]	[06 : 03: 05]	3	4
	[01 : 02: 04]	[00 : 05 : 03 : 06]	[06 : 05: 05]	[05 : 03: 06]	3	4
	[05 : 03: 01]	[00 : 05 : 03 : 06]	[03 : 03: 06]	[03 : 06: 05]	3	4

Aplicando a transformação armadilha matriz de Hadamard ao código convolucional de memoria unitária $(4, 3, 1)$, temos que o d_{free} do código original era 3, e na maioria das transformações a distancia livre não diminuiu,teve um aumento, ou seja $d'_{free} = \{G'_0\ G'_1\} = 4$, sendo elas a: A_1, A_3, A_4 e A_5.Ja na transformação A_1 o d'_{free} passou a ser 2.

Porém todos os códigos obtidos tiveram uma redução na dimensão do espaço de operação, onde um coluna das submatrizes geradoras G'_0 e G'_1 é nula.

aplicando novamente as transformação armadilha:A_1, ,A_2, A_3, A_4 e A_5 nas submatrizes geradores: $G'_0 = [05 : 03 : 06]$ e $G'_1 = [06 : 06 : 05]$, obtemos um novo código com as seguintes submatrizes G''_0 e G''_1, o resultado esta mostrado na tabela 6.5.

Portanto o que se verifica com relação ao novo código, e que não houve mudança, de diminuição do d'_{free}, para o d''_{free}, ou seja permaneceu o mesmo $d'_{free} = d''_{free}$, aplicando somente nas duas primeiras submatrizes geradoras da tabela 6.3.

Tabela 6.4: **Transformação de Hadamard(H_4), aplicada novamente no código de** $r = \frac{3}{4}$

Código	A	$B(H_4)$	G''_0	G''_1	d'_{free}	d''_{free}
(4, 3, 1)	[04 : 03: 01]	[00 : 05 : 03 : 06]	[03 : 03: 06]	[06 : 05: 03]	4	4
	[06 : 03: 01]	[00 : 05 : 03 : 06]	[06 : 03: 06]	[00 : 05: 03]	2	2
	[04 : 02: 01]	[00 : 05 : 03 : 06]	[03 : 05: 06]	[06 : 06: 03]	4	4
	[01 : 02: 04]	[00 : 05 : 03 : 06]	[06 : 05: 03]	[03 : 06: 06]	4	4
	[05 : 03: 01]	[00 : 05 : 03 : 06]	[05 : 03: 06]	[05 : 05: 03]	4	4

1. **Código com Taxa** $R = \frac{3}{8}$**:**

Para o código convolucional $(8, 3, 1)$ de matrizes geradoras:

$$\mathbf{G_0} = \begin{pmatrix} 360 & : 074 & 227 \end{pmatrix} e\ \mathbf{G_1} = \begin{pmatrix} 146 & : 073 & 303 \end{pmatrix}$$

e $d_{free} = \{G_0\ G_1\} = 8$, foram utilizadas as transformações armadilhas A_1, A_2, A_3, A_4 e A_5, ja apresentadas e as transformações armadilhas B que são as representações matriciais dos elementos da matriz de Hadamard H_8.

Na tabela 6.4, é apresentada a transformações armadilhas A_i onde i varia 1 ate 5, sendo elas do tipo 3 x 3, As submatrizes geradoras do código convolucional G_0 e G_1 e a distancia livre tanto do original como do novo código, a também as submatrizes geradoras do código convolucional G'_0 e G'_1.

Tabela 6.5: **Transformação de Hadamard(H_8),** $R = \frac{3}{8}$

Código	A	$B(H_8)$	G'_0	G'_1	d_{free}	d'_{free}
(4, 3, 1)	[04 : 03: 01]	[00:125:63:146:17:132:74:151]	[00:151:151]	[00:17:00]	8	0
	[06 : 03: 01]	[00:125:63:146:17:132:74:151]	[00:151:151]	[17:17:00]	8	4
	[04 : 02: 01]	[00:125:63:146:17:132:74:151]	[00:00:151]	[00:17:00]	8	0
	[01 : 02: 04]	[00:125:63:146:17:132:74:151]	[151:00:00]	[00:17:00]	8	0
	[05 : 03: 01]	[00:125:63:146:17:132:74:151]	[151:151:151]	[00:17:00]	8	4

O código convolucional de memoria unitária com taxa $r = \frac{3}{8}$ teve uma redução muito grande, se comparado com o código original, pois o d_{free} é igual a 8, enquanto que aplicada a transformação A_1, A_3 e A_4 o d'_{free} passou a ser 0, e aplicada a transformação A_1 e A_5 o d'_{free} passou a ser 4.

Apesar de ter alcançado o objetivo de redução da capacidade de correção, todos os códigos obtidos tiveram uma grande redução na dimensão do espaço de operação, pois quatro colunas das submatrizes geradoras do código G_0 e G_1 são nulas.

7 Conclusão

Neste trabalho foram apresentadas algumas transformações armadilha, Matriz de Hadamard aplicadas a alguns códigos convolucional de memoria unitária, sendo que algumas dessas aplicações foram capazes de corrigir a capacidade de correção de erros de alguns dos códigos convolucionais ótimos considerados.

Sendo feita uma abordagem básica sobre os sistema de comunicação, mostrando a descrição de cada um dos blocos, e também introduzindo conceitos e definições de classes de códigos tais como: código de arvore, código de treliça e códigos convolucionais, classifica-se, em seguida os sistemas criptográficos em: sistemas criptográfico convencionais e sistemas criptográfico de chave publica , dentro do sistema de chave publica, analisa-se o Kanapsak binário e criptosistema RSA convencional de chave publica. Na analise do método knapsak binário para sistemas criptográfico e chave publica, é feita a descrição do método do sistema criptográfico, e estabelecidas as propriedades das funções armadilhas, a finalidade da função armadilha representada pela matriz de Hadamard, foi de reduzir a distancia livre do código convolucional a um valor especifico ou desejado para a aplicação.

A Proposta do trabalho, é a aplicação da transformação armadilha a matriz de Hadamard, de inicio é analisada a matriz de Hadamard, do tipo Sylvestre da ordem de 4 e 8, em seguida é feita a aplicação da transformação armadilha matriz de Hadamard , e a analise do efeito dessa aplicação as submatrizes geradoras de códigos convolucionais de memoria unitária. Inicialmente a transformação, é aplicada a um código convolucional de taxa $R = 2/4$, constatou-se que em comparação o código convolucional de taxa $R = 2/8$, teve uma variação onde verificamos que no de taxa $R = 2/4$ o valor do d_{free} baixou de 5 para 2 e somente em duas situações o valor do d_{free} passou para 4, enquanto que no código convolucional de taxa $R = 2/8$, observou-se, que todas as aplicações da transformação, A_1, A_2, A_3, A_4 *e* A_5, o valor do d_{free} baixou de 10 para 8, tornando-se dessa forma uma constante nos valores.comparando dessa forma a mesma quantidade entradas, ou seja $k = 2$.

Já na aplicação da transformação de Hadamard ao código de taxa $R = 3/4$,

tivemos como resultado, e que apenas uma das transformações o valor do d_{free} baixo de 3 para 2, enquanto que na maioria das transformações, resultou em um d_{free} que passo de 3 para 4, ou seja não foi bom um resultado, sendo feito uma nova aplicação somente com as primeiras submatrizes encontradas a partir da transformação, finalidade de se verificar, o que resultaria.

A pois uma nova aplicação da transformação com as primeiras,novas submatrizes verificou-se que não houve nenhum avanço, obteve-se os mesmos valores do d_{free}, no teve melhora, ou seja preservou-se.

Aplicando o transformação a outro código, com a mesma quantidade de entrada do código anterior, cuja taxa é $R = 3/8$, tivemos um ótimo resultado pois em três das transformações o valor do d_{free} passou de 8 para 0, um declive enorme, que gerou satisfação mas interrogação, devido a esse declive, agora dois dessas transformações aplicadas a esse código o valor do d_{free}, passou de 8 para 4, descida aceitável.

Como todo trabalho de pequisa, os resultados de pesquisa, os resultados não completam e não finalizam as investigações, ou seja existem sempre as possibilidades de aprimoramento do trabalho apresentado e de sua extensão, analisando pontos não abordados. Dentre estes pontos são destacados:

- Aplicação da transformação armadilha matriz de Hadamard, a outros códigos $(n, k, 1)$ memória unitária e também a códigos (n, k, m), sendo $m \neq 1$.
- Analisar as classes de códigos convolucionais com proteção desigual, encontrada nesse trabalho, que foi o código com taxa $R = 3/8$, onde o valor do d_{free} passou de 8 para 0.
- Sugerir a aplicação das outras transformações armadilha: As representações matriciais dos elementos do grupo de permutação e As matrizes triangulares superiores, a códigos de memoria unitária $(n, k, 1)$, com taxas utilizadas neste trabalho.

REFERÊNCIAS

[1] BENCHIMOL, I. B. ***Módulo de treliça mínimo para códigos convolucionais***. Recife: UFPE, 2012.

[2] COSTELLO Jr., D. J. ***A construction technique for random - error - correting convolutional codes***. IEEE Trans. Inform. Theory, IT - 15 (5), pp 631 - 636, 1969.

[3] COSTELLO Jr., D. J. ***Construction of convolutional codes of sequential decoding***. PhD Thesis - University of Notre Dame, Notre Dame, IN, 1969.

[4] COSTELLO Jr., D. J. and LIN, S. ***Error control coding***. Upper Saddle River: Pearson Prentice Hall, 2nd ed., 2004.

[5] COUTINHO, S. C. ***Números inteiros e criptografia RSA***. Rio de Janeiro: IMPA, 2011.

[6] DIFFIE, W. and HELLMAN. M. E. Privacy and authentication: ***An introdution to cryptography***. Proceedings of the IEEE, vol. 67, No. 3, pp. 397 - 427, Nov. 1979.

[7] GILL, A. ***Linear sequential circuit - analysis, synthesis, and applications***. Mc Graw - Hill, New York, 1966.

[8] HAYKIN, S. ***Error control coding - Systems communications***. John Wiley & Sons, 4. ed, ISBN 0471178691, New York, 2001.

[9] JOHANNESSON, R. and ZIGANGIROV, K. S. ***Fundamentals of convolutional coding***. Wiley - IEEE Press, 1999.

[10] SYLVESTER, J. J. ***Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tesselated pavements in two or more colours, with applications to Newtons rule, ornamental tile-work, and the theory of numbers***, Phil. Mag., vol. 34, pp. 461-475, 1967.

[11] LAUER, G. S. ***Some optimal partial - unit - memory codes***. IEEE Trans. Inform. Theory, IT - 25 (2), pp 240 - 243, 1979.

[12] LEE, L. N. ***Short unit - memory byte - oriented binary convolutional codes having maximal free distance***. IEEE Trans. Inform. Theory, IT - 22 (3), pp 349 - 352, 1976.

[13] MASSEY, J. L. ***Catastrophic error - propagation in convolutional codes***. Proceedings of the 11th Midwest Circuit Theory Symposium, pp 583 - 587, Notre Dame, IN, 1968.

[14] MASSEY, J. L. and SAIN, M. K. ***Inverses of linear sequential circuits***. IEEE Trans. Comp., C - 17, pp 330 - 337, 1968.

[15] MOREIRA, J. C. and FARRELL, P. G. ***Essentials of error control coding***. John Wiley & Sons Ltd, 2006.

[16] PALAZZO, R. Jr.B.F. Uchoâ Filho, J. P.Arpasi, ***Notas de Aula - Codigos Convolucionais***. Departamento de Telecomunicações.FEEC-UNICAMP. Campinas - SP, 2005.

[17] PALAZZO, R. Jr. ***Cryptography Systems Based on Trellis Codes***. III Simpósio Brasileiro de Telecomunicações. S. J. dos Campos - SP, Setembro 02-04, 1985.

[18] PALAZZO, R. Jr. ***Propriedades das Funções Armadilhas e Analise de Cripto sistemas de Chaves Públicas Usando Códigos Convolucionais***. V Simpósio Brasileiro de Telecomunicações. Campinas - SP, Setembro 08-10, 1987.

[19] PALAZZO, R. Jr. ***Códigos de Treliça Fixos e variantes no Tempo***. Dissertação de Livre Docência. Universidade Estadual de Campinas. Campinas, 1987.

[20] PIEDADE, D. C. S.; SOUSA, E. V. de. FILHO, J. C. S. ***Análise de uma decodificação abrupta pelo algoritmo de Viterbi***. In: XXXVI - CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL- CNMAC, 36. 2016, Gramado. **Anais**.São Paulo: SBMAC, 2016.

[21] SANTOS, P. A. ***Uma proposta de um sistema criptográfico de chave pública utilizando códigos convolucionais clássicos e quânticos***. Campinas: UNICAMP, 2008.

[22] SCHLEGEL, C. and PÉREZ, L. ***Trellis and turbo coding***. Wiley - IEEE Press, 2004.

[23] SINGH, S. ***O livro dos códigos***. Rio de Janeiro, 2005.

[24] SOUSA, E. V. ;PIEDADE, D. C. S.; de. FILHO, J. C.***Analise da Representação Polinomial e Discreta dos Codificadores Convolucionais***.In:XVIII ENMC-National Meeting on Computational Modeling and VI ECTM - Meeting on Materials Science and Technology.19 a 21 de Outubro de 2016, UFPB - João Pessoa - PB

[25] VALDEVINO, A. ***Criptografia Caótica***. Brasília, 2010.

[26] VANSTONE, S. A. and ORSCHOT, P. ***An introduction to error correcting codes with applications***. Kluwer Academic Publishers, 2001.