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

DAYANE CRISTHNY SOUSA PIEDADE

CONTRIBUIÇÃO À CAPACIDADE DE CORREÇÃO DE ERROS EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA VIA TRANSFORMAÇÃO ARMADILHA

São Luís
2017

DAYANE CRISTHNY SOUSA PIEDADE

CONTRIBUIÇÃO À CAPACIDADE DE CORREÇÃO DE ERROS EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA VIA TRANSFORMAÇÃO ARMADILHA

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

Orientador: Prof. Dr. João Coelho Silva Filho

Universidade Estadual do Maranhão - UEMA

São Luís

2017

Piedade, Dayane Cristhny Sousa

Contribuição à capacidade de correção de erros em códigos convolucionais de memória unitária via transformação armadilha / Dayane Cristhny Sousa Piedade - 2017.

66 f.

Orientador: Prof. Dr. João Coelho Silva Filho.

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

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

CDU 004:621.3

DAYANE CRISTHNY SOUSA PIEDADE

CONTRIBUIÇÃO À CAPACIDADE DE CORREÇÃO DE ERROS EM CÓDIGOS CONVOLUCIONAIS DE MEMÓRIA UNITÁRIA VIA TRANSFORMAÇÃO ARMADILHA

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

Aprovado em 09 de Março de 2017

BANCA EXAMINADORA

Prof. Dr. João Coelho Silva Filho

Universidade Estadual do Maranhão - UEMA

Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva

Universidade Estadual do Maranhão - UEMA

Prof. Dr. Vandenberg Lopes Vieira

Universidade Estadual da Paraíba - UEPB

A meu Santo protetor, São José de Ribamar,
pelas bênçãos e proteção.

AGRADECIMENTOS

Eis que chegou a hora de expressar meus sinceros agradecimentos como forma de gratidão a todos que me ajudaram a alcançar essa vitória.

A Deus, por ter me proporcionado o maior e mais importante dos dons, a vida. Por ter me dado força espiritual e iluminado sempre meus caminhos.

Ao meu orientador e pai acadêmico, Professor Dr. João Coelho Silva Filho, pela competência, dedicação, disponibilidade e paciência ao me orientar neste e em outros trabalhos.

À minha família, em especial a meus pais, Dulce e Francisco, e meus irmãos, pelo incentivo, apoio e compreensão neste momento tão importante da minha vida. A minha sobrinha, Clarinha, com seu exemplo de vida, sempre me ensina a lutar mesmo quando as condições são adversas.

Aos meus amigos mais antigos e aos que durante o curso dividimos diversos momentos e que juntos pudemos concluir de forma prazerosa mais uma etapa de nossas vidas: Charles, David, Edson, Ismael, Luciano e Elzenir.

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

À secretária, Sara Martyns, com sua compreensão e competência sempre esteve disposta a nos ajudar.

À UEMA, em especial, a PECS, pela oportunidade de desenvolver e concluir com sucesso o mestrado.

À banca examinadora, pela disponibilidade em avaliar esta dissertação.

À FAPEMA pelo suporte financeiro.

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

“Quem como Deus?”

São Miguel Arcanjo

Na grandeza é preciso ser pequeno.

Dayane Cristhny

RESUMO

Este trabalho propõe uma matriz de transformação armadilha que ao ser aplicada às submatrizes geradoras, de um código convolucional de memória unitária, geram outras submatrizes geradoras capazes de reduzir a capacidade de correção de erros do código através da análise da distância livre. O 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 Geradora. Distância Livre.

ABSTRACT

This work proposes a matrix of trap transformation that, when applied to the generative submatrix, of a unitary memory convolutional code, generate other generating submatrices capable of reducing the error correction capacity of the code through the analysis of the free distance. The process of reducing the error correction capacity of the codes caused by the scrambling 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. Generating Matrix. Free Distance.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AV	Algoritmo de Viterbi
CSL	Circuito Sequencial Linear
CCP	Criptossistema de Chave Pública
MATLAB	MATrix LABoratory
MU	Memória Unitária
MUP	Memória Unitária Parcial
NP	Não Polinomial
P	Polinomial
PM	Problema da Mochila

Lista de Símbolos

F_q	Corpo finito
(n, k, m)	Código convolucional com n saídas, k entradas e m memórias
R	Taxa de um código
$\mathbb{Z}_2$	Grupo de Galois aditivo
d_{free}	Distância livre
t_{free}	Máxima capacidade de correção de erros
mdc	Máximo divisor comum
ϕ	Fluxo máximo
w	Peso de Hamming da palavra-código
G	Matriz geradora
A	Matriz de ordem $k \times k$
B	Matriz de ordem $n \times n$
T	Matriz triangular inferior de ordem $n \times n$

SUMÁRIO

1 INTRODUÇÃO

Os árabes foram um dos principais povos que contribuíram significativamente na evolução da *criptografia*, pois foram os primeiros a desenvolver a análise da cifra que depois vinha a ser conhecida como "*criptoanálise*, que é o ramo da criptologia que estuda formas de descodificar uma mensagem sem conhecer a chave", [5].

"Em 1839, Samuel Morse desenvolveu o popular código Morse, um alfabeto cifrado em sons curtos e longos", [24]. Morse inventou também o telégrafo, que fez uma revolução nos meios de comunicação. Com isso, a cifragem de mensagens se tornou uma necessidade absoluta.

"Com a invenção do computador, a área da criptografia realmente se desenvolveu, incorporando complexos algoritmos matemáticos", [21]. Os esforços em decifrar códigos formaram a base para a engenharia de computação moderna. "Com as rápidas evoluções nos meios de transmissão de dados, a forma de transmitir informações sigilosas foi constantemente se modificando", [26].

Atualmente, existem dois métodos pelos quais podem ser feitos os processos de *cifragem* e *decifragem* dos sistemas criptográficos, a saber: o método convencional e o método por chaves públicas. No método convencional, dois usuários que desejam se comunicar devem pré-estabelecer uma chave comum através de um canal seguro (meio físico que interliga os dois pontos) de envio e recepção de mensagens. Com N usuários, o número de chaves criptográficas necessárias é, no máximo, $N(N-1)/2$. Nos sistemas de chaves públicas, são utilizadas duas chaves, uma para cifrar e outra para decifrar. A chave de cifragem é enviada publicamente ao usuário que deseja transmitir uma mensagem cifrada. A chave de decifragem mantida secreta permite ao usuário receptor da mensagem cifrada decifrar a mesma.

Na criptografia contemporânea não se utiliza mais a suposição que qualquer sistema pode ser seguro, ou seja, somente seus criadores tem acesso à metodologia ou aos algoritmos internos do sistema. A ideia básica da utilização de chave pública é a eliminação da necessidade de utilização de um canal eficiente e de alta segurança para a distribuição de chaves. Do ponto de vista de implementação, os sistemas que usam chaves

públicas se fundamentam na existência de métodos diferentes para cifragem e decifragem.

É natural que ao transmitir um dado podem ocorrer problemas como interferências eletromagnéticas ou erros humanos, fazendo com que a mensagem recebida seja diferente daquela que foi enviada. A principal ideia da teoria dos códigos corretores de erros é codificar a informação inicial, adicionando informação redundante, de forma que, ao receber o sinal modificado seja possível, de alguma forma, recuperar a mensagem original.

A pesquisa é uma análise à capacidade de correção de erros dos códigos convolucionais utilizando a transformação armadilha proposta em códigos de memória unitária. Este processo proporciona um aumento no grau de privacidade da informação a ser enviada, devido a dois fatores: a resolução do Problema da Mochila para a determinação de códigos ótimos de memória unitária e a redução da capacidade de correção de erro dos códigos ocasionada pelo embaralhamento das colunas das submatrizes geradoras.

Neste contexto, o trabalho além de ser uma contribuição quanto à capacidade de correção de erros dos códigos convolucionais, contribui, também, a partir da conclusão obtida pelos resultados, a criptografia de chave pública e a transformação armadilha. Além disso será utilizado para as operações de distâncias livres, máxima capacidade de correção de erros e matrizes geradoras o software MATLAB.

Este trabalho encontra-se organizado da seguinte forma:

No capítulo 2, aborda-se conceitos importantes para a fundamentação da dissertação. Apresenta-se códigos convolucionais, definindo as principais formas de representação de codificadores e códigos convolucionais. Distância livre é um dos principais conceitos, ao qual concentra-se a ideia da proposta, estando também definido neste capítulo. Para o entendimento da decodificação dos códigos convolucionais é apresentado e exemplificado o método abrupta do algoritmo de Viterbi. Em seguida, mostra-se os tipos de sistema criptográficos inclusive o de chave pública que será abordado na aplicação deste trabalho, e códigos ótimo de memória unitária. Por fim, é definido e apresentado as propriedades das transformações armadilha, o que dará fundamento e consistência à transformação armadilha proposta.

No capítulo 3, apresenta-se a transformação armadilha proposta, Matriz Triangular Inferior, com definição e propriedades. Esta matriz possui uma estrutura bem definida, sendo capaz de diminuir a distância livre dos códigos convolucionais de memória unitária. Para isso, é feito a análise da transformação armadilha proposta, aplicando

esta às submatrizes geradoras dos códigos convolucionais de memória unitária com taxas $R = 2/4$ e $R = 2/8$. Os resultados e conclusão desta aplicação é discutido também neste capítulo, verificando o poder de correção dos códigos através da alteração da distância livre.

No capítulo 4, a transformação armadilha proposta é aplicada ao sistema criptográfico de chave pública com objetivo de corrigir o erro aplicado à mensagem a ser enviada.

Finaliza-se o trabalho com o capítulo 5, no qual encontra-se as contribuições, produção científica e sugestões para trabalhos futuros.

2 CÓDIGOS CONVOLUCIONAIS E SISTEMAS CRIPTOGRÁFICOS

Este capítulo apresenta algumas definições sobre códigos convolucionais e sistemas criptográficos, importantes para fundamentação do trabalho.

2.1 Codificador Convolucional

Um *Circuito Sequencial Linear* (CSL) possui como variáveis básicas a entrada, a saída e os estados (conteúdos dos registros de deslocamento). Neste tipo de circuito, a sequência de saída é uma aplicação linear da sequência de entrada e dos estados. Um **codificador convolucional** (n, k, m) pode ser implementado através de um circuito sequencial linear com n saídas, k entradas e m memória. "Em um circuito sequencial linear, os sinais escolhidos dentre os elementos de um corpo finito F_q são aplicados simultaneamente a todos os terminais de entrada em instantes discretos de tempo", [7]. Esse circuito consiste de uma rede de interconexões entre um número finito de componentes primitivos. Dois tipos destes componentes têm importantes funções em codificadores convolucionais: os **somadores** (que implementam adição módulo-q) e os **elementos da memória** para atrasar (ou armazenar) um sinal de entrada por um sinal de tempo.

O código gerado por um codificador convolucional é chamado de código convolucional (n, k, m). "O código convolucional é definido como o conjunto de palavras-código geradas pelo codificador para todas as possíveis sequências de informação", [1]. Os códigos convolucionais diferem dos códigos de bloco, pois contém memória e suas n saídas em uma certa unidade de tempo i não depende somente das k entradas naquele tempo, mas também dos m blocos de entradas anteriores. A taxa de um código é indicado por $R = k/n$.

A sequência de informação de um codificador convolucional pode ser escrita na forma

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

em que $u_i = (u_i^{(1)}, ..., u_i^{(k)})$ é o bloco de informação de comprimento k no instante de tempo i. Similarmente, a sequência codificada, chamada de palavra-código, é

$$v = (v_0, v_1, v_2, ...) = (v_0^{(1)}, v_0^{(2)}, ..., v_0^{(n)}, v_1^{(1)}, v_1^{(2)}, ..., v_1^{(n)}, ...),$$

em que $v_i = (v_i^{(1)}, ..., v_i^{(n)})$ é o bloco codificado de comprimento n no instante de tempo i. Cada bloco v_i depende do bloco u_i bem como de m blocos de informação passados $u_{i-1}, ..., u_{i-m}$. "O número total de elementos de memória para armazenar os bits passados utilizados pelo codificador é denotado por v", [18].

A Figura 2.1 mostra um codificador convolucional que consiste de um registro de deslocamento contendo duas memórias e dois somadores no grupo de Galois, $\mathbb{Z}_2$. A sequência de informação u é deslocada da esquerda para a direita de um par de bits por unidade de tempo, e duas saídas codificadas $v_i^{(1)}$ e $v_i^{(2)}$ são geradas pelos somadores em $\mathbb{Z}_2$. Portanto, a cada unidade de tempo i, o bit de informação corrente (a primeira célula) é u_i, os bits de informação passados são u_{i-1} e u_{i-2} (são a segunda e terceira células, respectivamente) e as duas saídas são $v_i^{(1)}$ e $v_i^{(2)}$. A primeira saída, $v_i^{(1)}$, é a soma em $\mathbb{Z}_2$ de u_i, u_{i-1} e u_{i-2}, enquanto que a segunda saída, $v_i^{(2)}$, é a soma de u_i e u_{i-1}. A sequência de bits codificados é multiplexada e transmitida pelo canal.

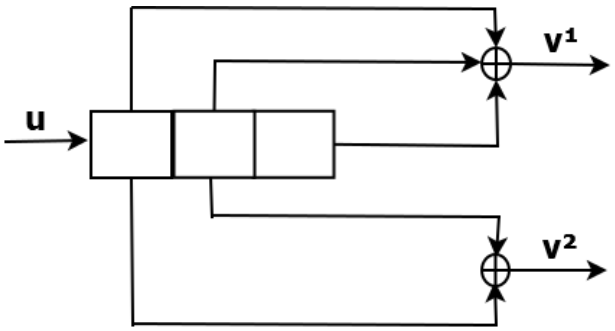

Figura 2.1: Codificador convolucional (2, 1, 2).
Fonte: Autora

Exemplo 2.1.1. Considere a sequência de informação $u = 1011001$. Assuma que as duas últimas células do codificador da Figura 2.1 contenham inicialmente zeros (0s). Assim, no instante $i = 0$, o bit de informação $u_0 = 1$ é codificado em dois bits de saída, $v_0^{(1)} = 1$ e $v_0^{(2)} = 1$. No instante $i = 1$, $u_0 = 1$ é deslocado para dentro da segunda célula. O bit de informação corrente $u_1 = 0$ gera dois bits codificados $v_1^{(1)} = 1$ e $v_1^{(2)} = 1$. No instante $i = 2$, $u_1 = 0$ é deslocado para dentro da segunda célula e $u_0 = 1$ é deslocado

para dentro da terceira célula. O bit de informação corrente $u_2 = 1$ gera dois bits codificados $v_2^{(1)} = 0$ e $v_2^{(2)} = 1$, e assim sucessivamente, obtendo a sequência codificada, $v = 111101000110111110$. É importante observar que os dois últimos pares de bits codificados são obtidos através do deslocamento de 0s após u_6, nos instantes $i = 7$ e $i = 8$.

2.2 Representações de Codificadores Convolucionais

Nesta seção, são apresentadas as principais representações dos codificadores convolucionais. Para cada uma das representações é mostrado um exemplo para efeito de compreensão. A sequência código obtida, v, em cada caso, é a mesma sequência obtida no Exemplo 2.1.1.

2.2.1 Representação Discreta

A representação de um codificador convolucional de taxa $R = k/n$ é dado por nk sequências de geradores de comprimento $m + 1$,

$$g_i^{(j)} = \left(g_{i,0}^{(j)}, g_{i,1}^{(j)}, \cdots, g_{i,m}^{(j)}\right), \quad \text{para} \quad i = 1, \cdots, k \quad \text{e} \quad j = 1, \cdots, n,$$

onde i representa a entrada, j a saída e m o número de memórias. As sequências de geradores $g_i^{(j)}$ são as respostas ao impulso do codificador.

A operação de codificação convolucional é a convolução discreta da sequência de informações com as sequências de geradores e é escrito como

$$v^{(j)} = \sum_{i=1}^{k} u^{(i)} * g_i^{(j)} \tag{2.1}$$

com $j = 1, \cdots, n$, onde $*$ é a operação de convolução.

A convolução discreta (2.1) pode ser expressa de forma compacta através de uma multiplicação de matrizes. As k sequências de informações podem ser escritas como sendo a sequência

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

onde $u_t = \left(u_t^{(1)}, \cdots, u_t^{(k)}\right)$ é o bloco de informação no instante de tempo t. Da mesma

forma, a palavra-código obtida a partir das n sequências codificadas é dada por

$$v^{(i)} = (v_0, v_1, v_2, \cdots) = \left(v_0^{(1)}, v_0^{(2)}, \cdots, v_0^{(n)}, v_1^{(1)}, \cdots, v_1^{(n)}, \cdots\right),$$

em que $v_t = \left(v_t^{(1)}, \cdots, v_t^{(n)}\right)$ é o bloco codificado no instante de tempo t. Assim, a codificação convolucional pode ser escrita como

$$v = u\ G, \tag{2.2}$$

onde G é uma matriz geradora semi-infinita da forma

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_{m-1} & G_m & \\ & & G_0 & \cdots & G_{m-2} & G_{m-1} & G_m \\ & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix},$$

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

$$G_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix},$$

em que $g_{i,l}^{(j)}$ é a sequência geradora entre a i-ésima entrada e a j-ésima saída no l-ésimo instante de tempo.

Exemplo 2.2.1. Considere a mensagem $u = 1011001$ aplicada ao codificador da Figura 2.1. Para este codificador, pode-se escrever os vetores conexão g_1 e g_2 para os dois somadores como:

$$g_1 = 111,\ g_2 = 110 \ \text{ e } \ g = 111110.$$

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

$$v = u \cdot G = (1011001) \begin{pmatrix} 11 & 11 & 10 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 11 & 11 & 10 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 11 & 11 & 10 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 11 & 11 & 10 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 11 & 11 & 10 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 11 & 11 & 10 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 11 & 11 & 10 \end{pmatrix} = 111101000110111110.$$

No Exemplo 2.2.1, uma sequência de informação u de comprimento $N = 7$ bits foi codificada em uma palavra-código v de comprimento $n = 18$ bits. Os quatro últimos bits na palavra-código correspondem a $m = 2$ zeros que são usados para retornar ao estado inicial do codificador. A primeira linha de $G, g = 111110$, é chamada sequência geradora do código e é a partir desta que obtém-se a matriz geradora, deslocando, na segunda linha, a primeira linha de duas posições para a direita. A terceira linha obtém-se através do deslocamento da segunda linha de duas posições para a direita, e assim sucessivamente.

Considerando um codificador (n, k, m), uma sequência de comprimento N é codificada em uma palavra código com comprimento $M = (N + m)$ blocos, ou seja, com $n(N+m)$ bits, em que os últimos nm bits são gerados a partir de m blocos de informação com todos os bits $0s$ anexados à sequência de informação. Portanto, a matriz geradora G é uma matriz com kN linhas e $n(m + N)$ colunas. A cada unidade de tempo, um conjunto sucessivo de k linhas é deslocado para a direita de n posições. Então, a matriz geradora captura o deslocamento da sequência de informação na operação de convolução implementada com o uso de registros de deslocamento.

2.2.2 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

$$g_i^{(j)}(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)} D + \cdots + g_{i,m}^{(j)} D^m,$$

onde $i = 1, \cdots, k$ e $j = 1, \cdots, n$.

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

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

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, então a operação de decodificação pode ser escrita como sendo

$$v^{(j)}(D) = \sum_{1=1}^{k} u_{(i)} g_i^{(j)}(D). \tag{2.3}$$

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

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

onde $g_i^{(D)}$ 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:

$$v(D) = u(D)G(D). \tag{2.4}$$

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 $j = 1, \cdots, n$.

Exemplo 2.2.2. Considere a mensagem

$$u(x) = 1 + x^2 + x^3 + x^6$$

aplicada pelo codificador da Figura 2.1. Para este codificador, pode-se escrever os vetores conexão $g_1(x)$ e $g_2(x)$ para os dois somadores como:

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

os polinômios geradores, em termo do operador de retardo D, são:

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

onde o termo de mais baixa ordem corresponde ao estágio de entrada do codificador. A sequência de saída do codificador é obtida por

$$v(D) = u(D)g_1(D) \text{ intercalado com } u(D)g_2(D).$$

Escrevendo a mensagem, $u(x)$, em termos do operador de retardo, tem-se:

$$u(x) = 1 + x^2 + x^3 + x^6, \text{ assim, } u(D) = 1 + D^2 + D^3 + D^6,$$

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

A sequência código v é obtida pelo somatório de $u(D)\ g_1(D)$ com $u(D)\ g_2(D)$. Assim, tem-se:

$$u(D)g_1(D) = (1 + D^2 + D^3 + D^6)(1 + D + D^2) = 1 + D + D^5 + D^6 + D^7 + D^8$$

$$u(D)g_2(D) = (1 + D^2 + D^3 + D^6)(1 + D) = 1 + D + D^2 + D^4 + D^6 + D^7$$

$$v = 111101000110111110.$$

Assim como na representação discreta, a representação polinomial apresentou os quatro últimos bits na palavra código correspondentes a $m = 2$ zeros que são usados para retornar ao estado inicial do codificador.

Em alguns casos a sequência final da representação discreta difere da representação polinomial. Este fato acontece porque os zeros necessários para o esvaziamento do registrador de deslocamento que produz bits adicionais à sequência código pode ocorrer apenas em uma representação. São apresentados exemplos em que estes casos acontecem em [25].

2.2.3 Representação Gráfica

A operação de codificação convolucional é melhor entendida pela representação gráfica. Neste trabalho apresenta-se duas representações gráficas que são úteis para o estudo de códigos convolucionais, a saber, o *diagrama de estados* e a *treliça*.

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, 16]. A Figura 2.2 mostra o diagrama de estados para o codificador (2, 1, 2) da Figura 2.1. Cada elipse no diagrama é um estado. Assim, o estado deste codificador no instante de tempo i é formado pelos bits contidos na primeira e segunda células e o estado no instante de tempo $i-1$ é formado pelos bits contidos na segunda e terceira células. Cada bit que entra no codificador produz uma saída (dois bits neste 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. 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. As setas indicam de onde sai e para onde chega cada par de bits.

Outra representação muito comum para o codificador convolucional é o **diagrama de treliça**. Este diagrama é uma representação do diagrama de estado do codificador evoluindo no tempo, ou seja, representa cada estado de tempo como um diagrama de estado particionado. Na Figura 2.3 é mostrado o diagrama de treliça para o codificador (2, 1, 2), o qual existem quatro possíveis estados. Assume-se o estado todo nulo (00) como estado inicial. De cada estado saem duas transições. Cada linha é acompanhada pelos correspondentes bits de saída.

2.3 Distância Livre de um Código Convolucional

O desempenho dos códigos convolucionais está diretamente relacionado dentre outros às propriedades de distâncias de código. A distância livre, definida por Costello, [2,

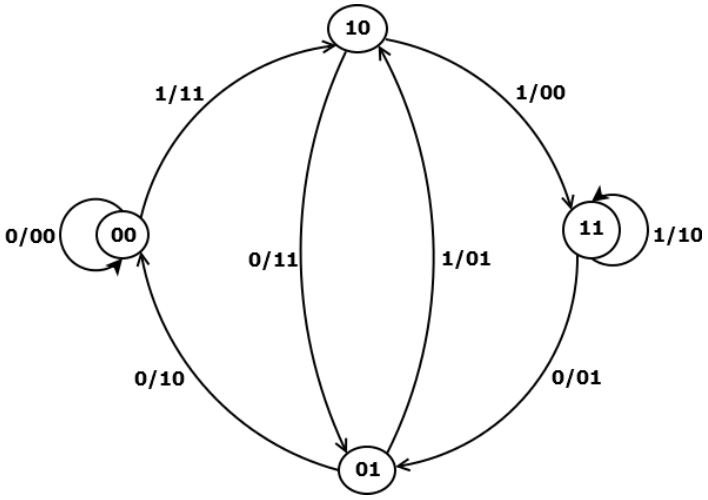

Figura 2.2: Diagrama de estados para o código convolucional (2, 1, 2).

Fonte: Autora

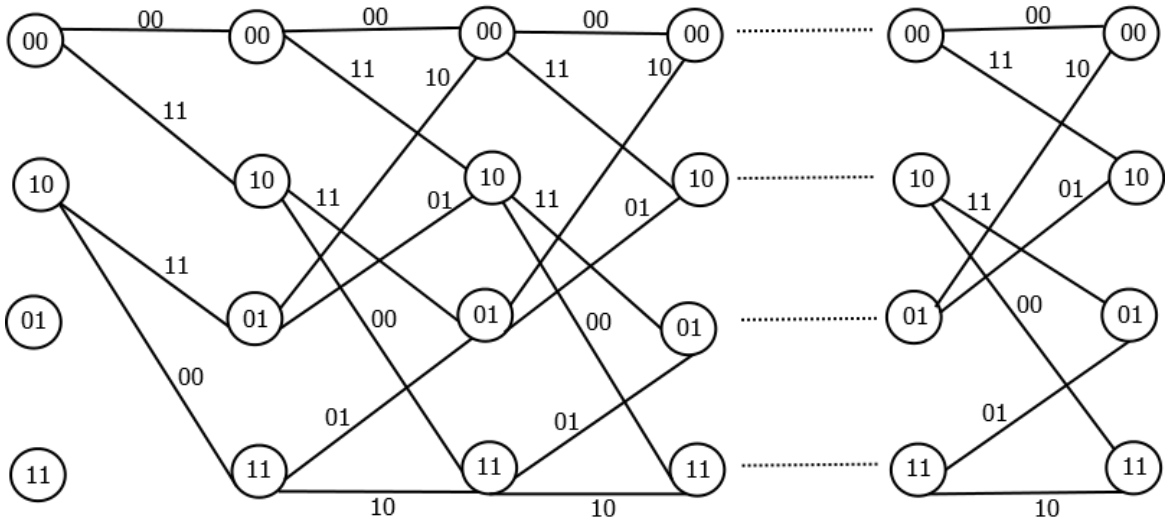

Figura 2.3: Diagrama de treliça para o código convolucional (2, 1, 2).

Fonte: Autora

3], é considerada a mais importante, pois é por meio desta que se estabelece a capacidade de correção ou de detecção de erros associada aos códigos convolucionais.

Para o entendimento da definição de distância livre de um código é necessário apresentar dois conceitos importantes, peso e distância de Hamming.

Definição 2.3.1. Define-se **peso de Hamming**, o número de dígitos não nulos da palavra-código.

Definição 2.3.2. A **distância de Hamming** entre as palavras-código corresponde ao número de posições onde estas duas palavras-código diferem.

Exemplo 2.3.1. Considere duas palavras-código v_1 e v_2 dadas por $v_1 = (0, 1, 1, 0, 1)$ e

$v_2 = (1, 1, 0, 1, 1)$. Os pesos de v_1 e v_2 são respectivamente 3 e 4, e a distância de Hamming entre v_1 e v_2 é 3.

Define-se como **distância livre** (d_{free}) de um código convolucional, "o caminho de menor peso de Hamming que diverge do estado S_0 em $t = 0$ e volta para S_0 pela primeira vez em algum instante futuro e permanece neste estado", [4], ou seja, distância livre é a distância de Hamming mínima entre quaisquer duas palavras-código distintas. Matematicamente, tem-se

$$d_{free} = min\{d_H(v, v^{'}) : u \neq u^{'}\} = min\{w_H(v) : u \neq 0\} = min\{w_H(uG) : u \neq 0\}, \quad (2.5)$$

onde v e $v^{'}$ são as sequências codificadas correspondentes às sequências de informação u e $u^{'}$, respectivamente. 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.

A obtenção do d_{free} de um código pode ser feita traçando-se dois caminhos distintos sobre a treliça a partir de duas transições que partam de um mesmo estado. Estes caminhos devem ser traçados de forma a se encontrarem novamente em um mesmo estado após o menor número possível de transições. O número de bits que diferirem entre as sequências dos dados de saída representadas por cada um dos caminhos será a distância livre.

Exemplo 2.3.2. Através do diagrama de estados da Figura 2.2 ou do diagrama de treliça da Figura 2.3, pode-se verificar que a distância livre do código gerado pelo codificador do código convolucional (2, 1, 2) é $d_{free} = 4$. O caminho de d_{free} é constituído de quatro transições de estados, a saber: $a = 00 \longrightarrow b = 10 \longrightarrow c = 11 \longrightarrow d = 01 \longrightarrow a = 00$, referentes à palavra-código $v = 11000110$ e à sequência de informação $u = 1100$.

Um código convolucional tem Distância Livre Ótima (DLO) se sua distância livre é igual ou superior a qualquer outro código convolucional de mesma taxa e mesmo comprimento de restrição de saída. Devido esta definição é possível definir a **máxima capacidade de correção de erros**.

Definição 2.3.3. A máxima capacidade de correção de erros t_{free} de um código convolucional é dada por

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

onde $\lfloor x \rfloor$ é a função piso dada pelo menor inteiro de x.

O parâmetro t_{free} é utilizado também para analisar o desempenho de algoritmos de decodificação baseados no princípio de máxima verossimilhança, compreendendo o algoritmo de Viterbi, que será apresentado na seção 2.5.

2.4 Códigos Convolucionais Específicos

Nesta seção, são apresentados três códigos convolucionais importantes, sendo o código convolucional de memória unitária utilizado no desenvolvimento da proposta deste trabalho.

2.4.1 Códigos Convolucionais 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]. Quando uma palavra-código é transmitida através de um canal ruidoso, existe uma probabilidade não nula de que um número finito de erros introduzidos pelo canal provoque um número infinito de erros de decodificação. Situação esta conhecida como propagação catastrófica de erros.

Para evitar a propagação catastrófica de erros, os codificadores de taxa $1/n$ devem satisfazer a condição, [14]:

$$mdc\ [g^{(j)}(D), j = 1, 2, \cdots, n] = 1, \tag{2.6}$$

onde *mdc* denomina o *máximo divisor comum*.

Um código convolucional é **catastrófico** se em seu diagrama de estado existe uma malha fechada de peso nulo.

Uma matriz geradora $G(D)$ é bf não catastrófica se, e somente se, "existir uma matriz inversa à direita $G^{-1}(D)$ apresentando apenas entradas polinomiais", [10, 22].

2.4.2 Códigos Convolucionais de Memória Unitária

Seja (n, k, m) um código convolucional com taxa $R = k/n$ e m memórias.

Sabe-se que um código convolucional (n_0, k_0, m) pode ser considerado como

um código ($n' = mn_0, k' = mk_0, m' = 1$), por representar

$$G_0' = \begin{bmatrix} G_0 & G_1 & \cdots & G_{m-1} \\ 0 & G_0 & \cdots & G_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G_0 \end{bmatrix} \text{ e } G_1' = \begin{bmatrix} G_m & 0 & \cdots & 0 \\ G_{m-1} & G_m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ G_1 & G_2 & \cdots & G_m \end{bmatrix}. \quad (2.7)$$

Os códigos representados em (2.7) foram definidos por Lee, [12], como **códigos de memória unitária**. Estes códigos são equivalentes, uma vez que para uma mesma sequência semi-infinita de dígitos de informação, produzem a mesma sequência semi-infinita codificada. Assim, se for definido a complexidade como sendo q^{mk}, ambos apresentando este mesmo valor, e para uma dada taxa, o valor máximo de d_{free} é obtido dentro do subconjunto de codificadores convolucionais com memória unitária.

Assim, o processo de codificação para tais códigos pode ser representado simplesmente por

$$v^{(j)} = u^j \cdot G_0 + u^{j-1} \cdot G_1, \quad \text{para} \quad j = 0, 1, 2, \quad (2.8)$$

onde $v^{(j)}$ é a sequência codificada, u^j é a sequência de informação e $v^{(j)} = 0, \forall \quad j < 0$.

2.4.3 Códigos Convolucionais de Memória Unitária Parcial

Existe uma classe de códigos convolucionais denominada **códigos convolucionais de Memória Unitária Parcial** (MUP) proposto por Lauer, [11]. Esta classe de códigos convolucionais possui a importante característica de conter codificadores mínimos atingindo a mesma distância livre que a dos códigos de memória unitária.

A propriedade de minimalidade resulta da submatriz G_1' tendo posto menor que k. Isso, quando comparando ao código de memória unitária, implica menor complexidade em termos do número de estados. Lauer definiu a complexidade de estado μ de um codificador de memória unitária como sendo o posto de G_1'.

Para um código MUP ser não catastrófico é suficiente que G_0 tenha posto completo e que as μ linhas não nulas de G_1 sejam linearmente independentes entre si e linearmente independentes das linhas de G_0. A importância de μ é que este determina o número 2^μ de estados no decodificador quando utilizado o *algoritmo de Viterbi*.

2.5 Decodificação dos Códigos Convolucionais

O algoritmo de decodificação mais conhecido e abordado nesta seção é o algoritmo de Viterbi.

2.5.1 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 convolucionais", [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 distância de Hamming constitui uma métrica.

Definição 2.5.1. [27] Dado dois elementos u e v do código, a **distância de Hamming** é definida por

$$d(u,v) = |\{i|u_i \neq v_i, 1 \leq i \leq n\}|\,. \tag{2.9}$$

O funcionamento da decodificação por máxima verossimilhança baseia-se em comparar a soma das distâncias de Hamming entre todos os dados da sequência de dados recebidos no receptor digital, e todos os dados de cada uma de todas as possíveis sequências que podem ser geradas no codificador convolucional do transmissor digital. A menor soma destas distâncias indica a possível sequência de dados que mais provavelmente teria sido gerada pelo codificador convolucional do transmissor. Graças à técnica de decodificação convolucional de Viterbi, é possível descartar a maior parte das sequências possivelmente geradas no codificador convolucional do transmissor e concentrar a pesquisa (comparação) apenas nas sequências de dados com maior probabilidade de serem as que resultariam na sequência de dados efetivamente recebidos no receptor digital. Devido a esta característica, o decodificador convolucional de Viterbi torna a decodificação convolucional por máxima verossimilhança viável de ser executada computacionalmente em sistemas de comunicação digital de alto desempenho.

Existem dois métodos pelo qual podem ser feitos o processo de decodificação pelo algoritmo de Viterbi, a decodificação abrupta e a suave. "A decodificação abrupta destaca-se pela simplicidade e baixa carga computacional enquanto que a decisão suave requer uma carga computacional maior em troca de melhor desempenho em termos de

capacidade de correção de erros", [20]. A seguir é apresentado um exemplo utilizando a decodificação abrupta.

A decodificação abrupta pelo algoritmo de Viterbi consiste em:

1. Iniciando do estado 00, compara-se a distância de Hamming do primeiro par de bits da sequência código (11) com as distâncias de Hamming das duas transições possíveis a partir do estado 00. As distâncias de Hamming obtidas são armazenadas (valores entre parênteses).

2. As distâncias de Hamming do próximo par de bits da sequência código (11) é comparada com as distâncias de Hamming das transições possíveis a partir dos estados alcançados após o passo anterior. As distâncias de Hamming encontradas são somadas àquelas obtidas nas transições anteriores.

3. O mesmo procedimento apresentado no passo 2 é repetido para o próximo par da sequência código (01). 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.

Observação 2.5.1. Os números entre parênteses citados nos itens $1, 2$ e 3 referem-se ao Exemplo 2.5.1.

Exemplo 2.5.1. Considere a sequência de informação original $u = 1011001$ que após ser codificada pelo codificador da Figura 2.1 obteve a sequência $v = 11110100011011$ e esta foi enviada.

Aplicando a decodificação abrupta pelo algoritmo de Viterbi, obtém-se o diagrama de treliça representado na Figura 2.4. Ao aplicar um erro no terceiro par de bits obtém-se a Figura 2.5. O caminho sobrevivente é traçado, nos dois casos, pela cor vermelha.

Na análise do último par de bits recebidos, na Figura 2.5, o caminho sobrevivente aponta uma distância de Hamming acumulada igual a 1, identificando um erro no terceiro par de bits. Este resultado mostra que a sequência decodificada com maior probabilidade de ter sido a sequência transmitida é a sequência $v = 11110100011011$, correspondente a mensagem $u = 1011001$. Assim, a sequência recebida apresenta um erro em relação à sequência decodificada, a distância de Hamming acumulada é 1.

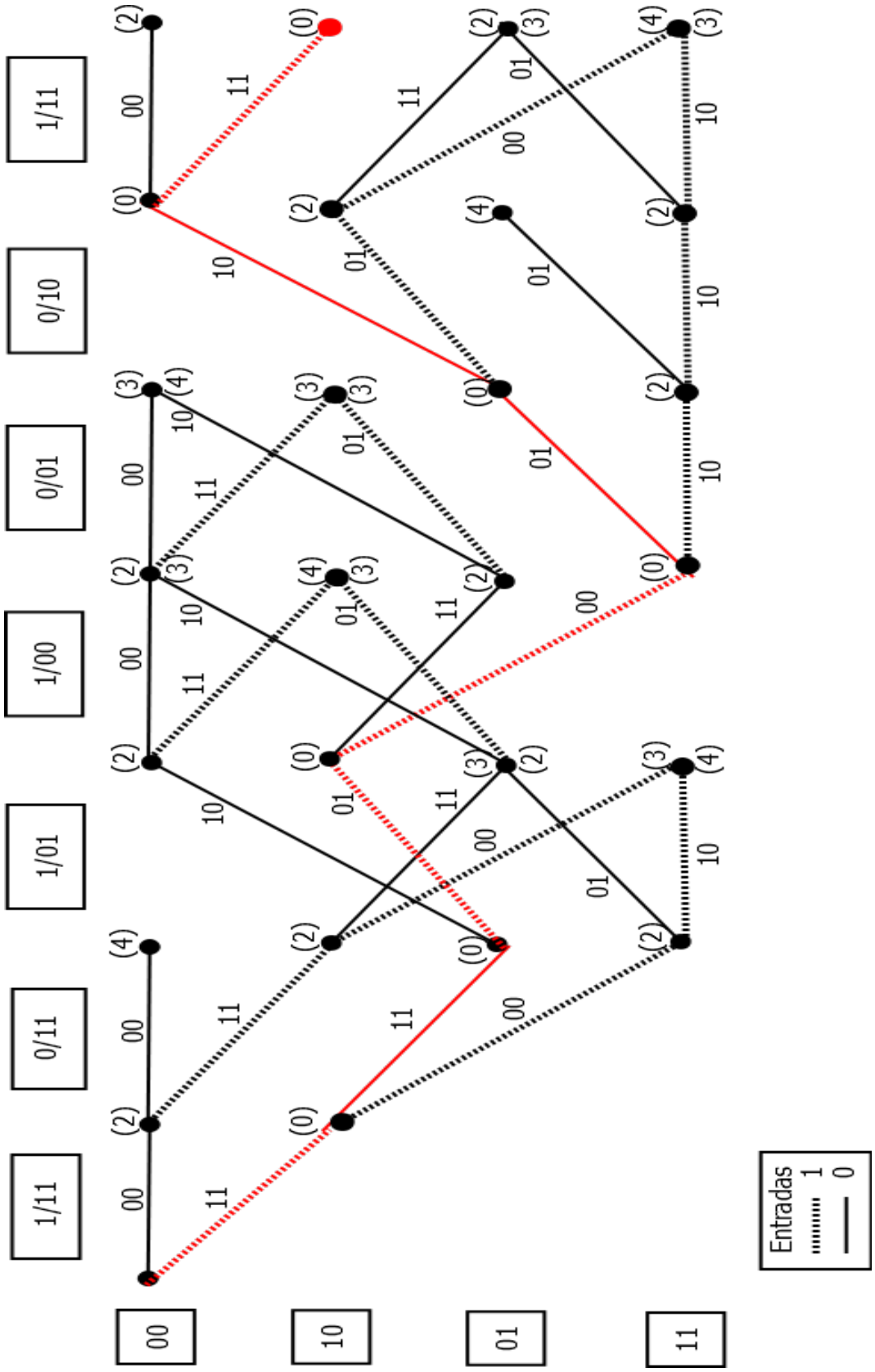

Figura 2.4: Treliça da decodificação abrupta pelo algoritmo de Viterbi.

Fonte: Autora

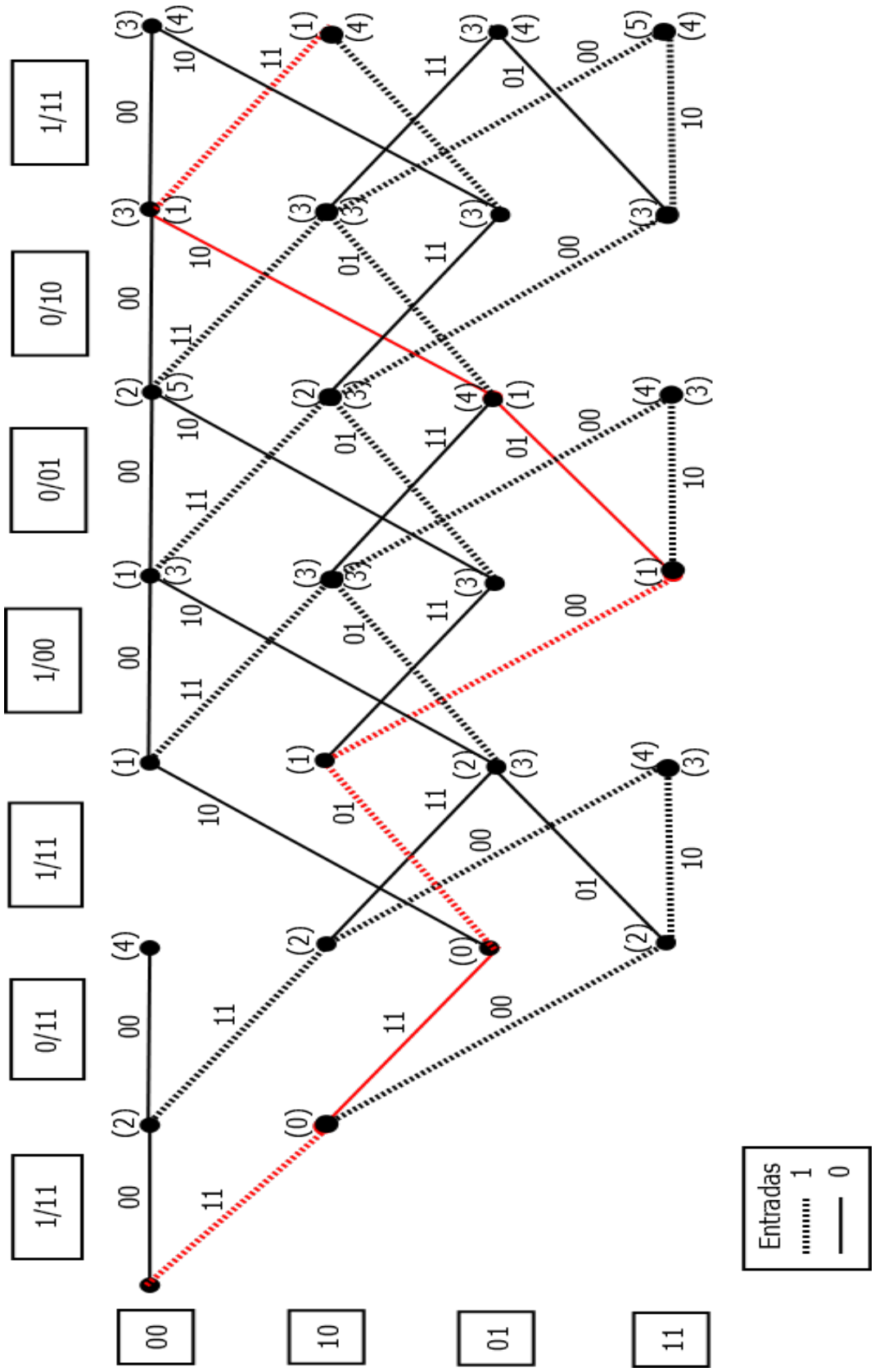

Figura 2.5: Treliça da decodificação abrupta pelo algoritmo de Viterbi com um erro no terceiro par de bits da mensagem codificada.

Fonte: Autora

2.6 Sistemas Criptográficos

Formalmente, um bf sistema criptográfico é uma classe de transformações $\{Sk\}_{k\in K}$ invertíveis

$$S_k : \overline{M} \rightarrow \overline{C},$$

onde $\overline{M}$ é o espaço das mensagens originais e $\overline{C}$ é o espaço das mensagens cifradas, e a chave k é selecionada de um espaço finito K que é denominado *espaço de chaves*.

Toda segurança de um sistema criptográfico está concentrada na chave utilizada para cifrar. Pode-se até supor que o criptoanalista conheça o espaço de chaves que está sendo utilizado, mas a probabilidade do mesmo descobrir a transformação utilizada é mínima.

A classificação dos sistemas criptográficos podem ser do tipo convencional e chave pública. Em ambos os sistemas existem dois problemas a serem resolvidos:

1. Privacidade: Objetiva evitar que a informação seja interceptada no canal por pessoas não autorizadas, os criptoanalistas. Na Figura 2.6 é mostrado um sistema criptográfico com privacidade.

2. Autenticidade: Busca evitar que informação seja alterada pelo criptoanalista. A Figura 2.7 apresenta um sistema criptográfico que ilustra este tipo de problema.

Em um sistema criptográfico envolvendo problema de privacidade, 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,

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

de modo a recuperar a mensagem original M.

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

Em um sistema criptográfico envolvendo problema de autenticidade, o criptoanalista não somente intercepta a mensagem, como também altera seu conteúdo. O receptor autorizado protege-se deste tipo de problema aceitando somente as mensagens cifradas que chegam através do canal, correspondendo à chave correta.

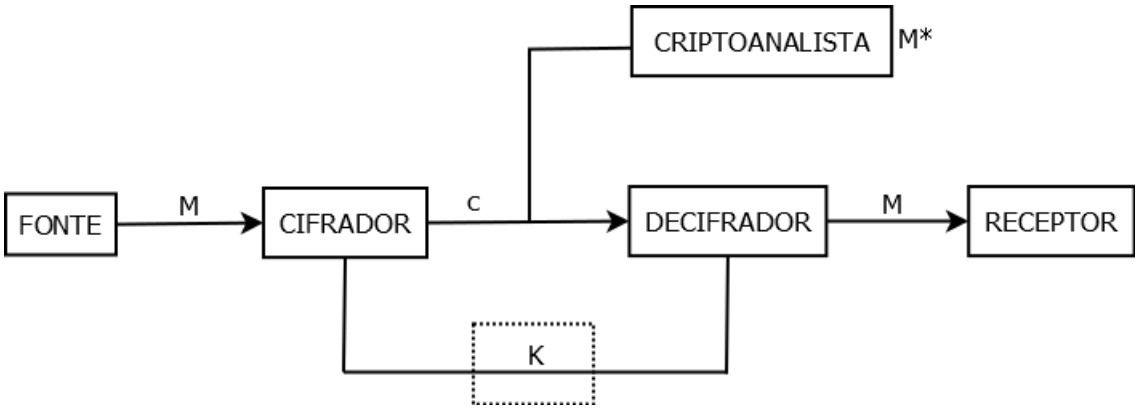

Figura 2.6: Criptossistema com privacidade.

Fonte: [9]

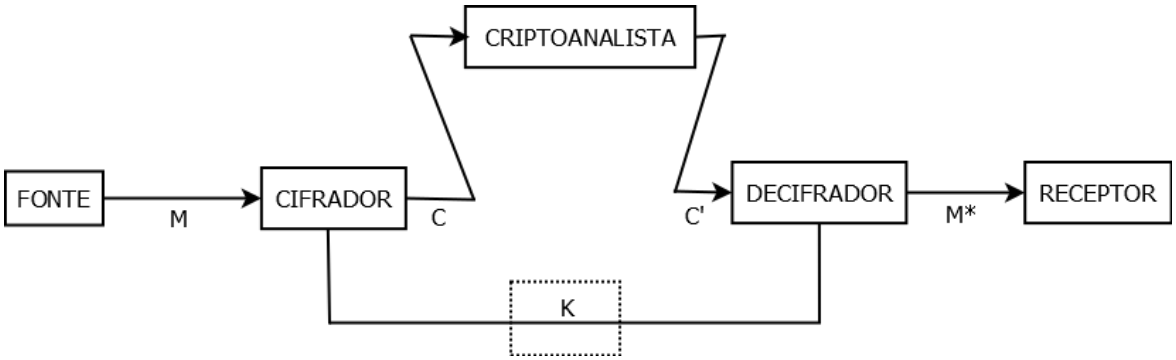

Figura 2.7: Criptossistema com autenticidade.

Fonte: [9]

2.6.1 Sistemas Criptográficos Convencionais

Em um sistema criptográfico convencional a chave deve ser enviada do transmissor para o receptor através de um canal seguro. Aí está uma das desvantagens dos sistemas convencionais, pois é necessário um canal seguro para o envio da chave, de modo a garantir a eficiência do sistema.

Dentre os algoritmos utilizados em criptografia convencional, pode-se citar: os de *substituição* e os de *transposição*.

1. Substituição: É caracterizado por uma transformação de substituição aplicada ao alfabeto. A decifragem é obtida através do mapeamento de cada letra do criptograma aplicada à transformação de substituição inversa.

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

$$M = (m_{00}, m_{01}, \cdots, m_{0(N-1)}) \cdots (m_{00}, m_{01}, \cdots, m_{0(N-1)}) \cdots,$$

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

2.6.2 Sistemas Criptográficos de Chave Pública

Com o objetivo de solucionar o problema dos sistemas convencionais, gerar necessariamente um canal seguro para o envio da chave, em 1976, Diffie e Helman, [6], lançaram um novo conceito em criptografia, a implementação que utiliza a técnica de chaves públicas.

Os criptossistemas de chave pública utilizam uma família de pares de transformações: (E_k, D_k), onde $k \in \{K\}$, E_k e D_k são mapas definidos por:

$$E_k : \{M\} \rightarrow \{C\} = E_k\{M\} \tag{2.10}$$

e

$$D_k : \{C\} \rightarrow \{M\} = D_k\{C\} \tag{2.11}$$

sobre um espaço de mensagens finito $\{M\}$, tal que:

1. Para todo $k \in \{K\}$, D_k é a transformação inversa de E_k, ou seja, para qualquer k e algum M, tem-se: $D_k E_k(M) = M$;

2. Para todo $k \in \{K\}$ e $m \in \{M\}$, as operações E_k e D_k são fáceis de se calcular, do ponto de vista computacional;

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

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

Assim, o sistema criptográfico de chave pública consiste essencialmente de duas partes: uma compreende a transformação para cifrar e outra para decifrar. Além disso, dado que uma das partes é conhecida, determinar a outra parte é um problema não tivial.

A propriedade quatro nos garante a existência de uma solução para computar os correspondentes pares de transformações inversas.

Como exemplo de um criptossistema de chave pública tem-se o *algoritmo RSA*. Proposto por Rivest, Shamir e Adleman (originando a sigla RSA) é o algoritmo de chave pública mais conhecido. É considerado um dos mais seguros, sua segurança baseia-se no fato de que não é fácil fatorar um número o qual é o produto de dois números primos grandes. "Dados dois números primos grandes p e q, multiplicá-los para encontrar n é muito mais fácil do que o contrário: dado n, encontrar os dois números primos p e q", [15]. A complexidade do algoritmo RSA está em gerar números primos grandes, com mais de cem dígitos, por exemplo.

2.6.3 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 P 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 NP. Neste caso, a complexidade é exponencial.

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 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 é denominado 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.

Considere, ainda, 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.

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.

Para a determinação de códigos ótimos de memória unitária é necessário resolver o problema da mochila, por sua vez este problema pertence à classe dos problemas não polinomiais completos (NP completo), ou seja, sua solução não é trivial, o que dificulta a determinação de bons códigos convolucionais, mas sua utilização é desejável em sistemas criptográficos convencionais e de chaves públicas.

A segurança do Criptossistema de Chave Pública (CCP) baseado em códigos convolucionais leva em consideração os seguintes fatos:

1. O embaralhamento das colunas das submatrizes geradoras do código convolucional faz com que os pesos de Hamming das palavras-código ramo diminua, causando a redução no poder de correção dos erros;

2. O problema da mochila deve ser resolvido.

Para obter resultados esperados no desempenho de um sistema criptográfico utilizando códigos convolucionais, é necessário que uma transformação armadilha seja utilizada.

2.7 Códigos Ótimos de Memória Unitária

Código ótimo é entendido como um código cuja função enumeradora dos pesos tem o menor número de palavras-código com valor igual ao do d_{free}.

Observação 2.7.1. Se pelo menos dois códigos apresentam o mesmo número de palavras-código com o valor d_{free}, então aquele que apresentar o menor número de palavras-código com distância $d_{free} + 1$ será considerado ótimo. Se persistir o fato que ambos os códigos tenham o mesmo número de palavra-código com distância $d_{free} + 1$, o método adotado será o de continuar o processo até que um destes apresente um menor número de palavras-código com distância $d_{free} + j$, sendo $j \geq 1$ um inteiro positivo.

Para o entendimento de códigos ótimos de memória unitária é preciso definir os conceitos de fluxo máximo e conservação de fluxo, pois "quando caracterizados como um problema de otimização combinatorial, os códigos ótimos de memória unitária satisfazem as propriedades destes conceitos", [17, 28].

Definição 2.7.1. (Fluxo Máximo - ϕ). O fluxo máximo 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:

$$\phi = n \cdot (q-1) \cdot q^{(k-1)}. \tag{2.12}$$

Definição 2.7.2. (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.

É possível estabelecer uma equivalência entre a determinação de bons códigos de memória unitária e o problema da mochila, observando os conceitos de códigos de memória unitária, fluxo máximo e conservação de fluxo.

Seja d um conjunto finito ordenado de distâncias de Hamming, $d_i = w_{0i} + w_{i0}$, onde w_{0i} e w_{i0} são os pesos de Hamming das palavras-código ramo que saem do estado 0 para o estado i e do estado i para o estado 0, respectivamente. Desse modo,

$$d = d_1, d_2, \cdots, d_{q^k-1},$$

onde

$$d_i = w_{0i} + w_{i0},$$

com $1 \leq i \leq q^k - 1$.

Determinar cada um desses valores, w_{0i} e w_{i0}, é o problema. Ao encontrá-los, é possível determinar as matrizes G_0 e G_1, respectivamente. Assim, o único parâmetro a saber é a distância livre do código convolucional, dado que, em geral, $d_i = d_{free} + i - 1$, para $1 \leq i \leq q^k - 1$. Dessa forma, os limitantes superiores da distância livre do código convolucional e de memória unitária serão muito importantes para a determinação de d. "A equação destes limitantes superiores", é, [17]:

$$d_{min} \leq n(m+1)(q-1)\left(\frac{q^{k-1}}{q^k-1}\right). \tag{2.13}$$

O menor valor do limitante da equação (2.13) é a *distância mínima irrestrita*, ou seja, o d_{free}. Considerando o conjunto d lexicográfico, tem-se que $d_1 = d_{free}$ e, em geral, $d_i = d_{free} + i - 1$, para todo i tal que $1 \leq i \leq q^k - 1$, com o d_{free} sendo obtido diretamente da equação (2.13).

A partir do conhecimento de d_1 e das informações de ϕ e q, o problema da mochila pode ser estabelecido por

$$\sum_{i=1}^{q^k-1} a_i d_i = (m+1)\phi, \tag{2.14}$$

onde $m = 1$, para os códigos de memória unitária, e a_i representa o número de vezes que o valor d_i aparece.

Na equação (2.14) utilizando uma representação vetorial para seus elementos, obtém-se uma caracterização matemática do problema da Mochila, dado por

$$a * d = (m+1)\phi. \tag{2.15}$$

Assim, para a determinação de bons códigos de memória unitária deve-se somente resolver o Problema da Mochila.

A cada vetor a, solução do problema da mochila apresentado, está associado a matriz geradora correspondente, como resultado da aplicação da técnica de "representação modular de códigos de bloco", [28]. A partir dos pesos de Hamming, d_i, é estabelecido uma forma de "determinação da matriz geradora através do conhecimento da distribuição dos pesos de Hamming das palavras-código", [19]. Portanto, resolver a equação (2.15) para códigos de memória unitária onde k assume valores grandes, equivale a determinar dentre os q^k possíveis subconjuntos de soluções as quais fornecerão os códigos desejados.

2.8 Transformações Armadilha

Transformações armadilha são funções que quando aplicadas às submatrizes geradoras originais do código, G_0 e G_1, reduzem o poder de correção deste código a um valor desejável ou mesmo fixado pela aplicação. Estas funções aplicadas às submatrizes geradoras, são feitas através de um par de matrizes invertíveis A e B, com dimensões $k \times k$ e $n \times n$, respectivamente. A matriz B tem por objetivo reduzir a distância livre, enquanto que a matriz A realiza o embaralhamento dos bits nas palavras-código ramo.

Uma vez aplicadas, estas transformações geram novas submatrizes geradoras, denominadas G'_0 e G'_1, dadas por

$$G'_0 = AG_0B \tag{2.16}$$

e

$$G'_1 = AG_1B. \tag{2.17}$$

As *propriedades das transformações armadilha* a serem apresentadas são estabelecidas com relação às condições de otimalidade dos códigos, [18, 21].

Sejam $G_i, 0 \leq i \leq m$, as submatrizes geradoras com dimensão $k \times n$ de um código convolucional, o mesmo pode ser um código de memória unitária, sobre F_2. Seja $\phi(G_i)$ o peso de Hamming total de cada G_i, dado por

$$\phi(G_i) = n2^{k-1}, \quad \text{com} \quad 0 \leq i \leq m.$$

Definição 2.8.1. Uma matriz G com dimensão $k \times n$, sobre F_2, é dita ser igualmente distribuída com relação aos pesos de Hamming se, e somente se, os $2^k - 1$ elementos não nulos gerados por G têm pesos de Hamming

$$\tilde{w} = \frac{\phi(G)}{2^k - 1}. \tag{2.18}$$

Se $2^k - 1$ divide $\phi(G)$, então $[\tilde{w}] = \tilde{w}$, isto é, $\tilde{w}$ é um número inteiro. Caso contrário, $[\tilde{w}] = \{|\tilde{w}| \pm j, \quad \text{onde} \quad j \geq 0\}$. Como os códigos de memória unitária são equivalentes aos códigos convolucionais, serão considerados apenas os de memória unitária onde G tem dimensão $2k \times n$. Segue, então, o Lema 2.8.1.

Lema 2.8.1. *Se o posto* $(G) = 2k$ *e* $2k \leq n$*, então tem* $2k$ *vetores linha linearmente independentes com peso de Hamming igualmente distribuídos e são capazes de gerar todas as* 2^{2k} *palavras-código também com pesos de Hamming igualmente distribuídos.*

Demonstração. Um código de memória unitária com taxa $R = k/n$ possui representação em treliça com 2^k estados e 2^k transições de cada estado. Desta forma, o número total de transições em uma janela de tempo é 2^{2k}. Seja $[\tilde{w}]$ a parte inteira de $\tilde{w}$. Então, existem

$$\frac{n!}{(n - [\tilde{w}])!\ [\tilde{w}]!}\ [\tilde{w}]! \tag{2.19}$$

vetores com peso de Hamming igual a $[\tilde{w}]$. Pode-se mostrar facilmente que existe um n_0 tal que,

$$n \geq n_0 \Rightarrow \ n!\ (n - [\tilde{w}])!\ [\tilde{w}]\ \geq 2k, \tag{2.20}$$

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

Como cada transição na treliça tem associada uma palavra-código ramo diferente, é possível encontrar um código de bloco $(k, 2n)$ tal que a matriz geradora seja constituída por vetores pesos de Hamming igualmente distribuídos, onde a distância mínima

é igual àquela do código de bloco $(k, 2n)$. Portanto, obtêm-se os $2k$ vetores linha (linearmente independentes) de G, de tal forma que G gera um espaço vetorial com 2^{2k} vetores com peso de Hamming igualmente distribuídos. □

Uma consequência imediata deste lema é que o código não é do tipo catastrófico. Entretanto, se $2k > n$, a base do espaço vetorial gerado por G possui dimensão $dim(G) \leq n$, e assim, pelo menos uma das linhas de G é combinação linear das demais, o que implica que nem todas as transições na treliça serão associadas a palavras-código diferentes.

Será estabelecida, agora, a condição de existência da transformação armadilha composta pelo par de transformações (A, B), sob a condição do Lema 2.8.1 supondo que o código utilizado seja ótimo.

Existem necessariamente duas condições. A primeira delas está relacionada com a aplicação da transformação armadilha a G_1, resultando em um novo código com mesma taxa e número de memórias. Em outras palavras, obtendo-se um código ótimo e supondo que as transformações armadilha A e B são aplicadas às submatrizes geradoras do código com taxa $R = k/n$ e $m + 1$ células, novos códigos com a mesma taxa serão obtidos. Decorrente disso, segue a proposição:

Proposição 2.8.1. *Considere G_i, com $0 \leq i \leq m$, submatrizes geradoras de um código convolucional ótimo não catastrófico sobre F_2. Sejam A e B matrizes invertíveis com dimensões $k \times k$ e $n \times n$, respectivamente. Então,*

$$d_{min}\{A[G_0, G_1, \cdots, G_m]B\} \leq d_{min}\{[G_0, G_1, \cdots, G_m]\},$$

onde A e B não são necessariamente matrizes unitárias.

Demonstração. Partindo da hipótese que $G = [G_0, G_1, \cdots, G_m]$ é a matriz geradora de um código ótimo, seja $d_{min}\{G\} = d_{min}$ e $G' = AGB$, onde a distância mínima de G' é igual a d'_{min}. Então

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

   ou

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

Seja S o conjunto de todas as possíveis transformações (A, B) sobre F_2. S pode ser dividido em três conjuntos:

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

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

Sabe-se ainda que G' é equivalente a G se

$$G' = AGB,$$

onde A e B são matrizes invertíveis com dimensões $k \times k$ e $n \times n$, respectivamente, cujos determinantes são iguais a 1. Então, (A, B) pertencem a S_2, e assim, a igualdade é válida em (2). É óbvio que se A e B pertencem a S_3, então a transformação resultante conduzirá a um código com capacidade corretora de erros menor. □

Há uma outra condição que é possível quando o código foi determinado de acordo com as condições do Lema 2.8.1 e que as transformações armadilha A e B são aplicadas às submatrizes geradoras do código com taxa $R = k/n$ e memória m. Estas aplicações resultam em um código com o mesmo número de células, porém, com taxa $\tilde{R} = k/q$.

Proposição 2.8.2. *Sejam G_i submatrizes geradoras, com dimensão $k \times n$, de um código convolucional ótimo não catastrófico sobre F_2, onde $0 \leq i \leq m$. Seja $d_{min}(k/n)$ a distância mínima deste código, isto é,*

$$d_{min}[G_0, G_1, \cdots, G_m] = d_{min}(k/n).$$

Então, existem matrizes A e B com dimensões $k \times k$ e $n \times q$, respectivamente, $(q > n)$ sobre F_2 transformando G_i em novas submatrizes $\tilde{G}_i$ com dimensões $k \times q$ tal que

$$d_{min}\{[\tilde{G}_0, \tilde{G}_1, \cdots, \tilde{G}_m]\} = d_{min}(k/q),$$

e

$$d_{min}(k/n) \leq d_{min}(k/q).$$

Demonstração. Seja $d_{min}(k,n)$ a distância mínima de um código de bloco (n,k), onde n denota o comprimento das palavras-código e k denota o comprimento dos bits de informação. É claro que $d_{min}(k/n) = d_{min}(k,n)$. A partir do Lema 2.8.1, tem-se que

$$d_{min}(k,n) \leq d_{min}(k,2n)$$

e que se $q > n$, onde

$$d_{min}(k,2q) \geq d_{min}(k,2n).$$

Assim, $d_{free}(k/n)$. Note que a igualdade é válida se G e G' são equivalentes onde $det(A) = 1$ e posto $(B^+) = n$, onde B^+ é uma submatriz geradora.

De forma análoga à Proposição 2.8.2, pode-se aplicar as transformações armadilha A e B às submatrizes geradoras do código com taxa $R = k/n$ e memória m para obter como resultado um código com o mesmo número de células e com taxa $\tilde{R} = p/q$. Assim segue que

Proposição 2.8.3. *Sejam G_i, com $0 \leq i \leq m$ e dimensão $k \times n$, submatrizes geradoras de um código convolucional ótimo não catastrófico sobre F_2. Seja $d_{min}(k/n)$ a distância mínima deste código, ou seja,*

$$d_{min}[G_0, G_1, \cdots, G_m] = d_{min}(k/n).$$

Então, existem matrizes A e B com dimensões $p \times k$ e $n \times q$, respectivamente, sendo $p \geq k$ e $q > n+1$ sobre F_2 transformando G_i em novas submatrizes $\tilde{G}_i$ com dimensões $p \times q$, tais que

$$d_{min}\{[\tilde{G}_0, \tilde{G}_1, \cdots, \tilde{G}_m]\} = d_{min}(p/q),$$

e

$$d_{min}(k/n) \leq d_{min}(p/q).$$

Demonstração. Segue da aplicação do Lema 2.8.1, da demonstração da Proposição 2.8.2 e do limitante superior da distância mínima dado por

$$d_{min}(k/n) \leq min\left\{\frac{2^{u-1}}{2^u - 1} \quad \frac{n}{k} \quad (u + km)\right\}.$$

Note que G' é equivalente a G se o posto $(A^-) = k$, posto $(A^+) = p$ e posto $(B^+) = n$, onde A^- e A^+ são submatrizes geradoras.

□

O código convolucional ótimo citado nas proposições anteriores pode ser de memória unitária ou não. Através destas três proposições é possível estabelecer as condições de existência das transformações armadilha.

3 TRANSFORMAÇÃO ARMADILHA PROPOSTA

"A capacidade de detecção e/ou correção de erros, bem como a otimalidade de um código está diretamente relacionada à distância livre (d_{free}) deste código", [28]. Existem muitas transformações que, quando aplicadas às submatrizes geradoras (G_0 e G_1) de um código, são capazes de reduzir a distância livre a níveis desejados, consequentemente, melhora na capacidade de correção de erro. Contudo, é possível observar que estas transformações não possuem uma lei de formação que as caracterizem.

Neste trabalho, é utilizada como proposta de transformação armadilha a ***Matriz Triangular Inferior***. Sua veridicidade está fundamentada nas propriedades das transformações armadilha apresentadas na seção 2.8.

A matriz triangular inferior possui uma estrutura bem definida, neste caso, estrutura de grupo, por este motivo antes de definir a matriz triangular inferior é preciso conceituar alguns fundamentos importantes da teoria de grupos.

Definição 3.0.1. (Operação Binária). Uma operação binária $*$ sobre um conjunto S é uma regra que associa algum elemento de S a cada par ordenado (a, b) de elementos de S ($a * b$ denotará o elemento associado a (a, b) através de $*$).

Uma operação binária sobre S deve associar a cada par ordenado (a, b) um elemento que está também em S. Essa associação é denominada fechamento, ou seja, S deve ser fechado sob a operação binária.

Definição 3.0.2. (Grupo). Seja G um conjunto munido de uma operação binária. Diz-se que $\langle G, * \rangle$ é grupo com respeito à operação $*$ se for verificado as propriedades:

1. A operação $*$ é associativa, ou seja, $g * (h * k) = (g * h) * k$ sempre que $g, h, k \in G$;
2. Existe $e \in G$ tal que $e * g = g * e = g$ sempre que $g \in G$: e é a identidade ou elemento neutro de G;
3. Se $g \in G$, então existe $g^{-1} \in G$ tal que $g * g^{-1} = g^{-1} * g = e$.

Quando o grupo G verifica a propriedade $g * h = h * g$ sempre que $g, h \in G$, diz-se que G é grupo comutativo ou abeliano.

Por exemplo, $(\mathbb{Z}, +)$, sendo que $\mathbb{Z}$ é o conjunto dos números inteiros relativos com a adição usual, é um grupo abeliano.

Definição 3.0.3. (Matriz Triangular Inferior). Uma matriz M é dita ser triangular inferior se todos os elementos acima da diagonal principal forem todos nulos, ou seja, se $a_{ij} = 0$ para $i < j$.

Uma matriz triangular inferior caracteriza-se por:

$$M = \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ a_{41} & a_{42} & a_{43} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}_{n \times n}.$$

Proposição 3.0.1. *As matrizes triangulares inferiores possuem algumas relevantes propriedades, a saber:*

1. *A matriz identidade é uma matriz triangular inferior;*
2. *O produto de matrizes triangulares inferiores origina uma matriz triangular também inferior;*
3. *O determinante de uma matriz triangular inferior é o produto dos elementos da sua diagonal principal;*
4. *Sendo uma matriz triangular inferior invertível, sua inversa é também uma matriz triangular inferior.*

De acordo com a definição de matriz inversa, uma matriz é invertível se e somente se o seu determinante for diferente de zero. Uma vez que o determinante de uma matriz triangular inferior é dado pelo produto dos elementos da diagonal principal, se um desses elementos for nulo, a matriz não é invertível, pois a multiplicação por esse

elemento resultaria em um produto também nulo, consequentemente, um determinante igual a zero.

As matrizes triangulares inferiores utilizadas neste trabalho têm como entradas elementos, linearmente independentes, do grupo de Galois $\mathbb{Z}_2$. Além disso, foi utilizado matrizes cuja diagonal é composta apenas por uns (1s). Assim, por suas propriedades, o determinante é igual a 1, e consequentemente, a invertibilidade está garantida. Foi utilizado para as operações de distâncias livres, máxima capacidade de correção de erros e matrizes geradoras, o software MATLAB.

3.1 Análise da Transformação Armadilha Proposta

Nesta seção, é desenvolvida a análise das transformações armadilha utilizando códigos convolucionais ótimos de memória unitária. Em cada caso, é especificado os códigos utilizados apresentando as submatrizes geradoras ($G_0, G_1, G_0^{'}$ e $G_1^{'}$), suas taxas e as distâncias livres (d_{free} e $d_{free}^{'}$).

Após ser aplicado as transformações armadilha a cada código, é apresentado e comentado os resultados obtidos.

Nas tabelas apresentadas nesta seção, todas as matrizes, as transformações armadilha e as submatrizes geradoras estão representadas na forma octal, onde cada linha da matriz é separada por dois pontos (:). Para ilustrar o que foi dito, considere a seguinte matriz dada em octal:

$$M = [12 : 05 : 03].$$

Sua representação binária é dada por:

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

Para análise da transformação armadilha proposta, o primeiro procedimento foi aplicar as transformações armadilha em cada submatriz separadamente, isto é,

$$G_0^{'} = AG_0B \quad \text{e} \quad G_1^{'} = AG_1B.$$

Os códigos convolucionais ótimos de memória unitária considerados possuem,

respectivamente, taxa $R = 2/4$, com submatrizes geradoras,

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

e $d_{free} = 5$; taxa $R = 2/8$ e submatrizes,

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

$d_{free} = 10$.

As transformações armadilha A utilizadas aos dois códigos são dadas pelas seguintes matrizes,

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ e } A_4 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

O Teorema 3.1.1 facilita a compreensão dos resultados obtidos na seção 3.2.

Teorema 3.1.1. *[23] Seja C um código convolucional (n, k, m), isto é, com k entradas e m memórias, gerado pela matriz*

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

Sejam Φ uma transformação linear e $\Phi(G)$ a matriz geradora dada por

$$\Phi(G) = \begin{pmatrix} \Phi(G_0) & \Phi(G_1) & \Phi(G_2) & \cdots & \Phi(G_m) & \\ & \Phi(G_0) & \Phi(G_1) & \cdots & \Phi(G_m - 1) & \Phi(G_m) \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

obtida através da substituição de cada uma das submatrizes, pela respectiva imagem, segundo a transformação linear Φ. O código convolucional C', gerado pela matriz $\Phi(G)$, é equivalente ao código C.

3.2 Resultados e Discussões

Para serem aplicadas ao código com taxa $R = 2/4$, as matrizes triangulares inferiores aqui consideradas como sendo transformação armadilha B são:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \text{ e } T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

Essas matrizes possuem determinantes iguais a 1 e posto completo, ou seja, posto $(T_i) = 4$, com $i = 1, 2, 3, 4$.

Código	G_0	G_1	A	B	$G_0^{'}$	$G_1^{'}$	d_{free}	$d_{free}^{'}$
(4, 2, 1)	15:03	14:07	A_1	T_1	17:01	14:05	5	3
(4, 2, 1)	15:03	14:07	A_1	T_2	13:01	04:15	5	4
(4, 2, 1)	15:03	14:07	A_1	T_3	01:03	14:07	5	3
(4, 2, 1)	15:03	14:07	A_1	T_4	11:07	04:13	5	3
(4, 2, 1)	15:03	14:07	A_2	T_1	01:17	05:14	5	3
(4, 2, 1)	15:03	14:07	A_2	T_2	01:13	15:04	5	4
(4, 2, 1)	15:03	14:07	A_2	T_3	03:01	07:14	5	3
(4, 2, 1)	15:03	14:07	A_2	T_4	07:11	13:04	5	3
(4, 2, 1)	15:03	14:07	A_3	T_1	16:01	11:05	5	3
(4, 2, 1)	15:03	14:07	A_3	T_2	12:01	11:15	5	4
(4, 2, 1)	15:03	14:07	A_3	T_3	02:03	13:07	5	4
(4, 2, 1)	15:03	14:07	A_3	T_4	16:07	17:13	5	6
(4, 2, 1)	15:03	14:07	A_4	T_1	17:16	14:11	5	5
(4, 2, 1)	15:03	14:07	A_4	T_2	13:12	04:11	5	4
(4, 2, 1)	15:03	14:07	A_4	T_3	01:02	14:13	5	3
(4, 2, 1)	15:03	14:07	A_4	T_4	11:16	04:17	5	3

Tabela 3.1: Transformação triangular inferior, $R = 2/4$

Fonte: Autora

Na Tabela 3.1 são apresentados os resultados na distância livre do código

$(4, 2, 1)$ obtidos pela aplicação das transformações armadilha nas submatrizes geradoras.

O d_{free} do código foi reduzido de 5 para 3 em 9 casos e em 5 casos para 4. Em apenas 1 caso foi maior que o d_{free} inicial e em outro permaneceu o mesmo.

Em seu caso de melhor excelência o código perdeu a capacidade de correção de erro, passando a corrigir 1 erro ao invés de 2, isto é, o d_{free} que era igual a 5 passou a ser igual a 3.

Para o código com taxa $R = 2/8$, as matrizes triangulares inferiores utilizadas como transformação armadilha B são:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix},$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \text{ e } T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Essas matrizes possuem determinantes iguais a 1 e posto completo, ou seja, posto $(T_i) = 8$, com $i = 1, 2, 3, 4$.

A Tabela 3.2 apresenta os resultados encontrados na aplicação das transformações armadilha nas submatrizes geradoras.

Códigos com $d'_{free} = 5$ foram obtidos, com isso a capacidade de corrigir erros passou de 4 ($d_{free} = 10$) para 2.

A maioria dos códigos encontrados possuem distância livre igual a 7, ainda assim, é um excelente resultado, com capacidade de corrigir 3 erros, apresentando poder de correção menor se comparado ao código original.

Código	G_0	G_1	A	B	$G_0^{'}$	$G_1^{'}$	d_{free}	$d_{free}^{'}$
(8, 2, 1)	370:077	174:237	A_1	T_1	310:217	164:257	10	7
(8, 2, 1)	370:077	174:237	A_1	T_2	130:023	014:063	10	5
(8, 2, 1)	370:077	174:237	A_1	T_3	130:301	034:241	10	6
(8, 2, 1)	370:077	174:237	A_1	T_4	270:105	254:245	10	7
(8, 2, 1)	370:077	174:237	A_2	T_1	217:310	257:164	10	7
(8, 2, 1)	370:077	174:237	A_2	T_2	023:130	063:014	10	5
(8, 2, 1)	370:077	174:237	A_2	T_3	301:130	241:034	10	6
(8, 2, 1)	370:077	174:237	A_2	T_4	105:270	245:254	10	7
(8, 2, 1)	370:077	174:237	A_3	T_1	107:217	333:257	10	10
(8, 2, 1)	370:077	174:237	A_3	T_2	113:023	077:063	10	7
(8, 2, 1)	370:077	174:237	A_3	T_3	231:301	275:241	10	6
(8, 2, 1)	370:077	174:237	A_3	T_4	375:105	011:245	10	7
(8, 2, 1)	370:077	174:237	A_4	T_1	310:107	164:333	10	7
(8, 2, 1)	370:077	174:237	A_4	T_2	130:113	014:077	10	5
(8, 2, 1)	370:077	174:237	A_4	T_3	130:231	034:275	10	6
(8, 2, 1)	370:077	174:237	A_4	T_4	270:375	254:011	10	8

Tabela 3.2: Transformação triangular inferior, $R = 2/8$

Fonte: Autora

3.3 Conclusão

Com os resultados apresentados nas Tabelas 3.1 e 3.2 é possível verificar que a transformação armadilha, matriz triangular inferior, proposta é capaz de diminuir, a valores considerados, a capacidade de correção dos códigos de memória unitária através da alteração da distância livre.

A utilização da transformação armadilha proposta neste trabalho é muito útil quando aplicada no processo de decodificação para recuperação de mensagens, em sistemas

criptográficos, pois a redução da distância livre de um código contribui em seu poder de correção de erros.

No capítulo a seguir é apresentado uma aplicação ao criptossistema de chave pública utilizando a transformação armadilha proposta com o objetivo de recuperar a mensagem original, após a atribuição de um erro na mesma.

4 APLICAÇÃO DA TRANSFORMAÇÃO ARMADILHA PROPOSTA AO CCP

A transformação armadilha proposta, neste trabalho, quando aplicadas às submatrizes geradoras de um determinado código tem por finalidade reduzir o poder de correção deste código a um valor desejado ou mesmo fixado pela aplicação, através da alteração da distância livre. Além deste procedimento, pode-se ainda adicionar um erro à mensagem cifrada, denominado v_e. Este vetor erro 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 utilizado na aplicação deste trabalho, proposto em [9, 21], é definido por:

$$\tilde{v}_t = \tilde{u}_t \, G_0 \oplus \tilde{u}_{t-1} \, G_1, \quad \text{com} \quad \tilde{u}_t = 0 \quad \text{para} \quad t < 0, \tag{4.1}$$

onde $\tilde{v}_t$ é a sequência codificada na entrada do canal e $\tilde{u}_t$ a sequência de informação. Substituindo as equações (2.16) e (2.17) em (4.1), obtém-se:

$$\tilde{v}_t = \tilde{u}_t \, G_0^{'} \oplus \tilde{u}_{t-1} \, G_1^{'}, \quad \text{com} \quad \tilde{u}_{t-1} = 0 \quad \text{para} \quad t \leq 0. \tag{4.2}$$

A equação (4.2) contém o processo de codificação e cifragem conjuntamente.

Adiciona-se um vetor erro ao vetor v_t. Este vetor erro corresponderá à quantidade de erro permissível ao código original, em que

$$v_{te} = v_t + v_i, \tag{4.3}$$

onde

v_{te} é o vetor correspondente à mensagem codificada/cifrada com a inserção do erro;

v_t é a mensagem cifrada/codificada;

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

O vetor v_i origina um vetor erro v_e na decodificação, cujo peso de Hamming é menor ou igual à capacidade de correção do código. O vetor erro na decodificação é obtido através da multiplicação do vetor erro pela transformação inversa de B.

Em um criptossistema de chave pública, as matrizes

$$G' = [G'_0 : G'_1] \tag{4.4}$$

são colocadas em uma lista pública. O número de erros adicionado à mensagem também pode ser divulgado. Na Figura 4.1 é representado o sistema de chave pública em diagrama de blocos.

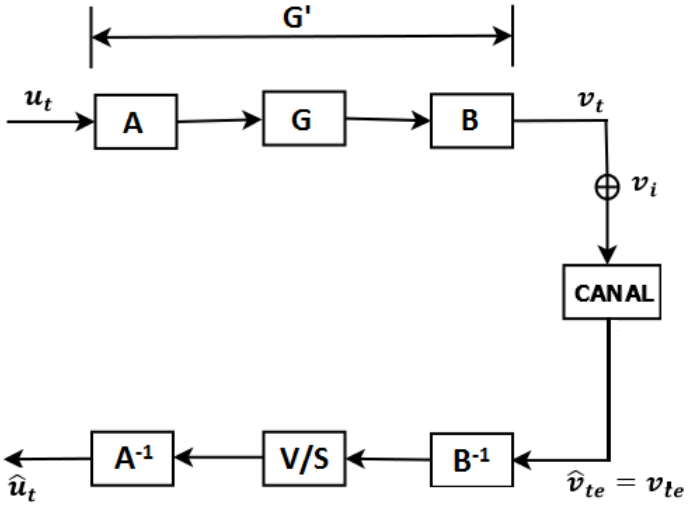

Figura 4.1: Diagrama em blocos do criptossistema de chave pública.

Fonte: [9, 21]

O processo de decodificação é aplicado ao vetor v_{te}, uma vez que a soma do vetor erro de transmissão ao vetor que contém a informação v_t é que segue através do canal.

Considerando-se que o canal seja livre de ruídos, no processo de decodificação empregado à sequência de saída do canal, $\hat{v}_{te}$ (valor estimado de v_{te}) é igual ao valor de v_{te} na entrada do canal. Utilizando-se a transformação inversa de B, B^{-1}, tem-se que

$$v_{te}\ B^{-1} = (u_t\ A)\ G_0 \oplus (u_{t-1}\ A)\ G_1. \tag{4.5}$$

Para obter u_t aplica-se o algoritmo de Viterbi. Este criptossistema utiliza a matriz B como transformação aplicada às submatrizes geradoras originais do código, cujo objetivo é reduzir a distância livre, enquanto que a matriz A realiza o embaralhamento dos bits nas palavras-código ramo.

A seguir é utilizado um procedimento de cifragem no sistema criptográfico, definido anteriormente, aplicando a transformação armadilha proposta. Na Figura 4.2 é apresentado o fluxograma da aplicação.

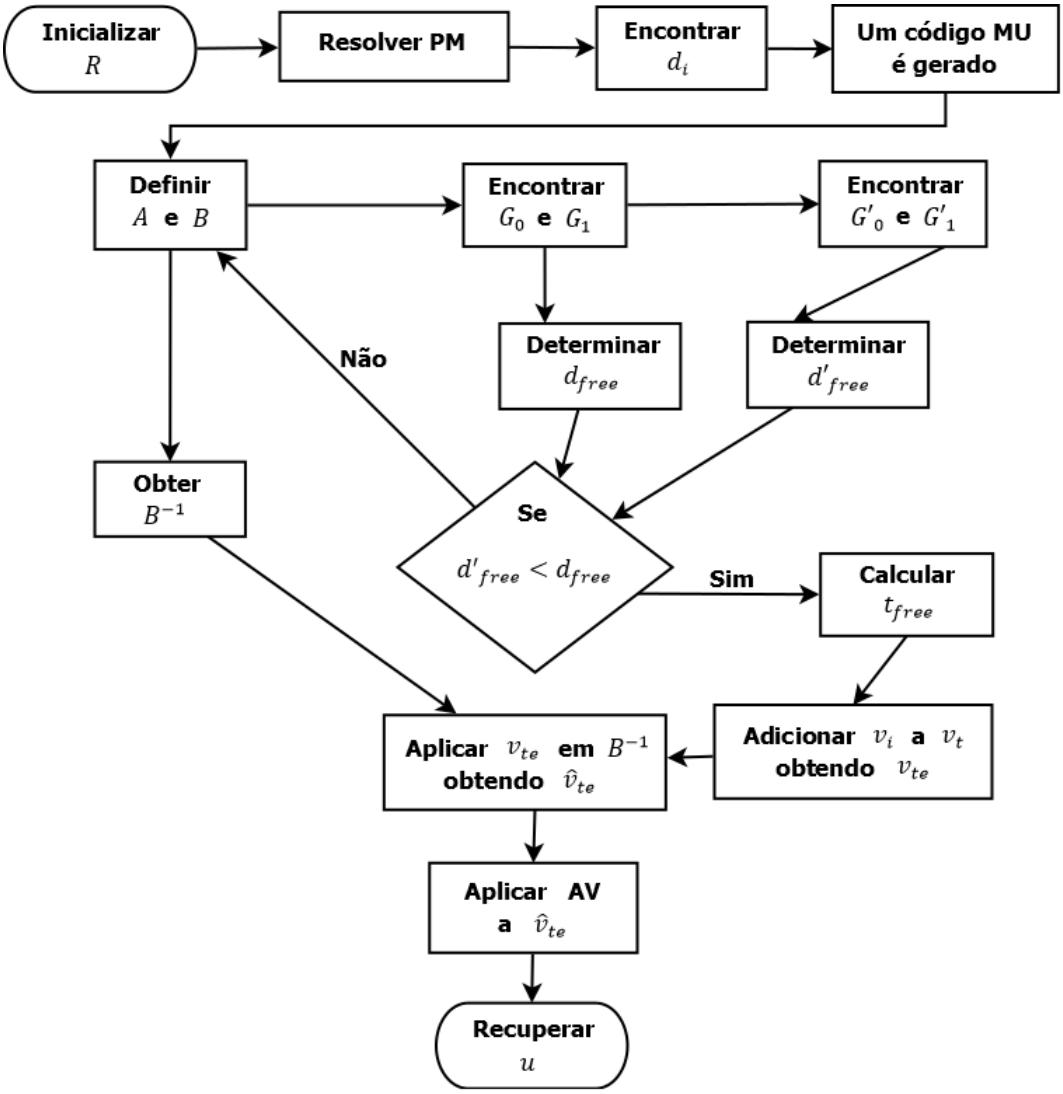

Figura 4.2: Fluxograma da aplicação.

Fonte: Autora

Considere um código ótimo de memória unitária com taxa $R = 2/4$, cujo codificador e diagrama de estados estão representados, respectivamente, nas Figuras 4.3 e 4.4. Este código pode ser encontrado pela resolução do problema da mochila. Da equação (2.15), têm-se que $d_1 = 5, d_2 = 6$ e $d_3 = 7$. Esta solução gera um código de memória unitária cujas submatrizes geradoras, G_0 e G_1, são dadas por:

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

onde a distância livre deste código é dada por $d_{free} = 5$.

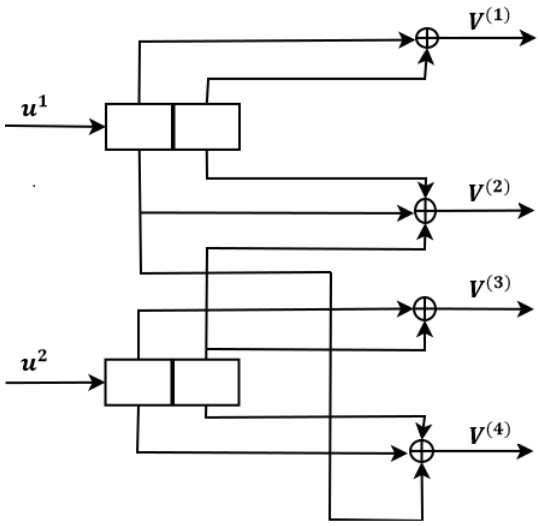

Figura 4.3: Codificador para o código convolucional (4, 2, 1).

Fonte: [21]

Sejam as transformações A e B dadas por:

$$A = A_4 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{e} \quad B = T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix},$$

e suas matrizes inversas, respectivamente, dadas por:

$$A^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{e} \quad B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Aplicando A e B em G_0 e G_1 como em (2.16) e (2.17), obtêm-se $G_0^{'}$ e $G_1^{'}$:

$$G_0^{'} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{e} \quad G_1^{'} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Como visto na seção 4.1, a distância livre deste novo código é igual a $d_{free}^{'} = 3$. Com isso, reduziu-se o poder de correção para apenas 1 erro, quando o código original era capaz de corrigir até 2 erros.

Considere a mensagem a ser transmitida $u = (10, 01, 11, 00)$, com o codificador da Figura 4.3 inicialmente resetado. Assim, a palavra a ser transmitida será

$$v_t = (1101, 1111, 1001, 1011).$$

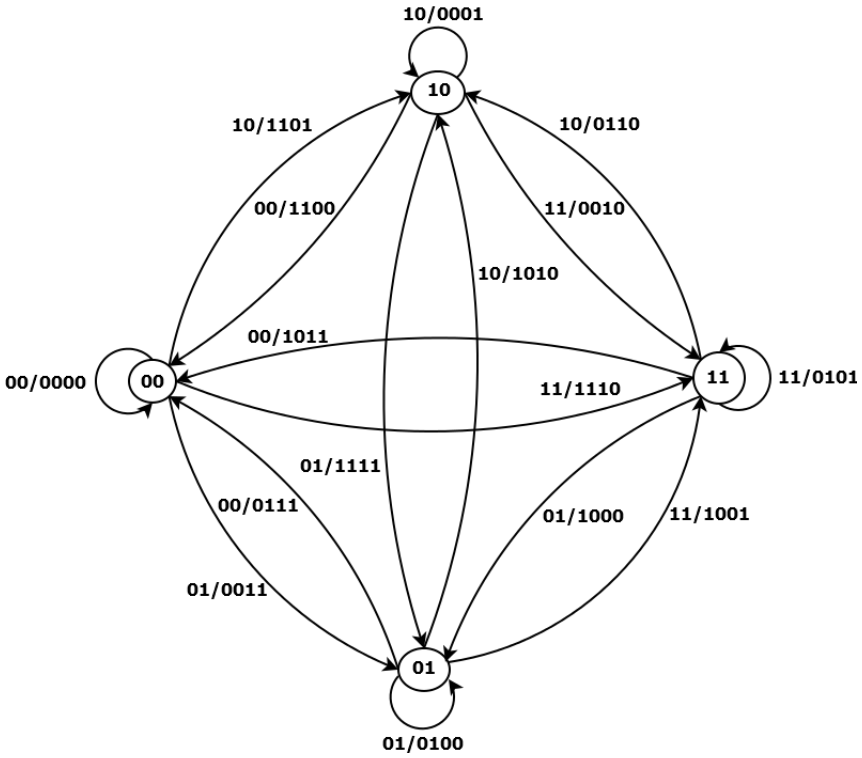

Figura 4.4: Diagrama de estados para o código convolucional (4, 2, 1).

Fonte: Autora

Adicionando o vetor erro de transmissão

$$v_i = (0010, 0000, 0000, 0000),$$

a palavra que percorrerá o canal será:

$$v_{te} = (1111, 1111, 1001, 1011).$$

Aplicando agora ao vetor v_{te} à transformação B^{-1} como mostra a equação (4.5), tem-se:

$$\hat{v}_{te} = (1011, 1011, 1101, 0111).$$

Em seguida é aplicado o processo de decodificação utilizando-se o algoritmo de Viterbi, cuja treliça está representado na Figura 4.5, obtendo assim a sequência de informação inicial

$$u_t = (10, 01, 11, 00),$$

conforme o caminho sobrevivente traçado pela cor vermelha.

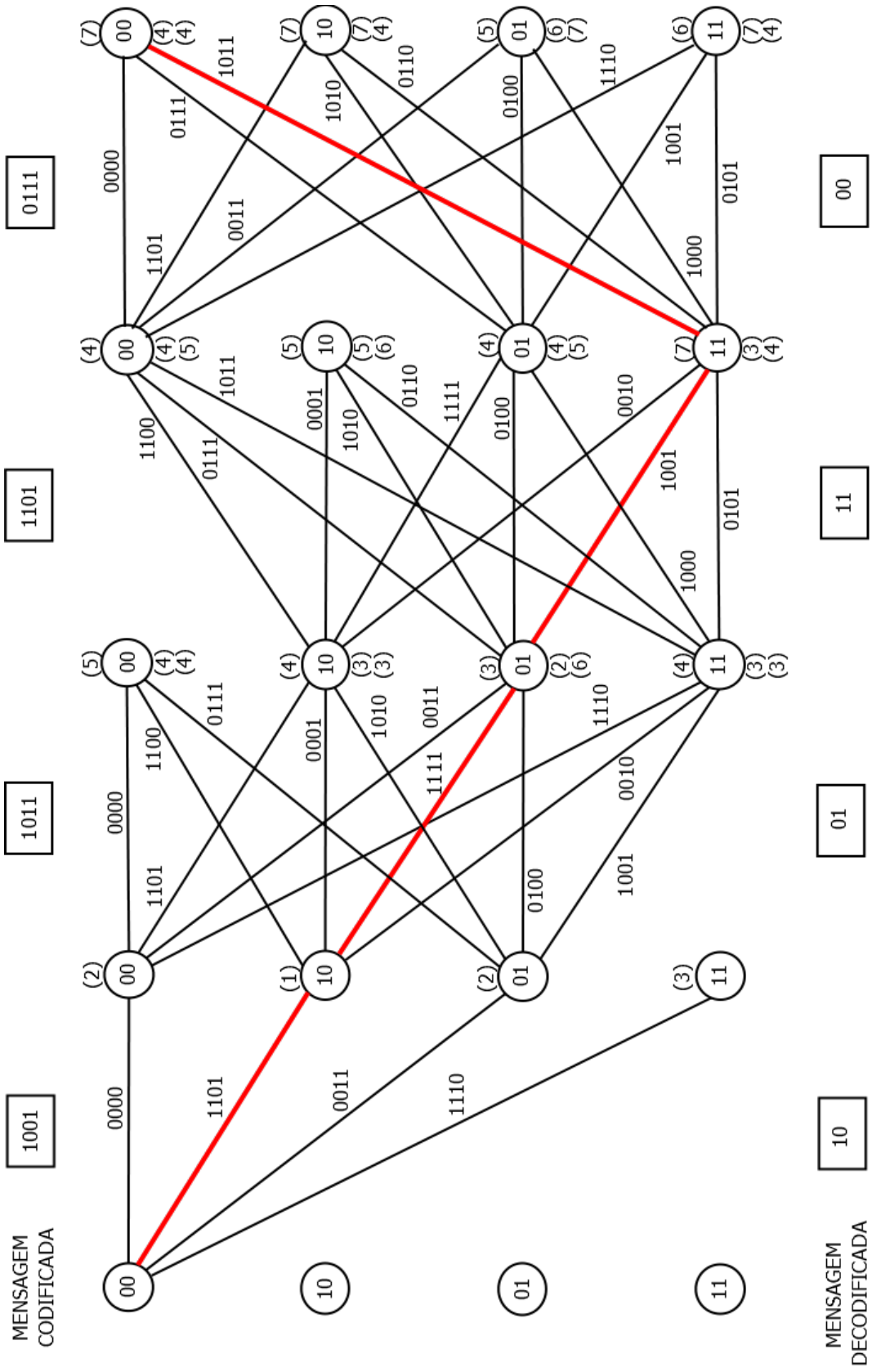

Figura 4.5: Treliça da decodificação da aplicação pelo método abrupta do algoritmo de Viterbi.

Fonte: Autora

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi apresentado uma proposta de transformação armadilha capaz de proporcionar a redução do poder de correção de erros dos códigos convolucionais ótimos de memória unitária, por meio da alteração da distância livre.

A transformação armadilha, matriz triangular inferior, aqui proposta, foi aplicada aos códigos convolucionais de memória unitária, com taxas $R = 2/4$ e $R = 2/8$. Encontrando como resultados algumas matrizes que mantiveram a distância livre do código obtido igual à do código original e outras que diminuíram o valor da distância livre a níveis desejados. Com os resultados obtidos para os códigos com as duas taxas, conclui-se que a transformação armadilha proposta apresentou bons resultados quanto à capacidade de correção de erros dos códigos convolucionais.

A proposta foi aplicada ao sistema criptográfico de chave pública com o objetivo de verificar sua eficácia, utilizando a matriz triangular inferior cuja capacidade de correção, apresentada na seção 4.1, é de 1 erro, quando o código original era capaz de corrigir 2 erros. Ao ser inserido um erro na mensagem a ser enviada, a transformação armadilha, matriz triangular inferior inversa, corrigiu esse erro e com a aplicação do algoritmo de Viterbi a mensagem foi decodificada obtendo-se a mensagem original. Confirmando, assim, a eficiência da transformação armadilha proposta.

O trabalho contribuiu à criptografia de chave pública utilizando códigos convolucionais e propondo uma transformação armadilha com o objetivo de diminuir a distância livre do código melhorando sua capacidade de correção de erros. Acreditamos que o objetivo proposto pela aplicação das transformações armadilha foi alcançado.

5.1 Produção Científica

Durante as pesquisas realizadas para esta dissertação, originaram-se dois artigos. O primeiro titulado *"Análise de uma Decodificação Abrupta pelo Algoritmo de Viterbi"*, sendo este apresentado no $XXXVI$ Congresso Nacional de Matemática Aplicada e Computacional (CNMAC 2016). O segundo artigo apresentado junto ao XIX

Encontro Nacional de Modelagem Computacional (ENMC 2016) com o título *"Análise da Representação Polinomial e Discreta dos Codificadores Convolucionais"*. Ambos estão disponíveis nos anais dos eventos.

5.2 Trabalhos Futuros

• Aplicar o trabalho com outras taxas, com possibilidade de generalização;

• Utilizar a transformação armadilha proposta em códigos de memória unitária parcial a nível de comparação aos códigos de memória unitária aplicados neste trabalho;

• Propor novas transformações armadilha verificando o poder de correção de erros dos códigos considerados.

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] HAICKEL, D. ***Uma proposição e análise de complexidade de um sistema criptográfico de chave pública utilizando códigos convolucionais de memória unitária***. Campinas: UNICAMP, 1991.

[10] JOHANNESSON, R. and ZIGANGIROV, K. S. ***Fundamentals of convolutional coding***. Wiley - IEEE Press, 2015.

[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] MIRITZ, R. C. ***Criptografia de chave pública***. Porto Alegre: UFRGS, 2000.

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

[17] PALAZZO Jr., R. ***New short constraint length convolutional codes derived from a network flow approach***. IEEE Trans. Comm., Brighton, U. K., 1985.

[18] PALAZZO Jr., R.; UCHÔA FILHO, B. F.; ARPASI, J. P. ***Notas de aula - Códigos convolucionais***, Departamento de Telemática. FEEC - UNICAMP, Campinas, São Paulo, 2005.

[19] PETERSON, W.; WELDON, E. J. ***Error-correcting codes***. MIT Press, Cambridge, Marsachussets, 1972.

[20] PIEDADE, D. C. S.; SOUSA, E. V. de. SILVA FILHO, J. C. Análise de uma decodificação abrupta pelo algoritmo de Viterbi. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 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] SILVA FILHO, J. C.; BORELLI, W. C. e MARQUES, E. M. R. ***Classe de equivalência de códigos de treliça via partições de reticulados***, VII ERMAC, Recife, 2007.

[24] SINGH, S. ***O livro dos códigos***. 5. ed. Rio de Janeiro: Record, 2005.

[25] SOUSA, E. V. de. PIEDADE, D. C. S.; SILVA FILHO, J. C. Análise da representação polinomial e discreta dos codificadores convolucionais. In: ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL, 19., 2016, João Pessoa. **Anais**... São Paulo: SBMAC, 2016.

[26] VALDEVINO, A. ***Criptografia Caótica***. Brasília: Universidade Católica de Brasília, 2010.

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

[28] YOUNG, M. C. P. ***Obtenção de códigos convolucionais ótimos de memória unitária por programação matemática***. Campinas: UNICAMP, 1989.