Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Polyane Alves Santos

Uma Proposta de um Sistema Criptográfico de Chave Pública Utilizando Códigos Convolucionais Clássicos e Quânticos

Campinas
2008

Polyane Alves Santos

Uma Proposta de um Sistema Criptográfico de Chave Pública Utilizando Códigos Convolucionais Clássicos e Quânticos

Dissertação de mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.
Área de concentração: Telecomunicações e Telemática.

Orientador: Prof. Dr. Reginaldo Palazzo Júnior

Campinas
2008

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Sa59p

Santos, Polyane Alves
Uma proposta de um sistema criptográfico utilizando códigos convolucionais clássicos e quânticos / Polyane Alves Santos. --Campinas, SP: [s.n.], 2008.

Orientador: Reginaldo Palazzo Junior.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Criptografia. 2. Teoria de codificação. 3. Codigos de controle de erros (Teoria da informação). I. Palazzo Junior, Reginaldo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: A proposal of a cryptographic system of public key using classical and quantum convolutional codes
Palavras-chave em Inglês: Cryptographic, Codification, Quantum error correction codes
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Carlos Eduardo Câmara, Yuzo Iano
Data da defesa: 15/07/2008
Programa de Pós Graduação: Engenharia Elétrica

Polyane Alves Santos
Graduada em Matemática – UESC-BA

Uma Proposta de um Sistema Criptográfico de Chave Pública Utilizando Códigos Convolucionais Clássicos e Quânticos

Dissertação de mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Telecomunicações e Telemática. Aprovada pela banca examinadora no dia 15 de Julho de 2008.

Banca Examinadora:

Prof. Dr. Reginaldo Palazzo Júnior
FEEC/UNICAMP

Prof. Dr. Yuzo Iano
FEEC/UNICAMP

Prof. Dr. Carlos Eduardo Camara
UFS – Itatiba

Campinas
2008

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Polyane Alves Santos

Data da Defesa: 15 de julho de 2008

Título da Tese: "Uma Proposta de Um Sistema Criptográfico de Chave Pública Utilizando Códigos Convolucionais Clássicos e Quânticos"

Prof. Dr. Reginaldo Palazzo Júnior (Presidente): _____

Prof. Dr. Carlos Eduardo Câmara: _____

Prof. Dr. Yuzo Iano: _____

Dedico aos meus pais, Maria da Glória e Antonio Eduardo, por suas sábias lições de esperança; sempre apoiando-me em minhas decisões. E aos meus irmãos, Edivan Alves e Élder Alves, por sempre acreditarem em mim, fornecendo-me o incentivo necessário para a realização dos meus sonhos.

Agradecimentos

Eis que chegou o momento de expressar os sinceros agradecimentos a todos adorados familiares e amigos, tanto aos antigos e queridos, quanto aos que se revelaram ao longo desse tempo.

Agradeço primeiramente a Deus, minha fonte de força suprema.

Minha dívida pessoal é com meu orientador, Prof. Reginaldo Palazzo Júnior, que me acolheu, sempre demonstrando acreditar em meu potencial, incentivando e auxiliando no desenvolvimento deste trabalho.

Aos professores da Banca Examinadora: Prof. Dr. Carlos Eduardo Camara e Prof. Dr. Yuzo Iano, pelas sugestões e leitura que levaram ao enriquecimento deste trabalho.

Aos professores de graduação Jaime Apaza, João Paulo Attie e Humberto Bortolossi, por acreditarem em mim, apoiando-me sempre que necessitei.

Aos amigos Ingrid Sampaio, Liliane Xavier, Germano José, Karoline Reis, Shirley Gomes, Jaqueline Cardoso, Débora Meneghetti, Carlos Tomé, Giuliano La Guardia, João Coelho, Márcia Braga, Igor Vellenich, a todos colegas do DT e a tantos outros que foram muito importantes durante esta caminhada.

Agradeço ao meu namorado, Cláudio Henrique, por caminhar sempre ao meu lado. Sou igualmente grata à sua família, em especial, à minha sogra Rita de Cássia pelo carinho que tem por mim.

Agradeço também à Capes, pelo suporte financeiro.

Agradecer a todos que ajudaram a alcançar essa vitória não é uma tarefa fácil. O problema não é decidir quem incluir, mas sim, decidir quem não mencionar. Por isso, a todos os meus amigos que, direta ou indiretamente, contribuíram com sua amizade e com sugestões efetivas para a realização deste trabalho, gostaria de expressar minha profunda gratidão.

A ciência humana de maneira nenhuma nega a existência de Deus. Quando considero quantas e quão maravilhosas coisas o homem compreende, pesquisa e consegue realizar, então reconheço claramente que o espírito humano é obra de Deus, e a mais notável.

Galileu Galilei

Resumo

A proposta de um sistema criptográfico de chave pública que utiliza códigos convolucionais de memória-unitária clássicos e quânticos apresentada neste trabalho, está baseada na utilização de transformações armadilha que, ao serem aplicadas às submatrizes reduzem a capacidade de correção de erros do código. Este processo proporciona um aumento no grau de privacidade da informação a ser enviada devido a dois fatores: para a determinação de códigos ótimos de memória unitária é necessário resolver o Problema da Mochila e a redução da capacidade de correção de erro dos códigos ocasionada pelo embaralhamento das colunas das submatrizes geradoras. São também apresentados neste trabalho, novos códigos convolucionais quânticos concatenados $[(4, 1, 3)]$.

Códigos Convolucionais Clássicos, Códigos Convolucionais Quânticos, Transformações Armadilhas, Sistema Criptográfico de Chave Pública, Problema da Mochila, Códigos Concatenados.

Abstract

The proposal of a cryptographic system of public key that uses classical and quantum convolutional codes of unit-memory presented in this work, is based on the use of trapdoors functions which when applied to submatrices reduce the capacity of correction of errors of the code. This process gives us an increase in the degree of privacy of information being sent, because of two factors, namely: to establish good unit-memory codes is necessary to solve the knapsack problem, and the reduction of the capacity of correcting errors of codes provided by scrambling the columns of generating submatrices. We also present in this work, news quantum convolutional codes $[(4, 1, 3)]$.

Key-words: Classical Convolutional Codes, Quantum Convolutional Codes, Functions Trapdoors, Cryptographic System of Public Key, Knapsack Problem, Concatenated Codes.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A criptografia é tão antiga quanto a escrita. Originou-se da necessidade de serem enviadas informações entre dois ou mais pontos sem que as mesmas fossem interceptadas ou alteradas no percurso entre os pontos de envio e recepção. Essencialmente, a criptografia consiste de um conjunto de técnicas que permitem tornar incompreensível, a pessoas não autorizadas, uma mensagem originalmente escrita com clareza, de forma a permitir que apenas o destinatário a decifre e compreenda.

De origem grega, a palavra Criptografia significa escrita secreta (kriptos = secreta e grafos = escrita). O imperador romano Júlio César fazia uso da criptografia para troca de mensagens com as tropas, nas campanhas do exército romano. Durante a Idade Média, o interesse pela criptografia diminuiu, bem como em muitas outras áreas de atividades intelectuais.

A arte da criptografia voltou a crescer com o início do Renascimento Italiano. Na época de Luís XIV da França, um código baseado em 587 chaves aleatoriamente selecionadas era utilizadas em mensagens governamentais. No início do século XIX, dois fatores impulsionaram o desenvolvimento da criptografia. O primeiro fator, foram as histórias de Edgar Allan Poe, tais como "The Gold Bug", que apresentavam mensagens em código, excitando assim, a imaginação dos leitores; o segundo, foi a invenção do telégrafo e do código Morse. Um fato interessante é que o Código de Morse foi o primeiro método de representação binária (traço e ponto) e foi aplicado em diversas áreas.

Com a I Guerra Mundial, foram construídas várias "Máquinas de Codificação" mecânicas que permitiam facilmente codificar e decodificar textos utilizando cifragens complexas e sofisticadas. A partir daí, surge a Criptografia Eletrônica.

Durante a II Guerra Mundial, uma máquina para cifrar, chamada Máquina de Hagelin, foi extensivamente utilizada pelo exército dos Estados Unidos. O livro "The Codebreakers, The Story of Secret Writing"[14], é o estudo mais completo sobre criptografia praticada desde os primórdios até a II Guerra Mundial.

Durante um bom tempo a criptografia foi utilizada exclusivamente pelos serviços militares e comunicações diplomáticas. Entretanto, com o grande desenvolvimento ocorrido nos setores

comerciais, industriais, negociações bancárias, etc., o interesse pela criptografia foi se tornou-se cada vez maior, com o objetivo de proteger os interesses de cada uma das partes componentes do sistema de comunicação.

Códigos inquebráveis e seguros tornaram-se ainda mais necessários com o advento dos computadores. Não só os arquivos precisam ser mantidos em sigilo, mas o acesso ao computador em si também deve ser gerenciado e regulado.

Na criptografia, existem dois métodos pelos quais podem ser feito o processo de cifragem e decifragem dos sistemas criptográfico, a saber: o método convencional e o método por chaves públicas.

No sistema convencional, dois usuários que desejam se comunicar, devem pré-estabelecer uma chave comum através de um canal seguro (meio físico que interliga os dois pontos) de envio e recepção de mensagens.

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, onde o número de chaves criptográficas necessárias é, no máximo, $N(N-1)/2$. O estabelecimento destas chaves, pode ser inviável, por exemplo, para um sistema com 100.000 usuários, pois, caso exista conexão entre todos os usuários o número total de chaves seria de 5 bilhões.

A criptografia contemporânea não é mais baseada em obscuridade, ou seja, não se utiliza mais a suposição que qualquer sistema pode ser seguro, ou seja, ninguém, exceto seus criadores, tem acesso à metodologia ou aos algoritmos internos do sistema. A idéia básica da utilização de chave pública é a eliminação da necessidade de utilização de um canal eficiente e de alta segurança para a distribuição de chaves.

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

Existem duas formas de implementar um sistema do tipo proposto por Diffie e Hellman. Na primeira, as duas partes envolvidas na comunicação trocam chaves entre si através do canal, permitindo às partes e somente às partes, estabelecerem uma chave comum que será utilizada para cifrar e decifrar. Na segunda, duas chaves são utilizadas, 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.

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

segurança.

Neste trabalho utilizamos do sistema criptográfico de chave pública que faz uso do método do Problema da Mochila. Este sistema é baseado nos códigos convolucionais clássicos de memória-unitária, buscando-se explorar todo seu grau de complexidade inerente ao processo de determinação de códigos ótimos e ao processo de decodificação.

Uma das novidades que este trabalho apresenta, é a proposta do sistema criptográfico de chave pública baseado no método do Problema da Mochila, onde os códigos utilizados são convolucionais quânticos. Outra contribuição é a construção de novos códigos convolucionais quânticos $[4, 1, 3]$ que possuem d_{free} maior do que o d_{free} do código convolucional quântico conhecido até o presente momento. Estes novos códigos serão utilizados no sistema criptográfico.

O trabalho está organizado como segue:

O **Capítulo 2** apresenta uma revisão sobre os códigos convolucionais. Também, é feita uma revisão sobre os métodos criptográficos existentes, desde os convencionais até os de chave pública.

No **Capítulo 3** é apresentado o problema de determinação de códigos ótimos de memória unitária e sua aplicação ao Problema da Mochila binário para o sistema criptográfico de chave pública.

No **Capítulo 4** são definidas as transformações armadilha utilizadas no trabalho e, em seguida são apresentados os resultados da aplicação destas transformações aos códigos convolucionais ótimos de memória unitária. As transformações armadilha de permutação são aplicadas aos códigos convolucionais ótimos de memória unitária com taxa $r = 2/3$, $r = 2/4$ e $r = 3/4$; as transformações armadilha triangulares superiores são aplicadas aos códigos convolucionais ótimos de memória unitária com taxa $r = 2/4$ e $r = 2/8$ e as transformações armadilha de Hadamard também são aplicadas aos códigos convolucionais ótimos de memória unitária com taxa $r = 2/4$ e $r = 2/8$. Após o processo de aplicação, é realizada uma análise acerca dos resultados obtidos em cada um dos casos e também a comparação entre o desempenho das transformações. Ainda, aplicamos as mesmas transformações armadilha à dois códigos com taxa $r = 3/6$, sendo destes, um convolucional de memória unitária e outro, um convolucional de memória unitária parcial. Em seguida, apresentamos os resultados obtidos e é realizada uma comparação entre o comportamento destes dois códigos ao sofrerem alterações através das transformações armadilha.

O **Capítulo 5** apresenta o processo de construção e geração do código convolucional quântico (CCQ) concatenado $\mathcal{C}$ $[4, 1, 3]$. Este CCQ concatenado é associado a um código convolucional clássico (CCC) concatenado, que por sua vez, é obtido a partir da concatenação de dois CCCs, $\mathcal{C}_1$ e $\mathcal{C}_2$. As transformações armadilha são aplicadas ao CCC $\mathcal{C}_2$ e, logo após, observamos o efeito causado no novo código convolucional quântico (CCQ) concatenado $\mathcal{C}'$.

Também, neste capítulo, apresentamos os novos CCQs concatenados $[4, 1, 3]$ encontrados. Além disso, realizamos em tais códigos o processo de aplicação das transformações armadilha para serem utilizados no sistema criptográfico.

No **Capítulo 6**, apresenta as considerações finais deste trabalho e também as propostas para trabalhos futuros.

Capítulo 2

Revisão de Códigos Convolucionais e Sistemas Criptográficos

2.1 Códigos Convolucionais

A diferença básica entre codificação de bloco e codificação convolucional, é que na codificação de bloco, o bloco codificado no instante de tempo i depende somente do bloco de informação do instante i, enquanto que na codificação convolucional, o bloco codificado no instante de tempo i depende não somente do bloco de informação do instante i, mas também de m blocos de informação anteriores. Devido a este fato, um codificador convolucional requer *memória*.

Um codificador convolucional (n, k, m) com k entradas, n saídas e m memórias (m, mais especificamente é o número máximo de registradores em uma certa entrada) pode ser implementado como um circuito lógico-sequencial. Em um *circuito sequencial linear* (CSL) [3], os sinais escolhidos dentre os elementos de um corpo finito F_q (na prática, consideramos somente o corpo binário F_2) são aplicados simultaneamente a todos os terminais de entrada em instantes discretos de tempo. Esse circuito consiste de uma rede de interconexões entre um número finito de componentes primitivos. Dois tipos destes componentes são considerados em codificadores convolucionais: os *somadores* (que implementam adição módulo-q) e os *elementos da memória* para atrasar (ou armazenar) um sinal de entrada por uma unidade de tempo.

As variáveis básicas de um circuito CSL são a entrada, a saída e os estados (conteúdos dos registros de deslocamento). Neste tipo de circuito, a seqüência de saída é uma aplicação linear da seqüência de entrada e dos estados.

Considere o circuito de codificação exibidos na Figura 2.1. Este codificador consiste de um registro de deslocamento contendo duas memórias e de dois somadores módulo-**2**. A seqüência de informação $\mathbf{u}$ é deslocada da esquerda para a direita de um bit por unidade de tempo, e duas saídas codificadas $\mathbf{v_i^{(1)}}$ e $\mathbf{v_i^{(2)}}$ são geradas pelos somadores módulo-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 a terceira células, respectivamente) e as duas saídas são $\mathbf{v_{i-1}}$

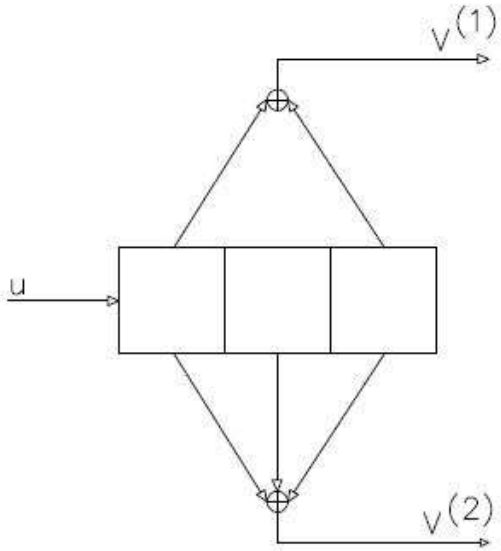

Figura 2.1: Codificador convolucional.

e $\mathbf{v_{i-2}}$. A primeira saída, $\mathbf{v_i^{(1)}}$, é a soma módulo-2 de u_i e u_{i-2}, enquanto que a segunda saída, $\mathbf{v_i^{(2)}}$, é a soma módulo-2 de u_i, u_{i-1} e u_{i-2}. A seqüência de bits codificados é então multiplexada e transmitida pelo canal.

Como um exemplo, considere a seqüência de informação $\mathbf{u} = 1011001$. Assuma que as duas últimas células inicialmente contenha 0s. Então, 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 $v_1^{(1)} = 0$ 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 segunda célula. O bit de informação corrente $u_2 = 0$ gera dois bits codificados $v_2^{(1)} = 0$ e $v_2^{(2)} = 0$, e assim sucessivamente, obtendo a seqüência codificada $\mathbf{v} = 11010010101111$. Observe que os dois últimos pares de bits codificados são obtidos através do deslocamento de 0s após u_6, nos instantes $i = 7$ e $i = 8$.

Em geral, no instante de tempo i, um bloco de informação com k bits é deslocado para dentro do codificador e um bloco de n bits é gerado na saída do codificador. A taxa de codificação é definida como sendo, $R = k/n$.

2.2 Representações de Codificadores Convolucionais

2.2.1 Representação Discreta

Um codificador convolucional de taxa $R = k/n$ é representado por nk seqüências de geradores $\mathbf{g_i^{(j)}} = (g_{i,0}^{(j)}, (g_{i,1}^{(j)}, \ldots, (g_{i,m}^{(j)})$, para $i = 1, \ldots, k$ e $j = 1, \ldots, n$, onde i representa a entrada, j a saída e m o número de memórias.

Evidentemente, a operação de codificação convolucional é a convolução discreta da seqüência de informação com as seqüências de geradores; é expressa como sendo

$$\mathbf{v}^{(j)} = \sum_{i=1}^{k} \mathbf{u}^{(i)} * \mathbf{g}_i^{(i)}, \tag{2.1}$$

para $j = 1, \ldots, n$, onde $*$ é a *operação de convolução*. Cada uma das n seqüências codificadas $\mathbf{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, \ldots)$, onde v_i^j tem comprimento n, para $j = 1, \ldots, n$ pode depender de cada uma das k seqüências de informação $\mathbf{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \ldots)$, onde u_i^j tem comprimento k, para $i = 1, \ldots, k$. A seqüência geradora composta para a i-ésima entrada do codificador é definida como

$$\mathbf{g_i} = (g_{i,0}^{(1)}, g_{i,0}^{(2)}, \ldots, g_{i,0}^{(n)}, g_{i,1}^{(1)}, \ldots, g_{i,1}^{(n)}, \ldots, g_{i,m}^{(1)}, \ldots, g_{i,m}^{(n)}). \tag{2.2}$$

A convolução discreta (2.1) pode ser expressa de uma forma mais compacta através uma multiplicação de matrizes. As k seqüências de informação podem ser escritas como sendo a seqüência de informação

$$\mathbf{u}^{(i)} = (\mathbf{u}_0, \mathbf{u}_1, \ldots) = (u_0^{(1)}, u_0^{(2)}, \ldots, u_0^{(k)}, u_1^{(1)}, \ldots, u_1^{(k)}, \ldots),$$

onde $\mathbf{u}_t = (u_t^{(1)}, \ldots, u_t^{(k)})$ é o bloco de informação no instante de tempo t. Da mesma forma, a palavra-código obtida a partir das n seqüências codificadas é dada por

$$\mathbf{v}^{(i)} = (\mathbf{v}_0, \mathbf{v}_1, \ldots) = (v_0^{(1)}, v_0^{(2)}, \ldots, v_0^{(n)}, v_1^{(1)}, \ldots, v_1^{(n)}, \ldots),$$

onde $\mathbf{v}_t = (v_t^{(1)}, \ldots, v_t^{(n)})$ é o bloco codificado no instante de tempo t. Portanto, a codificação convolucional pode ser escrita como

$$\mathbf{v} = \mathbf{uG}, \tag{2.3}$$

onde $\mathbf{G}$ é uma matriz geradora semi-infinita definida como

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

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

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

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

Exemplo 2.2.1. *A matriz geradora semi-infinita* $\mathbf{G}$ *para o codificador* $(2, 1, 2)$*, mostrada na Figura 2.1, é:*

$$\mathbf{G} = \begin{bmatrix} 11 & 01 & 11 & & & & \\ & 11 & 01 & 11 & & & \\ & & 11 & 01 & 11 & & \\ & & & 11 & 01 & 11 & \\ & & & \ddots & \ddots & \ddots & \end{bmatrix}. \tag{2.4}$$

A seqüência gerada pela seqüência de informação $\mathbf{u} = 1011001$ *é obtida pela multiplicação matricial a seguir:*

$$(1011001) \begin{bmatrix} 11 & 01 & 11 & & & & & & & \\ & 11 & 01 & 11 & & & & & & \\ & & 11 & 01 & 11 & & & & & \\ & & & 11 & 01 & 11 & & & & \\ & & & & 11 & 01 & 11 & & & \\ & & & & & 11 & 01 & 11 & & \\ & & & & & & 11 & 01 & 11 & \end{bmatrix}. \tag{2.5}$$

a qual produz $\mathbf{v} = 11010010101111$.

No Exemplo (2.2.1), uma seqüência de informação $\mathbf{u}$ de comprimento $N = 7$ bits foi codificada em uma palavra-código $\mathbf{v}$ de comprimento $M = 18$ bits. Os quatro últimos bits na palavra código correspondem a $m = 2$ zeros que são usados para retornar ao estado inicial do codificador. Pelo fato de existirem 2 saídas e 1 entrada, temos que

$$g_1^{(1)} = [101]$$

e

$$g_1^{(2)} = [111]$$

e assim, a primeira linha de $\mathbf{G}$ a saber, $g = 110111$, recebe o nome de *seqüência geradora do código* e é a partir desta que obtemos a matriz geradora da seguinte forma: a segunda linha obtemos através do deslocamento da primeira linha de duas posições para a direita. A terceira linha obtemos através do deslocamento da segunda linha de duas posições para a direita, e assim sucessivamente.

Ao considerarmos um codificador (n, k, m), uma seqüência de comprimento N é codificada em uma palavra-código com comprimento $M = (N + m)$ blocos, ou seja, com $n(N + m)$ bits,

onde os últimos nm bits são gerados a partir de m blocos de informação com todos os bits 0s anexados à seqüência de informação. Portanto, a matriz geradora $\mathbf{G}$ 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 seqüência de informação na operação de convolução implementada com o uso de registros de deslocamento.

2.2.2 Representação Polinomial

As seqüências de geradores $g_i^{(j)}$, com $i = 1, \ldots, k$ e $j = 1, \ldots, n$ podem ser representados por polinômios de grau finito tendo como variável o operador de retardo D. Os polinômios geradores para um codificador (n, k, m) são denotados por $\mathbf{g}_i^j(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)} D + \ldots + g_{i,m}^{(j)} D^m$, onde $i = 1, \ldots, k$ e $j = 1, \ldots, n$.

As seqüências de entrada e saída podem ser também escritas utilizando o operador de retardo: $\mathbf{u}^{(i)}(D) = u_0^i + u_1^i D + u_2^i D^2 + \ldots$ e $\mathbf{v}^{(i)}(D) = v_0^i + v_1^i D + v_2^i D^2 + \ldots$, respectivamente. A convolução no domínio temporal é equivalente à multiplicação no domínio das transformadas. Com isso, a operação de codificação pode ser escrita como sendo

$$\mathbf{v}^{(j)}(D) = \sum_{i=1}^{k} \mathbf{u}^{(i)} \mathbf{g}_i^{(j)}(D). \tag{2.6}$$

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

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

onde $g_i^{(j)}(D)$ é um polinômio de tal forma que, para um dado i e para todo $j = 1, \ldots, n$, 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 da seguinte maneira:

$$\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D). \tag{2.7}$$

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

2.2.3 Representação Gráfica

As representações gráficas proporcionam um melhor entendimento da operação de codificação convolucional. Nesta seção apresentaremos duas representações gráficas que são úteis

para o estudo de códigos convolucionais, a saber, o *diagrama de estados* e a *treliça*.

Diagrama de Estados

Um codificador convolucional é um circuito lógico linear e 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 (2.2), mostra o diagrama de estados para o codificador $(2, 1, 2)$. Assumiremos, 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 na primeira células e o estado no instante de tempo $i-1$ é formado pelos bits contidos na terceira e na 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 seqüência de transições de estados, começando por um estado conhecido. Em geral, existem 2^k transições saindo de cada estado, que correspondem à seqüência de informação $\mathbf{u}_i$, de comprimento de k. Para uma dada seqüência de informação, que correspondem à seqüência 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. Observe, na Figura 2.2, que pelo diagrama de estados a seqüência de saída gerada pelo codificador para a seqüência de informação $\mathbf{u} = 1011001$ é $\mathbf{v} = 11010010101111$.

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 2.3, onde existem quatro possíveis estados. Assumimos que o estado inicial é o estado todo nulo $\mathbf{a} = \mathbf{00}$. 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 a 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 $a = 00$, e correspondem à entrada de m blocos de bits zeros. Cada palavra-código é um caminho que começa no estado $a = 00$, no instante de tempo $t = 0$, e termina no estado $a = 00$.

2.3 Códigos Convolucionais Equivalentes

Um código convolucional pode ser gerado por vários codificadores, e isso motiva a introdução do conceito de equivalência.

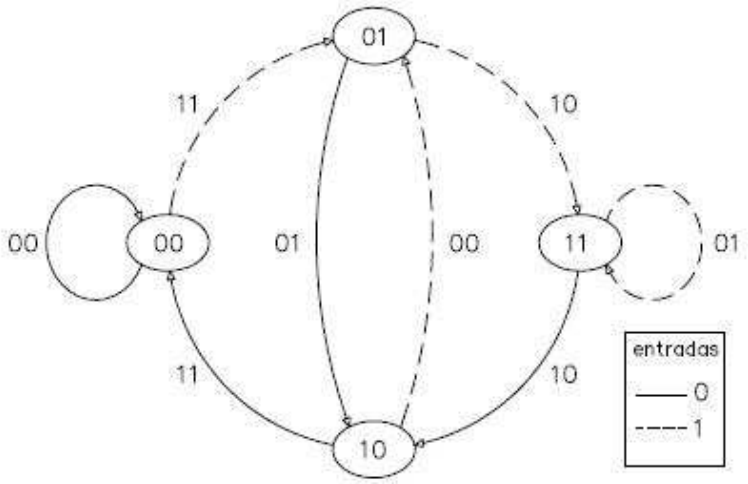

Figura 2.2: Diagrama de estados para o código convolucional (2,1,2).

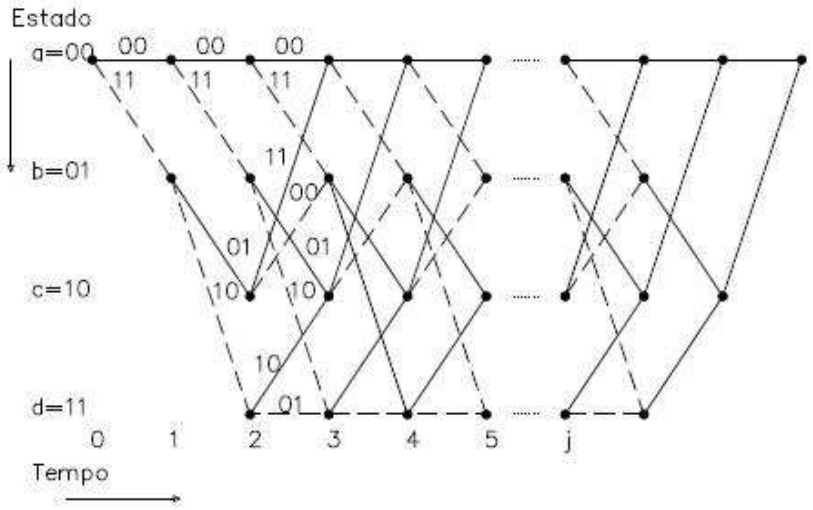

Figura 2.3: Diagrama de treliça para o código convolucional (2,1,2).

Dois codificadores são ditos *equivalentes* se eles geram o mesmo código.

Exemplo 2.3.1. *Neste exemplo construiremos um codificador equivalente ao codificador* $(2, 1, 2)$*, com taxa* $2/4$ *e com mesmo número total de memórias. A solução trivial é dada pela utilização da mesma matriz geradora discreta semi-infinita. Esta matriz, quando associada a um codificador* $(4, 2, 1)$*, pode ser escrita na forma matricial como sendo:*

$$\mathbf{G} = \begin{bmatrix} 1101 & 1100 & & & \\ 0011 & 0111 & & & \\ & 1101 & 1100 & & \\ & 0011 & 0111 & & \\ & & 1101 & 1100 & \\ & & 0011 & 0111 & \\ & & & \ddots & \ddots \end{bmatrix}, \tag{2.8}$$

ou na forma de matriz polinomial dada por:

$$\mathbf{G} = \begin{bmatrix} 1+D & 1+D & 0 & 1 \\ 0 & D & 1+D & 1+D \end{bmatrix}. \tag{2.9}$$

É possível construirmos um circuito lógico sequencial deste codificador partindo de qualquer uma das duas matrizes, como pode ser observado na Figura 2.4.

Podemos também obter códigos convolucionais equivalentes a partir de codificadores não triviais, ou seja, a partir de codificadores com diferentes matrizes geradoras.

2.4 Propriedades de Distância dos Códigos Convolucionais

O desempenho dos códigos convolucionais está diretamente ligado ao algoritmo de decodificação e também às propriedades de distância do código. As distâncias mais 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 distâncias é a distância livre, definida por Costello [24, 25], pois é através desta que podemos estabelecer a capacidade de correção ou de detecção de erros associada aos códigos convolucionais.

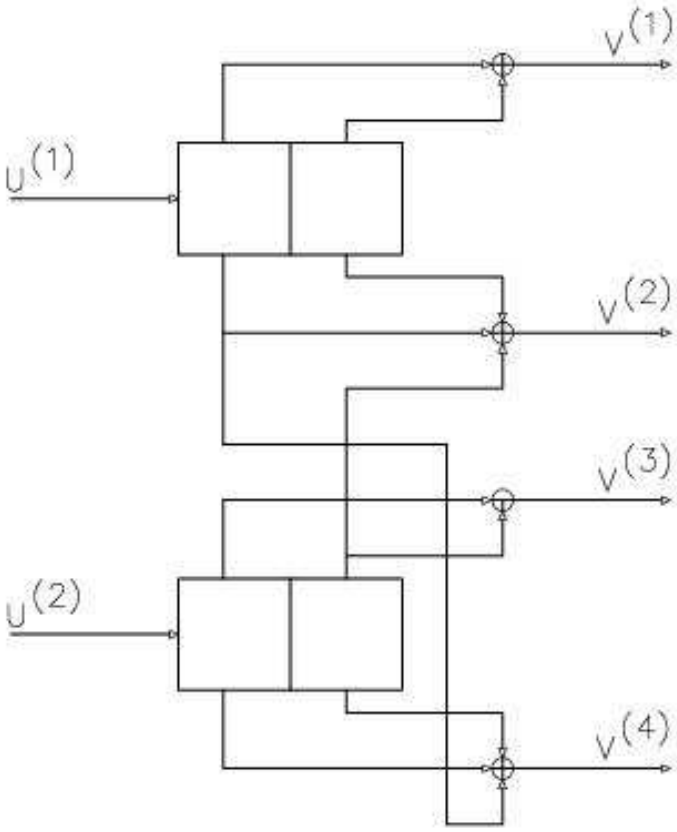

Figura 2.4: Codificador para o código convolucional (4,2,1).

2.5 Distância Livre

Define-se a distância livre, d_{free}, de um código convolucional como sendo a menor distância de Hamming entre quaisquer duas seqüências codificadas provenientes de duas seqüências de informação distintas. Simbolicamente, temos que

$$\begin{aligned} d_{free} &= \min\{d_H(\mathbf{v}, \mathbf{v}') : \mathbf{u} \neq \mathbf{u}'\} \\ &= \min\{w_H(\mathbf{v}) : \mathbf{u} \neq \mathbf{0}\} \\ &= \min\{w_H(\mathbf{uG}) : \mathbf{u} \neq \mathbf{0}\}, \end{aligned}$$

onde $\mathbf{v}$ e $\mathbf{v}'$ são as seqüências codificadas correspondentes às seqüências de informação $\mathbf{u}$ e $\mathbf{u}'$, respectivamente. Portanto, a distância livre é a distância de Hamming mínima entre quaisquer duas palavras-código distintas. Por causa da linearidade dos códigos convolucionais, a distância livre é também o menor peso da palavra-código não-nula.

Exemplo 2.5.1. *Através do diagrama de estados da Figura 2.2 ou do diagrama de treliça da Figura (2.3), 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 seqüência de informação* $\mathbf{u} = 100$*.*

Se a distância livre de um código convolucional $\mathcal{C}$ é superior a de qualquer outro código convolucional de mesma taxa e mesmo comprimento de restrição de saída, dizemos que $\mathcal{C}$ tem distância livre ótima (DLO).

A definição de distância livre nos permite definir a capacidade máxima de correção de erros:

Definição 2.5.1. *A máxima capacidade de correção de erros* t_{free} *de um código convolucional é dada por*

$$t_{free} = \lfloor \frac{d_{free} - 1}{2} \rfloor. \tag{2.10}$$

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 apresentado na próxima seção, e também o algoritmo de decodificação por síndromes.

2.6 Tipos de Codificadores Convolucionais

Consideraremos, agora, alguns tipos de codificadores convolucionais que serão importantes para o desenvolvimento do trabalho, a saber, *codificadores catastróficos e não catastróficos*, *codificadores de memória unitária* (MU) e *codificadores de memória unitária parcial* (MUP).

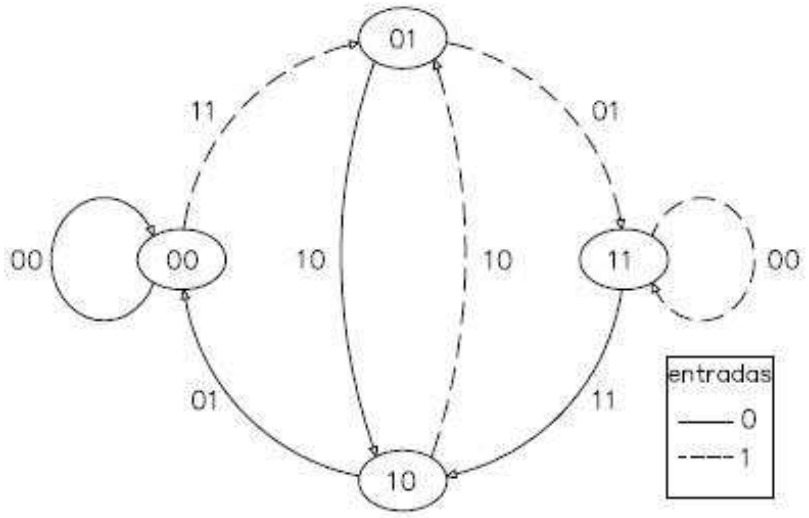

Figura 2.5: Diagrama de estados para o código convolucional catastrófico (2,1,2).

2.6.1 Codificadores Catastróficos e Não Catastróficos

Massey [6] denominou como codificadores *catastróficos* aqueles que geram uma palavra-código de peso finito para uma seqüência de informação de peso infinito. Nestes casos, existe uma probabilidade não nula de que quando esta palavra-código é transmitida através de um canal ruidoso, um número 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 [7] 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 $1/n$ devem satisfazer a condição:

$$\text{mdc}\left[\mathbf{g}^{(j)}(D), \quad j = 1, 2, \ldots, n\right] = 1, \tag{2.11}$$

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 fechada de peso nulo.

Exemplo 2.6.1 (Código Catastrófico)**.** *Para o codificador* $(2, 1, 2)$ *com polinômios geradores* $\mathbf{g}^{(1)}(D) = 1 + D$ *e* $\mathbf{g}^{(2)}(D) = 1 + D^2$*, temos mdc*$[\mathbf{g}^{(1)}(D), \mathbf{g}^{(2)}(D)] = 1 + D$*, e portanto o codificador é catastrófico.*

Considere a seqüência de informação $\mathbf{u}(D) = 1/(1 + D)$*, ou seja,* $\mathbf{u} = 1111....$ *Neste caso,*

as seqüências codificadas são $\mathbf{v}^{(1)} = \mathbf{u}(D)g^{(1)}(D) = 1$ *e* $\mathbf{v}^{(2)} = \mathbf{u}(D)g^{(2)}(D) = 1 + D$*, o que corresponde à seqüência codificada* $\mathbf{v} = 11010000\ldots$*. Se os três bits não nulos forem afetados por erros, a seqüência recebida será* $\mathbf{r} = 000000\ldots$*. Neste caso, o codificador irá produzir a seqüência toda nula como estimativa da seqüência de informação original, resultando portanto em um número infinito de erros de decodificação. Na Figura 2.5 é mostrado o diagrama de estados onde podemos observar que existe uma auto-malha de peso zero em torno do estado* 11*.*

2.6.2 Codificadores de Memória Unitária

Sejam $\mathbf{u_t}$ o vetor de entrada k-dimensional e $\mathbf{v_t}$ o vetor n-dimensional de dígitos codificados cujas componentes pertencem a F_q definidos por

$$\mathbf{u_t} = (u_{t1}, u_{t2}, \ldots, u_{tk}) \quad \text{e} \quad \mathbf{v_t} = (v_{t1}, v_{t2}, \ldots, v_{tn}),$$

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

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

$$\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1 + \ldots + \mathbf{u}_{t-m}\mathbf{G}_m, \tag{2.12}$$

onde, $t > 0$ e $\mathbf{u}_{-1} = \mathbf{0}$, e $\mathbf{0}$ 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.

Lee [4] mostrou que o codificador convolucional $(n' = mn_0, k' = mk_0, m' = 1)$ definido por

$$G'_0 = \begin{bmatrix} \mathbf{G_0} & \mathbf{G_1} & \cdots & \mathbf{G_{m-1}} \\ \mathbf{0} & \mathbf{G_0} & \cdots & \mathbf{G_{m-2}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{G_0} \end{bmatrix} \quad e \quad G'_1 = \begin{bmatrix} \mathbf{G_m} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{G_{m-1}} & \mathbf{G_m} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G_1} & \mathbf{G_2} & \cdots & \mathbf{G_m} \end{bmatrix}, \tag{2.13}$$

é equivalente ao código em (2.12), no sentido que a mesma seqüência codificadora binária semi-infinita, está associada à mesma seqüência de entrada semi-infinita. Lee denominou o código em (2.13) por *código de memória unitária* e definiu a *complexidade de estados* de um codificador binário como sendo 2^{mk_0}, pois existem 2^{mk_0} estados distintos do codificador.

A complexidade dos dois codificadores equivalentes é a mesma, porém, o valor máximo de d_{free} é alcançado dentro do subconjunto de codificadores convolucionais com memória unitária.

2.6.3 Códigos de Memória Unitária Parcial

Lauer [5] propôs uma classe de códigos convolucionais denominados códigos convolucionais de *memória unitária parcial* (MUP). Esta classe 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 implica menor complexidade em termos do número de estados quando comparado ao código de

memória unitária. Lauer definiu a complexidade de estado μ de um codificador de memória unitária como sendo o posto de $\mathbf{G'_1}$. Note que esta definição difere da definição de complexidade de estado, proposto por Lee.

Uma condição suficiente para um código MUP ser não catastrófico é 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*.

No próximo capítulo, apresentamos o problema referente à determinação de bons códigos convolucionais. O algoritmo para a obtenção de códigos convolucionais ótimos de memória unitária via o Knapsack combinatorial, faz uso de técnicas de otimização combinatorial quando tal problema é caracterizado com o de determinar o fluxo máximo em uma rede.

Esta associação permite nos relacionar os parâmetros de cada problema como segue:

- A representação da divisão do diagrama de estados é associada com a *rede de fluxos*;
- os estados são associados com os *nós*;
- a distância de Hamming de cada divisão no diagrama de estados particionado é associado com o *ramo de fluxo*.

Além disso, a propriedade de *conservação de fluxo* se mantém, ou seja, a soma das distâncias de Hamming dos ramos saindo de um dado estado intermediário é igual a somas da distâncias de Hamming entrando no dado estado; a soma das distâncias de Hamming saindo e voltando ao estado zero no diagrama de estados particionado é igual ao *fluxo máximo*.

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

Definição 2.6.1 (Código Ótimo)**.** *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-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 $d_{free}+1$, a política adotada será a 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.

Definição 2.6.2 (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:*

$$\phi = n.(q-1).q^{(k-1)} \tag{2.14}$$

Note que a equação (2.6.2) representa o peso de Hamming total de um código de bloco sobre F_q contendo q^k palavras-código de comprimento n.

Definição 2.6.3 (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.*

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

Em [10] foi proposto um algoritmo para encontrar códigos ótimos de memória unitária, que será apresentado no Capítulo 3.

2.7 Decodificação dos Códigos Convolucionais

O algoritmo de decodificação dos códigos convolucionais mais conhecido é o *algoritmo de Viterbi.*

2.7.1 O Algoritmo de Viterbi

Um decodificador de máxima verossimilhança (do inglês, Maximum Likelihood Decoder) para um determinado tipo de canal sem memória, escolhe como palavra-código aquela que maximiza o logaritmo da função probabilidade condicional conjunta de uma seqüência de saída, dado uma seqüência de informação na entrada do canal. Esta função é chamada *métrica* associada ao caminho.

Como o algoritmo de Viterbi [19, 20, 21] é um decodificador de máxima verossimilhança, este algoritmo escolhe o caminho na treliça com a maior métrica e assim, fornece a melhor seqüência estimada dentre todas as possíveis seqüências na treliça a partir da métrica acumulada.

Seja r a seqüência recebida e v a seqüência na entrada do canal, isto é, $r = (r_1, r_2, \ldots, r_N)$ e $v = (v_1, v_2, \ldots, v_N)$, onde r_i e v_i têm comprimento n.

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

$$\log P(r|v) = d(r, v) \log \frac{p}{1-p} + N \log(1-p) \tag{2.15}$$

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

O conjunto das possíveis transições de estados na treliça entre os instantes de tempo i e $i+1$ será definida como na seção (2.2.3), na janela de tempo i. Assim, 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 a nova métrica para todas as transições de estados na janela de tempo $i+1$, e acumular a 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.

2.8 Sistemas Criptográficos

Os sistemas criptográficos podem ser classificados como sendo do tipo convencional e do tipo chave pública.

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

2.8.1 Sistemas criptográficos convencionais

Em qualquer um dos sistemas criptográficos utilizados, ou seja, o convencional ou o de chave pública, existem dois tipos de problemas a serem resolvidos:

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 2.6 é mostrado um sistema criptográfico com privacidade.

2. A autenticidade: busca evitar que a informação seja alterada pelo criptoanalista. Na Figura 2.7 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.

Ilustramos um sistema criptográfico envolvendo o problema de privacidade, na Figura 2.6. 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.

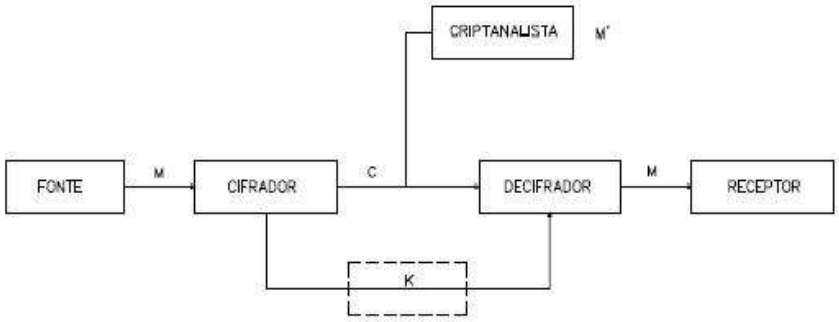

Figura 2.6: Criptossistema com privacidade.

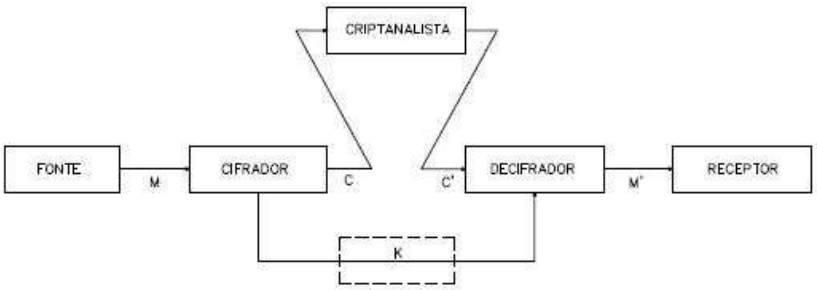

Figura 2.7: Criptossistema com autenticidade.

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*.

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

$$S_k : \overline{M} \to \overline{C}$$

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*.

No sistema criptográfico, toda segurança está concentrada na chave utilizada para cifrar. Podemos 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. Portanto, em um sistema criptográfico convencional a chave deve ser enviada do transmissor para o receptor através de um canal seguro. Aí reside 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.

Na Figura 2.7 observamos o motivo pelo qual um sistema criptográfico possa ser utilizado também para resolver o problema da autenticidade. Para este caso, 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.

Como um exemplo de um cripto-sistema convencional, ilustramos como cifrar uma mensagem representada por um vetor w de dimensão n. Para tal, basta multiplicar o vetor w por uma matriz invertível E de dimensão $l \times l$. A mensagem cifrada é obtida através do produto Aw. Se $D = E^{-1}$, então $w = DEw$. Assim, os processos de cifragem e decifragem requerem l^2 operações. Entretanto, para o cálculo de D e E, necessita-se inverter algumas matrizes que, no pior caso, apresenta complexidade de l^3 operações, donde, não é um problema de fácil solução.

Existem dois tipos de ataque que podem ser utilizados pelo criptoanalista:

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

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

Geralmente, na prática os ataques são desenvolvidos aos textos cifrados. Existem fundamentalmente dois tipos distintos de segurança exigidas para que se possa considerar um criptosistema seguro. Em alguns sistemas, a quantidade de informação útil em posse do criptoanalista, é insuficiente para que ele determine o par de transformação correspondente à cifragem e decifragem, mesmo que este possua grandes recursos computacionais. A este tipo de sistema diz-se que se tem uma *segurança incondicional*.

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

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

A segurança incondicional foi analisada com mais detalhes por Shannon em 1949, onde foi estabelecido que, se o criptoanalista possuir tempo de computação ilimitado, então o mesmo não necessita de um processo computacional eficiente. Shannon denominou o fato que em muitos textos interceptados uma única solução é possível de distância de unicidade N_0, que é o número de caracteres para se ter uma única solução, e através desta distância estabeleceu um modelo matemático. De acordo com esse modelo de cifragem aleatória tem-se que:

$$N_0 = \frac{H(k)}{D}, \tag{2.16}$$

onde $H(k)$ é a entropia da chave, ou seja, o comprimento da chave, e D é a redundância de linguagem medida em bits/caracter. A partir da equação (2.16) segue que:

$$H(k) = N_0 D, \tag{2.17}$$

onde N_0D representa o número de equações úteis para a solução do problema da chave [14].

Quando o número de equações é maior que a entropia da chave $H(k)$, como dado pela equação (2.17), uma única solução não é possível e, neste caso, o sistema não é incondicionalmente seguro.

No caso em que $H(k) = \infty$, então, pela equação (2.16) o valor de N_0 é infinito. Shannon demonstrou que, para este último caso, o valor de $N_0 = 28$ [2] é excelente para que se possa implementar um sistema, considerando, porém, que o criptoanalista tenha capacidade computacional limitada.

Dentre os algoritmos utilizados em criptografia convencional podemos citar: os de *substituição* e os de *transposição*. Para que possamos definir estes dois sistemas é necessário introduzir alguns conceitos básicos:

- Alfabeto - constitui um conjunto de símbolos ou letras para representar a mensagem.
- Mapa - Um mapa σ de um conjunto A em um conjunto B é uma relação que associa a cada elemento $a_1 \in A$ um elemento $b_1 \in B$.
- Substituição - É aplicada em um alfabeto através de uma mapa biunívoco σ. A este alfabeto associa-se uma possível substituição de letras mapeada por um dos possíveis conjuntos de σ_i, onde $i = 0 \ldots N-1$, obedecendo a certas propriedades convenientes.

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

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 seqüência caracterizada por:

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

onde o primeiro índice em m_{ij}, caracteriza o bloco da mensagem e o segundo caracteriza a posição de uma letra no bloco, [2].

Um outro sistema de cifragem bastante utilizado é o *Data Encryption Standard* (DES), o qual se propõe a proteger a comunicação de dados entre computadores. Tal sistema opera sobre blocos de 64 bits da mensagem original para a produção da mensagem cifrada. Um cifrador de blocos divide a mensagem original em blocos de tamanho fixo, e opera em cada bloco separadamente para produzir o texto cifrado, [2].

2.8.2 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 cifragem e decifragem são inseparáveis nestes sistemas estas chaves devem ser enviadas de forma bastante protegida. Com o objetivo de solucionar este problema, em 1976 Diffie e Hellman 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 2.8.

Existem dois tipos de sistemas de chave pública, a saber:

1. Os cripto-sistemas de chave pública;

2. Os sistemas com distribuição pública de chaves.

Tais sistemas diferem pouco no que se refere aos conceitos matemáticos. Neste trabalho, será abordado os sistemas de chave pública, que será descrito a seguir.

Cripto-sistemas de Chave Pública

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

$$E_k : \{M\} \rightarrow \{C\} = E_k\{M\} \quad e \tag{2.18}$$

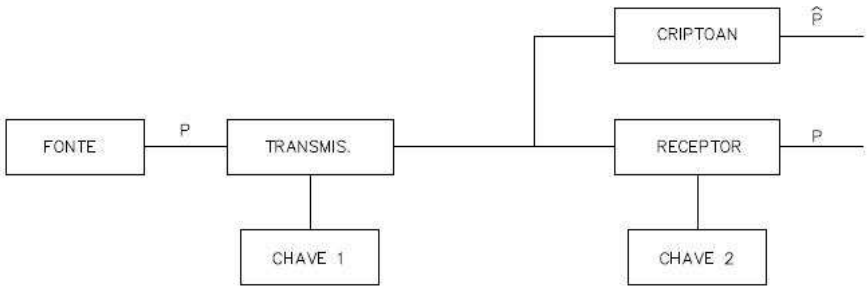

Figura 2.8: Fluxo de Informação em um sistema de chaves Públicas.

$$D_k : \{C\} \rightarrow \{M\} = D_k\{C\} \tag{2.19}$$

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, temos: $D_k E_k(M) = M$.

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.

Dessa forma, o sistema criptográfico em questão consiste essencialmente de duas partes: uma que 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 difícil.

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

Como exemplo de um cripto-sistema de chave pública temos o algoritmo RSA que será descrito a seguir.

O Algoritmo RSA

É o algoritmo de chave pública mais conhecido e foi proposto por Rivest, Shamir e Adleman, donde segue o nome RSA. É considerado um dos mais seguros e sua segurança baseia-se em um difícil problema de Teoria do Números. Para implementar o algoritmo RSA deve-se, primeiramente, gerar as chaves da seguinte maneira:

1. Escolher dois números primos muito grandes p e q;
2. Calcular $n = pq$, que será utilizado para codificar a mensagem;
3. Calcular a função $\phi(n) = (p-1)(q-1)$;
4. Escolher um número e tal que $1 < e < \phi(n)$, de forma que e e $\phi(n)$ sejam primos entre si;
5. Calcular d de forma que $d.e \equiv 1 mod(\phi(n))$.

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

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

$$C = E_{(e,N)}(M) = M^e mod(N).$$

Para o usuário A decifrar mensagem, ele utiliza o algoritmo de decifragem,

$$M = D_{(d,N)}(C) = C^d mod(N).$$

O RSA baseia-se no fato que, para encontrar dois números primos grandes, por exemplo de 100 dígitos, é computacionalmente fácil, porém, conseguir fatorar o produto de tais números é computacionalmente complexo.

2.8.3 Complexidade Computacional

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

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

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

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

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

dentre todos os NP. Esta classe de maior dificuldade de solução dentre os problemas NP são denominados NP Completos. Dentre estes problemas, encontra-se o problema da Mochila.

Podemos definir o problema da mochila da seguinte forma: 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 poderemos ilustrar um problema numérico: É possível escrever 31 como a soma de um subconjunto dos números,

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

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

$$31 = 10 + 9 + 12,$$

onde os números $[10, 9, 12]$, compõem um subconjunto do conjunto dado.

Assim, 31 pode ser representado como a soma do primeiro, terceiro e quarto números da lista.

Esse resultado também pode ser escrito como,

$$31 \rightarrow (1, 0, 1, 1, 0, 0).$$

E reciprocamente, dado $(1, 0, 1, 1, 0, 0)$ recuperamos 31.

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

O caminho inverso, isto é, a partir de f obter um subconjunto de elementos de $\{a_1\}$ cuja soma seja Y, 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.

Capítulo 3

Problema da Mochila Binário para Sistema Criptográfico de Chave Pública

Neste capítulo, será mostrado que para a determinação de códigos ótimos de memória unitária, é necessário resolver o Problema da Mochila como passo fundamental [12]. Como já foi citado, este problema pertence, no pior dos casos, à classe dos problemas não polinomiais completos (NP-completo), justificando assim a dificuldade na determinação de bons códigos convolucionais. Todavia, é desejável a sua utilização em sistemas criptográficos convencionais e também em sistemas de chaves públicas.

A primeira proposta de utilização de códigos corretores de erros em Criptografia foi feita por McEliece [15], surgindo inicialmente com o uso de códigos de bloco, mais especificamente com a utilização de códigos de Goppa. Esta proposta é baseada no fato de que estes códigos são difíceis de serem quebrados, pois sua alta eficiência de correção é destruída se as palavras do código tiverem algum embaralhamento dos bits que a compõem. Por outro lado, vários cripto-sistemas propostos são baseados em problemas bastante complexos da Teoria dos Números e Teoria Combinatorial.

A segurança do cripto-sistema de chave pública (CSCP) baseado em códigos convolucionais aqui proposto 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 diminuição no poder de correção dos erros;

2. O Problema da Mochila do tipo binário deve que ser solucionado.

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

Este capítulo, está organizado da seguinte forma: Na Seção 3.1, descrevemos o processo de determinação de códigos ótimos de memória unitária, através do algoritmo proposto por Palazzo [11]. Na Seção 3.2, apresentamos a proposta do esquema criptográfico que utiliza

códigos convolucionais de memória unitária. Na Seção 3.3, serão abordadas as propriedades das transformações armadilha para a implementação do sistema. Por fim, na Seção 3.4 será feita a análise de complexidade computacional do sistema.

3.1 Determinação de Códigos Ótimos de Memória Unitária

Em geral os códigos específicos ótimos e não-sistemáticos e os códigos ótimos de memória unitária satisfazem as propriedades de fluxo máximo e conservação de fluxo quando caracterizados como um problema de otimização combinatorial [12]. Como mencionado anteriormente, entendemos por código ótimo, um código cuja a função enumeradora dos pesos tem o menor número de palavras-código com valor do d_{free}.

Observando os conceitos de códigos de memória unitária, fluxo máximo e conservação de fluxo, estamos aptos a demonstrar a equivalência entre os problemas de determinação de bons códigos de memória unitária e o Problema da Mochila.

Considere que as conexões entre as células dos registros de deslocamento e os somadores mod q sejam arbitrárias, para uma dada taxa de transmissão $r = k/n$. Isto possibilita a geração de uma classe de códigos de memória-unitária, onde cada código pertence a esta classe pode ser representado por um diagrama de estados particionado, como mostrado na Figura 3.1.

Seja $\mathbf{d}$ 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 zero para o estado i e do estado i para o estado zero, respectivamente. Desse modo,

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

onde

$$d_i = w_{0i} + w_{i0}, \quad \text{com} \quad 1 \leq i \leq q^k - 1.$$

O problema que aqui nos defrontamos é o de encontrar cada um dos valores w_{0i} e w_{i0}. Ao descobrirmos esses valores, poderemos determinar as matrizes G_0 e G_1, respectivamente. Dessa forma, o único parâmetro que teremos que saber é a distância livre do código convolucional, dado que, em geral, $d_i = d_{free} + i - 1$, para $1 \leq i \leq q^k - 1$. Assim, os limitantes superiores da distância livre do código convolucional e de memória unitária serão muito importantes para a determinação de $\mathbf{d}$. Relembremos a equação destes limitantes superiores:

$$d_{min} \leq min_{p \geq 1}\{(q-1)(\frac{q^{k-1}}{q^k-1})\frac{n}{k}(p+km)\} \tag{3.1}$$

e

$$d_{min} \leq n(m+1)(q-1)(\frac{q^{k-1}}{q^k-1}). \tag{3.2}$$

O menor valor do limitante das equações (3.1) e (3.2) é a distância mínima irrestrita, ou seja, o d_{free}. Dado que o conjunto $\mathbf{d}$ é lexicográfico, temos que $d_1 = d_{free}$ e, em geral, $d_i = d_{free}+i-1$,

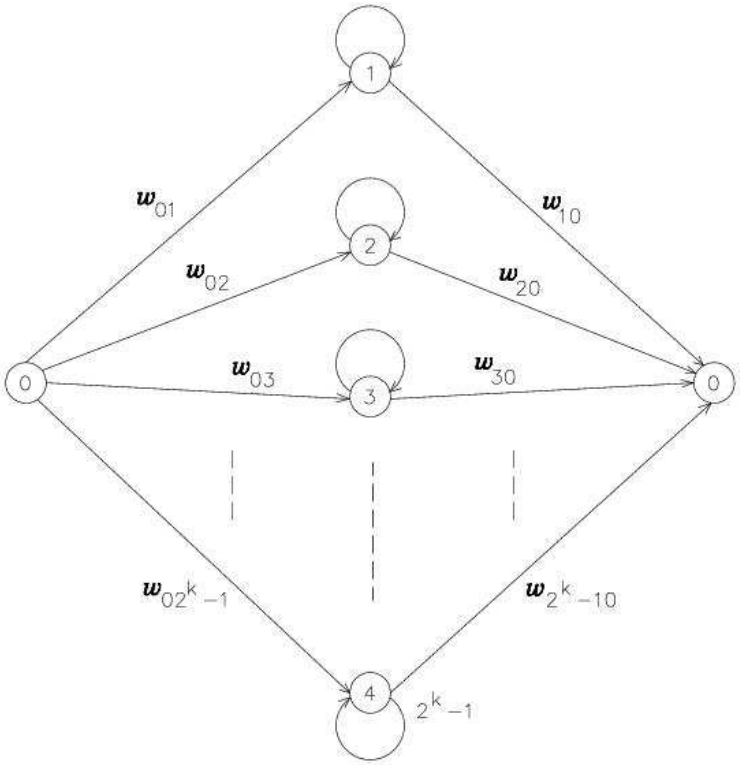

Figura 3.1: Diagrama de estados particionado de um código de memória unitária r = k/n.

para todo i tal que $1 \leq i \leq q^k - 1$, sendo que o d_{free} é obtido diretamente das equações (3.1) ou (3.2), dependendo do caso em questão.

A partir da propriedade de máximo fluxo, da equação (**??**) e da Figura 3.1, obtemos

$$\sum_{i=1}^{q^k-1} a_i d_i = (m+1)\phi, \tag{3.3}$$

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

Para um código de memória unitária com taxa $r = k/n$ fixada, os valores de máximo fluxo ϕ e do conjunto $\mathbf{d}$ são facilmente conhecidos através do emprego das equações (**??**), (3.1) e (3.2), respectivamente. Se utilizarmos uma representação vetorial para os elementos da equação (3.3), obtemos a caracterização matemática do Problema da Mochila:

$$\mathbf{a} * \mathbf{d} = (m+1)\phi. \tag{3.4}$$

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

Ao se determinar a solução da equação (3.4) através da representação modular de códigos de bloco lineares, podemos verificar se os pesos de Hamming d_i das palavras-código do código convolucional, com seus respectivos números de palavras, a_i, pertencem a uma dada matriz geradora. Se pertencerem, obtém-se as submatrizes geradoras G_0 e G_1. Portanto, resolver a equação (3.4) para códigos de memória unitária onde k assume valores grandes, equivale a determinar dentre os q^k possíveis subconjuntos de soluções aquelas que fornecerão os códigos desejados.

Essa forma de dependência exponencial com relação ao comprimento dos dados de entrada dos possíveis subconjuntos de soluções é a complexidade computacional respectiva ao melhor dos algoritmos conhecidos para resolver o Problema da Mochila. Tais conjuntos são definidos como sendo,

$$A_i = \{(w_{01}, w_{02}, w_{03}, \ldots, w_{(q^k-1)0}), (w_{10}, w_{20}, w_{30}, \ldots, w_{0(q^k-1)})\}$$

onde w_{0j} são os pesos de Hamming partindo do estado zero e chegando ao estado j, e w_{j0} são os pesos de Hamming partindo do estado j e chegando ao estado zero, onde $1 \leq i \leq q^{k-1}$.

Para explicitar os resultados anteriormente estabelecidos, apresentaremos um exemplo onde a solução do Problema da Mochila é trivial.

Exemplo 3.1.1. *Considere um código de memória unitária com taxa $r = 2/4$, sobre F_2. Tal código é equivalente a um código convolucional com memória $m = 2$ e taxa $R = 1/2$.*

Então, determinar um bom código de memória unitária com estes parâmetros é equivalente a resolver a equação,

$$a_1 d_1 + a_2 d_2 + a_3 d_3 = 16. \tag{3.5}$$

Utilizando a equação (3.1) ou (3.2), concluímos que este código tem distância livre $d_{free} = 5$ *e com isso, temos que* $d_1 = 5$, $d_2 = 6$ *e* $d_3 = 7$.

Sabendo os valores que d_i *assume e resolvendo a equação (3.5), temos que* $a_1 = 2$, $a_2 = 1$ *e* $a_3 = 0$. *Assim, os quatro conjuntos de soluções que satisfazem (3.5) são:*

- $A_1 = \{(4, 3, 1); (1, 3, 4)\}$;
- $A_2 = \{(4, 2, 2); (1, 3, 4)\}$;
- $A_3 = \{(4, 2, 2); (2, 3, 3)\}$;
- $A_4 = \{(2, 3, 3); (3, 2, 3)\}$.

Os subconjuntos A_3 *e* A_4 *produzem códigos com a mesma distância livre. Além disso, o* A_4 *é o subconjunto ótimo. Assim, uma possível solução para* G_0 *e* G_1 *é dada por*

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

3.2 Descrição do Método do Sistema Criptográfico

As funções armadilha são transformações aplicadas às submatrizes geradoras do código, G_0 e G_1, com a finalidade de reduzir o poder de correção deste código a um valor desejável ou mesmo fixado pela aplicação. Estas transformações aplicadas às submatrizes geradoras, são feitas através de um par de matrizes invertíveis A e B, sendo as dimensões destas matrizes, $k \times k$ e $n \times n$, respectivamente. Uma vez aplicadas, estas funções geram novas submatrizes geradoras, denominadas G'_0 e G'_1, dadas por:

$$G'_0 = AG_0B, \tag{3.6}$$

e

$$G'_1 = AG_1B. \tag{3.7}$$

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

Como mencionado, o processo de codificação para um código de memória unitária é definido por:

$$\tilde{v}_t = \tilde{u}_t G_0 \oplus \tilde{u}_{t-1} G_1, \quad \text{com} \quad \tilde{u}_t = 0 \quad \text{para} \quad t < 0, \tag{3.8}$$

onde $\tilde{v}_t$ é a sequência codificada na entrada do canal e $\tilde{u}_t$ a sequência de informação. Substituindo as equações (3.6) e (3.7) em (3.8), obtemos:

$$\tilde{v}_t = \tilde{u}_t G'_0 \oplus \tilde{u}_{t-1} G'_1, \quad \text{com} \quad \tilde{u}_{-1} = 0 \quad \text{para} \quad t \leq 0. \tag{3.9}$$

A equação (3.9) contém o processo de codificação e cifragem conjuntamente.

Adicionaremos agora um vetor erro ao vetor v_t. Este vetor erro corresponderá à quantidade de erro permissível ao código original, donde

$$v_{te} = v_t + v_i, \tag{3.10}$$

onde v_{te} é o vetor correspondente à mensagem codificada/cifrada com a inserção do erro, v_t é a mensagem cifrada/codificada e v_i é a quantidade de erros inseridos na mensagem no seu ponto de envio.

O vetor v_i origina um vetor de 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.

Apresentamos, na Figura 3.2, o sistema de chave pública considerado em forma de diagrama de blocos. Como o sistema criptográfico é de chave pública, as matrizes

$$G' = [G'_0 : G'_1] \tag{3.11}$$

são colocadas em uma lista pública, podendo também ser divulgado o número de erros que será adicionado à mensagem.

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

No processo de decodificação empregado à sequência de saída do canal, $\hat{v}_{te}$, considerando-se que o canal seja livre de ruídos, temos que $\hat{v}_{te}$ (valor estimado de v_{te}) é igual ao valor de v_{te} na entrada do canal. Utilizando-se então a transformação inversa de B, B^{-1}, como explicitado na equação (3.12), temos que

$$v_{te}.B^{-1} = (u_t A)G_0 \oplus (u_{t-1}.A)G_1. \tag{3.12}$$

Aplicando-se o algoritmo de Viterbi, obtém-se u_t. Para este cripto-sistema a matriz B é a 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.

Observe que para quebrar este cripto-sistema, é necessário que o criptoanalista primeiramente resolva o Problema da Mochila, relacionado a encontrar as submatrizes geradoras do código ótimo, G_0 e G_1, e depois encontrar as transformações A^{-1} e B^{-1}. Porém, estes fatos não apresentam soluções triviais.

Para ilustrar o procedimento de cifragem, considere o seguinte exemplo:

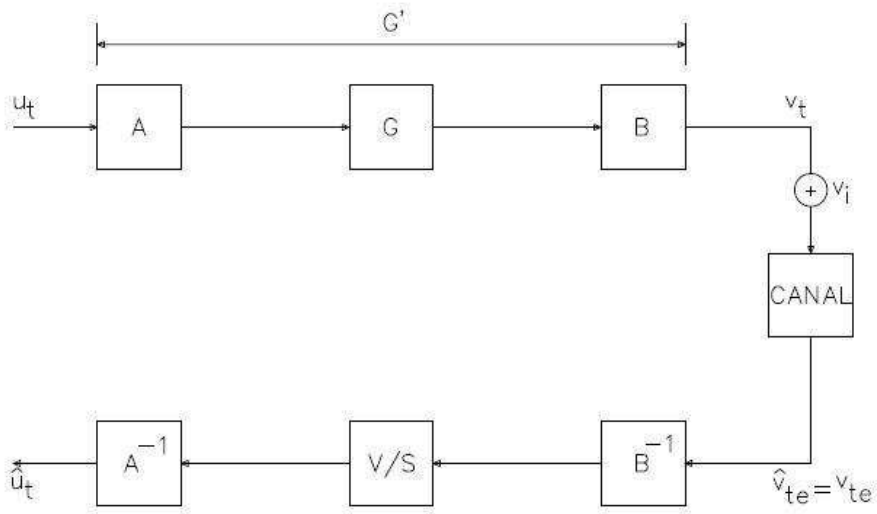

Figura 3.2: diagrama em blocos do cripto-sistema de chave pública.

Exemplo 3.2.1. *Considere um código ótimo de memória unitária com taxa* $r = 2/4$. *Este código pode ser encontrado pela resolução do Problema da Mochila. Da equação (3.4) temos que* $d_1 = 5$, $d_2 = 6$ *e* $d_3 = 7$. *Esta solução gera um código de memória unitária cujas submatrizes geradoras são dadas por:*

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

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

Sejam as transformações A e B dadas por:

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

e suas matrizes inversas, respectivamente, dadas por:

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

Aplicando A e B em G_0 *e* G_1 *como em (3.6) e (3.7), obtemos* G'_0 *e* G'_1*:*

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

A 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.*

Agora, suponha que se queira transmitir a mensagem $u = (10, 01, 11, 00)$. *Considere o codificador inicialmente resetado. Assim, a palavra a ser transmitida será*

$$v_t = (0111, 1101, 1001, 0011).$$

Se adicionarmos o vetor erro de transmissão $v_i = (1000, 0000, 0000, 0000)$, *a palavra que percorrerá o canal será:*

$$v_{te} = (1111, 1101, 1001, 0011).$$

Aplicamos agora ao vetor v_{te} *a transformação* B^{-1} *como mostra a equação (3.12) e obtemos*

$$v_{te} = (1110, 0011, 1111, 0111),$$

como mostra a equação (3.12). Em seguida, aplicamos o processo de decodificação utilizando-se o algoritmo de Viterbi, obtendo a sequência de informação $u = (10, 01, 11, 00)$.

3.3 Propriedades das Transformações Armadilha

Nesta seção serão estabelecidas as *propriedades das transformações armadilha* com relação às condições de otimalidade dos códigos empregados nos sistemas criptográficos.

Sejam G_i, $0 \leq i \leq m$, as submatrizes geradoras com dimensão $k \times n$ de um código convolucional, podendo este 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 3.3.1. *[13] 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*

$$\tilde{w}_H = \frac{\phi(G)}{2^k - 1}. \tag{3.13}$$

Se $2^k - 1$ divide $\phi(G)$, então $[\tilde{w}] = \tilde{w}$, ou seja, $\tilde{w}$ é um número inteiro. Caso contrário, $[\tilde{w}] = \{\tilde{w},\ |\tilde{w}| \pm j \quad \text{onde} \quad j \geq 0\}$. 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 3.3.1.

Lema 3.3.1. *[13] Se o posto*$(G) = 2k$ *e* $2k \leq n$, *então* G *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:

$$\frac{n!}{(n-[\tilde{w}])!.[\tilde{w}]!}[\tilde{w}]! \tag{3.14}$$

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

$$n \geq n_0; \quad n!(n-[\tilde{w}])!.[\tilde{w}] \geq 2k, \tag{3.15}$$

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 gerada seja constituída por vetores pesos de Hamming igualmente distribuídos, onde a distância mínima é igual àquela do código de bloco $(k, 2n)$. Portanto, obtemos os $2k$ vetores linha (linearmente independentes) de G, de tal forma que G gera um espaço vetorial com 2^{2k} vetores com peso de Hamming igualmente distribuídos. □

Uma consequência imediata deste lema é que o código não é do tipo catastrófico. Entretanto, se temos $2k > n$, 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 3.3.1 supõe-se quem o código utilizado seja ótimo.

Existem necessariamente duas condições. A primeira delas está relacionada com a aplicação da função armadilha a G_1, resultando num novo código com mesma taxa e número de memórias. Em outras palavras, obtendo-se um código ótimo e supondo que as funções 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 3.3.1. *[13] Sejam 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,*

$$dim\{A[G_0, G_1, \ldots, G_m]B\} \leq dim\{[G_0, G_1, \ldots, G_m]\},$$

onde A e B não são necessariamente matrizes unitárias.

Demonstração. Partimos da hipótese que $G = [G_0, G_1, \ldots, G_m]$ 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}. Tem-se, 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}.

Sabemos 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.

Sabemos 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 são determinantes 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 3.3.1 e que as funções 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 3.3.2. *[13] 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, ou seja,*

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

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

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

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 3.3.1, temos que $d_{min}(k,n) \leq d_{min}(k,2n)$ e que se $q > n$, donde $d_{min}(k,2q) \geq d_{min}(k,2n)$. Assim, $d_{free}(k/n)$. Note que a igualdade é válida se G e G' são equivalentes onde $det(A) = 1$ e $posto(B^+) = n$.

□

De forma análoga à Proposição 3.3.2, podemos 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 $\tilde{r} = p/q$. Disso segue que

Proposição 3.3.3. *[13] 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,*

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

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 dimensão $p \times q$, tal que

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

e

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

Demonstração. Segue da aplicação do Lema 3.3.1, da demonstração da Proposição 3.3.2 e do limitante superior da distância mínima dado por

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

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

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. Essas propriedades estão relacionadas ao aspecto de segurança desses cripto-sistemas quanto à sua complexidade computacional a ser visto posteriormente.

A seguir, calculamos a complexidade do modelo baseado no tempo de processamento para a quebra do sistema e serão feitas avaliações dos graus de dificuldade imposto ao criptoanalista.

3.4 Complexidade Computacional do Cripto-sistema

A complexidade deste cripto-sistema é obtida através da complexidade referente às principais etapas. A análise de complexidade parte da hipótese do pior caso, consideramos que o criptoanalista terá que realizar todos os passos necessários para chegar a uma possível solução.

Podemos dividir o problema em três etapas de complexidade:

1. A complexidade de resolver o Problema da Mochila, encontrando o código ótimo, cuja complexidade é um NP-completo;

2. A complexidade para encontrar a inversa das matrizes das transformações armadilha, A e B;

3. A complexidade referente ao tempo de processamento para a decodificação da mensagem, levando-se em consideração que o criptoanalista tenha posse das inversas das matrizes das transformações armadilha, A e B.

Observe que a complexidade do cripto-sistema não diminui quando tentamos encontrar as inversas das matrizes G'_0 e G'_1. Note que ao tentarmos encontrar a inversa da matriz geradora do código convolucional, temos que procurar a inversa da matriz semi-infinita. Pelo fato desta matriz não ser quadrada, pode não existir uma única matriz inversa de G', e a complexidade para resolver este problema torna-se equivalente a resolver os três passos citados acima, quando utilizamos k muito grande.

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

3.4.1 Complexidade do NP-Completo

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

$$\phi = n.q^{k-1}.(q-1), \tag{3.16}$$

e

$$d = \min\left\{\left[\frac{2^{(p-1)}}{(2^p-1)}\right].\frac{n}{k}.(p+k)\right\}, \tag{3.17}$$

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

$$2.\phi = a_0.d_0 + a_1.d_1 + \ldots + a_{(2^k-1)}.d_{(2^k-1)}. \tag{3.18}$$

$$\prod_{i=0}^{2^k-1}\left[\frac{2\phi}{d_i}\right] \cong \left[\frac{2\phi}{d_0}\right]^{2^k} \tag{3.19}$$

3.4.2 Complexidade da Inversão das Transformações Armadilha

Pelo processo de triangularização de matrizes, tem-se pelo menos:

$$2^{n(n-1)/2} \tag{3.20}$$

matrizes invertíveis.

Se o algoritmo mais eficiente para o cálculo da matriz inversa tem ordem de complexidade $nlogn$, então a complexidade envolvida nesta etapa será:

1. Caso as transformações armadilha A e B sejam da mesma ordem, temos:
$$2^{k(k-1)/2}k\log k. \tag{3.21}$$
2. Caso a ordem da transformação armadilha B seja muito maior que a ordem da transformação armadilha A, temos:
$$2^{n(n-1)/2}n\log n. \tag{3.22}$$

3.4.3 Complexidade do Algoritmo de Decodificação

Se o algoritmo utilizado no processo de decodificação for o algoritmo de Viterbi, então em cada estado são feitas 2^k comparações. Suponha que a quantidade de transições consideradas na treliça seja 4 e considere Δ como sendo o tempo gasto para cada comparação. E que o número total de comparações por transição de estado na treliça é 2^{2k}. A complexidade do algoritmo de decodificação é então:

$$4.\Delta.2^{2k}. \tag{3.23}$$

3.4.4 Complexidade Total

A complexidade total C do sistema é dada pela composição das equações (3.18), (3.21) ou (3.22) e (3.23). Logo, quando as transformações armadilha A e B apresentam a mesma ordem de grandeza, ou seja, $k \times n$, C é dado por

$$C = [2\phi/d]^{2^k-1}.[2^{k(k-1)/2}k\log k].4\Delta 2^{2k} \cong \Delta 2^{(2k+k2^k+k^2)}k\log k. \tag{3.24}$$

Note que:

- $[2\phi/d]$ é da ordem de 2^k, pois,
$$2\phi = n.2^{(k-1)} \quad e \quad d \cong [2^{(p-1)/2^p-1}](n/k)(p+k);$$
- $2.2^{k(k-1)/2}k\log k$ é da ordem de $2^{k^2}k\log k$;
- o termo $4\Delta 2^{2k}$ é da ordem de $\Delta 2^{2k.}$.

Quando a ordem de B é bem maior que a ordem de A, de forma análoga à equação (3.24), temos que C é da ordem de:

$$C \cong \Delta 2^{(2k+k2^k+(k/r)^2)}(k/r)\log(k/r). \tag{3.25}$$

Então, qualquer que seja o caso considerado a complexidade total do cripto-sistema proposto varia exponencialmente com o comprimento da entrada do sistema.

Capítulo 4

Transformações Armadilha

Neste capítulo apresentamos os resultados obtidos no processo de busca por transformações armadilhas a serem usadas no sistema criptográfico de chave pública.

Os códigos escolhidos são todos convolucionais ótimos de memória-unitária e foram utilizados aqueles com uma pequena complexidade para que pudéssemos observar o comportamento do cripto-sistema quanto à sua vulnerabilidade.

Antes da análise, as transformações armadilha a serem usadas serão definidas e comentadas com detalhes. Cada uma delas será aplicada em códigos específicos com diferentes taxas que serão citados e definidos quando necessário. A análise do comportamento desses códigos ocasionado pela aplicação das transformações armadilha A e B é apresentada de forma qualitativa, onde buscamos enfatizar os critérios adotados para as conclusões aqui apresentadas.

Este capítulo está organizado da seguinte maneira: Na Seção 4.1 são definidas as transformações armadilha utilizadas no desenvolvimento do trabalho. Na Seção 4.2 são apresentados os resultados obtidos com a aplicação das transformações armadilha aos códigos ótimos de memória-unitária. Na Seção 4.3 é feita uma comparação entre as transformações armadilha, onde analisamos o seu desempenho e observamos qual função armadilha é a que conduz ao melhor resultado. Finalmente, na seção 4.4 aplicamos a função armadilha de permutação em dois códigos com taxa $r = 3/6$, sendo um de memória-unitária (MU) e outro de memória-unitária parcial (MUP) e fazemos uma comparação entre os resultados obtidos.

4.1 Busca por Transformações Armadilha

No processo de busca por transformações que fossem capazes de diminuir a capacidade de correção de erros dos códigos convolucionais, encontramos muitas que, quando aplicadas às submatrizes G_0 e G_1 são capazes de reduzir o d_{free} a níveis desejados. Porém, observamos que estas matrizes não possuem uma lei de formação que as caracterizassem.

Devido a esse fato, passamos a analisar o efeito que algumas matrizes, com estruturas definidas, causam ao serem aplicadas em cada um dos códigos convolucionais que serão definidos

em cada caso separadamente.

Neste trabalho, utilizaremos três tipos de transformações, a saber:

- As representações matriciais dos elementos do *grupo de permutações*;
- As *matrizes triangulares superiores*, por possuírem uma estrutura de grupo, e
- As *matrizes de Hadamard*;

A seguir, apresentaremos e definiremos cada uma delas.

4.1.1 Grupo de permutações - S_n

Antes de definirmos o grupo de permutação, será necessário introduzir alguns conceitos fundamentais da teoria de grupos que são importantes para a teoria da codificação.

Definição 4.1.1 (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. Esta condição de que o elemento associado deve estar em S é conhecida como a condição de fechamento, ou seja, S deve ser fechado sob a operação binária.

Definição 4.1.2 (Comutatividade)**.** *Uma operação binária sobre um conjunto S é comutativa se $a * b = b * a \ \forall \ a, b \ \in \ S$.*

Definição 4.1.3 (Associatividade)**.** *Uma operação binária sobre um conjunto S é associativa se $(a * b) * c = a * (b * c)$ para quaisquer que sejam a,b e c $\in$ S.*

Após a apresentação destas definições, estamos aptos a conhecer a definição de grupo que é de grande importância para a compreensão da estrutura algébrica das transformações armadilha utilizadas neste trabalho.

Definição 4.1.4 (Grupo)**.** *Um grupo $\langle G, * \rangle$ é um conjunto G, junto com uma operação binária $*$ sobre G, tal que as seguintes propriedades são satisfeitas:*

1. *A operação binária é associativa;*
2. *Existe um elemento e em G tal que $e * g = g * e = g \ \forall \ g \in G$ (esse elemento e é um elemento identidade para $*$ sobre G);*
3. *Para cada g em G, existe um elemento g' em G com a propriedade de que $g' * g = g * g' = e$ (o elemento g' é um inverso de g com respeito à operação $*$).*

Quando a lei de composição considerada for uma "adição" diremos que o grupo em questão é um *grupo aditivo* ao passo que se a lei de formação for "multiplicação", nos referimos a ele como *grupo multiplicativo.*

Teorema 4.1.1. *Em todo grupo G, o elemento identidade é único. O inverso de cada elemento de G também é único.*

Veremos agora alguns exemplos de grupo:

Exemplo 4.1.1 (Grupo)**.**

1. **Grupo aditivo** $\mathbb{Z}_n$ *:Para qualquer inteiro positivo n, o conjunto $\mathbb{Z}_n$ com a operação de soma módulo n é um grupo aditivo.*

2. **Grupo aditivo de matrizes***: $M_{m\times n}(\mathbb{Z})$, o conjunto de todas as matrizes sobre $\mathbb{Z}$ é um grupo abeliano em relação à adição.*

3. **Grupos lineares de grau** n *: Indicaremos por $M_n(\mathbb{Q})$ o conjunto das matrizes sobre $\mathbb{Q}$ de ordem n, onde $\mathbb{Q}$ denota o conjunto dos racionais. Seja $GL(n,\mathbb{Q})$ o subconjunto das matrizes de $M_n(\mathbb{Q})$ que têm determinante não nulo, ou seja,*

$$GL(n,\mathbb{Q}) = \{A \in M_n(\mathbb{Q}) | det(A) \neq 0\}.$$

$GL(n,\mathbb{Q})$ é um grupo multiplicativo e é chamado grupo linear racional de grau n

Definição 4.1.5 (Subgrupo)**.** *Seja $(G,*)$ um grupo. Dizemos que um subconjunto $H \subset G$ é um subgrupo de G se, e somente se,*

1. *Para todo $a, b \in H \Rightarrow a * b \in H$, ou seja, H é fechado sob a operação binária de G;*

2. *$(H,*)$ também é um grupo, onde a operação binária é a mesma de G, porém restrita a H.*

Definição 4.1.6 (Permutações)**.** *Seja o conjunto $X = 1, 2, \ldots, n$. Uma permutação é uma bijeção*

$$\rho : X \to X.$$

Definição 4.1.7 (r-ciclo)**.** *Uma permutação $\rho \in S_n$ é chamada r-ciclo se existem elementos distintos $a_1, \ldots, a_r \in X = \{1, 2, \ldots, n\}$ tais que $\rho(a_1) = a_2$, $\rho(a_2) = a_3$, $\ldots$,$\rho(a_{r-1}) = a_r$, $\rho(a_r) = a_1$, e tais que $\rho(j) = j \ \ \forall \ \ j \in \{1, 2, \ldots, n\} \ \setminus \{a_1, \ldots, a_r\}$.*

Tal r-ciclo será denotado por $(a_1 \ldots a_r)$ onde r é o *comprimento do ciclo.* Os ciclos de comprimento 2 são também chamados *transposições.*

Exemplo 4.1.2 (r-ciclo)**.** *No grupo S_5,*

- $\begin{pmatrix} 12345 \\ 23451 \end{pmatrix}$ *é um* 5*-ciclo, denotado por* (12345)*; poderia também ser denotado por* (23451)*, ou* (34512)*, ou* (45123)*, ou* (51234)*.*
- $\begin{pmatrix} 12345 \\ 42135 \end{pmatrix}$ *é um* 3*-ciclo, denotado por* (143)*; poderia também ser denotado por* (431)*, ou* (314)*.*
- $\begin{pmatrix} 12345 \\ 42135 \end{pmatrix}$ *é um uma transposição, denotado por* (13) *e poderia ser também denotada por* (31)*.*
- $\begin{pmatrix} 12345 \\ 34521 \end{pmatrix}$ *não é um* r*-ciclo, qualquer que seja* r*.*

Definição 4.1.8 (Grupo de Permutações)**.** *O conjunto de todas as permutações forma um grupo sob composição, o qual é denotado por* S_n *(o grupo de permutações de* n *elementos). Existem* $n!$ *permutações de* X*.*

Como exemplo de grupo de permutação, veremos S_3, grupo de permutação de 3 elementos. Existem $3! = 6$ permutações.

Exemplo 4.1.3 (O Grupo S_3)**.** *Considere o conjunto* S_3 *dado por*

$$S_3 = \{id, (12), (13), (23), (123), (132)\},$$

onde a notação (abc) *representa a função definida por* $\mathrm{f}(a) = b$, $\mathrm{f}(b) = c$ *e* $\mathrm{f}(c) = a$*.*
Sejam $\alpha = (123)$ *e* $\beta = (12)$*; temos*
$\alpha^2 = (123)(123) = (132)$,
$\alpha^3 = (132)(123) = id$,
$\beta^2 = (12)(12) = id$,
$\beta\alpha = (12)(123) = (23)$,
$\alpha\beta = (123)(12) = (13)$,
$\alpha^2\beta = (132)(12) = (23)$.

Observe que foi verificado acima que α *e* β *geram o grupo* S_3*, isto é, todos os elementos do grupo são produtos finitos de fatores iguais a* α *ou* β*; foi também verificado que* $\alpha^3 = id$*,* $\beta^2 = id$ *e* $\beta\alpha = \alpha^2\beta \neq \alpha\beta$*.*

A seguir, apresentaremos um dos principais teoremas da teoria de grupos, o *Teorema de Cayley.*

Teorema 4.1.2 (Teorema de Cayley)**.** *Todo grupo é isomorfo um grupo de permutação de seus elementos.*

Existe uma grande variedade de grupos quanto a natureza de seus elementos. Apesar disso, o Teorema de Cayley afirma que cada grupo é isomorfo a algum grupo de permutação. Essa afirmação é também conhecida como Teorema da Representação: cada grupo pode ser representado concretamente como um grupo de permutações.

Uma consequência importante do Teorema de Cayley é que os grupos lineares gerais $GL(n, \mathbb{F})$, tem as mesmas propriedades que os grupos simétricos. Portanto, podemos representar grupos ou como subgrupos de $GL(n, \mathbb{F})$.

Teorema 4.1.3. *Seja $\mathbb{F}$ um corpo e n um inteiro positivo. Para toda permutação $\rho \in S_n$ vamos definir a matriz $n \times n$ dada por,*

$$A_\rho = (a_{ij}) = \begin{cases} 1 & se \quad \rho(i) = j \\ 0 & caso \quad contrário \end{cases}$$

A função $f : S_n \longrightarrow GL(n, \mathbb{F})$, definida por

$$f(\rho) = A_\rho$$

é um homomorfismo injetivo. Portanto, todo grupo de ordem n é isomorfo a um subgrupo de $GL(n, \mathbb{F})$.

Apresentamos, a seguir, um exemplo para melhor entendermos o Teorema 4.1.3.

Exemplo 4.1.4. *Considere o grupo de permutação S_3 do Exemplo 4.1.3 e M o subgrupo de $GL(n, \mathbb{F})$, composto pelas matrizes de dimensão 3×3. Sejam,*

$$\alpha \cong M_\alpha = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad e \quad \beta \cong M_\beta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Fazendo a associação entre os elementos do grupo temos:

- $\alpha^2 \cong M_{\alpha^2} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cong (132)$,
- $\alpha^3 \cong M_{\alpha^3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cong id$,
- $\beta^2 \cong M_{\beta^2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cong id$
- $\beta\alpha \cong M_{\alpha\beta} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cong (13)$

- $\alpha\beta \cong M_{\beta\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cong (23)$
- $\alpha^2\beta \cong M_{\alpha^2\beta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cong (23)$

4.1.2 Matriz de Hadamard

Outra transformação armadilha utilizada em nosso trabalho é a matriz de Hadamard, definida como:

Definição 4.1.9 (Matriz de Hadamard)**.** *É uma matriz quadrada H de ordem $n \times n$ composta de $+1$s e -1s tal que*

$$HH^T = nI.$$

Em outras palavras, o produto interno de duas linhas distintas de H é zero, ou seja, linhas distintas são ortogonais e o produto interno de qualquer linha com ela mesma é n. Dado que $H^{-1} = (1/n)H^T$, temos também que $H^TH = nI$, e deste modo as colunas tem as mesmas propriedades.

A seguir apresentaremos algumas matrizes de Hadamard de ordem, 1, 2, 4 e 8 do tipo *Sylvester* onde por convenção, ao invés de -1 escreveremos $-$.

$\bullet n = 1 \rightarrow H_1 = [1]$ $\qquad$ $\bullet\, n = 2 \rightarrow H_2 = \begin{bmatrix} 1 & 1 \\ 1 & - \end{bmatrix}$

$$\bullet n = 4 \rightarrow H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & - & 1 & - \\ 1 & 1 & - & - \\ 1 & - & - & 1 \end{bmatrix} \qquad \bullet\, n = 8 \rightarrow H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \end{bmatrix}$$

Agora, se trocarmos $+1$ por 0 e -1 por 1 em H_n conduz a uma matriz de *Hadamard binária* de ordem n e, é nesta forma que utilizaremos ao longo deste capítulo.

4.1.3 Matrizes triangulares superiores

Definição 4.1.10 (Matrizes Triangulares Superiores)**.** *Uma matriz M é dita ser triangular superior se todos os elementos abaixo da diagonal principal forem nulos, ou seja, se $a_{ij} = 0$ sempre que $i > j$.*

Proposição 4.1.1. *As matrizes triangulares superiores possuem as seguintes propriedades:*

1. *A matriz identidade é uma matriz triangular superior;*
2. *O produto de duas matrizes triangulares superiores resulta em outra matriz triangular superior;*
3. *O determinante de uma matriz triangular superior é o produto dos elementos da sua diagonal. Assim, uma matriz triangular superior é invertível se, e somente se, não tiver zeros na sua diagonal principal;*
4. *Se uma matriz triangular superior é invertível, sua inversa é também uma matriz triangular superior.*

As matrizes triangulares superiores que utilizamos neste trabalho têm como entradas elementos de F_2. Além disso, procuramos matrizes que possuam sua diagonal composta apenas por 1s. Assim, por suas propriedades, o determinante é igual a 1 e consequentemente, a invertibilidade está garantida.

4.2 Análise das Transformações Armadilha

Nesta seção, aplicamos as transformações armadilha que definimos na Seção (4.1), aos códigos convolucionais ótimos de memória-unitária. Em cada caso, especificamos os códigos a serem utilizados, apresentando as submatrizes geradoras, suas taxas e os seus respectivos $d'_{free}s$.

Após aplicarmos as transformações armadilha a cada código, apresentamos e comentamos os resultados obtidos, e depois realizaremos uma comparação entre os resultados.

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 (:). Como exemplo, considere a seguinte matriz:

$$G_0 = [13 : 06 : 03].$$

Sua representação octal é dada por,

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

4.2.1 Permutações

As representações matriciais dos elementos do grupo de permutação foram aplicadas como transformações armadilha aos códigos convolucionais ótimos de memória-unitária com taxas $r = 2/3$, $r = 2/4$ e $r = 3/4$.

O teorema 4.2.1 facilita compreensão dos resultados obtidos.

Teorema 4.2.1. *[26] Seja C um código convolucional (n, k, m), isto é, com k entradas e m memórias, gerado pela matriz*

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

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

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

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.

O primeiro procedimento foi aplicar as transformações armadilha em cada submatriz separadamente, isto é,

$$G'_0 = AG_0B \quad \text{e} \quad G'_1 = AG_1B.$$

Para o código com taxa $r = 2/3$, as transformações armadilha A são dadas pelas seguintes matrizes,

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad A_3 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad e \quad A_4 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{4.1}$$

Para as transformações armadilha B, utilizamos as representações matriciais dos elementos do grupo de permutação S_3.

Para o código com taxa $r = 2/4$, as transformações armadilha A são as mesmas utilizadas no código com taxa $r = 2/3$. Já as transformações armadilha B são as representações matriciais dos elementos do grupo de permutação S_4.

Com o objetivo de observar melhor o efeito causado pela aplicação da transformação armadilha A aos códigos convolucionais, para o código com taxa $r = 3/4$, estas transformações são constituídas das representações matriciais dos elementos do grupo de permutação S_3. E como transformação armadilha B, utilizamos as representações matriciais dos elementos do grupo S_4.

Efetuando essas transformações sobre as submatrizes geradoras dos códigos em consideração, observamos que não houve mudança alguma no d_{free} dos códigos e que suas características foram preservadas. Isso ocorre porque os códigos obtidos são equivalentes ao código original. Tais resultados são esclarecidos no Teorema 4.2.1.

Como os códigos obtidos não perderam a capacidade de correção de erros, mantendo o d_{free} igual ao do código original, concluímos que utilizadas desta forma as transformações armadilha de permutação não satisfazem aos critérios necessários para o modelo criptográfico apresentado.

Com o intuito de resolver o problema de obter códigos equivalentes, passamos a aplicar a função armadilha na matriz geradora G, dada por

$$G = [G_0 : G_1],$$

ao invés de aplicarmos nas submatrizes separadamente. Assim,

$$G' = A * G * B,$$

onde A e B são as transformações armadilha de ordem $k \times k$ (k = número de entradas) e $(m+1)n \times (m+1)n$ (m = memórias e n = número de saídas), respectivamente. A partir de G', obtemos novamente as submatrizes G'_0 e G'_1 separadas. A seguir, apresentamos um exemplo para ilustrar o que foi dito.

Exemplo 4.2.1. *Considere um código convolucional ótimo de memória unitária e taxa* $r = k/n = 2/4$. *Sejam* G_0 *e* G_1 *as submatrizes geradoras dadas por*

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

Seja

$$G = [G_0 : G_1] = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Sejam as transformações armadilha A *e* B, *a matriz identidade de ordem* 2×2 *e a matriz* 8×8 *dada pela representação matricial da permutação* $\rho = (512)(736)(84)$.

Aplicando essas transformações na matriz geradora temos

$$G' = A * G * B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$G' = A * G * B = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = [G'_0 : G'_1]$, *onde* G'_0 *e* G'_1 *são as novas submatrizes geradoras dadas por*

$$G'_0 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad e \qquad G'_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Procedendo desta forma, observamos que determinadas transformações são capazes de reduzir o valor da distância livre dos códigos considerados a níveis desejados. A seguir, apresentamos os resultados obtidos para cada código separadamente.

Código taxa r = 2/3

Considere o código convolucional $(3, 2, 1)$, cujas submatrizes geradoras são dadas por,

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

e $d_{free} = 3$. Como transformações armadilha A, foram utilizadas as matrizes A_1, A_2, A_3 e A_4, definidas na página 48. As transformações armadilha B, são as representações matriciais dos elementos do grupo de permutação S_6.

Diante dos resultados obtidos, observamos que na maioria dos casos a distância livre permaneceu a mesma. Porém, foram encontradas transformações armadilha capazes de diminuir a capacidade de correção do código, resultando em códigos capazes de detectar erros. Em algumas situações foram produzidos códigos catastróficos. Ocorreram situações em que o código resultante era um código de memória unitária parcial. Estes fatos podem ser observados na Tabela 4.1 e no exemplo a seguir.

Exemplo 4.2.2. *Seja B a representação matricial da permutação $\rho = (62)(34)$ e seja A a matriz identidade 2×2. Então, aplicando estas transformações na matriz G, obtemos as submatrizes geradoras*

$$G'_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad e \qquad G'_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Como pode ser verificado, esse novo código possui o $d_{free} = 1$ e é um código de memória unitária parcial.

Na Tabela 4.1, apresentamos a transformação armadilha A, a permutação associada à transformação armadilha B, representada por ρ, as submatrizes geradoras do código convolucional de memória unitária resultante, G'_0 e G'_1, e a distância livre do novo código, denotada por d'_{free}.

A	ρ	G'_0	G'_1	d'_{free}
02:01	(216543)	06:07	02:05	3
	(316425)	05:07	01:06	3
	(51)(6243)	03:05	02:07	3
	(6125)(34)	01:06	03:07	3
	(215634)	03:06	02:07	catastrófico
	(31)(42)	03:05	01:07	catastrófico
	(2156)(34)	02:07	06:03	3
	(3142)	05:03	01:07	catastrófico
	(26)(34)	07:06	00:07	1

Tabela 4.1: d_{free} dos códigos resultantes com taxa $r = 2/3$.

Código taxa r = 2/4

O código ótimo de memória unitária, com taxa $r = 2/4$ aqui considerado possui as submatrizes geradoras

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

e possui $d_{free} = 5$.

As transformações armadilha A aplicadas foram as mesmas utilizadas para o código com taxa $r = 2/3$, e as transformações armadilha B foram as representações matriciais dos elementos do grupo S_8.

Na Tabela 4.2, apresentamos a transformação armadilha A, a permutação associada à transformação armadilha B, representada por ρ, as submatrizes geradoras do código convolucional de memória unitária resultante, denotadas por G'_0 e G'_1, e o d'_{free}, ou seja, a distância livre do novo código.

Com a aplicação das transformações armadilha de permutação foram obtidos cinco tipos de resultados diferentes:

1. O d_{free} continuou o mesmo, mantendo a capacidade de correção de erros. Os códigos obtidos são equivalentes ao código original;

2. O d_{free} foi reduzido de 5 para 4. Com isso, passou a corrigir 1 erro;

3. Houve uma redução maior do d_{free}, ou seja, indo de 5 para 3. Assim, ao invés de corrigir até 2 erros, o novo código é capaz de corrigir apenas 1 erro;

4. A distância livre do código diminuiu de 5 para 2;

5. O código resultante é um código catastrófico.

Estes resultados podem ser observados nas Tabelas 4.2 e 4.3.

Outro fato observado é que a transformação A não interfere no valor do d_{free} dos códigos resultantes. tal fato é ilustrado na Tabela 4.2, onde foram aplicadas as mesmas transformações B para cada transformação A, e em todos os casos, independente de qual seja esta matriz, a distância livre continuou sendo a mesma. Percebe-se que a função da transformação A é apenas o de embaralhar os bits de informação sem, entretanto, alterar os elementos das submatrizes.

Código taxa R=3/4

Com o objetivo de analisar o efeito causado pela transformação A com $k > 2$, trabalhamos com o código convolucional ótimo de memória unitária e taxa $R = 3/4$. Tal código é gerado pelas submatrizes

A	ρ	G'_0	G'_1	d'_{free}
02:01	(3 1 2)(6 4 5)(7 8)	13:04	14:17	3
	(8 1 6 4 7)(5 2 3)	12:15	15:06	4
	(2 1 4 5 7 3 6 8)	17:12	01:16	3
	(5 1 8 4 3 2 6)	06:15	15:03	4
01:02	(3 1 2)(6 4 5)(7 8)	04:13	17:14	3
	(8 1 6 4 7)(5 2 3)	15:12	06:15	4
	(2 1 4 5 7 3 6 8)	12:17	16:01	3
	(5 1 8 4 3 2 6)	15:06	03:15	4
03:01	(3 1 2)(6 4 5)(7 8)	17:13	03:14	3
	(8 1 6 4 7)(5 2 3)	07:12	13:15	4
	(2 1 4 5 7 3 6 8)	05:17	17:11	3
	(5 1 8 4 3 2 6)	13:06	16:15	4
02:03	(3 1 2)(6 4 5)(7 8)	13:17	14:03	3
	(8 1 6 4 7)(5 2 3)	12:07	15:13	4
	(2 1 4 5 7 3 6 8)	17:05	01:17	3
	(5 1 8 4 3 2 6)	06:13	15:16	4

Tabela 4.2: d_{free} dos códigos resultantes com taxa $r = 2/4$.

A	ρ	G'_0	G'_1	d'_{free}
02:01	(51368)(742)	03:16	13:06	catastrófico
	(214)(635)	17:10	10:17	catastrófico
	(712368)(45)	13:06	12:15	5
	(216743)	16:11	12:07	5
	(81375)(24)	05:16	07:14	catastrófico
	(21847)(65)	04:13	17:11	3
	(41536)(78)	17:02	04:17	2

Tabela 4.3: Outros códigos com taxa $r = 2/4$ resultantes da aplicação da transformação permutação.

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

e possui distância livre igual a $d_{free} = 4$.

As transformações armadilha A e B utilizadas, foram as representações dos elementos dos grupos de permutação S_3 e S_8, respectivamente.

Nas Tabelas 4.4 e 4.5, apresentamos alguns elementos do grupo de permutação S_8, denotado ρ_B, cuja representação matricial é a transformação B. As transformações B são combinadas com todos os elementos do grupo S_3, representados por ρ_A. Nestas tabelas, também são dadas G'_0 e G'_1, as submatrizes resultantes após a aplicação de tais transformações, e a distância livre do novo código convolucional, d'_{free}.

ρ_A	ρ_B	G'_0	G'_1	d'_{free}
id	(7 1 8 5)(3 2 6 4)	06:16:15	14:02:05	2
	(4 1 2 5 8 6 3)	11:01:06	12:15:13	3
	(2 1 8)(7 3 6)	06:13:03	03:01:16	3
(2 1)	(7 1 8 5)(3 2 6 4)	16:06:15	02:14:05	2
	(4 1 2 5 8 6 3)	01:11:06	15:12:13	3
	(2 1 8)(7 3 6)	13:06:03	01:03:16	3
(3 2)	(7 1 8 5)(3 2 6 4)	06:15:16	14:05:02	2
	(4 1 2 5 8 6 3)	11:06:01	12:13:15	3
	(2 1 8)(7 3 6)	06:03:13	03:16:01	3
(3 1)	(7 1 8 5)(3 2 6 4)	15:16:06	05:02:14	2
	(4 1 2 5 8 6 3)	06:01:11	13:15:12	3
	(2 1 8)(7 3 6)	03:13:06	16:01:03	3
(3 1 2)	(7 1 8 5)(3 2 6 4)	15:06:16	05:14:02	2
	(4 1 2 5 8 6 3)	06:11:01	13:12:15	3
	(2 1 8)(7 3 6)	03:06:13	16:03:01	3
(1 3 2)	(7 1 8 5)(3 2 6 4)	02:07:11	15:03:05	2
	(4 1 2 5 8 6 3)	16:15:06	02:05:14	3
	(2 1 8)(7 3 6)	01:06:11	15:13:12	3

Tabela 4.4: d_{free} dos códigos resultantes com taxa $r = 3/4$.

Diante dos resultados apresentados nas Tabelas 4.4 e 4.5, podemos observar que ocorreram as seguintes situações:

1. Alguns códigos resultantes são catastróficos;

2. Existem códigos que permaneceram com a mesma capacidade de correção de erros, mantendo o mesmo valor do d_{free};

3. O valor da distância livre de alguns códigos foi reduzido de 4 para 3;

4. Outros códigos tiveram uma redução do valor da distância livre de 4 para 2, perdendo sua capacidade de correção de erros.

A	ρ	G'_0	G'_1	d'_{free}
04:02:01	(51368)(274)	10:04:13	07:03:11	3
	(214)(635)	12:13:07	05:02:14	3
	(712368)(45)	05:11:17	14:11:02	catastrófico
	(61278543)	01:14:13	13:11:05	3
	(61857324)	14:06:03	03:11:07	catastrófico
	(2184)(7563)	10:03:05	07:03:16	catastrófico

Tabela 4.5: Outros códigos com taxa $r = 3/4$ resultantes da aplicação da transformação de permutação.

Mais uma vez foi possível constatar que a matriz A não tem influência alguma na redução da distância livre do código. Podemos observar que esta transformação apenas faz uma permutação nas linhas das submatrizes G'_0 e G'_1.

4.2.2 Matrizes triangulares superiores

Uma outra transformação que utilizamos como armadilha B foram as matrizes triangulares superiores, devido às propriedades convenientes que estas possuem. Os códigos convolucionais ótimos de memória unitária considerados possuem, respectivamente, taxa $R = 2/4$, com submatrizes geradoras,

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

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

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

$d_{free} = 10$.

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

$$T_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_3 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_4 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

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

Observação 4.2.1. *Note que procuramos considerar matrizes que não possuem* 0*s em sua diagonal principal, com o objetivo de que as mesmas sejam invertíveis.*

Na Tabela 4.6, apresentamos os efeitos causados na distância livre do código $(4, 2, 1)$ pela aplicação das transformações consideradas nesta seção. Como pode ser observado, foram obtidos dois resultados diferentes:

1. O d_{free} do código foi reduzido de 5 para 4;

2. O código perdeu a capacidade de correção de erro, passando a corrigir apenas 1 erro ao invés de 2, ou seja, o d_{free} que era igual a 5 passou a ser igual a 3.

Novamente percebemos que a matriz A não interfere no processo de diminuição da capacidade de correção de erros, sendo sua única função o embaralhamento dos bits de entrada.

Código	G_0	G_1	A	B	G'_0	G'_1	d_{free}	d'_{free}
(4,2,1)	15:3	14:07	A_1	T_1	11:03	10:07	5	3
	15:3	14:07		T_2	11:02	10:05	5	3
	15:3	14:07		T_3	15:03	14:04	5	3
	15:3	14:07		T_4	17:02	16:07	5	4
	15:3	14:07		T_5	11:03	10:04	5	3
	15:3	14:07	A_2	T_1	12:03	17:07	5	3
	15:3	14:07		T_2	13:02	15:05	5	3
	15:3	14:07		T_3	16:03	10:04	5	3
	15:3	14:07		T_4	15:02	11:07	5	4
	15:3	14:07		T_5	11:13	10:04	5	3
	15:3	14:07	A_3	T_1	11:12	10:17	5	3
	15:3	14:07		T_2	11:13	10:13	5	3
	15:3	14:07		T_3	15:16	14:10	5	3
	15:3	14:07		T_4	17:05	16:11	5	4
	15:3	14:07		T_5	11:12	10:14	5	3
	15:3	14:07	A_4	T_1	03:11	07:10	5	3
	15:3	14:07		T_2	02:11	04:10	5	3
	15:3	14:07		T_3	03:15	04:14	5	3
	15:3	14:07		T_4	02:16	07:16	5	4
	15:3	14:07		T_5	03:11	04:10	5	3
	15:3	14:07	A_5	T_1	03:12	07:13	5	3
	15:3	14:07		T_2	02:13	05:15	5	3
	15:3	14:07		T_3	03:16	14:10	5	3
	15:3	14:07		T_4	02:15	07:11	5	4
	15:3	14:07		T_5	03:12	04:14	5	3

Tabela 4.6: Transformação triangular superior, $r = 2/4$.

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

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

$$T_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad T_4 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} e$$

$$T_5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Observe na Tabela 4.7, que ao aplicarmos as transformações A e B, os seguintes resultados foram obtidos:

1. Obtemos códigos com $d'_{free} = 8$, passando a ter a capacidade de corrigir apenas 3 erros.

2. A maioria dos códigos encontrados, têm $d_{free} = 7$.

Mais especificamente, só foram encontrados códigos com capacidade de correção de 3 erros, enquanto que o código original, é capaz de corrigir 4 erros.

Código	A	B	G'_0	G'_1	d_{free}	d'_{free}
(8,2,1)	A_1	T_1	364:057	152:210	10	7
		T_2	315:061	176:255	10	8
		T_3	204:071	166:357	10	7
		T_4	244:064	143:334	10	7
		T_5	267:053	174:335	10	7

Tabela 4.7: Transformação triangular superior,$r = 2/8$.

4.2.3 Hadamard

Foram utilizadas as matrizes de Hadamard de ordem 4 e 8 como função armadilha B para os códigos ótimos de memória unitária com taxa $R = 2/4$ e $R = 2/8$, respectivamente. Para possibilitar os cálculos em F_2, trocamos $-1's$ por $1's$ e os $1's$ por $0's$, tornando-as matrizes de Hadamard binárias.

Como função armadilha responsável pelo embaralhamento das linhas, onde as matrizes componentes são denotadas por A_i, utilizamos as matrizes 2×2:

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

As submatrizes geradoras do código $(4, 2, 1)$ são dadas por

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

e as geradoras do código $(8, 2, 1)$ dadas por

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

Nas Tabelas 4.8 e 4.9 encontram-se os resultados obtidos quando utilizadas as matrizes de Hadamard do tipo Sylvester, H_4 e H_8, e do tipo Paley, Hp_8, como sendo a transformação armadilha B. As matrizes estão em representação octal, sendo que o número que está antes dos dois pontos representa a primeira e depois a segunda linha da matriz. A matriz transformação A varia entre A_1, A_2 e A_3 anteriormente definidas.

Código	G_0	G_1	A	B	G'_0	G'_1	d_{free}	d'_{free}
(4,2,1)	15:3	14:07	A_1	H_4	03:05	05:00	5	2
			A_2		06:05	05:00	5	2
			A_3		03:06	05:05	5	2

Tabela 4.8: Transformação Hadamard de ordem 4, $r = 2/4$.

Como pode ser observado na Tabela 4.8 para o código de memória unitária e com taxa $R = 2/4$ o d_{free} teve uma considerável queda de 5 para 2 independentemente de qual seja a transformação armadilha A. Porém apesar de ter alcançado o objetivo de redução da capacidade de correção, todos os códigos obtidos tiveram uma redução na dimensão do espaço de operação, pois uma ou mais colunas das submatrizes G'_0 e G'_1 são nulas. Outro fato observado, é que dois dos três códigos resultantes são códigos de memória unitária parcial.

O mesmo padrão ocorre com o código convolucional de memória unitária com taxa $R = 2/8$ apresentado na Tabela 4.9 que perde a capacidade de correção de erro resultando em um d_{free} menor que o do código original. Neste caso, a distância livre inicial era $d_{free} = 10$ e, após a aplicação das transformações passou a ser $d'_{free} = 8$, indiferentemente de quais fossem as

Código	G_0	G_1	A	B	G'_0	G'_1	d_{free}	d'_{free}
(8,2,1)	370:037	174:237	A_1	H_8	360:231	252:146	10	8
			A_2		151:231	314:146	10	8
			A_3		360:151	252:314	10	8
			A_1	Hp_8	232:321	306:056	10	8
			A_2		113:321	350:056	10	8
			A_3		360:151	252:314	10	8

Tabela 4.9: Transformação Hadamard de ordem 8, $r = 2/8$.

matrizes A e B. Neste caso, também ocorre uma redução na dimensão do espaço em todos os casos observados.

4.3 Comparação entre as Transformações Armadilha

Os resultados da aplicação da representação matricial dos elementos do grupo de permutação como sendo a transformação armadilha nos códigos de memória unitária com taxa $r = 2/3$, $r = 2/4$ e $r = 3/4$, mostraram que é possível obter grandes diminuições na capacidade de correção de erros desses códigos. Da mesma forma, com a aplicação das transformações armadilha de Hadamard e matrizes triangulares superiores obtemos bons resultados na redução da capacidade de correção de erros dos códigos com taxa $r = 2/4$ e $r = 2/8$ apresentados. Observe que, em todas as situações, concluímos que tais diminuições são causadas pela aplicação da transformação B, já que a transformação A não influencia no resultado.

Para o código com taxa $r = 2/4$, as transformações armadilha que tiveram melhor desempenho foram as de permutação e as de Hadamard, ambas capazes de gerar códigos com distância livre igual a 2, ou seja, códigos que são apenas capazes de detectar erros.

Entretanto, para o código com taxa $r = 2/8$, a transformação armadilha triangular superior proporcionou melhores resultados, sendo que a maioria dos códigos obtidos atinge $d'_{free} = 7$, ao passo que a transformação de Hadamard gerou códigos com $d'_{free} = 8$.

É interessante observar que, para as transformações e os códigos aqui considerados, em nenhum dos casos houve aumento no valor da distância livre, ou seja $d'_{free} \leq d_{free}$.

4.4 Código com taxa r = 3/6 MU e MUP

Nesta seção consideraremos dois códigos com taxa $r = 3/6$, sendo um de memória unitária (MU) e outro de memória unitária parcial (MUP). Aplicaremos as mesmas transformações armadilha nos dois códigos para analisar o efeito que estas causam em cada um. O código MU utilizado, é gerado pelas submatrizes geradoras

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

Este código possui $d_{free} = 6$ e, consequentemente, é capaz de corrigir até 2 erros.

O código MUP foi obtido a partir de um código com taxa $r = 1/2$ e memória $m = 2$. Para isso, procedemos da seguinte forma:

Seja o código convolucional $C_1(2, 1, 2)$ com $d_{free} = 5$, com matriz geradora dada por

$$G1 = \begin{bmatrix} 11 & 01 & 11 \end{bmatrix}.$$

A partir da matriz geradora de C_1, obtemos um código convolucional correspondente $C_2(4, 2, 1)$, gerado pela matriz denotada por

$$G2 = \begin{bmatrix} 1101 & 1100 \\ 0011 & 0111 \end{bmatrix}.$$

Agora, utilizando a matriz do código $C_2(4, 2, 1)$, obtemos uma outra matriz que gera um código convolucional de memória unitária parcial C_3 com parâmetros $(n = 6, k = 3)$ e tem a seguinte forma:

$$G3 = \begin{bmatrix} 110111 & 000000 \\ 001101 & 110000 \\ 000011 & 011100 \end{bmatrix}.$$

A matriz $G3$ é a matriz geradora do código MUP. Este código tem distância livre igual a 5 e, portanto, é capaz de corrigir até dois erros.

Nas Tabelas 4.10 e 4.11, exibimos os resultados da aplicação da transformação armadilha de permutação tanto no código de memória unitária (MU) com taxa $r = 3/6$ quanto no código de memória unitária parcial (MUP) com taxa $r = 3/6$.

Na Tabela 4.10 são dadas a transformação armadilha A, a permutação ρ associada à transformação armadilha B, as submatrizes resultantes da aplicação das transformações armadilha A e B no código convolucional MU, G'_0 e G'_1 e, além disso, encontramos a distância livre do novo código, $d'_{free}G$

Da mesma forma, na Tabela 4.11, encontramos a transformação armadilha A, a permutação ρ referente a transformação armadilha B, as submatrizes resultantes da aplicação das transformações armadilha A e B no código convolucional MUP, denotados por GP'_0 e GP'_1 e $d'_{free}GP$ que é a distância livre do código resultante após aplicação das matrizes armadilha.

É importante ressaltar que as permutações ρ aplicadas aos dois códigos são as mesmas. Isso foi feito porque nosso objetivo é comparar o efeito que essas transformações causam nos dois códigos.

A	ρ	G'_0	G'_1	$d'_{free}G$
04:02:01	id	70:36:13	34:07:62	6
	(11 1)(8 2 4 12)(6 3 5 10)	03:77:51	33:20:45	5
	(9 11 3 2 10 12)(7 4 5)(6 8)	31:74:47	13:46:22	5
	(7 1 10 4 8 6 5 3 12)(2 9 11)	44:53:07	42:16:31	6
	(12 1 4 6 2 10 5 11 7 3 8 9)	31:63:16	41:54:46	catastrófico
	(10 1)(11 3 8 5 12 7)	70:66:11	16:62:63	5
	(8 1 7 5 10 6 11 12 3 2)	32:17:61	61:13:15	6
	(7 1 10)(8 3 6 4)(11 9 5 12)	64:63:14	62:31:35	5
	(12 1 8 9 7)(3 2 6 11)(5 10)	52:17:61	23:46:16	6
	(4 1 5 9 2 7 6 3 10 8)	17:51:61	14:33:46	catastrófico
	(12 1 9 6 4 5 10 3 11 2)(7 8)	42:37:14	47:06:74	5
	(9 1 10 3 8)(5 4 12 11)	70:66:11	34:07:47	5
	(8 1 7 3 12 10)(11 5)(9 6 2)	21:17:42	65:43:56	6
	(12 1 9 3 7 8)(11 5 10)(4 6)	62:23:14	51:36:56	6
	(10 1 11 12 8 5 4 6 2 9 3 7)	21:43:54	55:72:31	6
	(10 1 4 8 12 3 6)(5 11)(7 9)	25:63:16	45:23:13	4
	(9 1 11 7 5 6 12 10 8 3)	30:65:42	35:61:66	5
	(5 1 3 7 6 10 11 2 12 4)	43:65:54	32:07:64	6
	(12 1 3 5 8)(10 2 11 6 7 9 4)	46:74:73	45:34:02	3

Tabela 4.10: Transformação permutação aplicada ao código MU com taxa R = 3/6

.

Após o processo de aplicação das transformações armadilha ao código convolucional de memória unitária com taxa $R = 3/6$, observamos (veja Tabela 4.10) que foram obtidos os seguintes resultados:

1. Alguns códigos mantiveram a distância livre igual à distância livre do código original, ou seja, $d'_{free}G = 6$;

2. Alguns códigos, apesar de não perderem a capacidade de correção de erros, apresentaram diminuição de 1 unidade na distância livre, isto é, $d'_{free}G = 5$. Sendo assim, o código apenas deixou de detectar alguns erros em relação ao código original.

3. Alguns perderam a capacidade de correção de erros passando a ter $d'_{free}G = 4$, enquanto que a distância livre do código original era de 6. Estes códigos são capazes de corrigir apenas 1 erro;

4. Códigos resultantes com $d'_{free}G = 3$. A capacidade de correção deste código caiu pela metade quando comparado do código original;

5. Obtenção de códigos catastróficos que são muito interessantes para o nosso trabalho, como veremos mais adiante.

Como mencionado anteriormente, aplicamos as mesmas transformações armadilha no código convolucional de memória unitária parcial de taxa $r = 3/6$ e, com isso, os resultados obtidos foram:

1. Da mesma forma que no caso dos códigos MU, obtivemos códigos que mantiveram a distância livre igual à do código original, $d'_{free}GP = 5$, mantendo a capacidade de correção de erros;

2. Certos códigos tiveram uma perda na capacidade de correção de erros. Sua distância livre passou a ser igual a 4;

3. De forma análoga ao código MU, a aplicação das matrizes de transformações foram capazes de produzir um código catastrófico.

A	ρ	GP'_0	GP'_1	$d'_{free}GP$
04:02:01	id	67:15:03	00:60:34	5
	(11 1)(8 2 4 12)(6 3 5 10)	30:21:12	26:45:15	4
	(9 11 3 2 10 12)(7 4 5)(6 8)	14:03:25	70:62:21	5
	(7 1 10 4 8 6 5 3 12)(2 9 11)	01:06:65	66:25:20	4
	(12 1 4 6 2 10 5 11 7 3 8 9)	45:54:34	05:42:24	5
	(10 1)(11 3 8 5 12 7)	25:15:51	24:03:30	5
	(8 1 7 5 10 6 11 12 3 2)	14:64:22	64:05:54	4
	(7 1 10)(8 3 6 4)(11 9 5 12)	31:15:54	50:24:12	5
	(12 1 8 9 7)(3 2 6 11)(5 10)	34:64:62	05:12:24	5
	(4 1 5 9 2 7 6 3 10 8)	44:21:52	70:64:44	5
	(12 1 9 6 4 5 10 3 11 2)(7 8)	05:01:46	13:74:50	catastrófico
	(9 1 10 3 8)(5 4 12 11)	23:13:51	12:44:22	5
	(8 1 7 3 12 10)(11 5)(9 6 2)	05:44:20	32:50:42	4
	(12 1 9 3 7 8)(11 5 10)(4 6)	25:15:46	03:50:42	4
	(10 1 11 12 8 5 4 6 2 9 3 7)	07:16:24	24:11:51	5
	(10 1 4 8 12 3 6)(5 11)(7 9)	70:54:12	06:11:12	5
	(9 1 11 7 5 6 12 10 8 3)	26:06:12	50:26:45	4
	(5 1 3 7 6 10 11 2 12 4)	06:50:05	43:61:70	5
	(12 1 3 5 8)(10 2 11 6 7 9 4)	10:43:16	17:12:42	4

Tabela 4.11: Transformação permutação aplicada ao código MUP com taxa R = 3/6

.

4.4.1 Comparação entre MU e MUP

Comparando os efeitos que as transformações causaram aos dois códigos, observamos que o código convolucional de memória unitária é mais sensível a perturbações que o código convolucional de memória unitária parcial. Note que, enquanto o código MU tem uma diminuição

na capacidade de correção de erros bastante significativa, onde a distância livre de $d_{free} = 6$ do código original, passou a ser 5, 4 ou 3 no código resultante, o código MUP apenas tem uma redução no valor do d_{free} de 5 para 4. Graficamente, esse comportamento se deve ao fato que o d_{free} do código MU é o ponto de máximo da função convexa e o d_{free} do código MUP é o ponto de mínimo da função côncava.

Capítulo 5

O Código Quântico Concatenado

Neste capítulo, utilizamos o código convolucional quântico (CCQ) concatenado $[4, 1, 3]$ proposto por Almeida e Palazzo [17]. Este código tem $d_{free} = 9$ e é capaz de corrigir um erro quântico geral. Foi construído a partir da concatenação de dois códigos convolucionais clássicos (CCC), a saber, um $\mathcal{C}_1$ $(2, 1, 2)$, e outro $\mathcal{C}_2$ $(4, 2, 1)$ gerados a partir de uma mesma matriz geradora.

Analogamente ao capítulo anterior, aplicamos as transformações armadilhas no CCC concatenado $(4, 1, 3)$ com o intuito de diminuir a capacidade de correção do código. Entretanto, surpreendentemente, obtivemos como um dos resultados, CCCs concatenados com d_{free} maior que o d_{free} do CCC concatenado original, ou seja, $d_{free} > 9$. Os códigos concatenados obtidos possuem $d_{free} = 10$ ou $d_{free} = 11$ e, em sua maioria, o código com taxa $R = 2/4$ que o compõe é um código catastrófico. É importante mesmo sendo composto por um código catastrófico, o CCC resultante da concatenação é um código não catastrófico.

Este capítulo está organizado da seguinte forma: Na Seção 5.1, descrevemos todo o processo de construção do CCQ concatenado onde é feita a associação entre os CCCs e os CCQs. Na Seção 5.2, são apresentados todos os resultados obtidos com a aplicação da transformação armadilha ao código $(4, 1, 3)$. Na Seção 5.3, aplicamos a transformação armadilha de permutação aos códigos com $d_{free} > 9$ obtidos na Seção 5.2, e apresentamos os novos resultados.

5.1 Construção do CCQ [4,1,3]

Como mencionado anteriormente, para a construção do CCQ $[4, 1, 3]$ foram utilizados dois códigos convolucionais quânticos específicos $\mathcal{C}_1$ e $\mathcal{C}_2$. Estes códigos são capazes de corrigir, respectivamente, um erro de fase Z e um erro de bit X, para um mesmo conjunto de registros quânticos. Em seguida é realizada a concatenação destes dois códigos da seguinte forma: primeiro $\mathcal{C}_1$ foi usado para codificar o estado quântico de informação e depois $\mathcal{C}_2$ foi utilizado para codificar o estado quântico resultante da codificação por $\mathcal{C}_1$. O código $\mathcal{C}$ resultante desta concatenação é capaz de corrigir um erro quântico geral com geradores Z e X, para o mesmo conjunto

de registros quânticos protegidos por $\mathcal{C}_1$ e $\mathcal{C}_2$.

A seguir, veremos detalhadamente cada passo desse processo.

5.1.1 Um CCQ para o Canal Bit Flip

Para a obtenção de um CCQ para o canal bit flip foi utilizado o CCC $(2, 1, 2)$ com matriz geradora polinomial $\mathbf{G}(D) = [1 + D^2, 1 + D + D^2]$ na operação de codificação dos bits contidos em cada um dos kets da base de uma seqüência de registros quânticos que deve ser protegida da ação dos erros de um canal bit flip. Desta operação de codificação clássica, podemos obter a correspondente operação de codificação quântica:

$$\bigotimes_{t=0}^{+\infty} \mid u_t\rangle \rightarrow \bigotimes_{t=0}^{+\infty} \mid u_t + u_{t-2}, u_t + u_{t-1} + u_{t-2}\rangle, \tag{5.1}$$

onde $u_{-1} = u_{-2} = 0$.

Ao final de cada seqüência de bits contidos dentro dos kets da base são acrescentados mais dois bits 0s para serem codificados, pois, na prática, o interesse é sempre codificar uma seqüência finita de qubits. Esta condição garante que tenhamos sempre palavras-código clássicas válidas dentro dos kets e, consequentemente, garante a obtenção de palavras-código quântica válida.

Este CCQ herda todas as propriedades do CCC associado. Isto é, se um CCQ $[2, 1, 2]$ é não-catastrófico, continua com a mesma capacidade de correção de erros, ou seja, é capaz de corrigir até dois erros bit flip, X, em quaisquer dois qubits da palavra-código.

Exemplos N=1 e N=2

Seja N o comprimento da seqüência de informação. A seguir, será apresentado o CCQ descrito pela operação de codificação (5.1) para uma seqüência de informação composta de um e de dois registros quânticos.

Exemplo 5.1.1. *Suponha que se queira codificar apenas um registro quântico. O estado do sistema quântico a ser codificado é, portanto, o estado de um único qubit,* $|\Psi\rangle = a|0\rangle + b|1\rangle$. *A operação de codificação quântica é realizada através da codificação do bit dentro de cada um dos kets da base pelo CCC com matriz geradora dada por:*

$$G_{C_1} = \left[\begin{array}{ccc} 11 & 01 & 11 \end{array} \right].$$

Assim, temos a seguinte codificação quântica:

$$|0_L\rangle \rightarrow |000000\rangle$$

e

$$|1_L\rangle \rightarrow |110111\rangle.$$

Considerando que o estado inicial $a|0\rangle + b|1\rangle$ tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros X em quaisquer dos seis qubits das palavras-código.

Exemplo 5.1.2. *Suponha que se queira codificar dois registros quânticos. O estado do sistema quântico a ser codificado é, portanto, o estado de dois qubits, a saber, $|\Psi\rangle = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$. A operação de codificação quântica é realizada através da codificação dos dois bits contidos em cada um dos quatro kets da base pelo CCC com matriz geradora dada por:*

$$G_{C_1} = \begin{bmatrix} 11 & 01 & 11 & \\ & 11 & 01 & 11 \end{bmatrix}.$$

Assim, temos a seguinte codificação quântica:

$$|00_L\rangle \rightarrow |00000000\rangle,$$

$$|01_L\rangle \rightarrow |00110111\rangle,$$

$$|10_L\rangle \rightarrow |11011100\rangle \quad e$$

$$|11_L\rangle \rightarrow |11101011\rangle.$$

Considerando novamente que o estado inicial $a|0\rangle + b|1\rangle$ tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros X em quaisquer dos oito qubits das palavras-código.

5.1.2 Um CCQ para o canal Phase Flip

Há uma maneira fácil de tratar o canal phase flip como um canal bit flip. Para isso, suponha que ao invés de utilizarmos a base computacional $\{|0\rangle, |1\rangle\}$ para o qubit, utilizaremos a base conjugada $\{|+\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}, |-\rangle \equiv (|0\rangle - |1\rangle)/\sqrt{2}\}$. Com respeito a essa base, o operador Z leva o estado $|+\rangle$ ao estado $|-\rangle$ e vice-versa, isto é, este operador atua como se fosse um operador bit flip em relação aos símbolos $+$ e $-$. Assim, todas as operações necessárias, ou seja, codificação, detecção e a correção de erros, podem ser executadas exatamente como no canal bit flip, porém, em relação à base $\{|+\rangle, |-\rangle\}$, ao invés da base $\{|0\rangle, |1\rangle\}$. Usando este mesmo procedimento para os CCQs, a operação de codificação (5.1), descrita para um canal bit flip pode ser escrita para um canal na base $\{|0\rangle, |1\rangle\}$ como:

$$\bigotimes_{t=0}^{+\infty} |\, u_t\rangle \rightarrow \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2}(|0\rangle + (-1)^{(u_t+u_{t-2})}|1\rangle)(|0\rangle) + (-1)^{(u_t+u_{t-1})} \right\}, \tag{5.2}$$

onde $u_1 = u_2 = 0$.

Mais explicitamente,

$$\bigotimes_{t=0}^{+\infty} \mid u_t\rangle \rightarrow \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2} \Sigma^{(1,1)}_{(p_t,q_t)=(0,0)} (-1)^{(u_t+u_{t-2})p_t+(u_t+u_{t-1}+u_{t-2})q_t} |pt, q_t\rangle \right\}. \tag{5.3}$$

A operação (5.2) é denominada forma *compacta* (a palavra-código sempre pode ser escrita como um produto tensorial de blocos como este) e a operação (5.3) de forma *explícita*. Ambas as formas são equivalentes, mas para um CCQ e um canal quântico com erro geral, a forma compacta é mais útil para o entendimento da operação de decodificação e a forma explícita, mais útil para o entendimento da operação de geração.

Evidentemente, este CCQ também está associado ao CCC com matriz geradora polinomial $\mathbf{G}(D) = [1+D^2, 1+D+D^2]$ e, consequentemente, é um código não catastrófico com capacidade para a correção de qualquer grupo de até dois erros Z sobre a palavra quântica (5.3). Também podem ser corrigidas algumas palavras-código quânticas com mais de dois erros Z.

A natureza convolucional esta explícita nos bits presentes dentro dos kets do código, no CCQ para o canal bit flip. Já no CCQ para o canal phase flip, a natureza convolucional aparece de forma sutil na distribuição de *sinais* em frente aos kets do código.

Exemplos N=1 e N=2

A seguir, apresentamos o CCQ descrito pela operação de codificação (5.3) para uma seqüência de informação composta de um e de dois registros quânticos, respectivamente.

Exemplo 5.1.3. *Suponha que desejamos codificar apenas um registro quântico. De acordo com a operação de codificação (5.3), o CCQ pode ser convenientemente escrito pela base* $|+\rangle, |-\rangle$ *como sendo*

$$|0_L\rangle \rightarrow |+++++++\rangle \quad e$$

$$|1_L\rangle \rightarrow |--+---\rangle.$$

Na base $|0\rangle, |1\rangle$*, tal código é escrito como sendo*

$$|0_L\rangle \rightarrow \frac{1}{8}\{(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\}$$

$$|1_L\rangle \rightarrow \frac{1}{8}\{(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\}$$

Em outras palavras, cada estado da base foi codificado em uma superposição de sessenta e quatro estados, cada um contendo seis qubits. Considerando que o estado inicial $a|0\rangle+b|1\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros* Z *em quaisquer dos seis qubits da palavra-código. Assim, este código é capaz de corrigir dois erros de sinal em quaisquer dos seis blocos acima.*

Exemplo 5.1.4. *Suponha agora que desejamos codificar dois registros quânticos. De acordo com a operação de codificação (5.3), o CCQ pode ser convenientemente escrito pela base* $|+\rangle, |-\rangle$ *como:*

$$|00_L\rangle \rightarrow |++++++++\rangle$$

$$|01_L\rangle \rightarrow |++--+---\rangle$$

$$|10_L\rangle \rightarrow |--+---++\rangle$$

$$|11_L\rangle \rightarrow |---+-+--\rangle.$$

Na base $|0\rangle, |1\rangle$*, tal código é escrito como*

$$|00_L\rangle \rightarrow \frac{1}{16}\{(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\},$$

$$|01_L\rangle \rightarrow \frac{1}{16}\{(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\},$$

$$|10_L\rangle \rightarrow \frac{1}{16}\{(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\} \quad e$$

$$|11_L\rangle \rightarrow \frac{1}{16}\{(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\}.$$

Então, cada estado da base foi codificado em uma superposição de duzentos e cinquenta e seis estados, cada um com oito qubits. Considerando novamente que o estado inicial $|\Psi\rangle = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle$ *tenha sido perfeitamente codificado, podemos garantir a correção de até dois erros* X *em quaisquer dos oito qubits das palavras-código. Assim, o código é capaz de corrigir dois erros de sinal em quaisquer dos oito blocos.*

5.1.3 CCQ para o canal Quântico com Erro Geral

Após a construção destes dois códigos quânticos, $\mathcal{C}_1$ e $\mathcal{C}_2$ capazes de corrigir erros de fase Z e de bit X, respectivamente, para um mesmo conjunto de registros quântico, estamos praticamente prontos para concatená-los, obtendo o código concatenado $\mathcal{C}$.

Para que C_2 possa ser concatenado a C_1, devemos antes construir um código equivalente a C_1 com taxa 2/4. A opção trivial para esta tarefa é utilizar a matriz geradora

$$G_{C_1} = \begin{bmatrix} 11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix},$$

que é a matriz discreta semi-infinita associada à matriz polinomial $\mathbf{G}(D) = [1+D^2, 1+D+D^2]$, usada na construção de C_1. Agora, dois bits entrarão no codificador a cada instante de tempo e com cada registro de deslocamento conterá apenas uma memória de modo que o número total de memórias permaneça o mesmo número de C_1. Assim, o código equivalente a C_2 com taxa 2/4 será construído a partir da matriz geradora G_{C_1}, escrita na forma:

$$G_{C_2} = \begin{bmatrix} 1101 & 1100 & & & & \\ 0011 & 0111 & & & & \\ & & 1101 & 1100 & & \\ & & 0011 & 0111 & & \\ & & & & 1101 & 1100 \\ & & & & 0011 & 0111 \\ & & & & \ddots & \ddots & \ddots \end{bmatrix}.$$

Dessa forma, a operação de codificação quântica para o código C_2 $[4, 2, 1]$ será:

$$\bigotimes_{t=0}^{+\infty} | \, u_t\rangle \rightarrow \bigotimes_{t=0}^{+\infty} | \, u_t^1 + u_{t-1}^1, u_t^1 + u_{t-1}^1 + u_{t-1}^2, u_t^2 + u_{t-1}^2, u_t^2 + u_{t-1}^2 + u_t^1\rangle, \tag{5.4}$$

onde u_t^1, $u_t^2 \in \{0, 1\}$ e $u_{-1} = u_{-2} = 0$. Esta codificação quântica é a equivalente trivial com taxa 2/4 e memória unitária da operação (5.1). Devido à essa equivalência, a capacidade de correção do novo código é a mesma, ou seja, é possível garantir com $\mathcal{C}_2$ $[4, 2, 1]$ a correção de qualquer grupo com até dois erros X.

Concatenando-se os CCCs $(2, 1, 2)$ e $(4, 2, 1)$, respectivamente, associados aos CCQs $\mathcal{C}_1$ $[2, 1, 2]$ e $\mathcal{C}_2$ $[4, 2, 1]$, obtemos um novo CCC, com taxa $R = 1/4$ e matriz geradora dada por.

$$G_C = \begin{bmatrix} 1110 & 1000 & 1001 & 1011 & & & \\ & 1110 & 1000 & 1001 & 1011 & & \\ & & 1110 & 1000 & 1001 & 1011 & \\ & & & 1110 & 1000 & 1001 & 1011 \\ & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

O CSL deste CCC concatenado pode ser visualizado na Figura 5.1.

Este codificador concatenado não catastrófico possui três memórias *efetivas*, duas do primeiro codificador e uma do segundo. Assim, apenas oito estados (e não dezesseis, como poderia sugerir o número total de memórias do codificador) são gerados por este codificador. Tal fato pode ser observado através do seu diagrama de estados, apresentado na Figura (5.2).

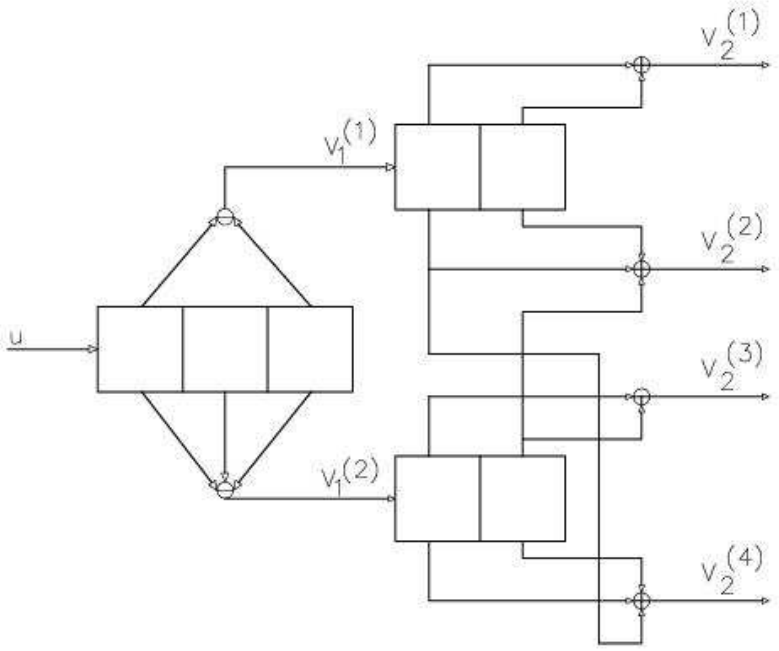

Figura 5.1: Concatenação dos codificadores (2,1,2) e (4,2,1).

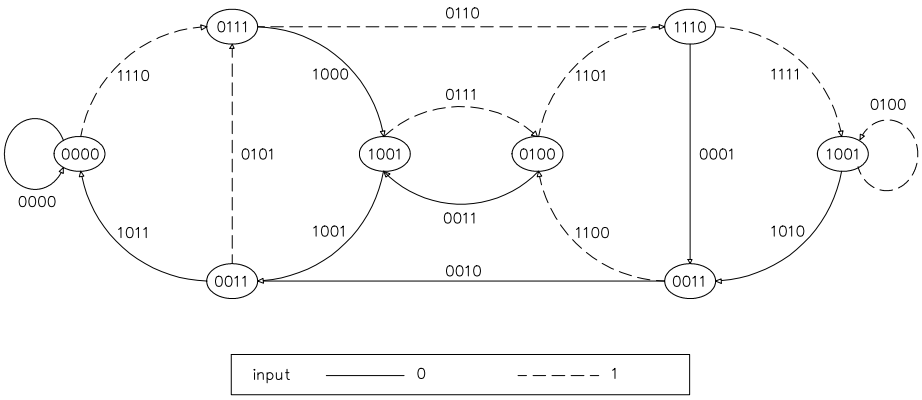

Figura 5.2: Diagrama de estados para o código convolucional clássico concatenado (4,1,3).

É fácil verificar neste diagrama de estados que são necessárias pelo menos quatro transições para partir do estado inicial 0000 e retornar ao mesmo estado ($0000 \rightarrow 0111 \rightarrow 1001 \rightarrow 0011 \rightarrow 0000$). O caminho mais curto é também o caminho do d_{free}. Em outras palavras, não há nenhum outro caminho não nulo que parta do estado inicial 0000 e retorne ao 0000, com peso menor que nove. A palavra-código correspondente é $1110 \rightarrow 1000 \rightarrow 1001 \rightarrow 1011$. Logo o d_{free} deste CCC é nove e, portanto, capaz de corrigir qualquer grupo com até quatro erros sobre a palavra-código.

O CCC $(4, 1, 3)$ pode ser usado na construção de um CCQ capaz de garantir a correção de um erro quântico geral com geradores X e Z em qualquer dos qubits da palavra-código quântica. Isto porque três condições são satisfeitas:

1. O CCC $(2, 1, 2)$ associado a $\mathcal{C}_1[2, 1, 2]$, garante a correção de um erro (no caso estudado garante a correção de até mesmo dois erros). Este erro clássico está associado ao erro quântico Z;

2. O CCC $(4, 2, 1)$, associado a $\mathcal{C}_2[4, 2, 1]$, garante a correção de um erro (no caso estudado garante a correção de até mesmo dois erros). Este erro clássico está associado ao erro quântico X;

3. O CCC $(4, 1, 3)$, associado à concatenação de $\mathcal{C}_2[4, 2, 1]$ a $\mathcal{C}_1[2, 1, 2]$, garante a correção de quatro erros. Estes erros clássicos estão associados aos erros quânticos $Z, X, Y(= XZ)$ e I, que constituem a base para um erro quântico geral.

Portanto, $\mathcal{C}_2[4, 2, 1]$ codifica o estado quântico resultante da codificação por $\mathcal{C}_1[2, 1, 2]$. O código $\mathcal{C}$, resultante desta concatenação, é um CCQ com taxa $R = 1/4$. Definindo o número de *memórias convolucionais quânticas* como sendo $m = T - 1$, onde T é o número de transições de unidades de tempo necessárias para a obtenção de uma palavra-código quântica válida, segue que a memória do CCQ concatenado é $m = 3$. Pode-se então dizer que $\mathcal{C}$ é um CCQ $[4, 1, 3]$.

A operação de codificação do CCQ concatenado para um canal quântico com erro geral pode agora ser convenientemente escrita na base $\{|0\rangle, |1\rangle\}$ como:

$$\bigotimes_{t=0}^{+\infty} \mid u_t\rangle \rightarrow \bigotimes_{t=0}^{+\infty} \left\{ \frac{1}{2} \Sigma^{(1,1)}_{(p_t,q_t)=(0,0)} (-1)^{(u_t+u_{t-2})p_t+(u_t+u_{t-1}+u_{t-2})q_t} \right.$$
$$\left. |p_t + p_{t-1}, p_t + p_{t-1} + q_{t-1}, q_t + q_{t-1} + p_t\rangle \right\}, \quad (5.5)$$

onde $u_{-1} = u_{-2} = 0$ e $p_{-1} = q_{-1} = 0$.

Foi necessário utilizar dois codificadores clássicos para este procedimento, pois o erro geral deste canal quântico possui dois geradores, Z e X.

Como o d_{free} do CCC associado é nove, esse CCQ não pode corrigir nenhum erro além do geral, o que nos leva a concluir que a distância quântica é três. Além disso, como os CCCs $(2, 1, 2)$, $(4, 2, 1)$ e $(4, 1, 3)$ são não-catastróficos, segue que o CCQ $[4, 1, 3]$ associado também é um código não-catastrófico.

Almeida [16] ressalta que, para a construção de um CCQ que corrija apenas um erro X ou apenas um erro Z, é necessário um CCC mais simples, de memória unitária, como por exemplo, $\mathbf{G}(D) = [1 + D, D]$, com $d_{free} = 3$. Porém, este CCC quando concatenado a um equivalente com taxa $R = 2/4$ gera um CCC com $d_{free} < 9$ e, assim, o CCQ $\mathcal{C}$, associado à concatenação, não garantiria a correção de um erro quântico geral.

5.2 Aplicando Transformações no CCC (4,1,3)

Utilizaremos o código CCQ $[4, 1, 3]$ como aplicação ao sistema criptográfico.

Para aplicar as transformações armadilha no CCC (4,1,3) procedemos da seguinte maneira: Antes da concatenação, aplicamos as matrizes A e B somente código $\mathcal{C}_2$, gerado pela matriz G_C, e observamos o decréscimo no valor de d_{free} do código (da mesma forma que procedemos no Capítulo 4) resultando no código $\mathcal{C}_2'$. A transformação armadilha A consiste da matriz identidade de ordem 2×2 e as transformações armadilha B utilizadas são aquelas provenientes das representações matriciais dos elementos do grupo de permutação S_8.

Após a obtenção do código $\mathcal{C}_2'$, concatenamos tal código ao código $\mathcal{C}_1$, dando origem ao código concatenado $\mathcal{C}'$.

Os resultados das aplicações podem ser observados na Tabela 5.1, onde são dadas a transformação armadilha A, a permutação associada a transformação armadilha B, denotada como ρ, as submatrizes G_0' e G_1' do código ($\mathcal{C}_2'$ código este resultante da aplicação das transformações armadilha no código $\mathcal{C}_2$), o d'_{free} que é a distância livre do código resultante, e o símbolo $(*)$, que denota o código é catastrófico.

Além disso, G_C denota a matriz geradora do código clássico concatenado. Esta matriz é representada da seguinte forma:

$$G_C = [G_0; G_1; G_2; G_3];$$

O ponto e vírgula separa as submatrizes geradoras. Como um exemplo, considere a matriz

$$G_C = [1110 \;\; 1000 \;\; 1001 \;\; 1011].$$

sua representação octal é dada por:

$$G_C = [16; 10; 11; 13].$$

Há também na Tabela 5.1, uma coluna com $d'_{free}C$, denotando a distância livre do código clássico concatenado resultante.

Observando a Tabela 5.1, temos os seguintes resultados, quando aplicadas as transformações no código $\mathcal{C}_2$:

1. Em alguns códigos o d'_{free} continuou o mesmo, ou seja, $d_{free} = 5$;

2. Em outros códigos, o d'_{free} caiu de 5 para 4;

3. Há códigos com $d'_{free} = 3$;

4. Há códigos resultantes catastróficos.

Depois disso, concatenamos os códigos $\mathcal{C}_1$ com os códigos resultantes $\mathcal{C}'_2$ obtendo o código clássico concatenado $\mathcal{C}'$ de mesma taxa $R = 1/4$ e mesma quantidade de memórias $m = 3$ do código concatenado clássico original, $\mathcal{C}$.

Como conseqüência, obtivemos os seguintes resultados tabulados na Tabela 5.1 (duas últimas colunas):

1. O d_{free} do novo código clássico concatenado $\mathcal{C}'$, continuou igual a 9, isto é, não houve mudança alguma na capacidade de correção de erros;

2. O código $\mathcal{C}'$ teve uma perda da capacidade de correção, pois seu d_{free} caiu de 9 para 8;

3. O código $\mathcal{C}'$ teve uma perda da capacidade de correção reduzida, indo de $d_{free} = 9$ para $d_{free} = 7$.

Além desses resultados citados acima, obtivemos outros, que mesmo não sendo o foco principal do trabalho, são de grande importância como veremos a seguir:

1. Em três casos houve aumento no d_{free}: o $d_{free} = 9$ passou a ser $d_{free} = 10$;

2. Em dois casos houve aumento no d_{free}: o $d_{free} = 9$ passou a ser $d_{free} = 11$.

Concluímos que, através do processo de aplicação de transformações armadilha no código $\mathcal{C}_2$ e concatenando-o com o código $\mathcal{C}_1$, podemos encontrar códigos clássicos concatenados com maior capacidade de correção de erros quando comparado ao código clássico concatenado original.

Ao analisar os resultados encontrados na Tabela 5.1, percebemos que não há uma relação direta entre o d_{free} do código $\mathcal{C}'_2$ e o d_{free} do código concatenado $\mathcal{C}'$ obtido, pois podemos encontrar situações em que $d_{free}(\mathcal{C}'_2) = 5$ resulta em um $d_{free}(\mathcal{C}) = 7$ ou 10 sendo que em sua maioria é igual a 9. Porém, observamos que em todos os casos em que o código $\mathcal{C}'_2$ é catastrófico, o d_{free} aumenta.

Após encontrar os novos CCCs concatenados, podemos obter os CCQs concatenados associados como foi feito na Seção 5.1.3.

5.3 Aplicando a Transformação de Permutação aos Novos CCCs (4,1,3)

Como visto na seção anterior, ao aplicar as transformações às submatrizes geradoras do código $\mathcal{C}_2$ com o intuito de encontrar códigos concatenados com capacidade de correção menor que a do código original, geramos alguns códigos com maior poder de correção de erros.

A	ρ	G'_0	G'_1	d'_{free}	G_C	$d'_{free}C$
02:01	id	15:03	14:07	5	16;10;11;13	9
	(5286374)	11:06	13:07	5	17;12;10;14	9
	(21564)(38)	15:02	14:17	3	17;01;00;03	7
	(425)	15:02	14:17	3	17;01;00;03	7
	(31278465)	13:05	05:13	$*$	16;13;05;16	11
	(61)(42)(7358)	17:14	04:13	3	03;03;17;10	8
	(213874)(56)	05:12	16:13	3	17;17;04;05	10
	(81674523)	11:16	13:06	5	07;03;01;15	9
	(526874)	15:06	12:07	5	13;13;14;15	10
	(61387)(52)	05:16	16:03	4	13;03;10;15	9
	(3162574)	16:11	06:13	5	07;04;14;15	9
	(51827364)	03:16	15:06	5	15;05;13;13	9
	(2164)(73)(58)	15:12	05:16	5	07;01;06;06	9
	(413865)(72)	01:16	02:17	3	17;10;06;06	9
	(4257)(836)	17:02	02:17	$*$	15;17;02;15	10
	(417328)(65)	03:14	15:13	5	17;12;04;06	9
	(215678)(43)	16:03	11:16	4	15;04;03;07	9
	(71583)(62)	15:07	06:11	5	12;10;03;17	9
	(214738)(56)	14:13	15:06	5	07;00;01;13	7
	(21743658)	06:13	07:12	$*$	15;06;07;15	11

Tabela 5.1: Transformação permutação oriunda do S_8 aplicada ao CCC concatenado.

Esses novos códigos com capacidade de correção maior que o original serão agora utilizados para o processo de codificação e então, novas transformações serão aplicadas a $\mathcal{C}_2$ para que possamos obter d_{free} menores no CCC obtido após a concatenação, e a partir daí utilizar o CCC concatenado resultante no sistema criptográfico.

Nas Tabelas 5.2, 5.3 e 5.4, o símbolo $(*)$ representa o código catastrófico.

5.3.1 O Código $\mathbf{CCC}_1$

Considere o CCC_1 $(4, 1, 3)$ com matriz geradora semi-infinita dada por

$$GC_1 = \begin{bmatrix} 1101 & 0110 & 0111 & 1101 & & \\ & 1101 & 0110 & 0111 & 1101 & \\ & & 1101 & 0110 & 0111 & 1101 \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix},$$

que é resultado da concatenação dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$. Este CCC concatenado tem $d_{free} = 11$ e, portanto, tem capacidade para corrigir até cinco erros clássicos.

As matrizes geradoras semi-infinitas dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$ são dadas, respectivamente, por:

$$G_{C_1} = \begin{bmatrix} 11 & 01 & 11 & & \\ & 11 & 01 & 11 & \\ & & 11 & 01 & 11 \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

e

$$G_{C_2} = \begin{bmatrix} 0110 & 0111 & & & \\ 1011 & 1010 & & & \\ & 0110 & 0111 & & \\ & 1011 & 1010 & & \\ & & 0110 & 0111 & \\ & & 1011 & 1010 & \\ & & & \ddots & \ddots \end{bmatrix},$$

onde $\mathcal{C}_1$ tem $d_{free} = 5$ e $\mathcal{C}_2$ é um código catastrófico. Aplicamos as transformações armadilha de permutação no novo código $\mathcal{C}_2$ obtendo os resultados apresentados na Tabela 5.2.

A Tabela 5.2 apresenta a transformação armadilha A, a permutação associada a transformação armadilha B, denotada como ρ, as submatrizes G'_0 e G'_1 do código $\mathcal{C}'_2$ que é o código resultante da aplicação das transformações armadilha ao código $\mathcal{C}_2$, a distância livre do código resultante, denotado por d'_{free}, G_C, que é a matriz geradora do CCC concatenado e $d'_{free}(C)$, a distância livre do CCC concatenado resultante.

Como pode ser observado na Tabela 5.2, após aplicar a transformação de permutação ao código $\mathcal{C}_2$, obtemos os seguintes resultados:

1. A capacidade de correção de erros do CCC obtido continua a mesma, ou seja, o $d'_{free} = 11$;
2. Houve uma diminuição na capacidade de correção de erros do código, sendo que o $d_{free} = 11$ passou a ser 10. Assim, o novo CCC concatenado passou a ser capaz de corrigir 4 erros.
3. O d_{free} do CCC passou de 11 para 9. Com isso, houve uma diminuição na sua capacidade de correção de erros.

Depois do processo de aplicação das transformações armadilha de permutação, podemos obter os CCQs equivalentes a cada um dos CCCs resultantes da mesma forma que foi realizado na Seção 5.1.

5.3.2 O Código CCC_2

Considere agora CCC_2 $(4, 1, 3)$. Este código possui matriz geradora semi-infinita dada por

$$GC_1 = \begin{bmatrix} 1101 & 1111 & 0010 & 1101 & & \\ & 1101 & 1111 & 0010 & 1101 & \\ & & 1101 & 1111 & 0010 & 1101 \\ & & & \ddots & \ddots & \ddots \end{bmatrix}.$$

A	ρ	G'_0	G'_1	d'_{free}	G_C	$d'_{free}C$
02:01	id	06:13	07:12	$*$	15;06;07;15	11
	(2143)(657)	03:15	13:14	5	16;12;02;07	9
	(8142367)	07:14	06:15	$*$	13;07;06;13	11
	(4123)(76)	14:07	07:14	$*$	13;14;07;13	11
	(815)(62)(37)	06:13	16:03	$*$	15;06;16;15	11
	(82576)	02:17	17:10	2	15;10;05;07	9
	(6185)(42)(37)	13:06	12:07	$*$	15;13;12;15	11
	(21375)(68)	12:17	15:02	3	05;00;07;17	9
	(218)(756)	12:07	15:06	5	15;14;13;13	10
	(5138)(247)	13:15	06:11	4	06;02;17;17	10

Tabela 5.2: Transformação permutação aplicada ao novo CCC_1 concatenado.

O CCC concatenado foi obtido a partir da concatenação dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$ e tem $d_{free} = 10$. As matrizes geradoras semi-infinitas dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$ são dadas, respectivamente, por:

$$G_{C_1} = \begin{bmatrix} 11 & 01 & 11 & & \\ & 11 & 01 & 11 & \\ & 11 & 01 & 11 & \\ & \ddots & \ddots & \ddots & \end{bmatrix}$$

e

$$G_{C_2} = \begin{bmatrix} 0110 & 0111 & & & \\ 1011 & 1010 & & & \\ & 0110 & 0111 & & \\ & 1011 & 1010 & & \\ & & 0110 & 0111 & \\ & & 1011 & 1010 & \\ & & \ddots & \ddots & \end{bmatrix},$$

onde $\mathcal{C}_1$ tem $d_{free} = 5$ e $\mathcal{C}_2$ é um código catastrófico. Aplicamos as transformações armadilha de permutação no novo código $\mathcal{C}_2$ obtendo os resultados apresentados na Tabela 5.3.

Na Tabela 5.3 apresentamos a transformação armadilha A, a permutação associada a transformação armadilha B, denotada ρ, as submatrizes G'_0 e G'_1 do código $\mathcal{C}'_2$ (que é o código resultante da aplicação das transformações armadilha no código $\mathcal{C}_2$) e a distância livre do código resultante, d'_{free}. G_C, é a matriz geradora do CCC concatenado e $d'_{free}(C)$ é a distância livre do CCC concatenado resultante.

Após efetuar a aplicação das transformações armadilha ao CCC os seguintes resultados foram obtidos:

1. O d'_{free} do código resultante permaneceu igual ao d_{free} do código original, ou seja, o novo código é capaz de corrigir até quatro erros clássicos;

2. Houve uma diminuição no valor do d'_{free}, de 10 para 9. Mesmo assim, continuou com a mesma capacidade de correção, isto é, pode corrigir até quatro erros clássicos;

3. O valor do d'_{free} do código obtido é 8, havendo uma redução na capacidade de correção de erros do código.

4. Como aconteceu com o CCC concatenado $(4, 1, 3)$, apresentado na Seção 5.2, houve um aumento no valor do d'_{free} do novo código, passando de 10 para 12. Com isso, o novo CCC concatenado tem a capacidade de corrigir até 5 erros clássicos.

A	ρ	G'_0	G'_1	d'_{free}	G_C	$d'_{free}C$
02:01	id	17:02	02:17	$*$	15;17;02;15	10
	(2143)(657)	17:01	10:17	2	16;06;01;07	9
	(8142367)	15:06	05:16	$*$	13;15;05;13	11
	(4123)(76)	17:04	02:17	$*$	13;17;04;13	10
	(815)(62)(37)	03:16	07:12	$*$	13;03;07;15	11
	(82576)	13:06	11:16	5	15;01;03;07	9
	(6185)(42)(37)	07:12	06:13	$*$	15;12;06;00	
	(21375)(68)	17:12	10:07	3	05;05;03;17	8
	(218)(756)	07:12	05:16	5	12;16;00;00	
	(5138)(247)	15:13	12:05	5	06;04;03;17	9

Tabela 5.3: Transformação permutação aplicada ao novo CCC_1 concatenado.

Após o processo de aplicação das transformações armadilha ao CCC, estamos aptos a obter os CCQs equivalentes aos CCCs resultantes, procedendo da mesma forma que foi feito na Seção 5.1.

5.3.3 O Código CCC_3

Considere o código CCC_3 $(4, 1, 3)$ com matriz geradora semi-infinita dada por

$$GC_1 = \begin{bmatrix} 1110 & 1011 & 0101 & 1110 & & \\ & 1110 & 1011 & 0101 & 1110 & \\ & & 1110 & 1011 & 0101 & 1110 \\ & & & \ddots & \ddots & \ddots \end{bmatrix}.$$

Como os códigos CCC_1 e CCC_2, apresentados anteriormente, este CCC concatenado foi obtido a partir da concatenação dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$. O CCC_3 tem $d_{free} = 11$, sendo capaz de corrigir até cinco erros clássicos.

As matrizes geradoras semi-infinitas dos códigos $\mathcal{C}_1$ e $\mathcal{C}_2$ são dadas, respectivamente, por:

$$G_{C_1} = \begin{bmatrix} 11 & 01 & 11 & \\ & 11 & 01 & 11 \\ & 11 & 01 & 11 \\ & \ddots & \ddots & \ddots \end{bmatrix}$$

e

$$G_{C_2} = \begin{bmatrix} 1011 & 0101 & & \\ 0101 & 1011 & & \\ & 1011 & 0101 & \\ & 0101 & 1011 & \\ & & 1011 & 0101 \\ & & 0101 & 1011 \\ & & \ddots & \ddots \end{bmatrix},$$

sendo que $\mathcal{C}_1$ possui $d_{free} = 5$ e $\mathcal{C}_2$ é um código catastrófico. Aplicamos as transformações armadilha de permutação no novo código $\mathcal{C}_2$ obtendo os resultados que podem ser observados na Tabela 5.3.3.

Na Tabela 5.4, apresentamos a transformação armadilha A, a permutação associada a transformação armadilha B, denotada como ρ, as submatrizes G'_0 e G'_1 do código $\mathcal{C}'_2$ que é o código resultante da aplicação das transformações armadilha no código $\mathcal{C}_2$, a distância livre do código resultante, representado por d'_{free}. Temos também, G_C, a matriz geradora do CCC concatenado e a distância livre do CCC concatenado resultante, denotada $d'_{free}(C)$.

Após aplicar as transformações armadilha ao CCC_3, obtivemos os seguintes resultados:

1. O d'_{free} do código resultante permaneceu o mesmo, ou seja, igual ao d_{free} do código original, sendo capaz de corrigir até cinco erros clássicos;

2. Houve uma diminuição no valor do d'_{free}, que passou 11 para 10, e assim, passou a ser capaz de corrigir até quatro erros clássicos;

3. O valor do d'_{free} do novo código passou a ser igual a 9, o que o torna capaz de corrigir até quatro erros clássicos;

4. O valor do d'_{free} do código obtido foi igual a 7, havendo uma redução na capacidade de correção de erros do código, ou seja, o novo código é capaz de corrigir até três erros clássicos.

Após o processo de aplicação das transformações armadilha ao CCC, estamos aptos a obter os CCQs equivalentes aos CCCs resultantes, procedendo da mesma forma que foi feito na seção (5.1).

A	ρ	G'_0	G'_1	d'_{free}	G_C	$d'_{free}C$
02:01	id	13:05	05:13	$*$	16;13;05;16	11
	(2143)(657)	15:12	03:15	4	07;04;12;16	9
	(8142367)	16:11	03:16	4	07;04;11;15	9
	(4123)(76)	17:12	03:15	5	15;04;00;16	7
	(815)(62)(37)	05:13	13:14	4	16;14;02;07	9
	(82576)	13:05	06:15	5	16;16;03;13	10
	(6185)(42)(37)	14:17	16:01	3	03;00;02;17	7
	(21375)(68)	15:03	05:16	5	16;10;00;13	7
	(218)(756)	17:11	10:07	3	06;06;01;17	9
	(5138)(247)	16:07	14:03	5	11;10;12;17	9

Tabela 5.4: Transformação permutação aplicada ao novo CCC_3 concatenado.

Capítulo 6

Conclusão

Neste trabalho foram apresentadas algumas transformações armadilhas capazes de propiciar a diminuição da capacidade de correção de erros dos códigos convolucionais ótimos de memória unitária. Além disso, através da aplicação destas transformações armadilhas, foram descobertos novos códigos convolucionais quânticos $[4, 1, 3]$ com d_{free} maior do que o já existente, apresentado por Almeida e Palazzo [17], que possui $d_{free} = 9$.

A seguir, apontamos as principais considerações finais de cada capítulo.

No Capítulo 2, fizemos uma abordagem geral sobre as ferramentas utilizadas para o desenvolvimento deste trabalho. Foram apresentados os conceitos básicos sobre códigos convolucionais e sistemas criptográficos.

No Capítulo 3 apresentamos o problema de determinação de códigos ótimos de memória unitária, que consiste na resolução do Problema da Mochila. Apresentamos também, o método do cripto-sistema de chave pública, que faz uso das transformações armadilhas e estabelecemos as propriedades destas transformações que tem a finalidade de reduzir a distância livre do código convolucional a um valor especificado ou desejado para a aplicação.

Além destes transformações, foi abordado também, que pode ser adicionado um grau a mais de dificuldade com a inserção de erros na mensagem no ponto de envio, gerando no ponto de recepção uma quantidade de erros na mensagem igual ao poder de correção do código original.

Ainda no Capítulo 3, realizamos a análise de complexidade total do sistema. Para isto, foram levados em consideração três fatores, a saber: resolver o Problema da Mochila, para encontrar o código convolucional ótimo de memória unitária; o grau de dificuldade para encontrar as matrizes A e B; e a complexidade do algoritmo de decodificação e decifragem. Assim, obtemos a complexidade do cripto-sistema de chave pública através de três fatores e verificamos que esta complexidade cresce exponencialmente com o comprimento da entrada do sistema k, e portanto torna o sistema seguro para aplicações com k muito grande.

O Capítulo 4 consiste dos resultados das aplicações das transformações armadilha aos códigos convolucionais. Foi abordado o processo de busca pelas transformações armadilha e foram apresentadas as transformações escolhidas, a saber: as representações matriciais dos elementos

do grupo de permutações, matrizes triangulares superiores e as matrizes de Hadamard.

As representações matriciais dos elementos do grupo de permutação foram aplicadas aos códigos de taxa $r = 2/3$, $r = 2/4$ e $r = 2/8$. Em todos os casos, obtivemos 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. As matrizes triangulares superiores, foram aplicadas aos códigos convolucionais de memória unitária, com taxas $r = 2/4$ e $r = 2/8$. Encontramos algumas matrizes que são capazes de reduzir a capacidade de correção de erros dos códigos. Da mesma forma, as matrizes de Hadamard aplicadas aos códigos convolucionais de memória unitária com taxa $r = 2/4$ e $r = 2/8$, forneceram códigos com capacidade de correção menor que a dos códigos originais. Após analisar o efeito causado por cada transformação armadilha, separadamente, fizemos a comparação entre os resultados.

Além disso, com o objetivo de analisar a vulnerabilidade dos códigos convolucionais MU e os códigos convolucionais MUP, aplicamos as mesmas transformações armadilha de permutação a estes dois códigos e observamos que os códigos convolucionais MU tem uma maior sensibilidade à perturbação que os códigos convolucionais MUP.

O Capítulo 5, apresenta o CCQ $[4, 3, 1]$ associado ao CCC $(4, 3, 1)$ que possui $d_{free} = 9$. Este CCQ é obtido a partir do código CCC que é resultante da concatenação de dois CCCs, sendo um $\mathcal{C}_1$ $(2, 1, 2)$ e outro $\mathcal{C}_2$ $(4, 2, 1)$. Aplicamos as transformações armadilha de permutação ao código $\mathcal{C}_2$ $(4, 2, 1)$, obtendo novos CCCs $\mathcal{C}_2'$. Em seguida concatenamos estes CCCs $\mathcal{C}_2'$ obtidos com o CCC $\mathcal{C}_1$ $(2, 1, 2)$, e assim, obtivemos novos CCCs $(4, 3, 1)$ concatenados. Como consequência, os seguintes resultados foram encontrados:

1. O d_{free} do novo CCC $(4, 3, 1)$ concatenado permaneceu igual ao d_{free} do CCC $(4, 3, 1)$ concatenado original;

2. O d_{free} do novo CCC $(4, 3, 1)$ concatenado foi menor que o d_{free} do CCC $(4, 3, 1)$ concatenado original;

3. O d_{free} do novo CCC $(4, 3, 1)$ concatenado foi maior que o d_{free} do CCC $(4, 3, 1)$ concatenado original.

Acreditamos, então, que o objetivo proposto pela aplicação das transformações armadilha foi alcançado. Além disso, obtivemos outros bons resultados: códigos concatenados com capacidade de correção maior que o código CCC $(4, 3, 1)$. Consequentemente, estes novos códigos CCCs $(4, 3, 1)$ são associados à novos códigos CCQs $[4, 3, 1]$.

Em seguida, utilizamos os códigos com d_{free} maior que o código original, ou seja $d_{free} = 10$ e $d_{free} = 11$, aplicando as transformações armadilha com o objetivo de reduzir a capacidade de correção de erros para a utilização destes no sistema criptográfico.

Dentre as possíveis linhas de trabalhos a serem realizados sugerimos:

- Análise de complexidade para a versão quântica do sistema criptográfico de chave pública apresentado;

- Expansão do trabalho para outras taxas, com possibilidade de generalização.
- Utilizar transformações armadilha pertencentes a grupos não-comutativos para a versão clássica;
- Utilizar codificadores convolucionais periodicamente variantes no tempo para a versão quântica.

Bibliografia

[1] W. Diffie, and M. E. Hellman,“ New direction in cryptography”, *IEEE Trans. Inform. Theory*, IT-22, pp. 644 – 654, Nov. 1976.

[2] W. Diffie, and M. E. Hellman, “Privacy and authentication: An introdution to cryptography”, *Proceedings of the IEEE*, vol. 67, No. 3, pp. 397 – 427, Nov. 1979.

[3] A. Gill, *Linear Sequential Circuit - Analysis, Synthesis, and Applications*, McGraw- Hill, New York, 1966.

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

[5] G. S. Lauer,“ Some optimal partial-unit-memory codes”, *IEEE Trans. Inform. Theory*, IT-25(2), pp 240 – 243, 1979.

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

[7] J. L. Massey and M. K. Sain, “Inverses of linear sequential circuits”, *IEEE Trans. Comp.*, C-17, pages 330 – 337, 1968.

[8] R. J. McEliece, “A public key system based on algebraic coding theory”, *JPL DSN Progress Rep.*, 1978.

[9] R. Palazzo, Jr., “Cryptographic systems based on trellis codes”, 3*rd Brazilian Symposium on Telecommunications*, São José dos Campos, Brazil, Sept. 2 – 4, 1985.

[10] R. Palazzo, Jr., *Códigos de Treliça Fixos e Variantes no Tempo*, Tese de Livre Docência, Universidade Estadual de Campinas. Campinas, Brazil, 1987.

[11] R. Palazzo, Jr., “Linear unit-memory codes - A knapsack problem?”, *2nd Swedish-USSR Intl. Workshop on Inform. Theory*, Granna, Sweden, 1985.

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

[13] R. Palazzo Jr., B. F. Uchôa Filho, J. P. Arpasi, *Notas de Aula - Códigos Convolucionais*, Departamento de Telemática, FEEC - UNICAMP, Campinas, São Paulo, 2005

[14] W. Finamore, "Criptografia", *International Symposium on Information and Coding Theory*, July 27 to August 1, Campinas, Brasil, 1987.

[15] R. J. McEliece."A public key system based on algebraic coding theory",*JPL DSN Progress Rep.*, 1978.

[16] A. C. A. de Almeida, *Códigos Convolucionais Quânticos Concatenados*, Tese de Doutorado, Universidade Outubro de Campinas (UNICAMP), Brasil, Outubro de 2004.

[17] A. C. A. de Almeida and R. Palazzo Jr., "A Concatenated [(4, 1, 3)] Quantum Convolutional Cod", 2004 IEEE *Information Theory Workshop*, San Antonio, Texas, 2004.

[18] C. C. Ribeiro. *Parallel Computer Models Combinatorial Algorithms*, School on Combinatorial Optmization, Rio de Janeiro, Brasil, Julho $8-9$, 1985.

[19] A. Dholakia, *Introduction to Convolutional Codes with Applications*, Kluwer Academic Publishers, Norwell, MA, 1994.

[20] A. J. Viterbi, "Convolutional codes and their performance in communication systems", *IEEE Trans. Comm.*, COM-19(5), pp $751-771$, 1971.

[21] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, New York, 1979.

[22] A. Garcia, Y. LEQUAIN *Elementos de Álgebra*. Projeto Euclides. IMPA, Rio de Janeiro, 2002.

[23] D. Haickel, *Uma propos ção e Análise de Complexidade de um Criptossistema de Chave Pública Utilizando Códigos Convolucionais de Memória Unitária.* , Dissertação de Mestrado, Universidade Outubro de Campinas (UNICAMP), Brasil, Abril de 1991.

[24] D. J. Costello Jr, *Construction of Convolutional Codes for Sequential Decoding*, PhD Thesis - University of Notre Dame, Notre Dame, IN, 1969.

[25] D. J. Costello Jr."A construction technique for random-error-correcting convolutional codes", *IEEE Trans. Inform. Theory*, IT-15(5), pp $631-636$, 1969.

[26] J. C. Silva Filho, W. C. Borelli e E. M. R. Marques, " Classe de equivalência de códigos de treliça via partiçõoes de reticulados", VII ERMAC, Recife, 2007.