Transformações Armadilhas Aplicadas aos Códigos Convolucionais Clássicos

Polyane Alves Santos e Reginaldo Palazzo Jr.

***Resumo*— Neste trabalho são apresentadas algumas transformações armadilhas que são aplicadas nas submatrizes dos códigos convolucionais clássicos, com o intuito de diminuir o d_{free}, para a obtenção de um alto grau de privacidade na informação a ser transmitida. Os códigos convolucionais utilizados são códigos ótimos de memória unitária que foram obtidos tendo como passo fundamental a resolução do Problema da Mochila.**

***Palavras-Chave*— Criptografia, Códigos Convolucionais, transformações Armadilhas.**

***Abstract*— This work presents some transformations trapdoors that are applied at submatrices of codes convolutionais with order to reduce the d_{fre}, to obtain a high degree of privacy in the information being transmitted. The codes convolutionais used are greats codes of unit memory which were obtained with the key step to solving the The Knapsack problem.**

***Keywords*— Encryption, codes Convolucionais, transformations Traps.**

I. Introdução

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

Na criptografia, existem dois métodos pelos quais podem ser feita a cifragem e a decifragem através dos criptossistemas, a saber, o método convencional e o método de chave pública. O primeiro modelo de chave pública na criptografia foi apresentado por Diffie e Hellman em 1976 [1]. Neste trabalho utilizamos a proposta de um criptossitema de chaves públicas que faz uso do método knapsack.Este sistema é baseado em códigos convolucionais de memória unitária aonde busca-se explorar todo seu grau de complexidade inerente ao processo de busca por códigos ótimos.

II. Códigos Convolucionais

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) [6], os sinais escolhidos de um corpo finito $GF(q)$ (na prática, o corpo binário $GF(2)$) são aplicados simultaneamente a todos os terminais de entrada em instantes discretos de tempo.

Os códigos convolucionais podem ter representações discreta, polinomial ou gráficas que podem proporcionar um melhor entendimento da operação de codificação convolucional.

Um codificador convolucional de taxa $R = k/n$ é representado por nk sequências 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 $i = 1, \ldots, n$. Como já é indicada pelo nome, a operação de codificação convolucional, é a convolução discreta da sequência de informação com as sequências de geradores, e é expressa como

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

para $j = 1, \ldots, n$ onde $*$ é a *operação de convolução*.

A codificação convolucional pode ser escrita como

$$\mathbf{v} = \mathbf{uG}, \tag{2}$$

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

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

onde os espaços em branco indicam zeros e

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

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

A. Distância Livre

Define-se a distância livre d_{free} de um código convolucional como sendo a menor distância de Hamming entre quaisquer duas sequências codificadas provinientes de dus sequências de informação distintas. Ou seja,

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

Polyane Alves santos e Reginaldo Palazzo Jr. Departamento de Telecomunicações, Universidade Estadual de Campinas, Campinas, Brasil, E-mails: polyane@dt.fee.unicamp.br, palazzo@dt@fee.unicamp.br.

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

B. Codificadores Catastróficos e Não Catastróficos

Massey [5] denominou como codificadores *catastróficos* aqueles que geram uma palavra-código de peso finito para uma sequência de informação de peso infinito.

Massey e Sain [3] 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:

$$\text{mdc}\left[\mathbf{g}^{(j)}(D), \quad j = 1, 2, \ldots, n\right] = 1 \tag{3}$$

onde mdc é o *máximo divisor comum*.

C. Código Convolucional 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 ao $GF(q)$ definidos por

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

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 $GF(q)$, geradoras de um código convoluciaonal (n_0, k_0) de memória m definido pela seguinte regra de codificação:

$$\mathbf{v}_t = \mathbf{u}_t\mathbf{G}_0 + \mathbf{u}_{t-1}\mathbf{G}_1 + \mathbf{u}_{t-m}\mathbf{G}_m, \tag{4}$$

e por convenção, $t > 0$ e $x_{-1} = \mathbf{0}$, onde $\mathbf{0}$ é definido como sendo a matriz linha com todos os seus elementos iguais a zero e a operação sendo realizada é sobre $GF(q)$.

Lee chamou o código definido por,

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

$$G'_0 = \begin{bmatrix} \mathbf{G_m} & \mathbf{G_0} & \cdots & \mathbf{G_0} \\ \mathbf{G_{m-1}} & \mathbf{G_m} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{G_m} \end{bmatrix}$$

de *código de memória unitária* e definiu a *complexidade de estados* de um codificador binário como sendo 2^{mk_0}, pois exitem 2^{mk_0} estados distintos do codificador.

III. Sistemas Criptográficos

Os sistemas Criptográficos podem classificados com 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 via sistemas convencionais.

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

1) A privacidade, tem por objetivo evitar que a informação seja interceptada no canal por pessoas não autorizadas, as quais são denominadas de criptoanalista.
2) A autenticidade, busca evitar que a informação seja alterada pelo criptoanalista.

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

A. Sistemas Criptográficos de Chave Pública

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

1) Os criptossistemas de chave pública e
2) Os sistemas com distribuição pública de chaves.

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

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

$$E_k : \{M\} \rightarrow \{C\} = E_k\{M\} \tag{5}$$

$$D_k : \{C\} \rightarrow \{M\} = D_k\{C\} \tag{6}$$

sobre um espaço de mensagens finito M, tal que:

1) Para cada $k \in \{K\}$, D_k é uma transformação inversa de E_k;
2) Para cada $k \in \{K\}$ e $m \in \{M\}$, as operações E_k e D_k são simples do ponto de vista computacional;
3) Para cada $k \in \{K\}$, é computacionalmente complexo descobrir a transformação D_k a partir de E_k;
4) É computacionalmente simples a obtenção do par de transformações inversas E_k e D_k.

IV. Knapsack Binário para Sistema Criptográfico de Chave Pública

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

A segurança do 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 o peso de Hamming da palavra código ramos diminua, causando a diminuição no poder de correção dos erros;
2) O fato de que o Problema da Mochila do tipo binário, tem que ser resolvido.

A. Descrição do Método do Sistema Criptográfico

As funções armadilhas são transformações aplicadas às submatrizes geradoras do código, G_0 e G_1, com a finalidade de reduzir o poder de correção deste código à uma valor desejável ou mesmo fixado pela aplicação.

Estas transformações aplicadas às submatrizes geradoras, são feitas através de um par de matrizes A e B, com dimensões $k \times k$ e $n \times n$, respectivamente.

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

$$G'_0 = AG_0B, \quad e \quad G'_1 = AG_1B. \tag{7}$$

Como o sistema cripográfico é de chave pública, as matrizes

$$G' = [G'_0 \; G'_1] \tag{8}$$

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

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

No processo de decodificação empregado à sequência de saída do canal, y_{te}, considerando que o canal é livre de ruído temos que y_{te} (valor estimado de y_{te}) é igual ao valor de y_{te} na entrada do canal. Utilizamos então a transformação inversa de B, como explicitado na equação (9)

$$y_{te}.B^{-1} = (x_tA)G_0 \oplus (x_{t-1}.A)G_1. \tag{9}$$

Aplicando o algoritmo de Viterbi, x_t é consequentemente obtido.

Observe que para que este criptosistema seja quebrado, é necessário que o criptoanalista primeiramente resolva o problema da Mochila relacionado a encontrar as submatrizes geradoras do código ótimo, G_0 e G_1, e depois encontrar as transformações A^{-1} e B^{-1}. Estes dois fatos não apresentam soluções triviais

V. Transformações Armadilhas

Nesta seção serão definidas e apresentadas transformações armadilhas a serem usadas no sistema criptográfico de chave pública, bem como os resultados obtidos com suas aplicções.

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 criptossistema quanto à sua vulnerabilidade.

Todas as matrizes aqui apresentadas, (tanto as transformações armadilhas, como as matrizes geradoras) estão na forma octal, onde cada linha é separada por dois pontos (:). Como exemplo, considere uma matriz

$$G = [13 : 06 : 03],$$

é a representação octal da matriz,

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

A. Grupo de Permutações - S_n

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

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

Definição 2 (Permutações): O conjunto de todas estas 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.

B. Matriz de Hadamard

Outra transformação armadilha utilizada é a matriz de Hadamar que é definida como:

Definição 3 (Matriz de Hadamard): É uma matriz quadrada H de ordem $n \times n$ composta de $+1's$ e $-1's$ tal que

$$HH^T = nI.$$

C. Matrizes Triangulares Superiores

Definição 4 (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 $i > j$.

VI. Análise das Transformações

Nesta seção, as transformações armadilhas já definidas, serão aplicadas aos códigos convolucionais que serão especificados em cada caso. Após essa aplicação, serão apresentados e comentados os resultados obtidos e finalmente será feita uma comparação entre estes resultados.

A. Permutações

As representações matriciais dos elementos do grupo de permutação foram aplicadas como transformações armadilhas nos códigos convolucionais ótimos de memória unitária de taxa $R = 2/3$, $R = 2/4$ e $R = 3/4$.

A seguir, definiremos algumas matrizes que serão utilizadas como matriz de transformação A, que serão utilizadas para os códigos de taxa $R = 2/3$ e $R = 2/4$

$$A_1 = [02 : 01] \quad , \quad A_2 = [01 : 02],$$
$$A_3 = [03 : 01] \quad e \quad A_4 = [02 : 03].$$

1) Código taxa R=2/3: Para o código convolucional $(3, 2, 1)$ de submatrizes geradoras

$$G_0 = [04 : 07] \quad e \quad G_1 = [05 : 03]$$

e $d_{free} = 3$, foram utilizadas as transformações armadilhas A_1, A_2, A_3 e A_4 já apresentadas. E as transformações armadilhas B que são as representações matriciais dos elementos do grupo de permutação S_6.

Aqui, na maioria dos casos a distância livre permaneceu a mesma. Porém, foram encontradas matrizes capazes de destruir a capacidade de correção do código, deixando-o apenas capaz de detectar erros. Em algumas situações o código obtido foi um código de memória unitária parcial.

Na Tabela I, 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}.

TABELA I

D_{free} DOS CÓDIGOS RESULTANTES COM TAXA $r = 2/3$.

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

B. Código taxa R=2/4

O código ótimo de memória unitária e taxa $R = 2/4$ aqui considerado possui as submatrizes geradoras

$$G_0 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad e \qquad G'_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

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

Na TabelaII a matriz A (em octal), a permutação referente a matriz B, as submatrizes geradoras G_0 e G_1 (também representadas em octal), e o d'_{free} que é a distância livre do código resultante.

TABELA II

D_{free} DOS CÓDIGOS RESULTANTES DE TAXA $2/4$

A	ρ	G'_0	G'_1	d'_{free}
02:01	(312)(645)(78)	13:04	14:17	3
	(81647)(523)	12:15	15:06	4
	(21457368)	17:12	01:16	3
	(5184326)	06:15	15:03	4
01:02	(312)(645)(78)	04:13	17:14	3
	(81647)(523)	15:12	06:15	4
	(21457368)	12:17	16:01	3
	(5184326)	15:06	03:15	4
03:01	(312)(645)(78)	17:13	03:14	3
	(81647)(523)	07:12	13:15	4
	(21457368)	05:17	17:11	3
	(5184326)	13:06	16:15	4
02:03	(312)(645)(78)	13:17	14:03	3
	(81647)(523)	12:07	15:13	4
	(21457368)	17:05	01:17	3
	(5184326)	06:13	15:16	4

Ao aplicar as matrizes de permutação foram obtidos resultados diferentes:

1) O d_{free} continuou o mesmo, mantendo a capacidade de correção de erros. Aqui, os códigos obtidos são equivalentes ao código original.
2) O d_{free} teve uma leve queda, reduzindo de 5 para 4. Com isso, passou a corrigir 1 erro e detectar alguns.
3) Houve uma redução maior do d_{free}, que de 5 caiu para 3. Assim, ao invés de corrigir até 2 erros, o novo código é capaz de corrigir apenas 1 erro.

Os dois últimos resultados podem ser observados na TabelaII.

Outro fato a ser observado é que a matriz A não interfere no valor do d_{free} dos códigos obtidos, pois independente de qual seja esta matriz, a distância livre continua a mesma. Percebe-se que o seu papel no processo é apenas de embaralhar os bits de informação sem, entretanto, alterar os elementos das submatrizes.

C. Código taxa R=3/4

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

$$G_0 = [03 : 04 : 16] \quad e \quad G_1 = [11 : 12 : 14]$$

e possui distância livre igual a 4.

As transformações armadilhas A e B aqui utilizadas, foram os elementos dos grupos de permutação S_3 e S_8, respectivamente.

Na Tabela (III) apresentamos alguns elementos do grupos de permutação S_8, denominado por ρ_B, cuja representação matricial é a transformação B que são combinadas com todos os elementos do grupo S_3, representados por ρ_A. Nela também estão (em octal) G'_0 e G'_1 que são as submatrizes resultantes após a aplicação de tais tranformações, e a distância livre do novo código convolucional.

TABELA III

D_{free} DOS CÓDIGOS RESULTANTES DE TAXA $3/4$

ρ_A	ρ_B	G'_0	G'_1	d'_{free}
id	(7185)(3264)	06:16:15	14:02:05	2
	(4125863)	11:01:06	12:15:13	3
	(218)(736)	06:13:03	03:01:16	3
(21)	(7185)(3264)	16:06:15	02:14:05	2
	(4125863)	01:11:06	15:12:13	3
	(218)(736)	13:06:03	01:03:16	3
(32)	(7185)(3264)	06:15:16	14:05:02	2
	(4125863)	11:06:01	12:13:15	3
	(218)(736)	06:03:13	03:16:01	3
(31)	(7185)(3264)	15:16:06	05:02:14	2
	(4125863)	06:01:11	13:15:12	3
	(218)(736)	03:13:06	16:01:03	3
(312)	(7185)(3264)	15:06:16	05:14:02	2
	(4125863)	06:11:01	13:12:15	3
	(218)(736)	03:06:13	16:03:01	3
(132)	(7185)(3264)	02:07:11	15:03:05	2
	(4125863)	16:15:06	02:05:14	3
	(218)(736)	01:06:11	15:13:12	3

Diante dos resultados podemos observar que ocorreram as seguintes situações:

1) Alguns códigos resultantes são catastróficos;
2) Houveram 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 caiu de 4 para 3;
4) Já outros códigos tiveram uma redução do valor da distância livre de 4 para 2, perdendo a capacidade de correção de erros.

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.

D. Matrizes Triangulares Superiores

Por suas propriedades, uma outra transformação que utilizamos como armadilha B, foram matrizes triangulares superiores. Os códigos convolucionais ótimos de memória unitária considerados foram de taxa $R = 2/4$ de submatrizes geradoras

$G_0 = [15 : 03] \quad e \quad G_1 = [14 : 07]$

As matrizes triangulares aqui consideradas como transformação B são,

$T_1 = [14 : 04 : 02 : 01] \quad , \quad T_1 = [17 : 07 : 03 : 01]$
$T_3 = [13 : 07 : 02 : 01] \quad \text{e} \quad T_4 = [13 : 05 : 03 : 01]$

TABELA IV
TRANSFORMAÇÃO TRIANGULAR SUPERIOR, $R = 2/4$

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

Como pode ser observado na Tabela (IV), foram obtidos dois resultados diferentes:

1) O d_{free} do código foi reduzido de 5 para 4 e assim, possou a ser capaz de corrigir apenas 1 erro e dectectar algumas palavras com até 2 erros;
2) O código perdeu a capacidade de correção de erro, passando a corrigir 1 erro ao invés de 2, seu d_{free} que era 5 passou a ser 3.

Mais uma vez, percebemos que a mtriz A não age de forma alguma na destruição da capacidade de correção de erro dos códigos, sendo sua única função o embraralhamento dos bits.

E. Hadamard

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

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

$$G_0 = [15 : 03] \quad e \quad G_1 = [14 : 07]$$

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

$$G_0 = [370 : 037] \quad e \quad G_1 = [174 : 237]$$

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

TABELA V
TRANSFORMAÇÃO HADAMARD DO TIPO SILVESTER DE ORDEM 4

Codigo	A	G'_0	G'_1	d_{min}	d'_{min}
(4,2,1)	A_1	03:05	05:00	5	2
	A_2	06:05	05:00	5	2
	A_3	03:06	05:05	5	2

Como pode ser observado na Tabela V para o código de memória unitária e 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.

TABELA VI
TRANSFORMAÇÃO HADAMARD DO TIPO SILVESTER DE ORDEM 8

Codigo	A	G'_0	G'_1	d_{min}	d'_{min}
(8,2,1)	A_1	360:231	252:146	10	8
	A_2	151:231	314:146	10	8
	A_3	360:151	252:314	10	8

TABELA VII
TRANSFORMAÇÃO HADAMARD DO TIPO PALEY DE ORDEM 8

Código	A	G'_0	G'_1	d_{min}	d'_{min}
(8,2,1)	A_1	232:321	306:056	10	8
	A_2	113:321	350:056	10	8
	A_3	360:151	252:314	10	8

De forma análoga, acontece com o código convolucional de memória unitária de taxa $R = 2/8$ apresentado nas Tabelas VI e VII que perde a capacidade de correção de erro resultando

em um d_{free} menor que o do código original. Neste, o $d_{free} = 10$ e após a aplicação das transformações o $d'_{free} = 8$ sem importar quem seja A e B. Aqui também ocorre uma redução na dimensão do espaço vetorial em todos os casos observados.

VII. CONCLUSÕES

Observamos que com a utilização de transformações armadilhas estruturadas como os elementos do grupo de permutação, as matrizes de Hamadard e as matrizes triangulares superiores , é possível a redução dos valores de d_{free} a níveis desejados. Concluimos também, que as tranformações armadilhas A não reduz a capacidade de correção de erro dos códigos, deixando tal ação a cargo da transformação B.

REFERÊNCIAS

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

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

[3] J. L. Massey and M. K. Sain. Inverses of linear sequential circuits. IEEE Trans. Comp., C-17, pages $330 - 337$, 1968.

[4] L. N. Lee. Short unit-memory byte-oriented binary convolutional codes having maxi- mal free distance. IEEE Trans. Inf. Theory, IT-$22(3)$, pages $349 - 352$, 1976.

[5] J. L. Massey. Catastrophic error-propagation in convolutional codes. Proc. of the 11th Midwest Symp. Cir. Th., pages $583 - 587$, Notre Dame, IN, 1968.

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