**UNIVERSIDADE DE SÃO PAULO**

**INSTITUTO DE MATEMÁTICA E ESTATÍSTICA**

# TRABALHO DE CONCLUSÃO DE CURSO

# Criptografia Pós-Quântica baseada em Códigos Corretores de Erros

Aluno: Gervásio Protásio dos Santos Neto

Orientador: Routo Terada

**Resumo**

Este trabalho de conclusão de curso consiste do estudo de criptossistemas baseados em códigos corretores de erros. Acredita-se que tais sistemas não são vulneráveis a ataques por computadores quânticos e seriam, portanto, alternativas viáveis para criptossistemas baseados no Problema do Logaritmo Discreto. Foca-se no criptossistema de McEliece, buscando expor e comparar as classes de códigos que são utilizadas na sua implementação, bem como introduzir os conceitos necessários de Teoria dos Corpos e Teoria dos Códigos que permitam uma análise matemática profunda. Estão inclusas também reduções de segurança que mostram que atacar este esquema criptográfico equivale a resolver problemas conhecidamente difíceis. Além disso, comenta-se os ataques genéricos que podem ser realizados contra ele.

**Palavras-chave:** criptografia pós-quântica, teoria dos Códigos, criptossistema de McEliece, códigos de Goppa, códigos MDPC.

# Sumário

**1 Introdução** **5**
1.1 Objetivos . . . . . 6
1.2 Métodos . . . . . 7
1.3 Organização do Texto . . . . . 7

**2 Códigos Corretores de Erros** **8**
2.1 Conceitos Básicos . . . . . 8
2.2 Códigos Lineares . . . . . 10

**3 Códigos de Goppa** **15**
3.1 Definição . . . . . 15
3.2 Matrizes Características . . . . . 17
3.2.1 Matriz Teste de Paridade . . . . . 17
3.2.2 Matriz Geradora . . . . . 21
3.3 Códigos de Goppa Binários Irredutíveis . . . . . 22
3.4 Decodificação de um CGBI . . . . . 25

**4 Criptossistema de McEliece** **28**
4.1 Componentes do Criptossistema . . . . . 28
4.2 Chave Pública e Chave Privada . . . . . 29
4.3 Criptografia e Decriptografia . . . . . 30
4.4 Reduções de Segurança . . . . . 30

**5 Códigos de Goppa Diádicos e Quase-Diádicos** **34**
5.1 Introdução . . . . . 34
5.2 Definições e Teoremas Básicos . . . . . 34
5.3 Códigos de Goppa Quase-Diádicos . . . . . 37

**6 Códigos LDPC e MDPC** **39**
6.1 Códigos LDPC . . . . . 39
6.1.1 Introdução . . . . . 39
6.1.2 Decodificação . . . . . 40
6.1.3 Uso no Criptossistema de McEliece . . . . . 43
6.2 Códigos MDPC . . . . . 43
6.2.1 Introdução . . . . . 43
6.2.2 Códigos MDPC Quase-Cíclicos - QC-MDPC . . . . . 44
6.2.3 Construindo códigos QC-MDPC . . . . . 44
6.2.4 McEliece QC-MDPC . . . . . 45

**7 Comparações entre variantes do McEliece** **47**

**8 Ataques ao McEliece** **49**

8.1 Ataque por conjunto de informação . . . . . 49

8.2 Ataque por palavra de peso mínimo . . . . . 52

8.3 Ataque por mensagem parcialmente conhecida . . . . . 52

8.4 Ataque por mensagem relacionada . . . . . 54

8.5 Ataque por reação . . . . . 54

8.6 Ataque por maleabilidade . . . . . 55

**9 Conclusões** **56**

# 1 Introdução

O assunto abordado neste trabalho de conclusão de curso (TCC) se enquadra nas áreas de Criptografia e Teoria da Computação e se interessa em pesquisar a fundo as alternativas baseadas em códigos corretores de erros para os sistemas criptográficas vigentes.

Atualmente, os padrões de encriptação mais populares no mundo são o RSA [R. L. Rivest, 1978] e criptossistemas baseados em curvas elípticas (em inglês: Elliptic Curves Criptossystems, ou **ECC**). [Miller, 1986]

A segurança destes modelos baseia-se no **Problema do Logarítmo Discreto** (PLD) [Terada, 2000], que de forma generalizada consiste em, dado um grupo abeliano $G$ e $g, s \in G$, calcular $t \in \mathbb{N}$ tal que

$$s = g^t.$$

No caso do RSA o grupo $G$ é o grupo multiplicativo de $\mathbb{Z}_n$, onde $n$ é o produto de dois primos. Para ECCs o grupo consiste de uma curva elíptica sobre um Corpo de Galois, munida de operações sobre os pontos e um ponto no infinito servindo como identidade. Comumente, e para os propósitos deste trabalho, assume-se que o corpo tem característica 2.

Acredita-se que resolver o PLD em computadores clássicos é difícil [Terada, 2000]. Não existe atualmente nenhum algoritmo polinomial para sua resolução.

Contudo, em 1994, Peter Shor propôs um algoritmo quântico que era capaz de resolver o PLD em tempo polinomial [Shor, 1997]. Se computadores quânticos vierem a se tornar viáveis e comercialmente disponíveis, o Algoritmo de Shor pode ser utilizado para facilmente descobrir as chaves secretas de sistemas que usam curvas elípticas ou RSA.

Conforme tornou-se claro que os métodos mais populares de segurança estariam ameaçados por possíveis avanços tecnológicos em Computação Quântica, a comunidade de segurança começou linhas de pesquisa em algoritmos criptográficos que não dependem do Problema do Logaritmo Discreto, a chamada *criptografia pós-quântica*.

Enquanto há diversas propostas para possíveis novos padrões de segurança pós-quântica, como por exemplo sistemas baseados em sistemas de variáveis multinomiais [Buchmann et al., 2004] ou sistemas baseados em hash [Merkle, 1979], uma das propostas mais promissora parece ser a de sistemas baseado em códigos corretores de erros [Augot et al., 2015].

O primeiro criptossistema baseado em códigos foi proposto por McEliece [McEliece, 1978] e leva seu nome. Mais tarde, Niederreiter [Niederreiter, 1986] defi-

niu um alteração do sistema de McEliece que pode ser interpretada como seu dual.

Ambas propostas receberam pouca atenção na época. Isso se deu por dois motivos, um técnico e outro de percepção. Do ponto de vista técnico, criptografia baseada em códigos clássica precisa de chaves muito maiores que RSA ou ECCs para atingir o mesmo nível de segurança. Portanto, utilizá-los era ineficaz quando alternativas mais baratas estavam disponíveis. Por outro lado, a comunidade de criptografia teve dificuldade em aceitar o uso de códigos corretores de erros (normalmente utilizados para garantir que informação será legível apesar de interferências) poderiam ser utilizados de forma segura para ofuscação de mensagens.

## 1.1 Objetivos

O principal objetivo deste trabalho é estudar a fundo o esquema criptográfico descrito por McEliece, suas fraquezas e pontos em que pode ser melhorado.

O criptossistema de McEliece, como apresentado em [McEliece, 1978] utiliza uma família de códigos corretores de erros conhecidos como Códigos de Goppa binários [Goppa, 1970]. Para entendê-lo, vamos estudar os fundamentos da teoria dos códigos e da teoria dos corpos finitos, que são essenciais para a compreensão dessa classe de códigos.

Então, vamos discutir os esforços para reduzir o tamanho das chaves nesse esquema, baseando-nos primariamente na dissertação de doutorado de Misoczki [Misoczki, 2013]. Nesta etapa, introduziremos duas classes de códigos corretores de erros que não possuem uma estrutura algébrica subjacente e analisaremos os efeitos de utilizá-los no lugar de códigos de Goppa. Veremos também uma possível modificação à estrutura dos códigos de Goppa que permite a geração de chaves mais compactas.

Será então feita uma comparação entre a descrição clássica do criptossistema do McEliece e as variantes abordadas, tendo como principais pontos de comparação o tamanho da chave e a segurança do esquema criptográfico. Algumas variantes que não foram exploradas serão citadas para fins contextuais.

Por fim, serão apresentados os ataques clássicos contra este sistema, juntamente com breves análises de suas eficácias.

## 1.2 Métodos

Este trabalho possui um cunho mais teórico, não focando em implementações, mas sim em resultados teóricos e algoritmos. Alguns resultados práticos clássicos serão retirados da literatura, principalmente no que diz respeito à segurança das implementações do criptossistema de McEliece e ao tamanho das chaves utilizadas.

Diversos artigos, tanto clássicos nas áreas de criptografia e teoria dos códigos quanto resultados mais recentes foram utilizados para o embasamento técnico do trabalho.

## 1.3 Organização do Texto

Na Seção 2 fazemos uma introdução à teoria dos códigos corretores de erros, seus conceitos e resultados básicos. Na Seção 3, introduzimos Códigos de Goppa genéricos e Códigos de Goppa binários e irredutíveis e damos seu algoritmo de decodificação. Na Seção 4, definimos formalmente o Criptossistema de McElice e apresentamos seus algoritmos de criptografia e decriptografia, bem como suas reduções de segurança. Na Seção 5 é introduzida a variante quase-diádica dos Códigos de Goppa e seu uso no Criptossistema de McEliece é explicado. Na Seção 6, introduzimos códigos LDPC e MDPC, bem como códigos quase-cíclicos e mostramos como podem seus usos no criptossistema de McEliece. Na Seção 6 comparamos as variantes apresentadas neste trabalho e a prosposta criptográfica original. Na Seção 8 descrevemos os possíveis ataques contra o criptossistema. Por fim, a Seção 9 contém as conclusões finais do trabalho.

# 2 Códigos Corretores de Erros

Códigos corretores de erros são estruturas que surgem inicialmente na teoria de processamento de sinais como formas de transmitir de forma segura informações por um canal sujeito a ruídos. A ideia básica é que, se queremos transmitir uma mensagem $m$ por um canal onde há probabilidade dessa mensagem ser alterada, introduzimos algum grau de redundância nela de tal forma que conseguimos recuperar a original, mesmo que alguns erros de transmissão ocorrerem.

Um exemplo simples de um código corretor de erro é que, se queremos transmitir um único bit, podemos replicá-lo $n$ vezes e então, o receptor conseguiria recuperar, com alta probabilidade, simplesmente assumindo que a mensagem original corresponde à maioria dos bits recebidos. Por exemplo, se queremos transmitir o bit 0, podemos transmitir "00000"e o receptor seria capaz de entender a mensagem original mesmo que recebesse algo como "10010".

Isto, contudo leva a grandes desperdícios. Supondo que haja um custo associado à transmissão de cada bit, o método descrito acima vai consumir $n$ vezes mais recursos do que a transmissão simples. Enquanto há um limite para o quão eficiente uma transmissão pode ser (dado pelo *Limite de Shannon* [Shannon, 1949]), é difícil atingir essa codificação ótima.

Códigos específicos tem propriedades de detecção e correção de erros diferentes e a sua teoria tem uma linguagem própria. Então, nesta seção vamos apresentar os conceitos básicos da Teoria dos Códigos e alguns resultados que serão úteis no desenvolvimento deste trabalho.

## 2.1 Conceitos Básicos

**Definição 2.1.** *Os caracteres que compõem o código e serão transmitidos por um canal de comunicação constituem o* ***alfabeto*** *do código. Costuma-se denotar um alfabeto genérico por $A$.*

É possível transformar $A$ em um espaço métrico se impormos sobre ele a chamada *métrica de Hamming*, definida como:

**Definição 2.2.** *Dados $u, v \in A^n$, a* ***distância de Hamming*** *entre $u$ e $v$ é dada por:*

$$d(u, v) = |\{i : u_i \neq v_i; 1 \leq i \leq n\}|$$

Esta métrica é essencial para a análise de quantos erros um código é capaz de detectar e corrigir. A partir dela, podemos introduzir o conceito de *distância mínima* de um código.

**Definição 2.3.** *Seja $C \subset A^n$ um código, definimos a* ***distância mínima*** *de $C$, denotada por $d_C$, como:*

$$d_C = \min\{d(u,v) : u, v \in C; u \neq v\}$$

Com essa noção, podemos provar o seguinte teorema:

**Teorema 2.1.** *Um código com distância mínima $d_C$ consegue detectar no máximo $d_C - 1$ erros e corrigir no máximo $\left\lceil \frac{d_C-1}{2} \right\rceil$ erros.*

*Demonstração.* Primeiro vamos notar que se $D(u,n)$ denota o conjunto de pontos de $A^n$ a uma distância no máximo $n$ de $u \in C$, então, para quaisquer $u, v \in C$, temos $D(u, \left\lceil \frac{d_C-1}{2} \right\rceil) \cap D(v, \left\lceil \frac{d_C-1}{2} \right\rceil) = \emptyset$.

Isso acontece pois, se temos $x \in D(u, \left\lceil \frac{d_C-1}{2} \right\rceil) \cap D(v, \left\lceil \frac{d_C-1}{2} \right\rceil)$, então, pelo fato da distância de Hamming ser uma métrica, temos:

$$d(u,v) \leq d(u,x) + d(x,v) \leq \frac{d_C - 1}{2} \leq d_C - 1$$

O que, pela definição de $d_C$, é um absurdo.

Então, se ao transmitir uma mensagem $x \in C$ ocorrem até $d_C - 1$ erros, obtendo $x'$ temos que, mapeando $x'$ para a palavra mais próxima em $C$, obteremos apenas $x$. Se ocorrerem $d_C$ ou mais erros, pode acontecer de mapearmos $x'$ para o original errado.

Quanto a correção de erros, de ocorrem no máximo $t = \frac{d_C-1}{2}$ erros e recebemos $y$, temos que existirá apenas um $x \in C$ tal que $y \in D(x,t)$ e podemos decodificar $y$ como $x$. □

Uma operação sobre códigos que será útil é a *mudança de alfabeto*. Podemos trocar o alfabeto de um código sem alterar seus parâmetros (comprimento de uma palavra, número de elementos e distância mínima).

Se $A$ e $B$ são dois conjuntos finitos tais que existe uma bijeção $f : A \to B$, então podemos definir:

$$\varphi : A^n \to B^n$$
$$(a_1, \ldots, a_n) \to (f(a_1), \ldots, f(a_n))$$

Temos que $\varphi$ é uma bijeção de $A^n$ em $B^n$ e que a distância de Hamming é preservada. Ou seja, se temos $t = d(u,v), u, v \in A^n$, então $d(\varphi(u), \varphi(v)) = t$.

Isso é útil pois nos permite transformar um alfabeto arbitrária em um com uma estrutura matemática bem definida. Por exemplo, se temos um código $C$ definido

sobre um alfabeto $A$ tal que $|A| = p$, podemos obter um código $C'$ sobre $\mathbb{Z}_p$, o que nos permite uma maior liberdade nas definições. Temos interesse no caso particular em que $p$ é um número primo, pois neste caso $\mathbb{Z}_p$ constitui um corpo.

Daqui em diante neste trabalho, vamos supor, a não ser que dito o contrário, que todos os códigos estão definidos sobre uma alfabeto correspondente ao inteiros módulo $n$, para algum $n$. Com essa hipótese, podemos definir de forma clara o seguinte conceito:

**Definição 2.4.** *O **peso** de uma palavra $u \in A^n$, denotado por $\omega(u)$, é o número de posições não nulas de $u$. Ou seja:*

$$\omega(u) = |\{i : u_i \neq 0\}|$$

O conceito de peso nos permite redefinir distância. Uma vez que $d(u, v)$ são as posições em que $u, v$ diferem, fica claro o seguinte resultado:

**Lema 2.1.** $d(u, v) = \omega(u - v)$

## 2.2 Códigos Lineares

Agora que temos os conceitos básicos, que podem ser aplicados a qualquer tipo de códigos, vamos nos concentrar na análise das propriedades de uma classe específica de códigos, muito úteis na prática e que serão os que utilizaremos na criptografia de McEliece: os códigos lineares:

**Definição 2.5.** *Seja $K$ um corpo finito. Dizemos que um código $C \subset K^n$ é um **código linear** se $C$ é um subespaço vetorial de $K^n$.*

Se temos que $|K| = q$, e a dimensão $\dim_K C = k$, então, como a base de $C$ terá $k$ elementos e teremos $q$ possibilidades para cada elemento, podemos concluir que $|C| = q^k$.

Podemos também estender o conceito de peso:

**Definição 2.6.** *O **peso de um código** $C$, denotado por $\omega(C)$ é o peso da menor palavra de $C$, ou seja:*

$$\omega(C) = \min\{\omega(u) : u \in C\}$$

Isso nos permite definir a distância mínima de um código de uma forma mais prática:

**Lema 2.2.** *Seja $C$ um código linear. Então $d_C = \omega(C)$.*

*Demonstração.* Como $d_C$ é a distância mínima de $C$, devem existir $x, y \in C$ tais que $d(x, y) = d_C$ e para todo $u, v$ em $C$, $d(u, v) \geq d_C$.

Pelo **Lema 2.1**, sabemos que $d(x, y) = \omega(x - y)$, mas por outro lado, como $C$ é um espaço vetorial, ele é fechado sob subtração e temos $v = x - y \in C$. Então, devemos ter que $\omega(v) = \omega(C)$. Caso contrário, haveria $v' \in C$ tal $\omega(v') < \omega(v)$, mas como $C$ é um espaço vetorial, isso significaria que existem $a, b \in C$ tais que $v' = a - b$ e portanto teríamos:

$$d(a, b) = \omega(v') < \omega(v) = d(x, y)$$

Um absurdo, dada a minimalidade de $d(x, y)$. □

Essa definição nos permite computar a distância mínima de um código de forma mais eficiente. Ao invés de calcular $\binom{|C|}{2}$ distâncias de Hamming, podemos simplesmente calcular $|C|$ pesos.

Até agora, temos pensado em códigos apenas como conjuntos, mas como notado acima, esses conjuntos possuem tamanho exponencial na dimensão do subespaço que os define. Isso significa que essa representação não é computacionalmente adequada.

Duas formas mais úteis de representar códigos lineares vem da observação que tanto a imagem quanto o núcleo de uma transformação linear são subespaços vetoriais. Então, podemos representar o subespaço $C$ como uma imagem de uma matriz ou como núcleo de uma outra matriz.

Seja $\dim_K C = k$ e $\{v_i\}_{1 \leq i \leq k}$ uma base de $C$. Então $C$ será a imagem da seguinte transformação linear:

$$T : K^k \to K^n$$
$$(x_1, \ldots, x_k) \to \sum_{i=1}^{k} x_i v_i$$

A forma matricial desta transformação é a seguinte matriz $k \times n$:

$$G = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & v_{13} & \ldots & v_{1n} \\ v_{21} & v_{22} & v_{23} & \ldots & v_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & v_{k3} & \ldots & v_{kn} \end{pmatrix}$$

Essa matriz $G$, chamada de ***matriz geradora*** de $C$ nos permite gerar a partir de um elemento externo ao código, uma palavra correspondente através da operação:

$$G : K^k \to K^n$$
$$x \to xG$$

Como, por definição, $k < n$, podemos interpretar essa operação como introduzindo $n - k$ redundâncias na mensagem original, que podem então ser usadas para identificar se houve ou não erros de transmissão.

Enquanto a matriz geradora nos permite obter os membros do código, ela não nos oferece uma forma de identificar se um dado vetor de $K^n$ é ou não elemento do código.

Para obter uma operação que nos permita identificar se um dado vetor pertence ou não a um subespaço de forma eficiente, precisamos definir o *dual* de um código.

**Definição 2.7.** *O dual de um código $C$ é um código $C^\perp$ tal que*

$$C^\perp = \{y \in K^n : \langle x, y\rangle = 0; \forall x \in C\}$$

*onde $\langle \cdot, \cdot \rangle$ representa o produto interno do espaço vetorial, definido como $\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$*

Temos que $C^\perp$ também é um espaço vetorial (o espaço *ortogonal* de $C$). Então, $C^\perp$ possui uma matriz geradora $H$ de dimensões $(n - k) \times n$ cujas linhas são os elementos da base de $C^\perp$.

Podemos definir o código $C$ como o núcleo da transformação:

$$H : K^n \to K^{n-k}$$
$$x \to Hx^T$$

Isso é formalizado pelo seguinte teorema.

**Teorema 2.2.** *Seja $C$ um código linear de dimensão $k$ e $C^\perp$ seu dual. Seja $H$ a matriz geradora de $C^\perp$. Então, $x$ pertence a $C$ se, e somente se, $Hx^T = 0$.*

*Demonstração.*
($\to$) Seja $h_i$ a i-ésima linha de $H$ e $x \in C$. Seja $y = Hx^T$. Temos:

$$y_i = \langle h_i, x\rangle$$

Como $x \in C$ e $h_i \in C^\perp$, temos $\langle h_i, x\rangle = 0$. Logo, $y_i = 0, 1 \leq i \leq n$, o que implica $y = 0$.

($\leftarrow$) Suponha que $Hx^T = 0$.

Então $\langle h_i, x\rangle = 0$ para $1 \leq i \leq n$. Isso significa que $x$ é ortogonal a todos os elementos da base de $C^\perp$.

Agora vamos usar as propriedades do produto interno. Temos que o produto $\langle \cdot, \cdot \rangle$ é *bilinear*, ou seja:

$$\langle u, v + \lambda w\rangle = \langle u, v\rangle + \lambda\langle u, w\rangle$$

Ademais, como $\{h_i\}_{1\leq i\leq n-k}$ é uma base de $C^\perp$ podemos escrever qualquer elemento $v \in C^\perp$ como $v = \sum_{i=1}^{n-k} \lambda_i h_i$, onde $\lambda_i$ são elementos do corpo sobre o qual estamos definindo os espaços $C$ e $C^\perp$.

Isso implica que, $\forall v \in C^\perp$, temos:

$$\begin{aligned}
\langle x, v\rangle = & \\
&= \left\langle x, \sum_{i=1}^{n-k} \lambda_i h_i \right\rangle \\
&= \sum_{i=1}^{n-k} \lambda_i \langle x, h_i\rangle \quad \text{(pela bilinearidade do produto interno)} \\
&= 0 \quad \text{(umas vez que } x \text{ é ortogonal à base)}
\end{aligned}$$

Logo, $x$ é ortogonal à todos os elementos de $C^\perp$, portanto $x \in (C^\perp)^\perp = C$. $\square$

A essa matriz $H$, que gera o dual de um código $C$ e pode ser usada para testar se um vetor está neste código, dá-se o nome de ***matriz de teste de paridade***.

**Definição 2.8.** *Seja $H$ uma matriz de teste de paridade de um código $C$ e $v \in K^n$. Chamamos o vetor $Hv^T$ de **síndrome** de $v$. Pelo teorema anterior, temos que um vetor pertence a $C$ se, e somente se, sua síndrome é zero.*

Podemos relacionar a matriz de teste de paridade com o peso de um código (e portanto, como provamos anteriormente, com sua distância mínima e capacidade de correção de códigos).

**Teorema 2.3.** *Seja $H$ a matriz de teste de paridade de um código $C$. Então $\omega(C) \geq t$ se, e somente se, quaisquer $t-1$ colunas de $H$ são linearmente independentes.*

*Demonstração.* Sejam $h^1, \ldots, h^n$ as colunas de $H$ e $x \in C$.

($\rightarrow$) Suponha que todo conjunto de $t-1$ linhas de $H$ é linearmente independente. Temos:

$$\sum_{i=1}^{n} h^i \cdot x_i = 0$$

Como $\omega(x)$ representa exatamente quantas entradas não nulas $x$ possui, se $\omega(x)$ for menor que $t$, teremos uma combinação nula de menos de $t-1$ colunas de $H$, contrariando nossa hipótese. Portanto, $\omega(x) \geq t, \forall x \in C$, o que implica que $\omega(C) \geq t$.

($\leftarrow$) Suponha agora que $\omega(C) \geq t$.

Por contradição, vamos supor que existem $t-1$ colunas de $H$ que são linearmente dependentes. Sejam elas $l_1, \ldots, l_{t-1}$. Logo, existem $y_1, \ldots, y_{t-1} \in K$, nem todos nulos tais que:

$$\sum_{i=1}^{t-1} l_i \cdot y_i = 0$$

Isso implica que o vetor $y$ que tem como a $l_i$-ésima entrada o valor $y_i$ e zero nas outras posições pertence a $C$, pois teremos $Hy^T = 0$. Contudo, temos $\omega(y) < t$, o que contraria a minimalidade de $\omega(C)$. □

O teorema acima nos dá um minorante para a distância mínima de um código. O corolário abaixo nos dá condições para um resultado exato

**Corolário 2.1.** *Seja $H$ a matriz de teste de paridade de um código $C$. Então $\omega(C) = t$ se, e somente se, quaisquer $t-1$ colunas de $H$ são linearmente independentes e existem $t$ colunas de $H$ que são linearmente dependentes.*

*Demonstração.* ($\rightarrow$) Suponha que $\omega(C) = t$. Então, pelo teorema acima, temos que todo conjunto de $t-1$ colunas de H são linearmente independentes. Ao mesmo tempo, devem haver pelo menos um conjunto de $t$ colunas de $H$ que são linearmente dependentes, caso contrário, o mesmo teorema nos diria que $\omega(C) \geq s+1$, um absurdo.

($\leftarrow$) Suponha agora que quaisquer $t-1$ colunas de $H$ são linearmente independentes e existem $t$ colunas de $H$ que são linearmente dependentes. O teorema 3 nos diz que $\omega(C) \geq t$, mas $\omega(C)$ não pode ser maior que $t$, pois isso implicaria que todo conjunto de $t$ colunas de $H$ seria linearmente independente, o que contradiz a hipótese. □

# 3 Códigos de Goppa

## 3.1 Definição

**Definição 3.1.** *Seja $K$ um corpo finito e $F$ uma extensão de $K$. Tomemos $\varphi(x) \in F[x]$ e $L = \{\alpha_0, \ldots, \alpha_{n-1}\} \subset F$, com $\alpha_i \neq \alpha_j$ se $i \neq j$ e $\varphi(\alpha_k) \neq 0$. Podemos definir o seguinte espaço vetorial sobre $K$:*

$$\Gamma_K(L, \varphi) = \left\{ c \in K^n : \sum_{k=0}^{n-1} c_k (\varphi(\alpha_k))^{-1} \cdot \frac{\varphi(x) - \varphi(\alpha_k)}{x - \alpha_k} = 0 \right\}$$

*$\Gamma_K(L, \varphi)$ é chamado de o **Código de Goppa** sobre $K$ associado ao conjunto $L$ (também chamado de **suporte** de $\Gamma_K$) e polinômio $\varphi$.*

Quando $K$, $L$ e $\varphi$ forem subentendidos ou puderem ser omitidos, vamos nos referir ao código referente a eles apenas como $\Gamma$.

Enquanto a definição clássica é útil na construção da matriz de teste de paridade dos códigos de Goppa, uma definição alternativa facilita a construção da sua matriz geradora. Ela é:

**Definição 3.2.**

$$\Gamma_K(L, \varphi) = \left\{ c \in K^n : \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \bmod \varphi(x) \right\}$$

**Proposição 3.1.** *A Definição 3.2 é equivalente à Definição 3.1*

*Demonstração.* Primeiro, vamos observar que como $\forall \alpha \in L$, $\varphi(\alpha) \neq 0$, para nenhum $\alpha \in L$ teremos $(x - \alpha) | \varphi(x)$. Isso ocorre pois, dado um polinômio $p(x)$, o polinômio $x - a$ divide $p(x)$ se, e somente se, $p(a) = 0$.

Junto com o fato de que $grau(\varphi) \geq grau(x - \alpha), \forall \alpha \in L$, teremos que $\text{mdc}(\varphi, x - \alpha) = 1$. Isso significa que, para todo $\alpha$ de $L$, o polinômio $x - \alpha$ é inversível módulo $\varphi$.

Uma vez que $\varphi(x) \equiv 0 \bmod \varphi$, podemos escrever:

$$\frac{1}{x - \alpha} \equiv \frac{-1}{\varphi(\alpha)} \left( \frac{\varphi(x) - \varphi(\alpha)}{x - \alpha} \right) \bmod \varphi(x)$$

Agora, seja $A = \left\{ c \in K^n : \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \bmod \varphi(x) \right\}$. Temos:

$$
\begin{aligned}
c \in A &\leftrightarrow \sum_{i=0}^{n-1} \frac{c_i}{x-\alpha_i} \equiv 0 \bmod \varphi(x) \\
&\leftrightarrow \sum_{i=0}^{n-1} \frac{-c_i}{\varphi(\alpha_i)} \left( \frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i} \right) \bmod \varphi(x) \equiv 0 \bmod \varphi(x) \\
&\leftrightarrow \sum_{i=0}^{n-1} -c_i \varphi(\alpha_i)^{-1} \left( \frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i} \right) \equiv 0 \bmod \varphi(x)
\end{aligned}
$$

Como o polinômio $\frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i}$ tem grau menor que o grau de $\varphi(x)$, uma vez que resulta da divisão de $\varphi(x)$ por outro polinômio, temos que o grau de $f(x) = \sum_{i=0}^{n-1} -c_i \varphi(\alpha_i)^{-1} \left( \frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i} \right)$ também será menor que o grau de $\varphi(x)$, pois ao somar polinômios não é possível obter um com grau superior a todos os termos da soma.

Contudo, como ainda temos que $f(x) \equiv 0 \bmod \varphi(x)$, temos que $\varphi(x) | f(x)$, o que implica em $f(x) = 0$, já que $grau(f) < grau(\varphi)$.

Assim, podemos completar a implicação:

$$
\begin{aligned}
c \in A &\leftrightarrow \sum_{i=0}^{n-1} -c_i \varphi(\alpha_i)^{-1} \left( \frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i} \right) \equiv 0 \bmod \varphi(x) \\
&\leftrightarrow \sum_{i=0}^{n-1} -c_i \varphi(\alpha_i)^{-1} \left( \frac{\varphi(x)-\varphi(\alpha_i)}{x-\alpha_i} \right) = 0 \\
&\leftrightarrow c \in \Gamma_K(L, \varphi)
\end{aligned}
$$

O que nos leva a concluir que $A = \Gamma_K(L, \varphi)$, mostrando a equivalência das definições.

□

A Definição 3.2 nos permite definir o *polinômio Síndrome* de uma palavra de $K^n$, um análogo da síndrome definida na seção anterior e que será útil durante o processo de decodificação e correção de erros.

**Definição 3.3.** *O **polinômio síndrome** de um vetor $c \in K^n$ é:*

$$
S_c(x) = \sum_{i=0}^{n-1} \frac{c_i}{x-\alpha_i}
$$

*Temos que $c \in \Gamma_K(L, \varphi) \leftrightarrow S_c(x) \equiv 0 \bmod \varphi(x)$*

## 3.2 Matrizes Características

### 3.2.1 Matriz Teste de Paridade

É interessante que possamos descrever a matriz de teste de paridade para um Código de Goppa genérico. Para tal, precisamos achar uma matriz que, ao ser multiplicada por um vetor em $\Gamma$ produza um vetor nulo. Como a definição de $\Gamma$ envolve o polinômio $\varphi$, vamos começar analisando os coeficiente de $\varphi$. Temos:

$$\varphi(x) = \sum_{i=0}^{\delta} \varphi_i \cdot x^i \text{ , onde } \delta \text{ é o grau de } \varphi \text{ e } \varphi_\delta \neq 0$$

Temos então:

$$\begin{aligned}\frac{\varphi(x) - \varphi(\alpha)}{x - \alpha} &= \sum_{i=0}^{\delta} \varphi_i \frac{x^i - \alpha^i}{x - \alpha} \\ &= \sum_{i=0}^{\delta-1} \left( \sum_{j=i+1}^{\delta} \varphi_j \alpha^{j-1-i} \right) x^i\end{aligned}$$

Para que $c = (c_0, c_1, \ldots, c_{n-1})$ esteja em $\Gamma$ é necessário que

$$\sum_{k=0}^{n-1} c_k (\varphi(\alpha_k))^{-1} \cdot \frac{\varphi(x) - \varphi(\alpha_k)}{x - \alpha_k} = 0$$

Substituindo a relação encontrada para os coeficientes de $\varphi$ em nossa condição obtemos:

$$\begin{aligned}&\sum_{k=0}^{n-1} c_k (\varphi(\alpha_k))^{-1} \cdot \frac{\varphi(x) - \varphi(\alpha_k)}{x - \alpha_k)} \\ &= \sum_{k=0}^{n-1} c_k (\varphi(\alpha_k))^{-1} \cdot \sum_{i=0}^{\delta-1} \left( \sum_{j=i+1}^{\delta} \varphi_j \alpha^{j-1-i} \right) x^i \\ &= \sum_{i=0}^{\delta-1} \left( \sum_{k=0}^{n-1} \left( \varphi(\alpha_k)^{-1} \sum_{j=i+1}^{\delta} \varphi_j \alpha_j^{j-1-t} \right) c_k \right) x^i\end{aligned}$$

Como essa expressão precisa ser igual a 0, é necessário que os coeficientes de todos os termos do polinômio sejam zero, o que significa que o segundo somatório precisa ser nulo. Este fato se traduz na seguinte equação:

$$\sum_{k=0}^{n-1}\left(\varphi(\alpha_k)^{-1}\sum_{j=i+1}^{\delta}\varphi_j\alpha_j^{j-1-t}\right)c_k = 0, \forall i \in [0, \delta-1]$$

Podemos usar a expressão acima para construir uma matriz. Interpretando a informação do somatório com índice $k$ como informação sobre as colunas e do somatório com índice $j$ como informação sobre o conteúdo das linhas, obtemos a seguinte matriz:

$$H = \begin{pmatrix} \varphi(\alpha_0)^{-1}\varphi_\delta & \dots & \varphi(\alpha_{n-1})^{-1}\varphi_\delta \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-1}+\varphi_\delta\alpha_0) & \dots & \varphi(\alpha_{n-1})^{-1}(\varphi_{\delta-1}+\varphi_\delta\alpha_{n-1}) \\ \vdots & & \vdots \\ \varphi(\alpha_0)^{-1}\sum_{j=1}^{\delta}\varphi_j\alpha_0^{j-1} & \dots & \varphi(\alpha_{n-1})^{-1}\sum_{j=1}^{\delta}\varphi_j\alpha_{n-1}^{j-1} \end{pmatrix}$$

Operações lineares feitas nas linhas dessa matriz não modificam o espaço gerado por ela, então podemos escaloná-la, multiplicando suas linhas por constantes e somando-as com outras linhas, para simplificar a matriz.

O processo a ser realizado é o seguinte:

1. Multiplica-se a primeira linha por $(\varphi_\delta)^{-1}$
2. Da segunda linha, subtraímos a primeira linha multiplicada por $\varphi_{\delta-1}$ e a multiplicamos por $(\varphi_\delta)^{-1}$
3. Da i-ésima linha subtraímos a primeira linha multiplicada por $\varphi_{\delta-i+1}$, a segunda linha multiplicada por $\varphi_{\delta-i+2}$ e assim sucessivamente até subtrairmos a linha anterior multiplicada por $\varphi_{\delta-1}$ e por fim, dividimos a linha por $\varphi_\delta$

Para fins ilustrativos, vamos realizar esse processo na primeira coluna da matriz $H$ (o que ocorre nas outras colunas é análogo, dada a estrutura simétrica da matriz).

Começamos com a coluna:

$$h_1 = \begin{pmatrix} \varphi(\alpha_0)^{-1}\varphi_\delta \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-1}+\varphi_\delta\alpha_0) \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-2}+\varphi_{\delta-1}\alpha_0+\varphi_\delta\alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1}\sum_{j=1}^{\delta}\varphi_j\alpha_0^{j-1} \end{pmatrix}$$

Multiplicamos a primeira linha por $\varphi_\delta^{-1}$ e obtemos:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-1} + \varphi_\delta \alpha_0) \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-2} + \varphi_{\delta-1}\alpha_0 + \varphi_\delta \alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1} \sum_{j=1}^{\delta} \varphi_j \alpha_0^{j-1} \end{pmatrix}$$

Da segunda linha, subtraímos a primeira linha multiplicada por $\varphi_{\delta-1}$ e obtemos:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}(\varphi_\delta \alpha_0) \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-2} + \varphi_{\delta-1}\alpha_0 + \varphi_\delta \alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1} \sum_{j=1}^{\delta} \varphi_j \alpha_0^{j-1} \end{pmatrix}$$

Em seguida, multiplicamos essa linha por $\varphi_\delta^{-1}$ resultando em :

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-2} + \varphi_{\delta-1}\alpha_0 + \varphi_\delta \alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1} \sum_{j=1}^{\delta} \varphi_j \alpha_0^{j-1} \end{pmatrix}$$

Agora, subtraímos da terceira linha a primeira multiplicada por $\varphi_{\delta-i+1} = \varphi_{\delta-3+1} = \varphi_{\delta-2}$ e obtemos:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 \\ \varphi(\alpha_0)^{-1}(\varphi_{\delta-1}\alpha_0 + \varphi_\delta \alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1} \sum_{j=1}^{\delta} \varphi_j \alpha_0^{j-1} \end{pmatrix}$$

Então, subtraímos da terceira linha a segunda multiplicada por $\varphi_{\delta-i+2} = \varphi_{\delta-1}$ e obtemos:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 \\ \varphi(\alpha_0)^{-1}(\varphi_\delta \alpha_0^2) \\ \vdots \\ \varphi(\alpha_0)^{-1} \sum_{j=1}^{\delta} \varphi_j \alpha_0^{j-1} \end{pmatrix}$$

Por fim, dividimos a linha por $\varphi_\delta$, obtendo:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 \\ \varphi(\alpha_0)^{-1}\alpha_0^2 \\ \vdots \\ \varphi(\alpha_0)^{-1}\sum_{j=1}^{\delta}\varphi_j\alpha_0^{j-1} \end{pmatrix}$$

Agora fica fácil ver que, ao repetir esse processo $\delta$ vezes, chegaremos a uma coluna da forma:

$$\begin{pmatrix} \varphi(\alpha_0)^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 \\ \varphi(\alpha_0)^{-1}\alpha_0^2 \\ \vdots \\ \varphi(\alpha_0)^{-1}\alpha_0^{j-1} \end{pmatrix}$$

Temos então que esse algoritmo de escalonamento nos dá uma matriz bem mais simples, que terá a seguinte forma:

$$H = \begin{pmatrix} \varphi(\alpha_0)^{-1} & \dots & \varphi(\alpha_{n-1})^{-1} \\ \varphi(\alpha_0)^{-1}\alpha_0 & \dots & \varphi(\alpha_{n-1})^{-1}\alpha_{n-1} \\ \vdots & & \vdots \\ \varphi(\alpha_0)^{-1}\alpha_0^{\delta-1} & \dots & \varphi(\alpha_{n-1})^{-1}\alpha_{n-1}^{\delta-1} \end{pmatrix}$$

Para fins de representação, é interessante pensar na matriz $H$ como sendo o produto de duas matrizes: uma matriz de Vandermond $V$ do vetor $(\alpha_0, \dots, \alpha_{n-1})$ e uma matriz diagonal $D$ cuja entrada $d_{ii} = \varphi(\alpha_i)^{-1}$. Ou seja:

$$\begin{aligned} H &= VD \\ &= \begin{pmatrix} 1 & \dots & 1 \\ \alpha_0 & \dots & \alpha_{n-1} \\ \vdots & & \vdots \\ \alpha_0^{\delta-1} & \dots & \alpha_{n-1}^{\delta-1} \end{pmatrix} \begin{pmatrix} \varphi(\alpha_0)^{-1} & & & \\ & \varphi(\alpha_1)^{-1} & & \\ & & \ddots & \\ & & & \varphi(\alpha_{n-1})^{-1} \end{pmatrix} \end{aligned}$$

É importante observar que as entradas de $H$ não são necessariamente elementos do corpo $K$, uma vez que tanto os coeficientes de $\varphi$ quanto os elementos do suporte $L$ são tomados de $F$, um corpo que é uma extensão de $K$. Contudo, se interpretamos $F$ como um espaço vetorial sobre $K$, podemos obter da matriz $H$ uma matriz $H'$ cujas entradas estão todas em $K$. Para tal, basta que para cada elemento de $H$,

escrevamos sua representação como um vetor coluna de $\dim_K F$ entradas (todas em $K$).

### 3.2.2 Matriz Geradora

Tomemos o polinômio síndrome de $c \in K^n$, que é $\sum_{i=0}^{n-1} \frac{c_i}{x-\alpha_i}$. Se chamamos o produto $\prod_{i=0}^{n-1}(x-\alpha_i)$ de $h(x)$, podemos escrever a síndrome como:

$$S_c(x) = \frac{b(x)}{h(x)} \text{ , para algum polinômio } b(x)$$

Se $c \in \Gamma$, temos $S_c(x) \equiv 0 \bmod \varphi(x)$, isso significa que $S_c(x) = \frac{b(x)\varphi(x)}{h(x)}$. Logo, temos:

$$b(x)\varphi(x) = h(x)\sum_{i=0}^{n-1}\frac{c_i}{x-\alpha_i} = \sum_{i=0}^{n-1} c_i \prod_{k\neq i}(x-\alpha_k)$$

A segunda igualdade vem do fato de que, para um dado $i$ o temo $(x-\alpha_i)$ de $h(x)$ é cancelado pelo denominador do outro termo.

Se derivamos a função $h(x)$, obtemos pela regra do produto a função $h'(x) = \sum_{i=0}^{n-1}\prod_{k\neq i}(x-\alpha_k)$, que é algo bem similar ao que temos na equação acima.

Se avaliamos $h'(x)$ em $\alpha_i$, obtemos:

$$h'(\alpha_i) = \sum_{j=0}^{n-1}\prod_{k\neq j}(x-\alpha_k) = \prod_{k\neq i}(\alpha_i-\alpha_k)$$

A segunda igualdade vale pois em todos os termos do somatório em que temos $j \neq i$, teremos no produtório associado o fator $(\alpha_i - \alpha_i)$, que tornará o produto 0.

Avaliando $b(\alpha_j)\varphi(\alpha_j)$ a relação com $h'(x)$ se torna explícita. Temos:

$$\begin{aligned} b(\alpha_j)\varphi(\alpha) &= \sum_{i=0}^{n-1} c_i \prod_{k\neq i}(\alpha_j-\alpha_k) \\ &= c_j \prod_{k\neq j}(\alpha_j-\alpha_k) \\ &= c_j h'(\alpha_j) \end{aligned}$$

Então podemos escrever $c_j$ na forma

$$c_j = \frac{\varphi(\alpha_j)}{h'(\alpha_j)} \cdot b(x)$$

Notando que o grau de $b(x)$ é no máximo $n-1-\delta$, uma vez que $\varphi(x)$ tem grau

$\delta$ e $b(x)\varphi(x) = \sum_{i=0}^{n-1} c_i \prod_{k\neq i}(x-\alpha_k)$ tem grau no máximo $n-1$, podemos escrever $b(x) = \sum_{i=0}^{n-1-\delta} x^i$ e $b(\alpha_j) = \sum_{i=0}^{n-1-\delta} \alpha_j^i$.

Assim, reescrevemos $c_j$ como:

$$c_j = \sum_{i=0}^{n-1-\delta} b_i \frac{\varphi(\alpha_j)(\alpha_j)^i}{h'(\alpha_j)}$$

Portanto, concluímos que para que $c$ esteja em $\Gamma_K(L,\varphi)$, deve existir $(b_0,\ldots,b_{n-1-\delta}) \in K^{n-1-\delta}$ tal que

$$c = (b_0,\ldots,b_{n-1-\delta}) \begin{pmatrix} \frac{\varphi(\alpha_0)}{h'(\alpha_0)} & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})} \\ \frac{\varphi(\alpha_0)}{h'(\alpha_0)}\cdot\alpha_0 & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})}\cdot\alpha_{n-1} \\ \vdots & & \vdots \\ \frac{\varphi(\alpha_0)}{h'(\alpha_0)}\cdot\alpha_0^{n-1-\delta} & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})}\cdot\alpha_{n-1}^{n-1-\delta} \end{pmatrix}$$

Logo, matriz geradora de um código de Goppa $\Gamma_K(L,\varphi)$ é:

$$G = \begin{pmatrix} \frac{\varphi(\alpha_0)}{h'(\alpha_0)} & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})} \\ \frac{\varphi(\alpha_0)}{h'(\alpha_0)}\cdot\alpha_0 & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})}\cdot\alpha_{n-1} \\ \vdots & & \vdots \\ \frac{\varphi(\alpha_0)}{h'(\alpha_0)}\cdot\alpha_0^{n-1-\delta} & \cdots & \frac{\varphi(\alpha_{n-1})}{h'(\alpha_{n-1})}\cdot\alpha_{n-1}^{n-1-\delta} \end{pmatrix}$$

## 3.3 Códigos de Goppa Binários Irredutíveis

Até agora nesta seção, lidamos com códigos de Goppa definidos sobre um corpo genérico $K$ e com um polinômio característico $\varphi(x)$ sem nenhuma propriedade especial. Contudo, no sistema de McEliece, como ele foi primeiro descrito [McEliece, 1978], são utilizados uma subclasse específica de códigos de Goppa, os *códigos binários irredutíveis* (CGBI).

**Definição 3.4.** *Um código de Goppa $\Gamma_K(L,\varphi)$ é dito **binário irredutível** se $K = \mathbb{F}_2$ (o corpo de dois elementos), $L \subset \mathbb{F}_{2^m}$ e $\varphi$ é um polinômio irredutível sobre $F_{2^m}[x]$.*

Para encontrar cotas inferiores e definir o algoritmo de decodificação e correção de erros para um CGBI, vamos precisar definir o *polinômio identificador de erros* de um vetor:

**Definição 3.5.** *Seja $c \in \mathbb{F}_2^n$ e $T_c = \{i : c_i = 1\}$ o conjunto de posições em que o vetor $c$ tem entradas não nulas. Definimos o **polinômio identificador de erros** de $c$ como:*

$$\sigma_c(x) = \prod_{i\in T_c}(x-\alpha_i)$$

*Se $c$ é o vetor nulo, definimos $\sigma_0(x) = 1$*

É fácil estabelecer uma relação entre o polinômio identificador de erros de um vetor e seu peso de Hamming: $grau(\sigma_c(x)) = \omega(c)$, uma vez que para cada entrada não nula do vetor adicionamos um termo no produtório, o que aumenta em 1 o grau do polinômio.

Vamos também estabelecer uma relação entre o polinômio síndrome de um vetor e seu polinômio identificador de erros:

**Lema 3.1.** *Seja $c \in \mathbb{F}_2^n$ e seja $\sigma_c{}'(x)$ a derivada do polinômio detector de erros de $c$. Então:*

$$S_c(x) \cdot \sigma_c(x) = \sigma_c'(x)$$

*Demonstração.* Vamos começar derivando o polinômio $\sigma_c(x) = \prod_{i \in T_c}(x - \alpha_i)$. Pela regra da cadeia obtemos:

$$\sigma_c'(x) = \sum_{i \in T_c} \prod_{j \neq i \in T_c} (x - \alpha_i)$$

Por outro lado, multiplicando $S_c(x)$ e $\sigma_c(x)$, obtemos:

$$\begin{aligned} S_c(x) \cdot \sigma_c(x) &= \prod_{i \in T_c}(x - \alpha_i) \cdot \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \\ &= \prod_{i \in T_c}(x - \alpha_i) \cdot \sum_{i \in T_c} \frac{c_i}{x - \alpha_i} \quad \text{(pois se } i \notin T_c\text{, então } c_i = 0) \\ &= \sum_{i \in T_c} \prod_{j \neq i \in T_c} (x - \alpha_i) \\ &= \sigma_c'(x) \quad \text{(como verificado acima)} \end{aligned}$$

□

Códigos de Goppa binários irredutíveis tem capacidades de correção de erro superiores aos códigos de Goppa genéricos. Sobre a distância mínima de códigos genéricos pode-se provar apenas que $d_\Gamma \geq \delta + 1$. Contudo, para um CGBI, podemos provar que $d_\Gamma \geq 2\delta + 1$.

Antes de provar esta cota, precisamos de mais um lema.

**Lema 3.2.** *Seja $p(x) \in \mathbb{F}_{2^m}[x]$ tal que $p(x) = \sum_{i=0}^{n} p_i x^i$. Então $(p(x))^2 = \sum_{i=0}^{2n} (p_i)^2 x^{2i}$*

*Demonstração.* Provamos isso por indução no número de termos de $p(x)$.

- **Caso base:** se $p$ tem apenas um termo, será da forma $p(x) = x^n$ e teremos $p(x)^2 = x^{2n}$. Se $p$ tem dois termos, é da forma $p(x) = x^n + x^m$ e teremos $p(x)^2 = x^{2n} + 2x^{mn} + x^{2m} = x^{2n} + x^{2m}$. De qualquer forma, vemos que os resultados se conformam à hipótese.

- **Hipótese de Indução:** Suponha que se $p$ tem $t$ termos então $(p(x))^2 = \sum_{i=0}^{n}(p_i)^2x^{2i}$ (onde apenas $t$ dos coeficientes $p_i$ são não nulos). Vamos provar que o resultado vale quando $p$ tem $t+1$ termos.

- **Passo:** Seja $p \in \mathbb{F}_{2^m}[x]$ com $t+1$ termos. Vamos definir

$$T = \{i \in [0, n] : p_i \neq 0\}$$

Podemos então escrever $p(x) = \sum_{i \in T} p_i x^i$. Se escolhermos $j \in T$, podemos reescrever $p$ como

$$\sum_{i \in T, i \neq j} p_i x^i + p_j x^j$$

.

Tomando $p(x)^2$, temos

$$\begin{aligned}
p(x)^2 &= \left( \sum_{i \in T, i \neq j} p_i x^i + p_j x^j \right)^2 \\
&= \left( \sum_{i \in T, i \neq j} p_i x^i \right)^2 + 2 \left( \sum_{i \in T, i \neq j} p_i x^i \right) (p_j x^j) + (p_j x^j)^2 \\
&= \left( \sum_{i \in T, i \neq j} p_i x^i \right)^2 + (p_j x^j)^2 \text{ (pois o corpo tem característica 2)} \\
&= \sum_{i \in T, i \neq j} (p_i)^2 x^{2i} + p_j^2 x^{2j} \text{ (pela hipótese de indução)} \\
&= \sum_{i \in T} p_i^2 x^{2i} \\
&= \sum_{i=0}^{2n} (p_i)^2 x^{2i}
\end{aligned}$$

A última igualdade é válida pois, mesmo tendo apenas $t+1$ termos não nulos no nosso polinômio, podemos escrevê-lo como o último somatório tomando $p_i = 0$ se $i \notin T$.

Isso conclui o passo e a prova por indução do lema.

□

**Teorema 3.1.** *A distância mínima de um código de Goppa binário irredutível é* $2\delta + 1$*, onde* $\delta$ *é o grau do polinômio característico do código.*

*Demonstração.* Pelo Lema 3.1, temos que, para $c \in \mathbb{F}_2^n, S_c(x) \cdot \sigma_c(x) = {\sigma_c}'(x)$. Além disso, pela definição de síndrome, temos que $c \in \Gamma \leftrightarrow S_c(x) \equiv 0 \bmod \varphi(x)$. Então, se $c \in \Gamma$, temos:

$$\begin{aligned} {\sigma_c}'(x) &\equiv S_c(x) \cdot \sigma_c(x) \bmod \varphi(x) \\ {\sigma_c}'(x) &\equiv 0 \bmod \varphi(x) \\ \therefore \varphi(x) &| {\sigma_c}'(x) \end{aligned}$$

Note que no polinômio ${\sigma_c}'(x)$, apenas as potências pares podem ter coeficientes não nulos. Isso ocorre pois, quando decrescemos os expoentes pares de $\sigma_c(x)$, sendo o corpo de característica 2, nulificamos esses termos, deixando apenas as potências pares em ${\sigma_c}'(x)$ (cujos coeficientes são resultados do decréscimo de expoentes ímpares de $\sigma_c(x)$). Podemos então escrever:

$${\sigma_c}'(x) = \sum_{i=0}^{2s} \sigma_i x^{2i} \text{ (para algum s} < n/2)$$

Então, pelo Lema 3.2, existe um polinômio $g(x)$ tal que ${\sigma_c}'(x) = g(x)^2$. Isso significa que temos $grau(g) = 2s$.

Como $\varphi(x)$ é irredutível, se $\varphi(x)|{\sigma_c}'(x)$, então $\varphi(x)|g(x)$, o que implica em $s \geq \delta$. Temos então:

$$\omega(c) = grau(\sigma_c(x)) \geq grau({\sigma_c}'(x)) + 1 = 2s + 1 \geq 2\delta + 1$$

Como $c$ é uma palavra arbitrária de $\Gamma$, temos $d_\Gamma \geq 2\delta + 1$.

□

## 3.4 Decodificação de um CGBI

Vamos agora descrever o Algoritmo de Patterson [Patterson, 1975], que decodifica eficientemente um código de Goppa binário irredutível.

Seja $c \in \mathbb{F}_2^n$ a mensagem recebida, $m \in \Gamma$ a mensagem enviada e $e \in \mathbb{F}_2^n$ o vetor de erros que foram acrescentados a $m$, temos:

$$c = m \oplus e$$

Tomando as síndromes, temos:

$$S_c(x) = S_{m \oplus e}(x) = S_m(x) + S_e(x) = S_e(x)$$

Tomemos agora $\sigma_e(x)$. Temos que podemos escrever este polinômio como a soma dos quadrados de dois outros polinômios, $\sigma_{par}$ e $\sigma_{impar}$, onde $\sigma_{par}$ é a raiz dos termos de $\sigma_e$ com coeficientes pares e $\sigma_{impar}$ é a raiz do polinômio obtido tomando-se os termos ímpares de $\sigma_e$, subtraindo-se 1 deles e tomando a raiz quadrada.

A definição pode ser um pouco confusa, então vamos ilustrá-la com um pequeno exemplo. Se temos que $\sigma_e(x) = x^7 + x^6 + x^4 + x^3 + x^2$, podemos escrever:

$$\sigma_e(x) = x(x^6 + x^2) + (x^6 + x^4 + x)^2$$

Se tomamos as raízes dos polinômios entre parênteses (o que podemos fazer, dado o Lema 3.2), escrevemos:

$$\sigma_e(x) = x(x^3 + x)^2 + (x^3 + x^2 + x)^2$$

E isso nos dá $\sigma_{impar} = (x^3 + x)$ e $\sigma_{par} = (x^3 + x^2 + x)$.

Podemos então escrever $\sigma_e = (\sigma_{par})^2 + x(\sigma_{impar})^2$. Se derivarmos, obtemos pela regra da cadeia que

$$\sigma_e{}' = 2(\sigma_{par})(\sigma'_{\text{par}}) + (\sigma_{impar})^2 + 2x(\sigma_{impar})(\sigma'_{\text{impar}}) = (\sigma_{impar})^2$$

Pelo Lema 3.1 temos:

$$\begin{aligned}
&\sigma_e \cdot S_e \equiv \sigma_e{}' \bmod \varphi(x) \\
&((\sigma_{par})^2 + x(\sigma_{impar})^2) S_e \equiv (\sigma_{impar})^2 \bmod \varphi(x) \\
&S_e \sigma_{par}^2 \equiv (1 + xS_e)\sigma_{impar}^2 \bmod \varphi(x)
\end{aligned}$$

Como $\varphi(x)$ é irredutível, podemos achar $T(x)$ tal que $T(x)S_e(x) \equiv 1 \bmod \varphi(x)$. Multiplicando a última equação por esse $T(x)$, obtemos:

$$\sigma_{par}^2 \equiv (T + x)\sigma_{impar}^2 \bmod \varphi(x)$$

Como $\mathbb{F}_2^n$ tem característica 2, existe $\tau(x)$ tal que $\tau(x)^2 = (T + x)$, o que nos permite escrever:

$$\sigma_{par}^2 \equiv \tau^2 \sigma_{impar}^2 \bmod \varphi(x)$$

Isto por sua vez implica que $\sigma_{par} \equiv \tau\sigma_{impar} \bmod \varphi(x)$.

Podemos usar o Algoritmo de Euclides estendido sobre corpos polinomiais para, tendo $\tau$ e $\varphi$, descobrir $\sigma_{impar}$ e $\sigma_{par}$, o que nos permite descobrir $\sigma_e$.

Pela definição de $\sigma_e(x)$, para um $\alpha_i \in L$, temos $\sigma_e(\alpha_i) = 0 \leftrightarrow (\sigma_e)_i = 0$, o que, pela definição de $e$, significa que a mensagem possui um erro na posição $i$ (é dai que o polinômio $\sigma$ recebe o nome de *identificador de erros*).

Isso nos permite derivar o seguinte algoritmo para decodificação de uma mensagem $c$ para uma palavra $m$ do nosso código:

1. Obter o polinômio $\sigma_e(x)$ da forma descrita acima
2. Para cada $\alpha_i \in L$:
   (a) se $\sigma_e(\alpha_i) = 0$:
      - $m_i = c_i + 1$

   (b) senão
      - $m_i = c_i$

# 4 Criptossistema de McEliece

Nesta seção, iremos descrever exatamente como códigos corretores de erros podem ser utilizados para criptografia apresentendo o esquema originalmente proposto por McEliece em [McEliece, 1978].

Nele, a criptografia, a ser discutida em mais detalhes adiante, consiste em transformar uma mensagem original em componente de um código e inserir erros aleatórios nela, de forma que alguém que venha a interceptá-la não seja capaz de entender seu conteúdo. A decriptografia consiste simplesmente de corrigir os erros presentes na mensagem e reverter a palavra do código de volta para a mensagem original.

Uma analogia que pode ser feita é pensar na mensagem com uma música contida em CD. A criptografia seria arranhar o CD de forma que ninguém fosse capaz de ouvir sua música. A decriptografia seria um processo pelo qual tiramos os arranhados do CD.

A ideia central deste esquema criptográfico, como definido em [Misoczki, 2013], é que escolhemos um código linear que permita, por meio da chave privada, um algoritmo eficiente de decodificação. A chave pública é uma versão disfarçada da chave privada e permite que apenas a codificação seja feita eficientemente.

Para fins de segurança, é interessante que a chave pública seja indistinguível de algo totalmente aleatório, o que torna difícil usá-la para obter-se a chave privada.

## 4.1 Componentes do Criptossistema

Seja $K$ um corpo finito que servirá de alfabeto para nosso código, $n$ o tamanho desejado das nossas mensagens criptografadas e $\delta$ o número de erros que queremos ser capazes de corrigir. Os seguintes elementos são necessários para que o esquema criptográfico em questão possa ser eficientemente implementado:

- **G**: uma matriz $k$ x $n$, geradora de um código linear de dimensão $k$
- $\boldsymbol{\psi}$: Uma estrutura capaz de corrigir $\delta$ erros do código gerado por G.
- **S**: Uma matriz $k$ x $k$, inversível
- **P**: Uma matriz de permutação $n$ x $n$

As matrizes $S$ e $P$ são utilizadas para "embaralhar"a matriz $G$, de forma a dificultar vazamento de informação decorrente de uma possível estrutura algébrica inerente a $G$. Por exemplo, como vimo com códigos de Goppa, a matriz $G$ tem uma estrutura intimamente ligada às informações que são necessária para a decodificação e precisamos mascarar a estrutura algébrica desta matriz se vamos utilizá-la de forma segura.

## 4.2 Chave Pública e Chave Privada

A partir de $G$, $S$ e $P$, geramos a matriz pública de codificação $\overline{G}$ da seguinte forma:

$$\overline{G} = SGP$$

A matriz $SG$ é uma matriz $k \times n$ em que cada linha é uma combinação linear das linhas da matriz $G$, então elas geram o mesmo subespaço, portanto uma palavra no código gerado por $SG$ também está no código gerado por $G$. Assim, um algoritmo que corrija erros de vetores de G, corrigirá erros em vetores de $SG$.

Contudo, como a matriz $P$ pode implicar em permutações de colunas, o código gerado por $\overline{G}$ não é necessariamente o mesmo de $G$. Algoritmos corretores de erros para $G$ podem falhar com vetores codificados em $\overline{G}$. Assim, quando vamos corrigir o erro de uma mensagem $m' = m\overline{G}$, precisamos primeiro multiplicá-la pela direita por $P^{-1}$ para garantir que nossa estrutura corretora de erros funcione corretamente.

O fato de a mensagem criptografada estar num subespaço diferente daquele no qual $\psi$ funciona adiciona mais uma camada de proteção ao sistema, pois significa que mesmo que um atacante descubra $\psi$, ainda não conseguirá decodificar a mensagem se não conhecer $P$.

A chave pública então consiste da matriz $\overline{G}$ e de $\delta$, uma vez que alguém que deseje encriptar seus dado usando McEliece deve saber quantos erros introduzir na informação codificada. Denotamos:

$$K_{\text{pub}} = (\overline{G}, \delta)$$

Já a chave privada consiste dos componentes do sistema, as informações com as quais um atacante malicioso seria capaz de descriptografar mensagens não destinadas a ele. Conhecimento de $S$, $P$, $G$ e $\psi$ levariam a isto, então eles são a chave privada. Denotamos:

$$K_{\text{priv}} = (G, \psi, S, P)$$

Em sua forma original o criptossistema usava como chave privada:

- $G$: uma matriz geradora de um código de Goppa binário irredutível
- $\psi$: O algoritmo de Patterson para decodificar tais códigos, juntamente com as informações necessárias para seu funcionamento: o polinômio característico $\varphi$ e o suporte $L$ usados para produzir a matriz $G$
- $S$: uma matriz não-singular aleatória

- $P$: uma matriz de permutação aleatória

## 4.3 Criptografia e Decriptografia

O algoritmo de criptografia consiste apenas em codificar a mensagem e introduzir um vetor de erros com peso adequado. Em pseudo-código:

**Algoritmo 1:** Algoritmo de Criptografia do McElice

**Data:** $m \in K^k$, a mensagem a ser criptografada
**Result:** $c \in K^n$, a mensagem criptografada
1 Calcule $m' = m\overline{G}$, a palavra do código correspondente a $m$ ;
2 Selecione um vetor aleatório $e \in K^n$ de peso $\delta$ ;
3 Calcule $c = m' \oplus e$
4 **return** $c$

A decriptografia, em pseudo-código, é:

**Algoritmo 2:** Algoritmo de Decriptogradia do McEliece

**Data:** $c = m\overline{G} \oplus e$, a mensagem criptografada
**Result:** $m$, a mensagem original
1 Calcule $\overline{c} = cP^{-1} = mSG + eP^{-1}$
2 Use o corretor de erros $\psi$ para remover os erros e obter $mSG$
3 Resolva o sistema linear sobredeterminado dado por $mSG = \overline{m}$, obtendo $m$
4 **return** $m$

É importante notar que como $P^{-1}$ também é uma matriz de permutação, o vetor $eP^{-1}$ também possui $\delta$ erros. Além disso, como observado na seção anterior, os códigos gerados por $SG$ e $G$ são iguais, então o decodificador $\psi$ deve funcionar corretamente.

## 4.4 Reduções de Segurança

Vamos agora mostrar que podemos reduzir a quebra do criptossistema de McEliece a problemas provadamente difíceis ou que temos evidências de serem difíceis. Adaptamos as definições e provas de [Misoczki et al., 2013] e [Faugere et al., 2013].

Vamos denotar por $\mathcal{F}_{n,k}$ uma família de códigos corretores de erro de tamanho $n$, dimensão $k$, capaz de corrigir $t$ erros sobre algum corpo $K$. Usamos a notação livremente nas definições e provas que seguem.

**Definição 4.1.** *Denotamos por $\mathcal{M}_{n,k}$ o conjunto de matrizes $k \times n$ sobre $K$. $\mathcal{M}_{n,k}$ é chamado de* ***espaço de chaves aparente*** *de $\mathcal{F}_{n,k}$, umas vez que qualquer matriz*

*que possa gerar um código da família $\mathcal{F}_{n,k}$ deve ser um elemento de $\mathcal{M}_{n,k}$, mas nem todo elemento de $\mathcal{M}_{n,k}$ é uma matriz geradora de $\mathcal{F}_{n,k}$.*

**Definição 4.2.** *Seja $\mathcal{K}_{n,k} \subset \mathcal{M}_{n,k}$ o conjunto de matrizes geradoras de $\mathcal{F}_{n,k}$. Chamamos $\mathcal{K}_{n,k}$ de **espaço de de chaves real** de $\mathcal{F}_{n,k}$, pois uma matriz $G$ é geradora de um código da família $\mathcal{F}_{n,k}$ se, e somente se, $G \in \mathcal{K}_{n,k}$*

Estas definições estão relacionadas ao problema de decisão de dada uma matriz $G \in \mathcal{M}_{n,k}$ decidir se ela é capaz de gerar um código em $\mathcal{F}_{n,k}$. Formalmente temos:

> Problema da Distinguibilidade de Códigos
> INSTÂNCIA: $G \in \mathcal{M}_{n,k}$
> PERGUNTA: $G$ pertence a $\mathcal{K}_{n,k}$?

O problema da distinguibilidade de códigos está em $\mathcal{NP}$ uma vez que dada uma matriz, podemos verificar eficientemente se ela de fato gera um código da família $\mathcal{F}_{n,k}$. Apesar dele não ter sido provado ser $\mathcal{NP}$-completo, acredita-se que não pode ser resolvido de forma eficiente para uma instância qualquer[Faugere et al., 2010, Misoczki, 2013].

Um programa capaz de resolver o problema da distinguibilidade de códigos para $\mathcal{F}_{n,k}$ é chamado de um *distinguidor*. Quando temos uma instância positiva do problema, o distinguidor dever ser capaz de "adivinhar" que a resposta é sim com probabilidade bem maior do que o faria para uma instância qualquer. Formalmente temos:

**Definição 4.3.** *Um programa $\mathcal{D} : \mathcal{M}_{n,k} \to 0, 1$ é um $(T, \epsilon)$-**distinguidor** para $\mathcal{K}_{n,k}$ contra $\mathcal{M}_{n,k}$ se possui tempo de execução $T$ e sua vantagem é tal que*

$$Vantagem(\mathcal{D}, \mathcal{K}_{n,k}) = |\Pr[\mathcal{D}(G) = 1 | G \in \mathcal{K}_{n,k}] - \Pr[\mathcal{D}(G) = 1]| > \epsilon$$

Vamos definir $\mathcal{S}(0, t)$ como a esfera de centro 0 e raio $t$ no espaço métrico induzido pela métrica de Hamming em $K^n$. Ou seja, $\mathcal{S}(0, t)$ são os vetores de $K^n$ de peso exatamente $t$.

Um *decodificador* é um programa capaz, de dado uma mensagem com erros codificada por um código linear, recuperar a mensagem original e o vetor de erros com alta probabilidade. Formalmente:

**Definição 4.4.** *Um programa $\mathcal{A} : \mathcal{M}_{n,k} \times K^n \to K^k \times \mathcal{S}(0, t)$ é um $(T, e)$-**decodificador** para $(\mathcal{M}_{n,k}, t)$ se possui tempo de execução $T$ e sua probabilidade de sucesso é*

$$Sucesso(\mathcal{A}) = \Pr[\mathcal{A}(G, mG + e) = (m, e)] > \epsilon$$

Um adversário do McEliece cujo espaço de chaves seja $\mathcal{K}_{n,k}$ é um programa que, dada uma mensagem criptografada por um código linear usado no McEliece, consegue recuperar a mensagem original com alta probabilidade. Formalmente:

**Definição 4.5.** *Um programa* $\mathcal{A} : \mathcal{M}_{n,k} \times K^n \to K^k \times \mathcal{S}(0,t)$ *é um* $(T,e)$-***adversário*** *para o McElice com* $\mathcal{K}_{n,k}$ *se possui tempo de execução* $T$ *e sua probabilidade de sucesso é*

$$Sucesso(\mathcal{A}, \mathcal{K}_{n,k}) = \Pr[\mathcal{A}(G, mG + e) = (m, e) | G \in \mathcal{K}_{n,k}] > \epsilon$$

Vale ressaltar que as probabilidades presentes nas definições acima dizem respeito à eficiência dos programas; é chance de eles realizarem corretamente a ação para a qual foram feitos.

Agora vamos provar um teorema relacionando nossas definições anteriores:

**Teorema 4.1.** *Se existe um* $(T,e)$*-adversário para o McElice com* $\mathcal{K}_{n,k}$ *então conseguimos construir ou um* $(T, e/2)$*-decodificador para* $(\mathcal{M}_{n,k}, t)$ *ou um* $(T+O(n^2), \epsilon/2)$*-distinguidor para* $\mathcal{K}_{n,k}$ *contra* $\mathcal{M}_{n,k}$*.*

*Demonstração.* Seja $A$ um $(T,e)$-**adversário** para o McElice com $\mathcal{K}_{n,k}$. Vamos definir o seguinte programa $\mathcal{D}$, que, como provaremos, será ou um decodificador ou um distinguidor conforme especificado.

**Algoritmo 3:** Programa $\mathcal{D}$

**Data:** $G \in \mathcal{M}_{n,k}$, $m \in K^k$

Selecione $e \in \mathcal{S}(0,t)$ aleatoriamente ;
**if** $\mathcal{A}(G, mG + e) = (m, e)$ **then**
    **return** *1*
**else**
    **return** *0*
**end**

Temos então:

$$\begin{aligned} \Pr[\mathcal{D}(G) = 1] &= \Pr[\mathcal{A}(G, mG + e) = (m, e)] \\ &= Sucesso(\mathcal{A}) \end{aligned}$$

Analogamente, temos:

$$\begin{aligned}\Pr[\mathcal{D}(G) = 1 | G \in \mathcal{K}_{n,k}] &= \Pr[\mathcal{A}(G, mG + e) = (m, e) | G \in \mathcal{K}_{n,k}] \\ &= \mathit{Sucesso}(\mathcal{A}, \mathcal{K}_{n,k})\end{aligned}$$

Então

$$\mathit{Vantagem}(\mathcal{D}, \mathcal{K}_{n,k}) = |\mathit{Sucesso}(\mathcal{A}, \mathcal{K}_{n,k}) - \mathit{Sucesso}(\mathcal{A})|$$

Particularmente temos que se $\mathit{Sucesso}(\mathcal{A}) > \epsilon/2$, encontramos um $(T, e/2)$-decodificador para $(\mathcal{M}_{n,k}, t)$. Se $\mathit{Sucesso}(\mathcal{A}) < \epsilon/2$, então $\mathit{Vantagem}(\mathcal{D}, \mathcal{K}_{n,k}) > \epsilon/2$ e temos um $(T + O(n^2), \epsilon/2)$-distinguidor para $\mathcal{K}_{n,k}$ contra $\mathcal{M}_{n,k}$, uma vez que o tempo de $\mathcal{D}$ leva é o mesmo de $\mathcal{A}$, acrescido do tempo necessário para calcular $mG + e$, que não excede $O(n^2)$.

□

Já vimos que um distinguidor eficiente provavelmente não existe, devido às hipóteses de dificuldade que existe sobre o problema da distinguibilidade de códigos. Também é improvável que um decodificador eficiente exista. Isso se deve ao seguinte problema de decisão:

> Problema da Decodificação por Síndrome
> INSTÂNCIA: $H \in \mathcal{M}_{n,k}$, um vetor $s \in K^n$
> PERGUNTA: Encontrar um vetor $e \in \mathcal{S}(0, t)$ tal que $eH^T = s$

Esse problema é $\mathcal{NP}$-completo [Berlekamp et al., 1978] e um decodificador eficiente pode ser adaptado para resolvê-lo eficientemente [Li et al., 1994].

Portanto, é improvável que exista um atacante eficiente contra o McEliece genérico, uma vez que isso implicaria em programas que decidem eficientemente problemas que conjectura-se serem difíceis.

Deve-se notar que essa redução de segurança é válida apenas sob a hipótese de que é difícil diferenciar um código em $\mathcal{F}_{n,k}$ de um código aleatório. Acredita-se que esse é o caso para a família de Códigos de Goppa com valores de $n$ não muito grandes [Faugere et al., 2013].

# 5 Códigos de Goppa Diádicos e Quase-Diádicos

## 5.1 Introdução

Códigos de Goppa quase-diádicos foram sugeridos em [Misoczki and Barreto, 2009] como uma família de códigos alternativa a códigos de Goppa binários irredutíveis no criptossistema de McEliece com uma forma de se obter chaves mais compactas. Estes códigos possuem estruturas simétricas que podem ser exploradas de forma a reduzir a complexidade de espaço necessária para armazenar as componentes do criptossistema.

Antes de discutir mais a fundo os impactos do uso dessa classe de códigos, precisamos de algumas definições.

## 5.2 Definições e Teoremas Básicos

**Definição 5.1.** *Dado um corpo $K$ e um vetor $h = (h_0, \ldots, h_{n-1}) \in K^n$, a* ***matriz diádica*** *$\Delta(h)$ é a matriz simétrica com componentes $\Delta_{ij} = h_{i \oplus j}$ ($i \oplus j$ denota o OU-exclusivo bit a bit ente $i$ e $j$). O vetor $h$ é chamado de a* ***assinatura*** *da matriz.*

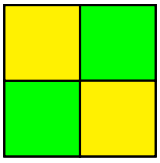

Figura 1: Matriz diádica

**Definição 5.2.** *Uma matriz* ***quase-diádica*** *é uma matriz (potencialmente não diádica) de blocos, cujos blocos componentes são matrizes diádicas.*

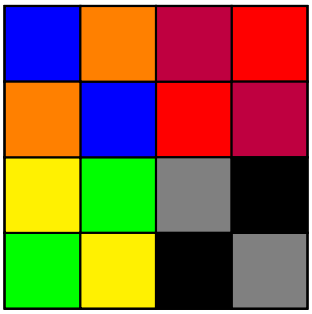

Figura 2: Matriz quase diádica. Cada bloco 4x4 é uma matriz diádica, bem como cada quadrado colorido.

**Definição 5.3.** *Seja $K$ um corpo finito e sejam $z = (z_0, \ldots, z_{k-1}) \in K^k$ e $L = (L_0, \ldots, L_{n-1}) \in K^n$ sequências disjuntas de elementos distintos. A* ***matriz de Cauchy*** *$\mathcal{C}(z, L)$ é a matriz $k \times n$ com entradas $\mathcal{C}_{ij} = 1/(z_i - L_j)$:*

$$\mathcal{C}_{ij} = \begin{pmatrix} \frac{1}{z_0 - L_0} & \cdots & \frac{1}{z_1 - L_{n-1}} \\ \vdots & \ddots & \vdots \\ \frac{1}{z_{k-1} - L_0} & \cdots & \frac{1}{z_{k-1} - L_{n-1}} \end{pmatrix}$$

Pode-se provar que é possível construir códigos de Goppa cujas matrizes de teste de paridade são matrizes de Cauchy:

**Teorema 5.1.** *O código de Goppa $\Gamma_K(L, \varphi)$ no qual $\varphi$ é mônico e da forma $\varphi(x) = (x - z_0) \ldots (x - z_{k-1})$, com todos os $z_i$ são distintos, tem como matriz de paridade a matriz de Cauchy $\mathcal{C}(z, L)$.*

A prova deste teorema foge ao escopo deste trabalho, mas encontra-se em [Tzeng and Zimmermann, 1975].

O teorema a seguir mostra que é possível construir matrizes de Cauchy diádicas sobre corpos de característica 2:

**Teorema 5.2.** *Seja $H$ uma matriz $n \times n$ sobre um corpo $K$ que é simultaneamente diádica com alguma assinatura $h \in K^n$ e de Cauchy com sequências $z, L \in K^n$, ou seja, temos $H = \Delta(h)$ e $H = \mathcal{C}(z, L)$. Então $K$ tem característica 2, $h$ satisfaz:*

$$\frac{1}{h_{i \oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$$

*e $z_i = 1/h_i + w$, $L_j = 1/h_j + 1/h_0 + w$, para algum $w \in K$.*

*Demonstração.* Como $H$ é diádica, ela também é simétrica, logo devemos ter:

$$\begin{aligned} \frac{1}{z_i - L_j} &= \frac{1}{z_j - L_i} \\ z_i - L_j &= z_j - L_i \\ L_j &= L_i + z_i - z_j \end{aligned}$$

Como isto vale para todo $i, j$, devemos ter que $L_i + z_i$ é constante, pois, por exemplo, temos $L_j = L_1 + z_1 - z_j = L_2 + z_2 - z_j$, o que implica $L_1 + z_1 = L_2 + z_2$. Vamos chamar de $\beta$ a constante $L_i + z_i$. Isso nos permite escrever: $L_j = \beta - z_j$.

Substituindo isso na definição de que $H_{ij} = 1/(z_i - L_j)$ obtemos:

$$H_{ij} = \frac{1}{z_i + z_j - \beta} \quad \text{e} \quad H_{ii} = \frac{1}{2z_i - \beta}$$

Se tomamos $i = j$, temos, pela definição de matriz diádica $H_{ii} = h_{i\oplus i} = h_0$ (pois ou OU-exclusivo de um número com ele mesmo é 0). Isso significa que a diagonal da matriz diádica é constante e devemos ter:

$$\frac{1}{2z_i - \beta} = h_0, \forall i$$

Isso significa que ou temos que todos os $z_i$ são iguais, ou que, para todo $i$, $2z_i = 0$, o que implica que o corpo tem característica 2. Como ter todos os $z_i$ iguais contraria a definição de uma matriz de Cauchy, devemos ter que $K$ tem característica 2.

Como $K$ tem característica 2, se $x \in K$, temos $2x = x + x = 0$, o que implica em $x = -x$. Então, podemos escrever $-\beta = \beta$. Isso nos permite concluir que $\beta = 1/h_0$. Logo,

$$H_{ij} = \frac{1}{z_i + z_j + 1/h_0}$$

Usando a definição de matriz diádica, temos:

$$\begin{aligned} H_{ij} &= h_{i\oplus j} \\ \frac{1}{H_{ij}} &= \frac{1}{h_{i\oplus j}} \\ \frac{1}{h_{i\oplus j}} &= z_i + z_j + \frac{1}{h_0} \end{aligned}$$

Tomando $j = 0$ e usando que o OU-exclusivo entre um número e 0 é o próprio número, podemos concluir que:

$$\frac{1}{h_i} = z_i + z_0 + \frac{1}{h_0}$$

Ou, isolando $z_i$:

$$z_i = \frac{1}{h_i} + z_0 + \frac{1}{h_0}$$

(Vale lembrar, novamente, que $z_i = -z_i$ e $1/h_i = -1/h_i$, pois $K$ tem característica 2.)

Substituindo esta definição de $z_i$ em $\frac{1}{h_{i\oplus j}} = z_i + z_j + \frac{1}{h_0}$ nos permite escrever:

$$\frac{1}{h_{i\oplus j}} = \frac{1}{h_i} + z_0 + \frac{1}{h_0} + \frac{1}{h_j} + z_0 + \frac{1}{h_0} + \frac{1}{h_0} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$$

Isto prova a propriedade desejada de $h$.

Por fim, se tomamos $w = z_0 + 1/h_0$, podemos escrever $z_i = 1/h_i + w$ e:

$$\begin{aligned} L_j &= \beta - z_j \\ &= \beta + z_j \\ &= \frac{1}{h_0} + \frac{1}{h_j} + w \end{aligned}$$

Isto conclui a prova. □

O que este teorema nos dá é uma caracterização de matrizes de Cauchy diádicas. Para construir uma, basta escolher um $h$ que respeite a equação $\frac{1}{h_{i\oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$.

Em [Misoczki and Barreto, 2009], os autores apresentam um algoritmo para essa construção. Enquanto o algoritmo completo foge do escopo deste trabalho, apresentamos a intuição por trás dele:

Começamos escolhendo um $h_0$ aleatório, e para cada $i$ que é uma potência de 2 menor do que $n$ (a dimensão da matriz), escolhe-se um valor aleatório para $h_i$.

Para os outros valores, usa-se a fórmula:

$$h_{i+j} = \frac{1}{\frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}}$$

Isto funciona pois temos que, se $i$ é uma potência de 2 e $0 < j < i$, então $i \oplus j = i + j$.

Então, resumidamente, o algoritmo gera aleatoriamente as entradas correspondentes a 0 e à potências de 2 e usa esse valores, juntamente com a fórmula dada pelo **Teorema 5.2**, para determinar os demais.

A matriz resultante será a matriz de teste de paridade de um código de Goppa sobre $K$ que terá suporte $L$ e polinômio característico $\varphi(x) = (x - z_0) \dots (x - z_{n-1})$, com $L$ e $z$ respeitando as equações descritas no **Teorema 5.2**. Isso nos leva intuitivamente à seguinte definição:

**Definição 5.4.** *Um código de Goppa* ***diádico*** *é um código de Goppa que admite uma matriz de teste de paridade diádica.*

## 5.3 Códigos de Goppa Quase-Diádicos

Os autores de [Misoczki and Barreto, 2009] explicam que não se pode usar uma matriz de teste de paridade na forma de uma matriz de Cauchy para especificar o código a ser usado no criptossistema de McEliece pois isso tornaria muito fácil a recuperação do polinômio característico $\varphi(x)$.

Então, eles modificam o algoritmo usado para gerar matrizes de Cauchy diádicas para que obtenha-se, na verdade, matrizes *quase-diádicas*.

Isso nos leva a definição:

**Definição 5.5.** *Um código de Goppa é* ***quase-diádico*** *(denotado QD-Goppa) se admite uma matriz de teste de paridade quase-diádica.*

Como cada bloco de uma matriz quase-diádica é diádico, esses blocos podem ser representados de forma compacta pelas suas assinaturas.

Então, num código de Goppa quase-diádico, ao invés de ser necessário armazenar a matriz como um todo, pode-se guardar apenas as assinaturas correspondentes aos blocos diádicos.

Os autores provam ainda que o fator de redução obtido pelo seu método é igual ao número de erros que se deseja que o código gerado seja capaz de corrigir.

Em [Misoczki and Barreto, 2009], os autores mostram ainda que a substituição de códigos de Goppa binários irredutíveis por códigos de Goppa quase-diádicos leva também a melhoras expressivas nos tempos de geração de códigos, de codificação e decodificação. Isso, consequentemente, também implica tempos de criptografia e decriptografia melhores para o criptossistema de McEliece.

# 6 Códigos LDPC e MDPC

## 6.1 Códigos LDPC

### 6.1.1 Introdução

Códigos LDPC (sigla em inglês para *low density parity check*, ou, teste de paridade de baixa densidade) foram inicialmente propostos por Gallager em tese de doutorado [Gallager, 1962]. São, em contraste com códigos algébricos, como códigos de Goppa, desprovidos de uma estrutura algébrica subjacente. Um código LDPC é caracterizado principalmente pelo fato de que sua matriz de teste de paridade é esparsa.

Códigos LDPC também admitem um representação na forma de um grafo bipartido que recebe o nome de **grafo de Tanner**. Neste grafo, as partições correspondem às linhas e colunas da matriz de teste de paridade. Os vértices correspondentes às linhas são chamados de *vértices de checagem*, enquanto os correspondentes às colunas são chamados de *vértices de variáveis*. Há uma aresta entre um vértice de checagem $c_i$ e um de vértice de variável $j$ se a entrada $ij$ da matriz teste de paridade for 1.

Por exemplo, se um código LDPC é representado pela a seguinte matriz de teste de paridade $H$:

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

Então o grafo de Tanner correspondente será:

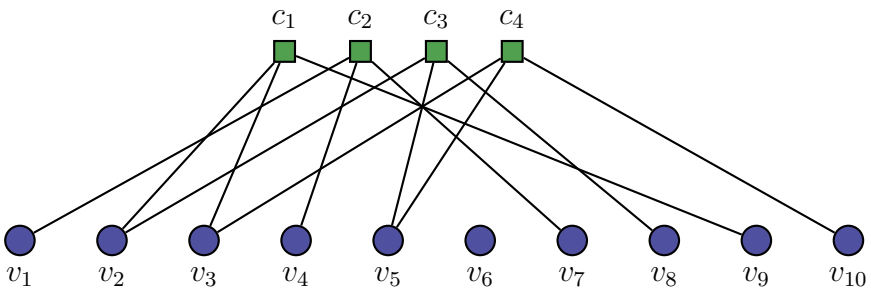


Figura 3: Grafo de Tanner para matriz $H$: vértices de checagem em verde e de variáveis em azul.

### 6.1.2 Decodificação

Códigos LDPC admitem um algoritmo de decodificação simples, conhecido como *Bit-Flipping Algorithm*.

Suponha que temos uma matriz de teste de paridade $H$, com linhas $h_i$, e recebemos uma mensagem $m$. Se $m$ pertencer ao código, a síndrome de $m$ é 0. Isso implica que $\langle m, h_i \rangle = 0$, para toda linha de $H$. Dizemos que $m$ *satisfaz* as checagens de $H$.

Caso $m$ contenha algum erro, para alguma linha $h_i$, teremos $\langle m, h_i \rangle \neq 0$. Nesse caso, dizemos que $m$ *viola* a $i$-ésima checagem de $H$.

Analisando o código sob a perspectiva do grafo de Tanner, temos que o grafo tem um vértice de variável para cada bit de $m$ e um vértice de checagem para cada checagem de $H$. Neste contexto, $m$ satisfaz uma checagem se, sendo $c_i$ o vértice de checagem correspondente, os valores dos vértices de variável que são seus vizinhos somam 0 (módulo 2).

Por exemplo, se temos a matriz de teste de paridade e o grafo de Tanner associado do exemplo anterior e recebemos a mensagem $m = (1, 0, 0, 1, 0, 0, 0, 1, 1, 0)$ temos: $v_1 = 1$, $v_2 = 0$, $v_3 = 0$, $v_4 = 1$, $v_5 = 0$, $v_6 = 0$, $v_7 = 0$, $v_8 = 1$, $v_9 = 1$, $v_{10} = 0$. As checagens são:

$$
\begin{aligned}
c_1 &= v_2 \oplus v_3 \oplus v_9 = 0 \oplus 0 \oplus 1 = 1 \\
c_2 &= v_1 \oplus v_4 \oplus v_7 = 1 \oplus 1 \oplus 0 = 0 \\
c_3 &= v_2 \oplus v_5 \oplus v_8 = 0 \oplus 0 \oplus 1 = 1 \\
c_4 &= v_3 \oplus v_5 \oplus v_{10} = 0 \oplus 0 \oplus 0 = 0
\end{aligned}
$$

Então podemos dizer que, neste exemplo, a mensagem $m$ satisfaz as checagens $c_2$ e $c_4$ e viola as checagens $c_1$ e $c_3$. Isto é equivalente a dizer que $\langle m, h_2 \rangle = \langle m, h_4 \rangle = 0$ e $\langle m, h_3 \rangle, \langle m, h_1 \rangle \neq 0$.

Dizemos que uma variável está envolvida em uma violação se uma das equações de checagem da qual ela faz parte não é igual a zero. No exemplo anterior, temos que $v_2$ faz parte de duas violações pois $c_1 \neq 0$ e $c_3 \neq 0$ e $v_2$ é necessária para o cálculo de ambas.

O algoritmo *Bit-Flipping* realiza todas as checagens e mantém quantas vezes cada variável está presente em violações. Toda variável que estiver envolvida em mais do que $t$ violações (onde $t$ é um limite pré-definido) tem seu valor invertido (de 0 pra 1, ou vice-versa). O processo é então repetido até que todas as checagens sejam satisfeitas ou um número máximo de iterações se passe.

Algoritmicamente temos:

**Algoritmo 4:** *Bit-Flipping*

**Data:** mensagem $m$, limite $t$ e número máximo de iterações $max$
**Result:** mensagem corrigida

```
1  for it = 0; it < max; it = it + 1 do
2      Inicializar n (uma para cada variável) contadores count_i como 0   // cout_j
         conta em quantas violações uma variável está envolvida
3      Inicializar r (uma para cada checagem) variáveis c_i como 0
4      for i = 1; i ≤ r; i = i + 1                    // para cada vértice de chegagem
5       do
6          for j ∈ Ajd(c_i)                  // para cada variável envolvida na checagem
7           do
8              c_i = c_i ⊕ m_j    // adicione o valor de do bit correspondente à soma
9          end
10         if c_i ≠ 0                                  // se a checagem for violada
11          then
12             for j ∈ Ajd(c_i)                       // para cada variável envolvida
13              do
14                 count_j = count_j + 1        // incremente o contador de checagens
                     violadas por essa variável
15             end
16         end
17     end
18     if todos os c_i são 0                    // Todas as checagens são satisfeitas
19      then
20         return m                                // a mensagem faz parte do código
21     else
22         for j = 1; j ≤ r; j = j + 1                    // para todas as variáveis
23          do
24             if count_j > t   // se ela se envolve em mais violações que o limite
25              then
26                 m_j = m_j ⊕ 1                      // inverte o valor da variável
27             end
28         end
29     end
30     return Erro          // falha se algoritmo demora mais que max iterações
31 end
```

No algoritmo acima, assumimos que a matriz de teste de paridade do código tem $r$ linhas e $n$ colunas. Assumimos também que se $c_i$ é uma checagem, então $\text{Adj}(c_i)$ são os índices das variáveis envolvidas nesta checagem. No exemplo anterior, teríamos $\text{Adj}(c_1) = 2, 3, 9$, por exemplo.

Para exemplificar o algoritmo, vamos usar os valores de $m$ e $H$ usados em exemplos anteriores e adotaremos o limite $t = 1$. Como já vimos anteriormente, temos as seguintes checagens:

$$
\begin{aligned}
c_1 &= v_2 \oplus v_3 \oplus v_9 = 0 \oplus 0 \oplus 1 = 1 \\
c_2 &= v_1 \oplus v_4 \oplus v_7 = 1 \oplus 1 \oplus 0 = 0 \\
c_3 &= v_2 \oplus v_5 \oplus v_8 = 0 \oplus 0 \oplus 1 = 1 \\
c_4 &= v_3 \oplus v_5 \oplus v_{10} = 0 \oplus 0 \oplus 0 = 0
\end{aligned}
$$

A única variável que está envolvida em mais que $t$ violações é $v_2$, então invertemos o segundo bit de $m$, obtendo (1,1,0,1,0,0,0,1,1,0).

As checagens agora são:

$$
\begin{aligned}
c_1 &= v_2 \oplus v_3 \oplus v_9 = 1 \oplus 0 \oplus 1 = 0 \\
c_2 &= v_1 \oplus v_4 \oplus v_7 = 1 \oplus 1 \oplus 0 = 0 \\
c_3 &= v_2 \oplus v_5 \oplus v_8 = 1 \oplus 0 \oplus 1 = 0 \\
c_4 &= v_3 \oplus v_5 \oplus v_{10} = 0 \oplus 0 \oplus 0 = 0
\end{aligned}
$$

Como todas foram satisfeitas, encerramos o algoritmo e devolvemos (1,0,0,1,0,0,0,1,1,0).

O valor do limite $t$ influencia a probabilidade da decodificação ter sucesso. Discussões sobre o valor ótimo encontram-se em [Gallager, 1962, Misoczki et al., 2013].

### 6.1.3 Uso no Criptossistema de McEliece

Códigos LDPC apresentam uma forma de lidar com o problema do tamanho de chaves no criptossistema de McEliece. Enquanto o esquema clássico requer chaves proibitivamente grandes, códigos LDPC, por serem inerentemente esparsos, permitiriam uma redução das chaves.

Sua utilização no lugar de códigos de Goppa binários irredutíveis como família de códigos do criptossistema de McEliece foi primeiro sugerida em [Rosenthal et al., 2000]. Nesta proposta, as matrizes $S$ e $P$ utilizadas no McEliece precisam ser esparsas. Contudo, no mesmo artigo os autores apontam falhas na segurança do sistema que o tornam impróprio para uso.

Outras propostas foram feitas, usando variações de códigos LDPC, em [Baldi et al., 2006, Baldi et al., 2007, Baldi and Chiaraluce, 2007], mas também foram quebradas, por motivos similares aos da proposta de [Rosenthal et al., 2000].

Uma variante do McEliece usando códigos LDPC proposta em [Baldi et al., 2008] abre mão de um pouco do poder de redução de chaves desses códigos para obter melhor segurança. Os ataque usadas em propostas anteriores baseadas em códigos LDPC não tiveram sucesso em quebrá-la [Misoczki, 2013].

## 6.2 Códigos MDPC

### 6.2.1 Introdução

Enquanto códigos LDPC tem o potencial de diminuir o tamanho da chave do criptossistema de McEliece, propostas anteriores de sua utilização falharam em conciliar esta redução com a manutenção da segurança do sistema.

Então, em [Misoczki et al., 2013] foi feita a proposta de se utilizar códigos MDPC, sigla em inglês para *moderate density parity check*, ou, teste de paridade de densidade moderada. Tais códigos tem matrizes de teste de paridade que, mesmo não sendo densas, são menos esparsas que as de códigos LDPC.

Códigos LDPC e MDPC são muito similares, mas enquanto as linhas da matriz de teste de paridade de um código LDPC tem um peso constante pequeno (normalmente menos que 10), em códigos MDPC assume-se que o peso das linhas é da ordem de $O(\sqrt{n \log n})$. [Misoczki et al., 2013]

Para decodificação, pode-se utilizar o mesmo algoritmo de *Bit-Flipping* que é usado para decodificar códigos LDPC. O aumento da densidade da matriz de teste de paridade diminui a capacidade de correção de erro desta família de códigos. Isto, contudo, não é problemático num contexto criptográfico, onde o interesse não é corrigir *muitos* erros, apenas corrigir um número adequado para garantir a segurança

do sistema.

No caso de uma falha do algoritmo de decodificação, uma implementação do criptossistema de McEliece pode requisitar que a mensagem original seja criptografada e enviada novamente. A chance de falha em ambos os envios é baixa.

### 6.2.2 Códigos MDPC Quase-Cíclicos - QC-MDPC

Apesar de levar a reduções, se o criptossitema de McEliece usar códigos MDPC puros, o tamanho da chave ainda é proibitivo [Misoczki et al., 2013]. Uma forma de se conseguir chaves de tamanhos ainda menores é utilizar a versão **quase-cíclica** dos códigos MDPC:

**Definição 6.1.** *Um código linear $C$ de dimensão $k$ sobre $\mathbb{F}_2^n$ é **quase-cíclico** (QC) se existe um inteiro $\eta$ tal que todo shift circular de $\eta$ bits de uma palavra de $C$ produz outra palavra de $C$.*

Por exemplo, seja $C$ um código quase-cíclico sobre $\mathbb{F}_2^{10}$ no qual $\eta = 3$. Se $(1,1,1,1,1,0,0,0,0,0)$ é uma palavra de $C$, então $(0,0,0,1,1,1,1,1,0,0)$ (obtido por um shift circular de 3 posições para a direita), também o é.

Para aplicações criptográficas, é especialmente interessante o caso em que $n$ é um múltiplo de $\eta$, ou seja, existe um $p$ tal que $n = \eta p$. Neste caso, se $H$ é a matriz de teste de paridade do código, então $H$ é da forma:

$$H = [H_1|H_2|\ldots,H_\eta]$$

Na equação acima, cada $H_i$ é uma matriz $(n-k) \times p$ que tem a propriedade de que a segunda linha é um shift circular de um único bit da primeira linha, a terceira é um shift da segunda e assim sucessivamente.

### 6.2.3 Construindo códigos QC-MDPC

Ao gerar códigos QC-MDPC para uso no criptossistema de McEliece, estamos interessados em gerar códigos quase-cíclicos em que $n = \eta \cdot p$ e $p = r$. Isso significa que as matrizes serão da forma descrita em 6.2.2.

Para gerar um código QC-MDPC aleatório, com dimensão $k$ e cujas linhas tenham algum peso constante $\omega$, começamos escolhendo a primeira linha da matriz de teste de paridade como sendo um vetor aleatório de tamanho $n$ e peso $\omega$. Então, geramos a segunda linha a partir da primeira por meio de um shift circular de $r$ posições. Realizamos o mesmo processo para as outras $r-2$ linhas.

Para obter a matriz de teste geradora G a partir da matriz quase-cíclica $H$, utilizamos os blocos de $H$. Assumindo que $H_\eta$ é não-singular, construímos G como:

$$G = \left( \begin{array}{c|c} & (H_\eta^{-1} \cdot H_0)^T \\ & (H_\eta^{-1} \cdot H_1)^T \\ I & \vdots \\ & (H_\eta^{-1} \cdot H_{\eta-1})^T \end{array} \right)$$

Onde $I$ é a matriz identidade $k \times n$. Temos que $G$ tem $k + r = n$ colunas e $r(\eta - 1) = n - r = k$ colunas, tendo as dimensões necessárias para ser a matriz geradora do código. É também simples de verificar que $HG^T = 0$, mostrando que $G$ realemente gera o código para o qual $H$ é a matriz de teste de paridade.

### 6.2.4 McEliece QC-MDPC

A primeira prosposta de uso de códigos MDPC no criptossistema de McEliece foi feito em [Misoczki et al., 2013]. Neste caso, as matrizes $S$ e $P$, utilizadas para esconder a estrutura algébrica da matriz geradora $G$ e esta é ausente em códigos MDPC.

Desta forma, os algortimos de criptografia e decriptografia são muito similares aos clássicos, mas sem as matrizes $S$ e $P$. Eles são:

**Algoritmo 5:** Algoritmo de Criptografia do McElice

**Data:** $m \in K^k$, a mensagem a ser criptografada
**Result:** $c \in K^n$, a mensagem criptografada
1 Calcule $m' = mG$, a palavra do código correspondente a $m$ ;
2 Selecione um vetor aleatório $e \in K^n$ de peso $\delta$ ;
3 Calcule $c = m' \oplus e$
4 **return** $c$

**Algoritmo 6:** Algoritmo de Decriptogradia do McEliece

**Data:** $c = mG \oplus e$, a mensagem criptografada
**Result:** $m$, a mensagem original
1 Use o corretor de erros $\psi$ para remover os erros e obter $mG$
2 Resolva o sistema linear sobredeterminado dado por $mG = \overline{m}$, obtendo $m$
3 **return** $m$

Nos algoritmos acima, a estrutura corretora de erros $\psi$ é o Algoritmo *Bit-Flipping*, que utiliza a matriz de teste de paridade do código.

A utilização de códigos QC-MDPC permite chaves menores para o criptossistema. Não termos que armazenar as matrizes $S$ e $P$ contribui para a diminuição. Temos também que a estrutura quase esparsa da matriz de teste de paridade de

códigos MDPC permite que apenas as entradas não nulas da matriz sejam armazenadas. Além disso, como essa matriz é quase-cíclica, é necessário armazenar apenas a primeira linha dela e o fator de *shift* $\eta$ para ser possível reconstruir a matriz completa. Isso significa que a matriz de teste de paridade (da qual podemos derivar a matriz geradora) pode ser representada apenas pelas entradas não nulas de sua primeira linha.

# 7 Comparações entre variantes do McEliece

Nos capítulos anteriores descrevemos o criptossistema de McEliece em sua versão original, que utiliza códigos de Goppa binários irredutíveis, bem como versões modificadas que funcionam com códigos de Goppa quase-diádicos (QD-Goppa) e códigos MDPC quase-cíclicos (QC-MDPC).

O principal motivo pelo qual foram desenvolvidas alternativas ao McEliece é o tamanho da chave produzida pelo esquema clássico. Para atingir um nível de segurança de 80 bits (ou seja, ser equivalente a um criptossistema de chave privada com chave de 80 bits), a descrição original necessita de uma chave de cerca de 460 mil bits [Bernstein et al., 2008]. Uma chave RSA, para o mesmo nível de segurança, tem 1024 bits [Barker et al., 2007].

Quase-ciclicidade ou quase diadicidade foram propriedades comumente exploradas para tentar reduzir o tamanho da chave, uma vez que permitem representar uma matriz por apenas algumas de suas partes [Misoczki, 2013]. Isso significa que pode-se obter uma representação linear da matriz (baseada, por exemplo em sua primeira linha, como no cado dos QC-MDPC), ou invés de uma representação quadrática (guardar toda a matriz).

Um esquema usando códigos BHC quase-cíclicos foi proposta por [Gaborit, 2005], mas foi revelado que a redução de chave acarretava também em uma redução de segurança que tornava impraticável o uso dessa solução [Otmani et al., 2010]. Uma outra proposta, usando uma classe de códigos de Reed-Solomon chamada de códigos alternantes, foi feita em [Berger et al., 2009], mas foi quebrada, por motivos similares a anterior em [Faugere et al., 2010].

Como citado no capítulo 6, a maioria das variantes do McEliece que buscavam reduzir o tamanho das chaves por meio de códigos LDPC foram quebradas [Otmani et al., 2010, Baldi and Chiaraluce, 2007, Rosenthal et al., 2000]. A de [Baldi et al., 2008], permanece segura.

As variantes que foram descritas neste trabalho permitem uma grande redução do tamanho da chave. Uma comparação dos tamanhos (em bits) é feita na tabela abaixo, retirada de [Misoczki et al., 2013].

| Nível de Segurança | QC-MDPC | QD-Goppa | Goppa Clássico |
|---|---|---|---|
| 80 | 4801 | 20480 | 460647 |
| 128 | 9857 | 32768 | 1537536 |
| 256 | 32771 | 65536 | 7667855 |

Tabela 1: Comparação entre tamanho de chave do criptossistema clássico e as variantes apresentadas neste trabalho

A variante quase-diádica de Goppa pareceia promissora, uma vez que não era vul-

nerável aos ataques descritos em [Otmani et al., 2010, Faugere et al., 2010] que tiveram sucesso em quebrar variantes do criptossistema de McEliece usando códigos algébricos modificados para permitir tamanhos menores de chave [Misoczki, 2013]. Contudo, um novo ataque, descrito em [Faugere et al., 2016], usando bases de Gröebner, teve sucesso em atacá-la. Até o presente momento, não foi publicada uma possível defesa contra este ataque.

A variante quase-cíclica MDPC não somente produz chaves menores, como parecia resistente devido à ausência de estrutura algébrica desta família de códigos. Em [Misoczki et al., 2013], os autores apresentam reduções de segurança que mostram que, sob uma hipótese de indistinguibilidade, não era possível atacar esta variante.

Contudo, descobriu-se um ataque que explora o fato de que o algoritmo de decodificação de códigos MDPC pode não conseguir decodificar uma mensagem [Guo et al., 2016]. Este ataque foi capaz de eficientemente recuperar a chave privada para implementações do QC-MDPC-McEliece feita usando os parâmetros recomendados pela literatura.

Existem outras propostas baseadas em códigos MDPC [Baldi et al., 2016], usando outras técnicas tanto para codificação e decodificação (portanto, com impactos diferentes na criptografia e decriptografia). Ainda não é claro se elas são vulneráveis ao ataque de [Guo et al., 2016].

Enquanto a prosposta clássica usando códigos de Goppa binários irredutíveis pode apresentar algumas vulnerabilidades [Faugere et al., 2013], ela permanece, no geral, segura. O único empecilho para sua adoção continua sendo o tamanho inviável de chave.

# 8 Ataques ao McEliece

Nesta seção vamos apresentar alguns dos ataques que podem ser utilizados contra o criptossistema de McEliece. Alguns vão focar na recuperação da mensagem original dada a mensagem criptografada, outros em tentar recuperar a chave privada a partir da pública.

Em todos os ataques abaixo assumimos estar trabalhando com mensagens e textos cifrados pertencentes a $\mathbb{F}_2^k$ e $\mathbb{F}_2^n$, respectivamente.

Os ataques aqui descritos foram adaptados de [McEliece, 1978, Engelbert et al., 2007, Bernstein et al., 2008]. Nota-se que todos eles são do tipo *chosen ciphertext attack* (CCA), uma classe de ataques na qual o atacante tem acesso ao texto cifrado e pode modificá-lo [Terada, 2000].

## 8.1 Ataque por conjunto de informação

Este é o ataque o mais simples que pode ser feito ao criptossistema de McEliece e é independente da escolha de código, uma vez que não se preocupa com nenhuma estrutura subjacente que a chave possa possuir. Ele é comumente usado como subrotina de outros ataques.

Seja $m$ a mensagem original. Se o código usado tem dimensão $k$ e gera textos de tamanho $n$, temos que a chave pública $\overline{G}$ tem dimensão $k$ x $n$. Sejam $m' = m\overline{G}$ a palavra codificada e $c = m' \oplus e$ o texto criptografado enviado. Um atacante que busca recuperar $m$ a partir de $c$ pode fazê-lo adivinhando $k$ entradas em $c$ que são idênticas às respectivas posições em $m'$ (equivalentemente: adivinhando $k$ entradas nulas de $e$).

O atacante primeiro escolhe $k$ entradas de $c$ que ele acredita estarem livres de erro. Seja $I = \{i_1, i_2, \ldots, i_k\}$ um conjunto de posições de $c$ que estejam livres de erro, ou seja, índice tais que $c_{i_t} = {m'}_{i_t}$, com $1 \leq t \leq k$. Chamamos $I$ de um **conjunto de informação**.

Depois o atacante monta a matriz $A$ de dimensão $k$ x $k$ em que a $t$-ésima linha de $A$ é igual à $i_t$-ésima linha de $\overline{G}$.

Para um dado $j$, ${m'}_j = \langle m, \overline{g}_j \rangle$, onde $\overline{g}_j$ é a $j$-ésima linha de $\overline{G}$, então, pelas nossa definições, devemos ter:

$$mA = ({m'}_{i_1}, {m'}_{i_2}, \ldots, {m'}_{i_k})$$

Se a matriz $A$ for não-singular, então o sistema linear definido pela equação acima tem uma única solução, que será exatamente a mensagem original, em sua forma decriptografada. Se $A$ não for inversível, o atacante tenta construir um outro

conjunto de informação até que tenha sucesso.

Algoritmicamente temos:

**Algoritmo 7:** Decodificação por conjunto de informação

**Data:** $c = m' \oplus e$ (texto encriptado), $\overline{G}$ (chave pública)

**Result:** $m$ (mensagem original)

1 $I \leftarrow \{i_1, i_2, \ldots, i_k\}$, índices sem erros;
2 Inicializar matriz $A$, $k$ x $n$ ;
3 **for** $i_t$ ***in*** $I$ **do**
4 $\quad a_t = \overline{g}_{i_t}$
5 **end**
6 **if** *$A$ for singular* **then**
7 $\quad$ Volte para o passo **1**
8 **end**
9 Resolver o sistema linear $mA = (m'_{i_1}, m'_{i_2}, \ldots, m'_{i_k})'$
10 **return** $m$

Se o vetor de erros $e$ tem peso de Hamming $\omega$, então a probabilidade de escolher uma posição em $c$ que não contém erros é $(1 - \omega/n)$, o que significa que a probabilidade de encontrar um conjunto de informação é:

$$\left(1 - \frac{\omega}{n}\right)^k$$

Se $X$ é a variável aleatória que representa o número de tentativas que tem que ser feitas para se obter um conjunto de informação, temos:

$$\Pr[X = t] = \left(1 - \frac{\omega}{n}\right)^k \left(1 - \frac{\omega}{n}\right)^{k(t-1)}$$

Isso significa que $X$ segue uma distribuição geométrica com parâmetro $(1-\omega/n)^k$ e, em média, são necessárias $(1 - \omega/n)^{-k}$ tentativas para se encontrar um conjunto de informação.

Para valores práticos de $n$, $k$ e $t$, esse número é bem alto. Por exemplo, se temos $n = 1024$, $t = 50$ e $k = 524$ (valores baixos para os parâmetros do McElice), o número esperado de tentativas ultrapassa dez bilhões.

Vamos apresentar um pequeno exemplo para ilustrar o ataque.

Suponha que nosso corpo seja o $\mathbb{F}_2$ e tenhamos $n = 8$, $k = 3$ e a seguinte chave pública:

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

Se nossa mensagem original for $m = (1\ 0\ 1)$, então $m'$ será:

$$m' = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Se tomamos o vetor de erro como sendo $e = (0\ 0\ 1\ 0\ 0\ 1\ 0\ 0)$, a mensagem a ser enviada será:

$$c = m' \oplus e = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Em média, após $(1 - 2/8)^{-3} = 2.370370...$ tentativas vamos conseguir encontrar um conjunto de informação, por exemplo $\{1, 2, 4\}$ e vamos montar a seguinte matriz $A$:

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

Temos ainda que $(m'_{i_1}, m'_{i_2}, \ldots, m'_{i_k}) = (0\ 0\ 1)$.

Para recuperar a mensagem original, precisamos resolver o sistema:

$$\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

As equações são:

$$\begin{aligned} x \oplus z &= 0 \\ y &= 0 \\ z &= 1 \end{aligned}$$

Fica claro então que a solução é $(1\ 0\ 1)$, que é exatamente nossa mensagem original.

## 8.2 Ataque por palavra de peso mínimo

A ideia deste ataque é recuperar o erro $e$ usado na criptografia. Uma vez feito isso, o ataque por conjunto de informação descrito anteriormente se torna mais simples, umas vez que pode-se escolher de forma determinística e correta quais posições da mensagem cifrada não sofreram alterações.

Ele consiste buscar palavras com peso de Hamming baixo em um novo código.

Seja $\overline{G}$ a matriz geradora do código $C$, $m$ a mensagem original e $c$ a mensagem criptografada. Vamos definir um novo código $C'$ cuja matriz geradora é

$$\begin{pmatrix} \overline{G} \\ c \end{pmatrix}$$

Então vamos sistematicamente enumerar as palavras de peso de Hamming baixo em $C'$ em busca da que tiver o menor peso.

Se esta palavra for $z$, devemos ter que:

$$c = m\overline{G} \oplus z$$

Isso vale pois, caso contrário, teríamos outra mensagem $\mu$ que estaria mais próxima de $c$ do que $m$ está. Isso significaria que durante o processo de decriptografia, $c$ seria corrigido para a palavra $\mu$, o que implicaria no não funcionamento do sistema.

Portanto, $z$ deve ser o erro que foi usado durante a criptografia.

Este ataque é, contudo, computacionalmente inviável, dado que o problema de decidir se um dado código possui uma palavra com um dado peso $\omega$ é $\mathcal{NP}$-completo [Berlekamp et al., 1978].

## 8.3 Ataque por mensagem parcialmente conhecida

Seja $m \in \mathbb{F}_2^k$ a mensagem original. Suponha que conhecemos alguns dos bits de $m$ e seja $\mathcal{I} \subset \{1, \ldots, k\}$ as posições destes bits. Vamos representar o complemento de $\mathcal{I}$ (as posições dos bits *desconhecidos*) por $\mathcal{J}$.

O vetores $m_{\mathcal{I}} \in \mathbb{F}_2^{|\mathcal{I}|}$ é um subvetor de $m$ cujas posições são as de $m$, mas apenas nas posições presentes em $\mathcal{I}$. Definimos analogamente o vetor $m_{\mathcal{J}}$.

Por exemplo, se $m = (1, 1, 0, 1, 1)$ e $\mathcal{I} = \{2, 4\}$ e $\mathcal{J} = \{1, 3, 5\}$, temos:

$$
\begin{aligned}
m_{\mathcal{I}} &= (m_2, m_4) \\
&= (1, 1) \\
m_{\mathcal{J}} &= (m_1, m_3, m_5) \\
&= (1, 0, 1)
\end{aligned}
$$

Se nossa matriz de chave pública é $\overline{G}$, podemos definir a matriz $\overline{G}_{\mathcal{I}}$ que será uma matriz composta pelas linhas $g_i$ de $G$ tal que $i \in \mathcal{I}$. Analogamente definimos $\overline{G}_{\mathcal{J}}$.

Continuando o exemplo e tomando os mesmos conjuntos $\mathcal{I}$ e $\mathcal{J}$, e $\overline{G}$ da forma:

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

Teremos:

$$
\overline{G}_{\mathcal{J}} = \begin{pmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \text{ e } \overline{G}_{\mathcal{I}} = \begin{pmatrix}
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

Podemos então escrever a equação que é chave para este ataque:

$$
m = m_{\mathcal{I}}\overline{G}_{\mathcal{I}} \oplus m_{\mathcal{J}}\overline{G}_{\mathcal{J}}
$$

Podemos usar essa equação para resestruturar nosso texto cifrado da seguinte forma:

$$
\begin{aligned}
c &= m\overline{G} \oplus e \\
c &= m_{\mathcal{I}}\overline{G}_{\mathcal{I}} \oplus m_{\mathcal{J}}\overline{G}_{\mathcal{J}} \oplus e \\
c \oplus m_{\mathcal{I}}\overline{G}_{\mathcal{I}} &= m_{\mathcal{J}}\overline{G}_{\mathcal{J}} \oplus e \\
c' &= m_{\mathcal{J}}\overline{G}_{\mathcal{J}} \oplus e
\end{aligned}
$$

Como conhecemos $c$, $m_{\mathcal{I}}$ e $\overline{G}_{\mathcal{I}}$, podemos facilmente calcular $c'$. Agora o problema de recuperar a mensagem original se resume a recuperar $m_{\mathcal{J}}$.

Para tal, podemos usar o ataque por conjunto de informação tendo como entradas $c'$ e $\overline{G}_{\mathcal{J}}$. Nesse caso, o ataque será bem mais eficiente, pois a mensagem que queremos

recuperar não tem mais tamanho $k$, mas sim tamanho $|\mathcal{J}| < k$. Isso significa que basta escolher $|\mathcal{J}|$ posições de $c'$ que são livres de erro, o que é, probabilisticamente, mais fácil do que escolher $k$ posições corretas de $c$.

O número médio de tentativas até um conjuntos de informação viável ser encontrado cai de $(1 - \omega/n)^{-k}$ para $(1 - \omega/n)^{-|\mathcal{J}|}$, um decréscimo exponencial.

## 8.4 Ataque por mensagem relacionada

Suponha que duas mensagens, $m_1$ e $m_2$ foram criptografadas usando o criptossistema de McEliece e que se conhece a diferença $\Delta_m$ entre elas, ou seja, temos a informação:

$$\Delta_m = m_1 \oplus m_2$$

Sejam $c_1 = m_1\overline{G} \oplus e_1$ e $c_2 = m_2\overline{G} \oplus e_2$ as mensagens cifradas produzidas a partir de $m_1$ e $m_2$, respectivamente. Como os erros usados no processo de criptografia são escolhidos independentemente e forma aleatória, temos que, com alta probabilidade, os vetores $e_1$ e $e_2$ são diferentes.

Neste ataque vamos explorar a informação previamente adquirida para obter dados sobre vetores de erro, e assim facilitar o ataque por conjunto de informação.

Temos:

$$\begin{aligned}
c_1 \oplus c_2 &= (m_1\overline{G} \oplus e_1) \oplus (m_2\overline{G} \oplus e_2) \\
c_1 \oplus c_2 &= (m_1 \oplus m_2)\overline{G} \oplus (e_1 \oplus e_2) \\
c_1 \oplus c_2 &= \Delta_m\overline{G} \oplus (e_1 \oplus e_2) \\
c_1 \oplus c_2 \oplus \Delta_m\overline{G} &= e_1 \oplus e_2
\end{aligned}$$

Se conhecermos $c_1$ e $c_2$, como, por hipótese, sabemos $\Delta_m$ e $\overline{G}$ é pública, podemos calcular $e_\oplus = e_1 \oplus e_2$. Dada a aleatoriedade e independência de $e_1$ e $e_2$ temos que se uma posição de $e_\oplus$ é nula, esta mesma entrada é provavelmente nula em $e_1$ e $e_2$ também.

Esse conhecimento adquirido sobre os vetores de erro usados na criptografia facilita o ataque por conjunto de informação, por aumentar a probabilidade de escolhermos posições inalteradas nas mensagens cifradas.

## 8.5 Ataque por reação

Neste ataque supomos que o atacante pode mandar uma mensagem criptografada $c$ para um servidor, que por sua vez responde **ACK** se conseguir decriptografar a

mensagem e **ERR** caso contrário.

A ideia do ataque é repetidamente alterar a mensagem original $c$ e pedir para o servidor decriptografá-la, explorando o padrão de respostas para conseguir informações que nos auxiliem a recuperar a mensagem original.

Seja $c^i$ a mensagem obtida invertendo-se o $i$-ésimo bit de $c$ e $e$ o erro usado para produzir $c$. Enviamos cada $c^i$, $1 \leq i \leq n$, para o servidor. Se a resposta for **ERR**, significa que devemos ter introduzido um erro a mais na mensagem, o que impossibilitou o servidor de realizar corretamente a criptografia, logo, devemos ter que $e_i = 0$. Analogamente, se a resposta for **ACK**, devemos ter suprimido um erro, uma vez que o servidor ainda conseguiu decriptografar a mensagem, e devemos ter $e_i = 1$.

Em $O(n)$ requisições ao servidor seremos capaz de recuperar $e$ e usar o ataque por conjunto de informação para recuperar a mensagem original.

## 8.6 Ataque por maleabilidade

O objetivo deste ataque não é obter informação sobre a chave secreta ou sobre a mensagem original, mas sim produzir, a partir de um texto cifrado $c$ um novo texto cifrado $c'$, que corresponde a uma mensagem $m'$ que de alguma forma se relaciona a mensagem original $m$.

Tomemos:

$$\begin{aligned} c' &= c \oplus (1, 1, \ldots, 1) \cdot \overline{G} \\ &= (m\overline{G} \oplus e) \oplus (1, 1, \ldots, 1) \cdot \overline{G} \\ &= (m \oplus (1, 1, \ldots, 1))\overline{G} \oplus e \\ &= \widetilde{m}\overline{G} \oplus e \end{aligned}$$

Neste caso, $\widetilde{m}$ é a negação bit a bit da mensagem original $m$. Este ataque então nos permite obter o texto cifrado que corresponde à negação (bit a bit) da mensagem original.

Esta informação pode ser usada, por exemplo, em algum ataque de sabotagem, onde o texto cifrado original é substituído pelo que resulta do ataque. Isso levaria o portador da chave privada a decodificar uma mensagem que é o inverso da enviada.

# 9 Conclusões

Muito úteis em telecomunicações, fundamentados por conceitos de álgebra abstrata e álgebra linear, códigos corretores de erro são estruturas interessantes com grande aplicabilidade e uma rica teoria.

Após a proposta de utilizá-los em um contexto criptográfico [McEliece, 1978] permanecer dormente por algumas décadas, a busca por alternativa pós-quânticas a problemas baseados no logaritmo discreto levou a um novo interesse na área.

A criptografia baseada em códigos corretores de erros possui um grande potencial para substituir o RSA e ECCs em um possível mundo em que o advento de computadores quânticos tornaram o uso destes esquemas criptográficos inviável [Augot et al., 2015].

Reduções a problemas provadamente $\mathcal{NP}$-completos e ou possivelmente difíceis fundamentam sua segurança [Misoczki, 2013, Faugere et al., 2013, Berlekamp et al., 1978] e evidenciam que, para um caso genérico, esse esquema criptográfico é difícil de se atacar.

Ataques genéricos, como os descritos na seção 8, são muitas vezes exponenciais e ou dependem de informações extras, cuja disponibilidade é improvável, ou só veem aplicações em contextos restritos e cuja formulação pode parecer artificial.

Contudo, o grande tamanho das chaves [Bernstein et al., 2008, Misoczki, 2013] ainda torna sua implementação em larga escala desafiadora, principalmente em contextos em que memória é um recurso crítico.

Tentativas de diminuição dessas chaves levaram a variantes do McEliece com vulnerabilidades que puderam ser exploradas a ponto de quebrar a criptografia [Otmani et al., 2010, Faugere et al., 2010].

Este trabalho, baseando-se na tese de doutorado de Misoczki [Misoczki, 2013], deu enfase às variantes que utilizavam códigos de Goppa diádicos e códigos QC-MDPC. Enquanto as ideias apresentadas são interessantes e as propostas pareciam promissoras, ambas foram quebradas durante o desenvolvimento deste projeto [Faugere et al., 2016, Guo et al., 2016].

Contudo, ainda há tentativas de produzir novas variantes, menos propensas a ataques e que ainda produzam chaves pequenas [Baldi et al., 2016, Baldi et al., 2008].

Além disso, há um grande volume de publicações recentes na área, evidenciando o interesse da comunidade acadêmica no tópico e a crença de que ainda é possível obter uma implementação prática e segura de criptossistemas baseados em códigos corretores de erro.

# Referências


[Augot et al., 2015] Augot, D., Batina, L., Bernstein, D. J., Bos, J., Buchmann, J., Castryck, W., Dunkelman, O., Güneysu, T., Gueron, S., Hülsing, A., et al. (2015). Initial recommendations of long-term secure post-quantum systems. *Available at pqcrypto.eu.org/docs/initial-recommendations.pdf*.

[Baldi et al., 2008] Baldi, M., Bodrato, M., and Chiaraluce, F. (2008). A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *International Conference on Security and Cryptography for Networks*, pages 246–262. Springer.

[Baldi and Chiaraluce, 2007] Baldi, M. and Chiaraluce, F. (2007). Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *2007 IEEE International Symposium on Information Theory*, pages 2591–2595. IEEE.

[Baldi et al., 2006] Baldi, M., Chiaraluce, F., and Garello, R. (2006). On the usage of quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *2006 First International Conference on Communications and Electronics*.

[Baldi et al., 2007] Baldi, M., Chiaraluce, F., Garello, R., and Mininni, F. (2007). Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *2007 IEEE International Conference on Communications*, pages 951–956. IEEE.

[Baldi et al., 2016] Baldi, M., Santini, P., and Chiaraluce, F. (2016). Soft McEliece: MDPC code-based McEliece cryptosystems with very compact keys through real-valued intentional errors. *arXiv preprint arXiv:1606.01040*.

[Barker et al., 2007] Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2007). Nist special publication 800-57. *NIST Special Publication*, 800(57):1–142.

[Berger et al., 2009] Berger, T. P., Cayrel, P.-L., Gaborit, P., and Otmani, A. (2009). Reducing key length of the McEliece cryptosystem. In *Progress in Cryptology–AFRICACRYPT 2009*, pages 77–97. Springer.

[Berlekamp et al., 1978] Berlekamp, E. R., McEliece, R. J., and Van Tilborg, H. C. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386.

[Bernstein et al., 2008] Bernstein, D. J., Lange, T., and Peters, C. (2008). Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography*, pages 31–46. Springer.

[Buchmann et al., 2004] Buchmann, J. A., García, L. C. C., Döring, M., Engelbert, D., Ludwig, C., Overbeck, R., Schmidt, A., Vollmer, U., and Weinmann, R.-P. (2004). Post-quantum signatures. *IACR Cryptology ePrint Archive*, 2004:297.

[Engelbert et al., 2007] Engelbert, D., Overbeck, R., and Schmidt, A. (2007). A summary of McEliece-type cryptosystems and their security. *J. Mathematical Cryptology*, 1(2):151–199.

[Faugere et al., 2013] Faugere, J.-C., Gauthier-Umana, V., Otmani, A., Perret, L., and Tillich, J.-P. (2013). A distinguisher for high-rate McEliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844.

[Faugere et al., 2016] Faugere, J.-C., Otmani, A., Perret, L., De Portzamparc, F., and Tillich, J.-P. (2016). Structural cryptanalysis of McEliece schemes with compact keys. *Designs, Codes and Cryptography*, 79(1):87–112.

[Faugere et al., 2010] Faugere, J.-C., Otmani, A., Perret, L., and Tillich, J.-P. (2010). Algebraic cryptanalysis of McEliece variants with compact keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 279–298. Springer.

[Gaborit, 2005] Gaborit, P. (2005). Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91.

[Gallager, 1962] Gallager, R. (1962). Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28.

[Goppa, 1970] Goppa, V. D. (1970). A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30.

[Guo et al., 2016] Guo, Q., Johansson, T., and Stankovski, P. (2016). A key recovery attack on MDPC with CCA security using decoding errors. In *22nd Annual International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT), 2016.*

[Li et al., 1994] Li, Y. X., Deng, R. H., and Wang, X. M. (1994). On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273.

[McEliece, 1978] McEliece, R. J. (1978). A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116.

[Merkle, 1979] Merkle, R. (1979). *Security, authentication and public-key systems - A certified digital signature.* PhD thesis, Stanford University.

[Miller, 1986] Miller, V. (1986). Use of elliptic curves in cryptography. *Advances in Cryptology (CRYPTO85)*, pages 417–426.

[Misoczki, 2013] Misoczki, R. (2013). *Two Approaches for Achieving Efficient Code-Based Cryptosystems.* PhD thesis, Université Pierre et Marie Curie-Paris VI.

[Misoczki and Barreto, 2009] Misoczki, R. and Barreto, P. S. (2009). Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, pages 376–392. Springer.

[Misoczki et al., 2013] Misoczki, R., Tillich, J.-P., Sendrier, N., and Barreto, P. S. (2013). MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE.

[Niederreiter, 1986] Niederreiter, H. (1986). Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166.

[Otmani et al., 2010] Otmani, A., Tillich, J.-P., and Dallot, L. (2010). Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3(2):129–140.

[Patterson, 1975] Patterson, N. J. (1975). The algebraic decoding of Goppa codes. *Information Theory, IEEE Transactions on*, 21(2):203–207.

[R. L. Rivest, 1978] R. L. Rivest, A. Shamir, L. M. A. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[Rosenthal et al., 2000] Rosenthal, J., Monico, C., and Shokrollahi, A. (2000). Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory (ISIT'2000).*

[Shannon, 1949] Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

[Shor, 1997] Shor, P. W. (1997). Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1481–1509.

[Terada, 2000] Terada, R. (2000). *Segurança de dados: criptografia em redes de computador*. Editora Blucher.

[Tzeng and Zimmermann, 1975] Tzeng, K. and Zimmermann, K. (1975). On extending Goppa codes to cyclic codes (corresp.). *IEEE Transactions on Information Theory*, 21(6):712–716.