**UNIVERSIDADE DE BRASÍLIA**
**FACULDADE DE TECNOLOGIA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

# QC-MDPC MCELIECE: UMA IMPLEMENTAÇÃO OTIMIZADA DE UMA NOVA VARIANTE MCELIECE

**HOMERO DE OLIVEIRA MARTINS**

**ORIENTADOR: ANDERSON CLAYTON ALVES NASCIMENTO**

**DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA**

**PUBLICAÇÃO: 569/14 DM PPGEE**
**BRASÍLIA/DF: JULHO – 2014**

## FICHA CATALOGRÁFICA

MARTINS, HOMERO DE OLIVEIRA.
QC-MDPC McEliece: uma implementação otimizada de uma nova variante McEliece, [Distrito Federal] 2014.
66 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2014). Dissertação de Mestrado - Universidade de Brasília. Faculdade de Tecnologia. Departamento de Engenharia Elétrica. Programa de Pós-Graduação em Engenharia Elétrica.

1. Criptografia pós-quântica
2. Criptografia baseada em códigos
3. Teoria de códigos
4. Decodificação eficiente

I. ENE/FT/UnB. II. Título (série)

## REFERÊNCIA BIBLIOGRÁFICA

MARTINS, H. O. (2014). QC-MDPC McEliece: uma implementação otimizada de uma nova variante McEliece. Dissertação de Mestrado em Engenharia Elétrica, Publicação 569/14 DM PPGEE, Departamento de Engenharia Elétrica, Programa de Pós-Graduação em Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 66 p.

## CESSÃO DE DIREITOS

AUTOR: Homero de Oliveira Martins.
TÍTULO: QC-MDPC McEliece: uma implementação otimizada de uma nova variante McEliece.

GRAU: Mestre ANO: 2014



**Homero de Oliveira Martins**

**DEDICATÓRIA**

*Para Patrícia e João Pedro, com amor.*

*"Encryption is a powerful defensive weapon for free people. It offers a technical guarantee of privacy, regardless of who is running the government. It's hard to think of a more powerful, less dangerous tool for liberty."*

*Esther Dyson*

**AGRADECIMENTOS**

Esta dissertação de mestrado foi feita sob a supervisão do professor Anderson Clayton Alves Nascimento. Gostaria, portanto, de agradecê-lo por todo o constante apoio que sempre recebi desde meu ingresso como aluno do programa de mestrado em engenharia elétrica da Universidade de Brasília, bem como pelo importante conhecimento transmitido em várias disciplinas por ele ministradas nas áreas de criptografia, segurança e teoria da informação.

Também muito colaborou com a definição do tema do trabalho e desenvolvimento da pesquisa o professor Diego de Freitas Aranha. Agradeço a ele por toda a paciência na orientação e acompanhamento do trabalho ao longo do último ano. Sem a sua experiência, intervenção e direcionamento a pesquisa não teria alcançado o objetivo e muito menos os resultados satisfatórios.

Concomitante com a condição de aluno de mestrado, continuei respondendo por minhas atribuições como analista legislativo de informática na Câmara dos Deputados. Portanto, agradecimentos também devem ser registrados ao amigo Sandro Heleno de Sene Trindade que na condição de diretor sempre apoiou a minha situação de aluno e dessa maneira me incentivou a continuar meus estudos.

Um muito obrigado também a Fabiano Peruzzo Schwartz, que enquanto aluno de doutorado do mesmo departamento de engenharia elétrica, foi quem primeiro me incentivou para que eu tomasse a decisão de ingressar no mestrado.

**RESUMO**

**QC-MDPC MCELICE: UMA IMPLEMENTAÇÃO OTIMIZADA DE UMA NOVA VARIANTE MCELIECE**

**Autor: Homero de Oliveira Martins**

**Orientador: Anderson Clayton Alves Nascimento**

**Programa de Pós-Graduação em Engenharia Elétrica**

**Brasília, Julho de 2014.**

Esta dissertação apresenta a implementação de uma versão otimizada de uma variante McEliece. O criptossistema de McEliece é um exemplo de criptografia baseada em códigos que representa uma alternativa aos criptossistemas atuais mais populares e comerciais, pois até o presente momento ele é imune à computação quântica. Possui algoritmos rápidos e simples, porém uma desvantagem sua é o tamanho das chaves com as quais trabalha. Ao substituir os códigos Goppa da proposta McEliece original por códigos LDPC e MDPC é possível conseguir chaves muito menores. Ao aplicar técnicas de programação tais como a paralelização de operações e ao mesmo tempo utilizar decodificadores eficientes de códigos LDPC é possível alcançar bons resultados e um desempenho otimizado do criptossistema baseado em códigos provando que ele deve de fato ser levado em consideração como um forte candidato substituto para o RSA e o DSA enquanto os computadores quânticos surgem para facilmente computar logaritmos discretos e fatorar grandes números inteiros.
**Palavras-chaves**: criptografia pós-quântica, criptografia baseada em códigos, teoria de códigos e decodificação eficiente.

**ABSTRACT**

**QC-MDPC McEliece: an Optimized Implementation of a New McEliece Variant**

**Author: Homero de Oliveira Martins**

**Supervisor: Anderson Clayton Alves Nascimento**

**Post-Graduation Program in Electrical Engineering**

**Brasília, July of 2014.**

This paper presents the implementation of an optimized version of a McEliece variant. The McEliece cryptosystem is an example of code-based cryptography which is an alternative to the most popular and commercial cryptosystems nowadays as it is believed to be immune to quantum computing. It has simple and fast algorithms, but its drawback is the size of the keys it has to deal with. By substituting the Goppa codes of the McEliece original proposal by LDPC and MDPC codes it's possible to achieve much smaller keys. And by applying programming technics such as parallelization of operations and also utilizing efficient decoders of LDPC codes it's possible to achieve really good results and optimal performances of the code-based cryptosystem showing that it really has to be considered as a strong substitute to RSA and DSA as quantum computers emerge to easily compute discrete logarithms and factor large integers.
**Key-words**: post-quantum cryptography, code-based cryptography, coding-theory and efficient decoding.

**SUMÁRIO**

**LISTA DE ALGORITMOS**

**LISTA DE TABELAS**

**LISTA DE FIGURAS**

## LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

AES – Advanced Encryption Standard

AVX – Advanced Vector Extensions

AVX2 – Advanced Vector Extensions 2

CA – Certificate Authority

DES – Data Encryption Standard

DOOM – Decoding One Out Of Many

DSA – Digital Signature Algorithm

ECDSA – Elliptic Curve Digital Signature Algorithm

ISD – Independent Segment Decoding

LDPC – Low Density Parity Check

MDPC – Medium Density Parity Check

NIST – National Institute of Standards and Technology

RRE – Reduced Row Echelon

QC – Quasi-Cyclic

QC-MPDC – Quasi-Cyclic Medium Density Parity Check

QSC – Qary Symmetric Channel

RAM – Random Access Memory

RSA – Rivest, Shamir e Adleman

SIMD – Single Instruction Multiple Data

UPC – Unsatisfied Parity Check

# 1 INTRODUÇÃO

## 1.1 CONTEXTUALIZAÇÃO E FORMULAÇÃO DO PROBLEMA

A criptografia contemporânea lida com questões tais como sigilo, encriptação de mensagens, distribuição de chaves, assinaturas digitais, geradores de números pseudoaleatórios e funções de resumo criptográfico. Existem inúmeras aplicações que fazem uso da criptografia contemporânea, como por exemplo: assinatura digital de documentos, votação eletrônica e sigilo de transações financeiras na internet. O objetivo da criptografia contemporânea é projetar sistemas que atinjam requisitos de segurança independentes do modo como um adversário poderá utilizá-los. Para tal, a única premissa assumida é que o adversário possui um computador e está restrito a operações em tempo polinomial. Algoritmos de tempo polinomial são considerados rápidos. Segundo Terr, um algoritmo é considerado de tempo polinomial se o número de passos necessários para o algoritmo ser executado por completo para uma determinada entrada é assintoticamente no máximo $n^k$ ($O(n^k)$) para algum inteiro não negativo $k$, onde $n$ é a complexidade da entrada, o que, afirma Sipser (2006, p. 249), intuitivamente quer dizer que o número de passos é menor ou igual a $n^k$ desconsiderando uma diferença determinada por algum fator constante.

Formalmente, segundo Sipser (2006, p. 249), sejam $f$ e $g$ funções $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. Dizemos que $f(n) = O(g(n))$ se existem inteiros positivos $c$ e $n_0$ tal que para todo inteiro $n \geq n_0$

$$f(n) \leq cg(n).$$

A criptografia pós-quântica preocupa-se com a pesquisa de primitivas criptográficas que são imunes tanto aos computadores quânticos quanto aos computadores atuais.

A segurança de muitos criptossistemas, incluindo-se o popular RSA, encontra-se fundamentada na premissa de que a fatoração de números grandes é difícil de ser efetuada. Existem outros criptossistemas bastante difundidos cuja segurança reside na dificuldade do cálculo de logaritmos discretos.

Em 1994, Peter Shor (Shor, 1997) apresentou um algoritmo capaz de fatorar números em tempo polinomial num computador quântico, assim como um algoritmo para o cálculo do logaritmo discreto. Porém, isto não significa que a criptografia esteja com os seus dias contados a partir do momento que os computadores quânticos deixarem de fazer parte dos

modelos teóricos e dos sofisticados experimentos laboratoriais e passarem a integrar o nosso dia a dia. Com certeza este momento significará apenas a recomendável aposentadoria de alguns de nossos criptossistemas mais populares tais como o RSA, o DSA e o ECDSA, devido à impossibilidade de uso de forma segura, pois a segurança deles reside na dificuldade de fatoração de números grandes e do cálculo de logaritmos discretos.

Porém, enquanto o advento do computador quântico não ocorre, pesquisadores criptógrafos vêm testando e estudando criptossistemas considerados como candidatos substitutos para aqueles que terão sua utilização fortemente desaconselhada, de modo que existam alternativas seguras para se continuar, por exemplo, realizando transações financeiras de forma confiável através de sistemas computacionais.

A tabela abaixo apresenta a situação atual de muitos criptossistemas com relação aos algoritmos quânticos:

Tabela 1.1–Situação de segurança de criptossistemas clássicos em relação a computadores quânticos. Fonte: Hallgren e Vollmer (2009, p. 16).

| **Criptossistema** | **Quebrado por algoritmo quântico?** |
|---|---|
| Criptografia de chave pública RSA | Quebrado |
| Troca de chaves Diffie-Hellman | Quebrado |
| Criptografia de curvas elípticas | Quebrado |
| Troca de chaves Buchmann-Williams | Quebrado |
| Homomorfismo algébrico | Quebrado |
| Criptografia de chave pública McEliece | Não quebrado ainda |
| Criptografia de chave pública NTRU | Não quebrado ainda |
| Criptografia de chave pública sobre reticulados | Não quebrado ainda |

Segundo Bernstein (2009, p. 1), existem classes importantes de criptossistemas que acredita-se serem resistentes tanto aos computadores clássicos quanto aos quânticos:

> - criptografia baseada em hash *"hash-based cryptography"*, cujo exemplo é o criptossistema de assinatura digital de chave pública de Merkle (1979) baseado em árvores de hash*;*
>
> - criptografia baseada em códigos *"code-based cryptography",* como exemplo temos o criptossistema de chave pública de McEliece (1978) baseado em códigos Goppa*;*

- criptografia baseada em reticulados *"lattice-based cryptography"*, que tem como exemplo o criptossistema de chave pública "NTRU" de Hoffstein-Pipher-Silverman (1998)*;*

- critptografia baseada em equações multivariadas quadráticas *"multivariate-quadratic-equations cryptography"*, como exemplo há o criptossistema de chave pública "HFE$^{v-}$" de Patarin (1996)*;*

- criptografia de chave secreta *"secret-key cryptography"*, cujo exemplo é a cifra "Rijndael" de Daemen-Rijmen (1998), renomeada para "AES".

A criptografia baseada em códigos, além de ser considerada uma alternativa viável para aplicações futuras devido a sua natureza "pós-quântica", também é capaz de proporcionar benefícios para as aplicações atuais devido a seu desempenho algorítmico, o qual é várias ordens de complexidade inferior àquelas apresentadas pelos criptossistemas tradicionais.

O criptossistema de McEliece (1978) corresponde à primeira proposta de um criptossistema baseado em códigos, apresentado originalmente usando códigos de Goppa. Sua segurança é suportada por duas premissas: a indistinguibilidade da família de códigos e a dificuldade de decodificação de um código linear genérico.

Apesar de seu desempenho e de acreditar-se ser imune ao computador quântico, este criptossistema não é entendido nem utilizado como um substituto do RSA. Segundo Overbeck e Sendrier (2009, p. 95), isto ocorre parcialmente devido ao tamanho de sua chave pública, que vai de 100 kilobytes a vários megabytes. A outra razão, para eles, é a falta de publicidade do criptossistema. Bernstein (2009, p. 3) afirma que para níveis de segurança reais tais como 128 bits, o RSA trabalha com chaves de alguns milhares de bits, enquanto as chaves do McEliece se aproximam do milhão de bits para este mesmo nível de segurança, o que, segundo ele, é a razão para não estarmos utilizando o McEliece em vez do RSA.

Portanto, nos últimos anos o criptossistema de McEliece é um dos que vêm sendo estudado por pesquisadores criptógrafos numa tentativa de encontrar códigos alternativos que sejam substitutos dos códigos de Goppa, de tal maneira a se conseguir chaves com tamanhos consideravelmente menores. Este trabalho corresponde à implementação de uma proposta recente de Misozki, Tillich, Sendrier e Barreto (2013) na qual níveis de segurança atuais são alcançados com chaves de dimensões próximas às dos criptossistemas de chave pública atualmente difundidos. Preocupa-se também com a otimização desta implementação para que apresente tempos de processamento melhorados de tal maneira a evidenciar e reforçar

a proposta de McEliece adaptada como candidata a alternativa viável caso a computação quântica de fato se estabeleça.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo geral

Este trabalho tem por objetivo principal implementar e otimizar a proposta de Misoczki, Tillich, Sendrier e Barreto (2013) acerca de uma variante do criptossistema de McEliece de tal maneira a alcançar tempos de execução inferiores aos encontrados na proposta.

## 1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em cinco capítulos, incluindo este capítulo.

No capítulo dois é apresentada uma visão geral do referencial teórico, objetivando a compreensão das tecnologias e conceitos. São abordados os seguintes temas: (i) Criptografia, (ii) Grupos, Anéis e Corpos, (iii) Códigos; (iv) Criptografia baseada em Códigos; (v) QC-MDPC McEliece, uma variante do Criptossistema de Chave Pública de McEliece.

O capítulo três detalha a implementação, o capítulo quatro descreve os resultados obtidos e o capítulo cinco discute os pontos de maior importância envolvendo o tema deste estudo, assim como aborda algumas conclusões e apresenta sugestões de pesquisas futuras que podem ser desenvolvidas a partir das ideias apresentadas como resultados desta pesquisa.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é feita a abordagem do conteúdo teórico envolvido na compreensão da pesquisa desenvolvida. O capítulo descreve a criptografia e seus conceitos básicos e trata da criptografia de chave pública. A seguir aborda tópicos de álgebra ao tratar das definições de grupos, anéis e corpos e descreve o que são códigos lineares e códigos LDPC. No item seguinte a criptografia é relacionada aos códigos, onde fala-se sobre a criptografia baseada em códigos e o criptossistema de chave pública McEliece é dado como exemplo. Em seguida descreve-se o QC-MDPC McEliece, que é uma variante do criptossistema de chave pública McEliece e que corresponde ao alvo deste estudo. Sobre o QC-MDPC McEliece são explicados os seus conceitos básicos, encriptação e decriptação e é discutida a decodificação eficiente e questões de segurança.

## 2.1 CRIPTOGRAFIA

### 2.1.1 Conceitos básicos

O modelo do canal de comunicação normalmente estudado em criptografia encontra-se apresentado na Figura 2.1. Neste modelo Stinson (2006, p.2) define o objetivo fundamental da criptografia, que é permitir que duas pessoas, usualmente denominadas Alice e Bob, possam se comunicar através de um canal inseguro de tal maneira que um adversário, Oscar, não consiga entender o que está sendo conversado. O canal inseguro pode ser, por exemplo, uma linha telefônica ou uma rede de computadores. A informação que Alice envia a Bob, um texto claro que pode ser uma mensagem escrita ou uma imagem, por exemplo, possui estrutura completamente arbitrária. Alice encripta a mensagem fazendo uso de uma chave predefinida e envia a mensagem cifrada através do canal. Oscar, ao espionar o canal e interceptar a mensagem cifrada, é incapaz de determinar o conteúdo desta. Porém, Bob, que tem conhecimento da chave utilizada por Alice, pode decriptar a mensagem cifrada e recuperar seu texto original.

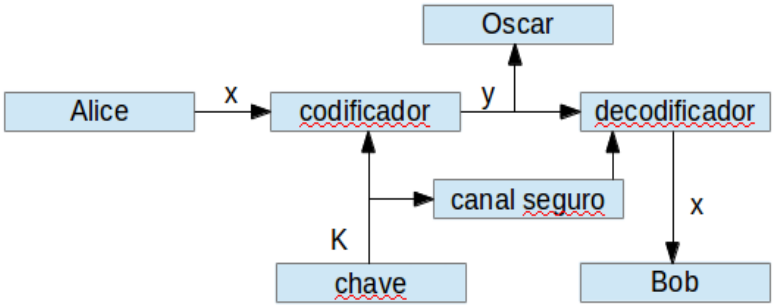


Figura 2.1–Canal de comunicação. Fonte: Stinson (2006, p. 2).

As ideias apresentadas no parágrafo anterior podem ser representadas formalmente por meio de notação matemática. Um criptossistema corresponde a uma quíntupla ($\mathcal{P}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{E}$, $\mathcal{D}$) onde as seguintes condições são satisfeitas:

- $\mathcal{P}$ é um conjunto finito de possíveis textos simples de mensagens;
- $\mathcal{C}$ é um conjunto finito de possíveis textos cifrados de mensagens;
- $\mathcal{K}$ é um conjunto finito de possíveis chaves;
- Para cada $K \in \mathcal{K}$, existe uma regra de encriptação $e_K \in \mathcal{E}$ e uma regra de decriptação correspondente $d_K \in \mathcal{D}$. Cada $e_K : \mathcal{P} \rightarrow \mathcal{C}$ e $d_K : \mathcal{C} \rightarrow \mathcal{P}$ são funções tais que $d_K(e_K(x)) = x$ para todo texto claro $x \in \mathcal{P}$.

### 2.1.2 Criptografia de Chave Pública

No modelo apresentado na Figura 2.1, Alice e Bob escolhem secretamente o valor da chave $K$. Esta chave é então empregada tanto na regra de encriptação $e_K$ quanto na regra de decriptação $d_K$. De fato, normalmente $d_K$ é igual a $e_K$ ou é facilmente derivada em tempo polinomial a partir dela segundo Stinson (2006, p. 161). Um criptossistema deste tipo é conhecido como criptossistema de chave simétrica e a descoberta tanto de $d_K$ como de $e_K$ tornam o sistema inseguro.

Stinson (2006, p. 161) lembra que uma desvantagem inerente aos criptossistemas de chave simétrica quando utilizados exclusivamente é que torna-se obrigatória a comunicação prévia da chave $K$ entre Alice e Bob por intermédio de um canal seguro antes da transmissão de qualquer mensagem cifrada entre ambos, o que em prática pode ser difícil

de ser conseguido.

Segundo Stinson (2006, p. 161), num criptossistema de chave pública, dado o conhecimento de $e_K$ a determinação de $d_K$ é inviável computacionalmente. Neste caso, a regra de encriptação $e_K$ corresponde a uma chave pública que pode ser divulgada em algum lugar, um diretório por exemplo. Portanto, o criptossistema de chave pública tem como vantagem o fato de qualquer pessoa, inclusive Alice, poder enviar uma mensagem encriptada para Bob por intermédio da regra $e_K$ sem a necessidade de comunicação prévia através de um canal seguro da chave secreta acordada por ambos. Bob, por sua vez, será a única pessoa capaz de decriptar a mensagem utilizando-se da regra $d_K$, que é chamada de chave privada.

Deve ser observado que a função pública de encriptação utilizada por Bob, $e_K$, deve ser fácil de ser computada. No entanto, o cálculo da função inversa, que corresponde à operação de decriptação, deve ser difícil para qualquer pessoa que não seja Bob. Uma função que é fácil de ser calculada, mas difícil de ser invertida, é dita uma função candidata de única via. No âmbito da encriptação é necessário que $e_K$ seja uma função de única via injetiva. Porém, num criptossistema de chave pública não é suficiente apenas que uma função de mão única injetiva seja empregada na encriptação de mensagens, pois $e_K$ deve ser de mão única para todos menos para Bob, que precisa ser capaz de decriptar as mensagens que recebe. É preciso, portanto, que Bob conheça uma informação secreta que permita que ele inverta $e_K$ facilmente. Ou seja, Bob consegue decriptar as mensagens endereçadas a ele porque tem conhecimento extra de um segredo, digamos $K$, que o possibilita fazer uso da função de decriptação $d_K$.

A ideia do criptossistema de chave pública foi apresentada por Diffie e Hellman em 1976 num artigo no qual os autores desafiavam a comunidade de criptólogos a apresentar um algoritmo criptográfico que atendesse aos requisitos de um sistema de chave pública. Em 1977, Rivest, Shamir e Adleman propuseram o famoso criptossistema RSA, que representou uma das primeiras respostas bem-sucedidas ao desafio de Diffie e Hellman (1976). Rivest, Shamir e Adleman (1978) publicaram sua proposta no ano seguinte, a qual desde então é a mais aceita, implementada e difundida solução para a encriptação com chave pública. Vários outros criptossistemas de chave pública foram propostos; em todos a segurança encontra-se fundamentada em diferentes problemas computacionais, de acordo com Stinson (2006, p. 161). Destes, o mais importante é o RSA (e suas variantes), no qual

a segurança dos criptossistemas reside na dificuldade de fatoração de números inteiros grandes; o criptossistema de El Gamal (1984) (e variações como os criptossistemas de curva elíptica), nos quais a segurança está no problema do logaritmo discreto e o criptossistema de McEliece, cuja segurança está na premissa da dificuldade de decodificação de um código linear genérico.

Stallings (2011, p. 37) define um criptossistema com segurança incondicional *"se o texto cifrado produzido pelo criptossistema não contém informação que possibilite determinar univocamente o texto claro correspondente, independente de quanto texto cifrado estiver disponível. Ou seja, independente de quanto tempo ou recurso um adversário tiver disponível, é impossível para ele decifrar a mensagem simplesmente porque a informação necessária não está contida nela"*. É importante observar que um criptossistema de chave pública nunca pode garantir segurança incondicional. Isto porque um adversário, ao observar uma mensagem cifrada $y$, pode por sua vez encriptar sucessivas mensagens utilizando a chave pública $e_K$ até que ele encontre a mensagem única $x$ tal que $y = e_K(x)$. Esta mensagem $x$ corresponde à decriptação de $y$.

## 2.2 GRUPOS, ANÉIS E CORPOS

Todas as definições abaixo foram retiradas de Milne (2013).

### 2.2.1 Definições Básicas

Um **grupo** é definido como um conjunto $G$ com uma operação binária

$$(a,b) \mapsto a * b : G \times G \to G$$

satisfazendo as seguintes condições:

**G1**: (associatividade) para todo $a, b, c \in G$,

$$(a * b) * c = a * (b * c);$$

**G2**: (existência de um elementro neutro) existe um elemento $e \in G$ tal que

$$a * e = a = e * a$$

para todo $a \in G;$

**G3**: (existência de inversos) para todo $a \in G$, existe um $a' \in G$ tal que

$$a * a' = e = a' * a.$$

Dois grupos $(G, *)$ e $(G', *')$ são ditos **isomórficos** se existe uma correspondência de um para um $a \leftrightarrow a'$, $G \leftrightarrow G'$ tal que $(a * b)' = a' *' b'$ para todo $a, b \in G$.

A **ordem** $|G|$ de um grupo $G$ é sua cardinalidade. Um grupo finito cuja ordem é uma potência de um número primo $p$ é chamado de um $p$-grupo.

Seja $S$ um subconjunto não vazio de um grupo $G$. Se

**S1**: $a, b \in S \Rightarrow ab \in S$, e

**S2**: $a \in S \Rightarrow a^{-1} \in S$,

então a operação binária em $G$ faz de $S$ um grupo, pois **S1** implica que a operação binária em $G$ define uma operação binária $S \times S \rightarrow S$ em $S$, que é automaticamente associativa. Por suposição $S$ contém, pelo menos, um elemento $a$, seu inverso $a^{-1}$ e o produto $e = aa^{-1}$. Por fim, **S2** garante que os elementos inversos de $S$ estão em $S$.

Um subconjunto não vazio $S$ satisfazendo **S1** e **S2** é chamado de **subgrupo** de $G$.

Dados um subconjunto $S$ de um grupo $G$ e um elemento $a \in G$, sejam

$$aS = \{as \mid s \in S\}$$

$$Sa = \{sa \mid s \in S\}.$$

Pela lei associativa, $a(bS) = (ab)S$ e, portanto, este conjunto pode ser denotado sem ambiguidade por $abS$.

Quando $H$ é um subgrupo de $G$, os conjuntos da forma $aH$ são chamados de c**lasses laterais à esquerda** de $H$ em $G$, e os conjuntos da forma $Ha$ são ditos **classes laterais à direita** de $H$ em $G$. Como $e \in H$, $aH = H$ se e somente se $a \in H$.

Um **grupo abeliano**, ou **grupo comutativo**, é um grupo $(G, *)$ no qual

$$a * b = b * a$$

para todo $a, b \in G$.

Um **monoide** é um conjunto $G$ com uma operação binária

$$(a,b) \mapsto a * b : G \times G \to G$$

satisfazendo as seguintes condições:

**G1**: (associatividade) para todo $a, b, c \in G$,

$$(a * b) * c = a * (b * c);$$

**G2**: (existência de um elementro neutro) existe um elemento $e \in G$ tal que

$$a * e = a = e * a$$

para todo $a \in G;$

**G3**: (fechamento) para todo $a, b \in G,$

$$a * b \in G.$$

Pela definição, um monoide para o qual todo elemento possui um elemento inverso é um grupo.

Um **anel** é um conjunto $A$ com duas operações binárias adição "+" e multiplicação "." tal que o anel $(A, +, .)$ é um grupo abeliano sobre a adição, ou seja:

**A1**: para todo $a, b, c \in A$

$$(a + b) + c = a + (b + c);$$

**A2**: para todo $a, b \in A$

$$a + b = b + a;$$

**A3**: existe um elemento neutro $0 \in A$ tal que

$$a + 0 = a$$

para todo $a \in A;$

**A4**: para todo $a \in A$ *existe um* $a' \in A$ tal que

$$a + a' = 0.$$

o anel $(A, +, .)$ é um monoide sobre a multiplicação, portanto:

**M1**: para todo $a, b, c \in A$

$$(a \,.\, b) \,.\, c = a \,.\, (b \,.\, c);$$

**M2**: existe um elemento neutro $1 \in A$ tal que

$$a \,.\, 1 = a$$

para todo $a \in A;$

**M3**: para todo $a, b \in A$

$$a \,.\, b \in A;$$

e além disso o anel $(A, +, .)$ tem a multiplicação distributiva sobre a adição, ou seja:

**AM**: para todo $a, b, c \in A$

$$a \,.\, (b + c) = a \,.\, b + a \,.\, c.$$

Se além disso, a multiplicação for comutativa

$$a \,.\, b = b \,.\, a.$$

o anel $(A, +, .)$ é dito **anel comutativo**.

Um **corpo** é um anel $A$ tal que todo elemento não nulo é invertível.

Um **corpo finito**, ou **corpo de Galois**, é um corpo que contém um número finito de elementos, chamado de ordem. Corpos finitos só existem quando sua ordem é uma potência de um número primo $p^k$, onde $p$ é um número primo e $k$ é um inteiro positivo. Para cada potência de um número primo existe um corpo finito com ordem correspondente à potência e todos os corpos de uma determinada ordem são ditos **isomórficos**.

$\boldsymbol{F}_q$ é um corpo finito com $q$ elementos onde $q$ é uma potência de um número primo.

## 2.3 CÓDIGOS

### 2.3.1 Códigos Lineares

As definições todas encontradas neste tópico estão em McEliece (2004).

Um código $(n,k)$-linear sobre $\boldsymbol{F}_q$ é um subespaço $k$-dimensional do espaço vetorial $n$-dimensional $\boldsymbol{V}n(\boldsymbol{F}_q) = \{(x_1, \ldots, x_n): x_i \in \boldsymbol{F}_q\}$; $n$ é o comprimento do código e $k$ a dimensão. A taxa do código é dada pela razão $k/n$.

Uma vantagem dos códigos lineares é a facilidade com que eles são especificados. Um código $(n,k)$-linear $\mathcal{C}$ pode ser descrito completamente por qualquer conjunto $k$ de palavras código linearmente independentes $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, pois cada palavra código é uma das $q^k$ combinações lineares $\sum_{i=1}^{k} a_i x_i$ , $a_i \in \boldsymbol{F}_q$. Se dispusermos as palavras código numa matriz $G_{k \times n}$, $G$ é dita a matriz geradora de $\mathcal{C}$.

Outra vantagem dos códigos lineares é a facilidade de codificação. Um código $(n,k)$-linear possui $q^k$ palavras código e, portanto, pode ser utilizado para a comunicação de qualquer uma das $q^k$ mensagens. Se considerarmos que estas mensagens estão indexadas por $q^k$ $k$-tuplas $\mathbf{u} = (u_1, u_2, \ldots, u_k) \in \mathbf{V}_k(\boldsymbol{F}_q)$ e que as linhas de $G$ são linearmente independentes, então uma regra bastante simples de codificação que mapeia mensagens $\mathbf{u}$ em palavras código $\mathbf{x}$ é

$$\mathbf{u} \rightarrow \mathbf{u}G$$

onde $\mathbf{u}G$ corresponde à multiplicação do vetor $\mathbf{u}_{1 \times k}$ pela matriz $G_{k \times n}$.

O mapeamento acima pode ser ainda mais simplificado utilizando-se o fato que se $G$ é uma matriz geradora de $\mathcal{C}$, então toda matriz linha equivalente a $G$ também é. Sabe-se que toda matriz é linha equivalente a uma matriz escalonada linha reduzida (RRE), e portanto todo código linear tem uma matriz geradora RRE, onde uma matriz RRE sobre um corpo $\boldsymbol{F}$ tem as seguintes propriedades:

(a) o elemento mais à esquerda diferente de 0 em cada linha é 1,
(b) toda coluna contendo o elemento mais esquerda igual a 1 tem todos os outros

valores igual a 0,

(c) se o elemento mais à esquerda diferente de zero na linha $i$ ocorre na coluna $t_i$, então $t_1 < t_2 < \ldots < t_r$.

Existe uma matriz ainda mais importante associada a cada código linear, denominada matriz de verificação de paridade.

Se $\mathcal{C}$ é um código $(n,k)$-linear em $\boldsymbol{F}_q$, então uma verificação de paridade para $\mathcal{C}$ é uma equação da forma

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = 0,$$

que é satisfeita para todo $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{C}$. O conjunto de todos os vetores $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ para o qual a equação é satisfeita para todo $\mathbf{x} \in \mathcal{C}$ é ele próprio um subconjunto de $\mathbf{V}n(\boldsymbol{F}_q)$ denotado por $\mathcal{C}^{\perp}$ e chamado de código dual de $\mathcal{C}$. Um resultado da álgebra linear afirma que $\mathcal{C}^{\perp}$ tem dimensão $n - \dim(\mathcal{C}^{\perp})$, ou seja, $\mathcal{C}^{\perp}$ é um código $(n,n-k)$-linear sobre $\boldsymbol{F}_q$. Uma matriz de verificação de paridade para $\mathcal{C}$ é definida como sendo uma matriz geradora para $\mathcal{C}^{\perp}$. Portanto:

seja $\mathcal{C}$ um código $(n,k)$-linear sobre $\boldsymbol{F}_q$. Uma matriz $H$ com a propriedade $H\mathbf{x}^{\mathrm{T}} = 0$ se, e somente se, $\mathbf{x} \in \mathcal{C}$ é chamada de matriz de verificação de paridade de $\mathcal{C}$.

Para falar da decodificação de códigos lineares seja $\boldsymbol{A}_Y$ o alfabeto de saída. Consideremos que $\boldsymbol{A}_Y = \boldsymbol{F}_q$, ou seja, os alfabetos de entrada e saída são idênticos. Portanto, se $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{V}_n(\boldsymbol{F}_q)$ é transmitido, o vetor recebido $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ também estará em $\mathbf{V}_n(\boldsymbol{F}_q)$ e a diferença $\mathbf{z} = \mathbf{y} - \mathbf{x}$ é chamada de padrão de erro. Se $z_i \neq 0$ então um erro ocorreu na i-ésima coordenada.

A matriz de paridade descrita acima é uma ferramenta útil para a decodificação. Se $\mathbf{x}$ é transmitido, $\mathbf{x}$ é uma palavra código e portanto $H\mathbf{x}^{\mathrm{T}} = 0$. Se houve algum erro, ou seja, se $\mathbf{z} \neq 0$, então muito provavelmente $H\mathbf{y}^{\mathrm{T}} \neq 0$. O vetor $\mathbf{s} = H\mathbf{y}^{\mathrm{T}}$ é chamado de síndrome e uma questão importante acerca da síndrome é que ela depende apenas do padrão de erro e não da palavra código transmitida, pois

$$\mathbf{s} = H\mathbf{y}^{\mathrm{T}} = H(\mathbf{x} + \mathbf{z})^{\mathrm{T}} = H\mathbf{x}^{\mathrm{T}} + H\mathbf{z}^{\mathrm{T}} = 0 + H\mathbf{z}^{\mathrm{T}} = H\mathbf{z}^{\mathrm{T}}$$

O receptor, porém, não está interessado em **z**; ele quer saber **x**. Mas como ele conhece **y** e **x** = **y** – **z**, ele pode considerar o problema de encontrar **z**.

A síndrome apresenta alguma informação sobre **z**, mas não o suficiente. Isto porque para um dado $\mathbf{s} \in \mathbf{V}_{n-k}(\boldsymbol{F}_q)$, o conjunto de soluções para $H\mathbf{z}^{\mathrm{T}} = \mathbf{s}$ corresponde a uma classe lateral do código $\mathcal{C}$, que é um subconjunto de $\mathbf{V}_n(\boldsymbol{F}_q)$ da forma

$$\mathcal{C} + \mathrm{z}_0 = \{\mathbf{x} + \mathrm{z}_0 \colon \mathbf{x} \in \mathcal{C}\}.$$

Existem $q^{n-k}$ classes laterais de $\mathcal{C}$, correspondendo às $q^{n-k}$ síndromes possíveis **s**, cada classe lateral contendo exatamente $q^k$ elementos. Portanto, uma vez que o receptor computa **s**, ele reduz sua busca por **z** de $q^n$ para $q^k$ possibilidades, que são os elementos da classe lateral correspondendo a **s**.

Para se fazer distinção dentre estes $q^k$ candidatos para **z**, no entanto, é necessário o conhecimento de informação adicional acerca do canal. Se considerarmos um canal q-ário simétrico (qSC), por exemplo, no qual se **X** e **Y** são vetores aleatórios modelando, respectivamente, a entrada e a saída do canal, então **Y** = **X** + **Z**, onde $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)$ é um vetor aleatório cujos componentes são variáveis aleatórias independentes e identicamente distribuídas com distribuição normal

$$\mathrm{P}(Z = 0) = 1 - (q - 1)\varepsilon,$$
$$\mathrm{P}(Z = \mathbf{z}) = \varepsilon \text{ se } \mathbf{z} \neq 0.$$

Para este canal é fácil se distinguir padrões de erro, pois se $\mathbf{z} \in \mathbf{V}_n(\boldsymbol{F}_q)$, então

$$\mathrm{P}(Z = \mathbf{z}) = [1 - (q - 1)\varepsilon]^{n - wH(z)}\varepsilon^{wH(z)},$$

onde *wH(z)* é o *peso de Hamming* de **z**, definido como o número de componentes diferentes de zero de **z** ou, de maneira alternativa, *wH(z)* é o número de erros ocorrendo em **z**. Se $\varepsilon \ll 1/q$, o lado direito da última equação é uma função decrescente de *wH(z)* e o **z** mais provável é aquele com menor peso.

O algoritmo abaixo ilustra o funcionamento do decodificador de síndrome, onde o passo 2 deste algoritmo representa um esforço considerável.

**Algoritmo 2.1** O algoritmo de decodificação de síndrome para um qSC. Fonte McEliece (2004, p. 145).

*1. Calcule a síndrome* $\mathbf{s} = H\mathbf{y}^{\mathrm{T}}$.

*2. Encontre um vetor de distância mínima na classe lateral correspondente de* $\mathbf{s}$. *Chame-o de* $\mathbf{z}_0$.

*3. A palavra código de saída é dada por* $\bar{\mathbf{x}} = \mathbf{y} - \mathbf{z}_0$.

### 2.3.2 Códigos LDPC

Este tópico traz definições e conceitos encontrados em Moon (2005).

Um $(n,k)$-código de bloco $\mathcal{C}$ sobre um alfabeto de $q$ símbolos corresponde a um conjunto de $q^k$ vetores de dimensão $n$ chamados de palavras código ou vetores código. Existe um codificador associado ao código que mapeia a mensagem, uma $k$-tupla $\mathbf{m} \in \mathcal{A}^k$ a sua palavra código associada.

Um código de bloco $\mathcal{C}$ sobre um corpo $\boldsymbol{F}^q$ de $q$ símbolos de comprimento $n$ e com $q^k$ palavras código é dito um $(n,k)$-código q-ário linear se e somente se suas $q^k$ palavras código formam um subespaço vetorial de dimensão $k$ do espaço vetorial de todas as $n$-uplas $\boldsymbol{F}^n{}_q$. O valor $n$ é dito o comprimento e $k$ é a dimensão do código. Para um código linear a soma de duas palavras código é também uma palavra código, assim como a combinação linear de palavras código é uma palavra código.

Códigos de baixa densidade de verificação de paridade LDPC (*low density parity check*) são uma classe de códigos de bloco lineares. A denominação vem da característica de sua matriz de verificação de paridade, que contém apenas uns poucos elementos 1 em comparação com os muitos que são 0. Sua maior vantagem é que eles possibilitam um desempenho próximo à capacidade de muitos tipos de canais e algoritmos com complexidade linear de tempo para decodificação. Além disso, são bastante adequados para implementações que fazem forte uso do paralelismo.

Eles foram originalmente propostos por Gallager em 1962. Curiosamente, apesar de pertencerem à categoria dos melhores códigos existentes, foram ignorados durante bastante tempo e permaneceram no ostracismo por cerca de 30 anos.

O peso de um vetor binário é definido como o número de seus elementos diferentes de

zero. Desta definição infere-se o significado de peso de coluna de uma matriz, assim como o de peso de linha. Uma matriz geradora LDPC é regular se os pesos de coluna forem iguais, assim como os pesos de linha. Para gerar um código LDPC regular, um peso de coluna $w_c$ é definido, normalmente um inteiro pequeno tal como $w_c = 3$, e valores para $N$ (o tamanho do bloco) e $M$ (a redundância). Então uma matriz $M$ x $N$ é gerada com peso $w_c$ em cada coluna e $w_r$ em cada linha. Para atingir um peso de linha uniforme $w_r$ devemos ter que $w_cN = w_rM$. Isto significa que cada bit participa em $w_c$ verificações e cada verificação envolve $w_r$ bits. Tal código regular é dito um ($w_c$,$w_r$,$N$)-código.

Existe um grafo associado a cada matriz de verificação de paridade, chamado de *grafo de Tanner*, o qual contém dois conjuntos de nós. O primeiro conjunto consiste em $N$ nós que representam os $N$ bits da palavra código e os nós deste conjunto são chamados de nós de bits. O segundo conjunto é composto de $M$ nós, chamados de nós de verificação. O grafo possui uma aresta entre o n-ésimo nó de bit e o m-ésimo nó de verificação se e somente se o n-ésimo bit está envolvido na m-ésima verificação, ou seja, se $A_{mn} = 1$, sendo $A$ a matriz de verificação de paridade. Portanto, o grafo de Tanner corresponde a uma representação gráfica da matriz de verificação de paridade e será utilizado para dar uma ideia de um algoritmo decodificador. A Figura 2.2 ilustra um grafo de Tanner para a matriz $A$ dada como exemplo.

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

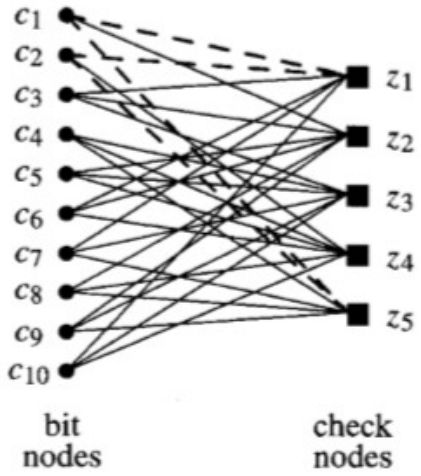


Figura 2.2–Grafo de Tanner para a matriz A. Fonte: Moon (2006, p. 638)

Vamos considerar o seguinte algoritmo iterativo de decodificação.

**Decodificação abrupta (*hard decoding*)**: Para cada bit $c_n$ compute as verificações de paridade para aquelas verificações que são influenciadas por $c_n$. Se o número de verificações não nulas exceder algum limite (por exemplo, o número de verificações de paridade é diferente de zero), então o respectivo bit é considerado errado, este bit errado é invertido e a correção prossegue.

Este esquema simples é capaz de corrigir mais de um erro. Suponha que $c_n$ esteja errado e outros bits influenciando sua verificação de paridade também estejam errados. Disponha o grafo de Tanner correspondente de tal maneira que $c_n$ esteja na raiz, desconsiderando a possibilidade de existência de ciclos no grafo. Na Figura 2.3, suponha que os bits realçados estejam errados. Os bits que se conectam a verificações de paridade conectadas à raiz estão na camada 1. Os bits conectados a verificações de paridade da camada 1 estão na camada 2 e assim sucessivamente. Portanto, a decodificação deve proceder das folhas (o topo da figura). No instante em que a correção alcançar $c_n$, outros bits errados já devem ter sido corrigidos. Logo, bits e verificações de paridade que não estão conectados diretamente a $c_n$ mesmo assim são capazes de influenciar $c_n$.

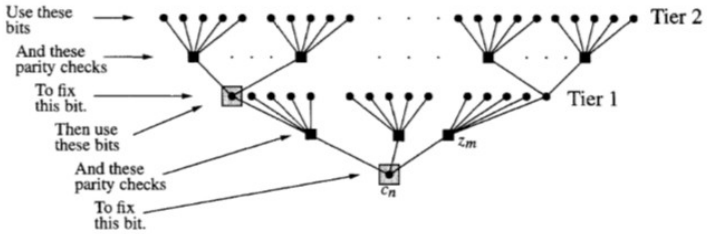


Figura 2.3–Árvore de verificação de paridade associada a um grafo de Tanner. Fonte: Moon (2005, p. 644).

## 2.4 CRIPTOGRAFIA BASEADA EM CÓDIGOS

### 2.4.1 Ideia geral

Em termos gerais, na criptografia baseada em códigos quando Alice quer mandar uma mensagem para Bob ela faz uso da chave pública deste para encriptar a mensagem. A esta mensagem cifrada é adicionado um erro aleatório. E somente Bob, com sua chave privada, é capaz de decriptar a mensagem, remover o erro introduzido e recuperar o conteúdo original enviado por Alice. Isto corresponde ao que está mostrado na figura abaixo.

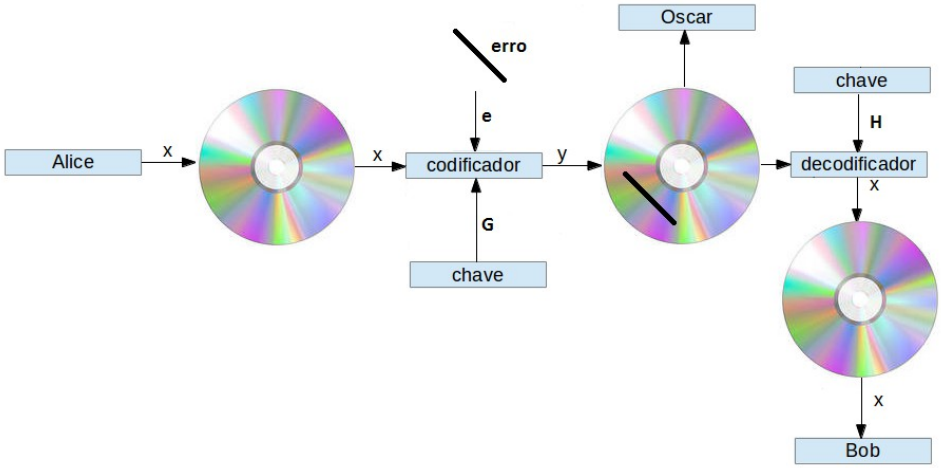


Figura 2.4–Criptografia baseada em códigos.

### 2.4.2 Criptossistemas e códigos

Para dirimir possíveis dúvidas, já que o tema envolve encriptação/decriptação de mensagens através de criptossistemas baseados em códigos, os termos decodificação e

decriptação estão empregados neste documento com o mesmo significado, que é o de recuperar o texto claro da mensagem encriptada, o que é o caso quando emprega-se um algoritmo decodificador de códigos LDPC, por exemplo, para decriptar uma mensagem cifrada.

Overbeck e Sendrier (2009, p. 94) definem os criptossistemas baseados em códigos como "*os criptossistemas nos quais a primitiva algorítmica (a função de uma única via subjacente) utiliza um código corretor de erros $C$. Esta primitiva pode consistir na adição de erro a uma palavra de $C$ ou em computar a síndrome relativa a uma matriz de verificação de paridade de $C$*".

O primeiro criptossistema de chave pública baseado em códigos foi proposto por Robert J. McEliece em 1978. Nele a chave privada corresponde a um código de Goppa binário aleatório e irredutível e a chave pública é uma matriz geradora aleatória de uma versão aleatoriamente permutada deste código. A mensagem cifrada é uma palavra código a qual alguns erros são adicionados e apenas o proprietário da chave privada, que corresponde ao código de Goppa, é capaz de remover esses erros e recuperar a mensagem original. Após quase quatro décadas de existência alguns parâmetros tiveram que ser ajustados, mas nenhum ataque conhecido até o presente momento representa alguma ameaça ao criptossistema de McEliece, mesmo levando-se em consideração a computação quântica.

Overbeck e Sendrier (2009, p.94) afirmam que "*como em qualquer criptossistema, em criptografia baseada em códigos há uma relação de compromisso entre eficiência e segurança"*. Segundo eles, não existe nenhuma aplicação prática conhecida de criptossistemas baseados em códigos, talvez por causa do tamanho grande requerido para as chaves públicas, que podem variar de 100 kilobytes a muitos megabytes, mas, com certeza, pela falta de publicidade destes criptossistemas num contexto onde não há demanda urgente por soluções alternativas. De qualquer maneira, apesar do tamanho de suas chaves, o criptossistema de McEliece é bastante rápido, pois tanto a operação de encriptação quanto a de decriptação possuem baixa complexidade. Além disso, com relação à segurança é bastante conhecida a dificuldade de decodificação de um código aleatório linear, que corresponde a um problema da teoria de codificação cujas soluções todas têm tempo exponencial. Ademais, os códigos Goppa apresentam como característica a indistinguibilidade, uma questão de segurança necessária para os criptossistemas ditos assimétricos, pois segundo Berlekamp (1978) é provado que caso um adversário não

consiga distinguir um código Goppa de um código aleatório, então ele é desafiado a decodificar um código linear genérico, um problema NP-completo.

Na teoria da complexidade computacional, NP denota o conjunto de problemas que são decidíveis em tempo polinomial por uma máquina de Turing não determinística.

Uma máquina de Turing determinística possui uma função de transição que, para um determinado estado e símbolo sob a cabeça de leitura, especifica o símbolo a ser escrito na fita, a direção na qual a cabeça deve se mover (esquerda ou direita) e o seu próximo estado. Uma máquina de Turing não determinística difere no fato que várias ações diferentes podem ser aplicadas para uma mesma combinação de estado e símbolo.

Segundo Sipser (2006), a classe de complexidade NP-completo é o subconjunto de problemas de decisão em NP de tal maneira que todo problema em NP pode sofrer uma redução em tempo polinomial a um dos problemas NP-completo.

Uma redução em tempo polinomial é uma redução que é computável por uma máquina de Turing determinística em tempo polinomial.

### 2.4.3 O criptossistema de chave pública de McEliece

Em 1978 McEliece (1978, p. 114) escreveu na introdução do seu artigo de apresentação de sua proposta de criptossistema de chave pública: "*Usando o fato que existe um algoritmo rápido de decodificação para códigos Goppa, o que não acontece para códigos lineares genéricos, construímos um criptossistema de chave pública que parece ser seguro e ao mesmo tempo permite taxas extremamente rápidas de dados, o que é ideal para redes de comunicação multiusuário, tais como as vislumbradas pela NASA para a distribuição de dados espaciais adquiridos*".

Além de ser mais rápido que o RSA nas tarefas de encriptação e decriptação, até hoje a proposta original do criptossistema de chave pública de McEliece continua imune a ataques, tendo sido necessário apenas um ajuste nos seus parâmetros de segurança. O Algoritmo 2.2 representa a ideia do criptossistema.

---

**Algoritmo 2.2** O criptossistema de chave pública McEliece. Fonte Overbeck e Sendrier (2009, p. 97).

---

**Parâmetros:** $n, t \in \mathbb{N}, t \ll n$

**Geração das chaves:** dados *n, t* crie as seguintes matrizes:

$G_{k \times n}$: *matriz geradora de um código* $\mathcal{G}$ *sobre* $\mathbf{F}$ *com dimensão k e distância mínima* $d \geqslant 2t + 1$ *(na proposta original um código de Goppa binário irredutível)*

$S_{k \times k}$: *matriz aleatória binária não singular*

$P_{n \times n}$: *matriz aleatória de permutação*

*Então, compute a matriz* $G^{PUB}{}_{k \times n} = S.G.P$

***Chave pública:*** $(G^{PUB}, t)$

***Chave privada:*** $(S, D_{\mathcal{G}}, P)$, *onde* $D_{\mathcal{G}}$ *é um algoritmo de decodificação eficiente para* $\mathcal{G}$

***Encriptação*** $(E(G^{PUB}, t))$***:*** *para encriptar uma mensagem* $\mathbf{m} \in \mathbf{F}^k$ *escolha aleatoriamente um vetor* $\mathbf{z} \in \mathbf{F}^n$ *de peso t e compute a mensagem cifrada* **c** *como:*

$$c = mG^{PUB} \square z.$$

***Decriptação*** $(D(S, D_{\mathcal{G}}, P))$***:*** *para decriptar uma mensagem cifrada* **c** *inicialmente calcule*

$$cP^{-1} = (mS)G \square zP^{-1}$$

*e aplique o algoritmo de decodificação* $D_{\mathcal{G}}$*. Como msG tem uma distância de Hamming de no máximo t para* $cP^{-1}$ *obtemos a palavra código*

$$mSG = D_{\mathcal{G}}(cP^{-1}).$$

*Podemos portanto calcular a mensagem* $m = (mSGG^{-1})S^{-1}$.

### 2.4.4 QC-MDPC McEliece, uma variante do criptossistema de chave pública McEliece

Misoczki, Tillich, Sendrier e Barreto (2013, p. 1) afirmam que todos os criptossistemas baseados na dificuldade de fatoração ou no cálculo do logaritmo discreto podem ser atacados em tempo polinomial com um computador quântico executando os algoritmos propostos por Shor (1997). Sobre este assunto Bernstein, Buchman e Dahmen (2009) apresentam extensas informações. Ainda segundo Misoczki, Tillich, Sendrier e Barreto (2013, p. 1), "*acredita-se que a criptografia baseada em códigos é resistente à computação quântica e, portanto, ela é considerada como uma alternativa para aplicações futuras*". Independente de sua natureza pós-quântica, os criptossistemas baseados em códigos possuem outra vantagem disponível para aplicações atuais: sua eficiência algorítmica muitas ordens de complexidade inferior aquela dos criptossistemas

tradicionais.

Por eficiência algorítmica entendem-se as propriedades do algoritmo relacionadas à quantidade de recursos por ele utilizados. Portanto, um algoritmo pode ser julgado eficiente com respeito ao tempo ou ao espaço de memória ocupado. Um algoritmo é considerado eficiente se consome quantidade de recursos inferior a um valor tido como aceitável. E segundo Cormen, Leiserson, Rivest e Stein (2009, p.12), diferentes algoritmos escritos para resolver um mesmo problema normalmente diferem enormemente em sua eficiência. Segundo estes autores, a análise de um algoritmo significa a predição dos recursos que serão por ele requisitados. Ocasionalmente, memória ou largura de banda deverão ser levados em consideração, porém na maioria das situações o que se pretenderá medir é o tempo de execução. Normalmente ao analisar vários algoritmos candidatos para solucionar um problema, pode-se identificar aquele mais eficiente. Tal análise poderá indicar mais de um candidato viável, assim como vários algoritmos poderão ser descartados por terem desempenho considerado insatisfatório.

Apesar de apresentarem algoritmos eficientes, os criptossistemas baseados em códigos tem como desvantagem o tamanho grande de suas chaves. Misoczki, Tillich, Sendrier e Barreto (2013) citam o trabalho de Gaborit (2005) para afirmar que neste caso o tamanho das chaves pode ser significantemente reduzido com a escolha de códigos com um grande grupo de automorfismo, como os códigos quase-cíclicos (QC). Porém, Faugère, Otmani, Perret e Tillich (2010) mostram que um ataque algébrico estruturado pode quebrar a maioria dos criptossistemas baseados nestes códigos, tendo em vista que eles possuem uma estrutura algébrica que permite a um adversário construir um sistema algébrico de equações que pode ser solucionado, pois a natureza quase-cíclica do código escolhido colabora com uma redução drástica do número de incógnitas do sistema a ser resolvido. Este tipo de ataque é exponencial e pode ser facilmente evitado através da escolha de parâmetros mais conservadores. E códigos sem qualquer estrutura algébrica frustram completamente a tentativa.

Os códigos com baixa densidade de verificação de paridade (LDPC) descritos por Gallager (1962) são bons candidatos para resguardar o criptossistema do ataque descrito no parágrafo anterior. São códigos sem nenhuma estrutura algébrica que satisfazem uma propriedade combinatorial bastante simples: eles admitem uma matriz de verificação de paridade esparsa e esta esparsidade é usada para a decodificação. Vários trabalhos sugerem

a adoção destes códigos no criptossistema de McElice, como por exemplo Monico, Rosenthal e Shokrollahi (2000) e Baldi e Chiaraluce (2007).

O trabalho de Monico, Rosenthal e Shokrollahi (2000) analisa a utilização de um código LDPC no criptossistema de McEliece: a chave privada corresponde a uma matriz esparsa de verificação de paridade $H$ de um código $\mathcal{C}$ com peso constante de linha $w$; a chave pública é uma matriz densa geradora $G' = S.G.P$ de um código $\mathcal{C}'$, onde $S$ é uma matriz embaralhadora, $G$ é uma matriz geradora de $\mathcal{C}$ e $P$ é uma matriz de permutação. Porém, é possível encontrar palavras-código de baixo peso no dual de $\mathcal{C}'$, fato que é suficiente para a reconstrução da matriz de verificação de paridade esparsa que decodifica de maneira eficiente.

Baldi e Chiaraluce (2007) sugerem uma correção para este problema, que consiste na substituição da matriz de permutação $P$ por uma matriz $Q$ inversível com peso pequeno constante de linha $m$ e escolhendo-se $S$ esparsa. Definindo-se apropriadamente os valores de $w$ e $m$ torna a tarefa de encontrar palavras código de peso $wm$ em $\mathcal{C}'$ completamente inviável. No entanto, Otmani, Tillich e Dallot (2010) mostraram que escolhas inapropriadas para as estruturas destas matrizes permitem que o criptossistema seja quebrado. Este ataque pode ser prevenido, segundo Baldi, Bodrato e Chiaraluce (2008), através de uma matriz $S$ densa e uma construção mais genérica para $Q$.

Misoczki, Tillich, Sendrier e Barreto (2103, p. 2) observam que para a utilização de códigos LDPC no criptossistema de McEliece não se faz necessária a substituição da matriz de permutação por uma matriz com peso de linha constante e pequeno. Basta aumentar suficientemente o comprimento e o peso da linha da matriz esparsa secreta de verificação de paridade para se evitar todos os ataques de mensagens conhecidos que façam uso de algoritmos padrão de decodificação, bem como de recuperação de chave que busquem palavras código de baixo peso no dual do código público. Eles propõem, portanto, a substituição dos códigos LPDC por códigos de média densidade de verificação de paridade MDPC (*medium density parity check*), os quais admitem uma matriz de verificação de paridade que é moderadamente esparsa. Para exemplificar, para uma taxa de código de ½ e 80 bits de segurança, eles definem a matriz secreta de verificação de paridade como sendo 4800 x 9600 com peso de linha igual a 90, ao passo que códigos LDPC usados na prática para a correção de erros tem peso de linha normalmente menor

que 10. Neste caso do exemplo é possível corrigir apenas 84 erros, enquanto qualquer código LDPC de mesmo tamanho é capaz de corrigir de 700 a 800 erros. Apesar da capacidade de correção de erros dos códigos MDPC ser significantemente menor quando comparada com a dos códigos LDPC, o número de erros passível de ser introduzido na mensagem ainda é grande o suficiente para evitar a decriptação desta por meio de algoritmos padrão utilizados para correção de erros em códigos lineares genéricos.

Além disso, acrescentam ao código proposto uma estrutura quase-cíclica (QC), o que confere às chaves tamanhos compactos, no exemplo, para o nível de segurança de 80 bits a chave tem apenas 4800 bits, e ressaltam que "*o estado da arte indica que uma estrutura quase-cíclica, por si só, não implica um benefício significativo para o adversário. Todos os ataques anteriores ao criptossistema de McEliece são baseados na combinação de uma estrutura quase-cíclica com alguma informação algébrica sobre o código*".

#### 2.4.4.1 Definições

Para o correto entendimento do criptossistema de chave pública baseada em códigos QC-MDPC McEliece proposto por Misoczki, Tillich, Sendrier e Barreto (2013) seguem várias definições que podem ser encontradas no correspondente documento.

Um **código binário (*n*,*r*)-linear** $\mathcal{C}$ de comprimento $n$, dimensão $n - r$ e codimensão $r$ corresponde a um subespaço vetorial $(n - r)$ dimensional de $\boldsymbol{F}_n^2$. Ele é gerado pelas linhas de uma matriz $G \in \boldsymbol{F}_2^{(n-r) \times n}$ chamada matriz geradora de $\mathcal{C}$. E também é o núcleo de uma matriz $H \in \boldsymbol{F}_2^{r \times n}$ conhecida como matriz de verificação de paridade de $\mathcal{C}$. Em álgebra linear, o núcleo de uma transformação linear $L:V \rightarrow W$ entre dois espaços vetoriais $V$ e $W$ é o conjunto de todos os elementos $v \in V$ tal que $L(v) = 0$, sendo $0$ o vetor nulo em $W$.

Um **código (*n*,*r*)-linear** é considerado **quase-cíclico** (QC) se existir um inteiro $n_0$ tal que qualquer deslocamento circular de uma palavra-chave por $n_0$ posições dá como resultado uma outra palavra-chave.

Um **código (*n*,*r*,*w*)-MDPC** é um código linear de comprimento $n$, codimensão $r$ e que admite uma matriz de verificação de paridade com peso de Hamming constante de suas linhas igual a $w$.

Para a construção de um **código (*n*,*r*,*w*)-QC-MDPC** escolhe-se uma palavra aleatória de

comprimento $n = n_0 p$ e peso *w* que corresponde à primeira linha da matriz *H*. As outras *r* – 1 linhas são obtidas por intermédio de *r* – 1 deslocamentos quase-cíclicos, de tal maneira que cada bloco $H_i$ tem peso de linha $w_i$ tal que $w = \sum_{i=0}^{n_0-1} w_i$ . A matriz geradora *G* pode ser calculada a partir dos blocos $H_i$ como $G = [I \mid Q]$ , onde *I* é a matriz identidade e $Q = \begin{bmatrix} (H^{-1}_{n_0-1} . H_0)^T \\ (H^{-1}_{n_0-1} . H_1)^T \\ \vdots \\ (H^{-1}_{n_0-1} . H_{n_0-2})^T \end{bmatrix}$ .

### 2.4.4.2 Encriptação e decriptação

As operações de encriptação e decriptação McEliece baseadas num código (*n*,*r*,*w*)-QC-MPDC com capacidade para correção de *t* erros são definidas como:

1) ***Geração das chaves***: crie um vetor aleatório binário *h* de peso *w*. Ele corresponde à primeira linha da matriz *H* e as demais *r* – 1 linhas são obtidas através de deslocamentos quase-cíclicos de *h*. Obtenha a matriz *G* correspondente a *H* na forma reduzida escalonada. A chave pública é *G* e a chave privada é *H*.

2) ***Encriptação***: para encriptar uma mensagem *m* crie um vetor aleatório binário *e* de peso menor ou igual a *t*. A mensagem cifrada *x* é dada por $x \leftarrow mG + e$ .

3) ***Decriptação***: para decriptar *x* em *m*, calcule $mG \leftarrow \Psi_H(mG + e)$ , onde $\Psi_H$ é um decodificador para o código QC-MDPC que possui conhecimento da matriz *H*. A mensagem *m* é recuperada a partir das primeiras (*n* – *r*) posições de *m*G.

### 2.4.4.3 Decodificação eficiente

A decriptação é a etapa que mais consome tempo na criptografia baseada em códigos. Portanto, a opção pelo algoritmo decodificador mais apropriado é crucial para a obtenção de um desempenho desejável na tarefa de decodificação. Para decodificar códigos LDPC existem duas categorias de algoritmos: a composta por algoritmos rápidos e simples, por exemplo o de inversão de bits (*bit-flipping)*, que tem menor capacidade de correção de

erros e a daqueles mais elaborados e com uma maior capacidade corretora, como por exemplo o algoritmo soma-produto. Misoczki, Tillich, Sendrier e Barreto (2013) sugerem a utilização dos algoritmos de inversão de bits (*bit-flipping*) para os códigos MDPC, tendo em vista que estes têm baixa complexidade, são iterativos e rápidos, e o ganho em termos da baixa complexidade de decodificação mais que supera a pequena melhora na capacidade de correção de erros provida pelo segundo grupo.

Em termos gerais, todos os algoritmos que usam a técnica de inversão de bits seguem a mesma ideia. Primeiramente, a cada iteração eles computam a síndrome da mensagem a ser decodificada. A seguir são computados os números de equações de verificação de paridade não satisfeitas associadas a cada bit da mensagem. Cada bit associado a mais que $b$ equações não satisfeitas é invertido. Este processamento é repetido até que a síndrome seja completamente zerada ou até que seja alcançado um número limite de iterações, quando considera-se que o processo de decodificação falhou. A grande diferença entre os diversos decodificadores é na maneira como o limite $b$ é calculado, o que pode variar desde considerar o maior número de equações não satisfeitas até o pré-cálculo de limites baseados em parâmetros do código.

Em seu artigo, Misoczki, Tillich, Sendrier e Barreto (2013) apresentam uma variante do algoritmo de inversão de bits de Gallager (1962) que se aplica aos códigos MDPC. A diferença é na definição do limiar $b$. Três possíveis alternativas são:

I. Gallager (1962) pré-computa o valor de $b$ a cada iteração.

II. Huffman e Pless (2003) consideram b como o valor máximo $\text{Max}_{\text{upc}}$ de equações de verificação de paridade não satisfeitas.

III. Misoczki, Tillich, Sendrier e Barreto (2013) calculam $b$ como $b = \text{Max}_{\text{upc}} - \delta$, para um $\delta$ inteiro, pequeno e positivo.

A alternativa II proporciona uma melhor capacidade de correção de erros que a alternativa I, com o respectivo aumento do número de iterações do algoritmo. A alternativa III combina as vantagens de I e II. Ela reduz a quantidade de iterações obtida por II na medida que muito mais bits são invertidos em cada iteração. Se o algoritmo falha na tentativa de decodificar, $\delta$ é decrementado de 1 e o processo é reiniciado. Obviamente quando $\delta = 0$ estamos na alternativa II, assegurando pelo menos sua capacidade de correção de erros. O

valor ótimo para $\delta$ é determinado empiricamente. Para os parâmetros sugeridos na seção 2.4.4.4, Tabela 2.2, $\delta \approx 5$ é um valor interessante que reduz o número de iterações de $\sim$ 65 para $\lesssim 10$.

O Algoritmo 2.3 representa a proposta de decodificação de Misoczki, Tillich, Sendrier e Barreto (2013, p. 7), onde *upc* é a abreviação de verificações de paridade não satisfeitas (*unsatisfied parity check*) e **0** significa um vetor zero de dimensão *r*.

---

**Algoritmo 2.3** Variante do algoritmo de *bit-flipping* para decodificação de códigos MDPC. Fonte: Misoczki, Tillich, Sendrier e Barreto (2013, p. 7).

---

**Entrada**: $y = m\mathrm{G} + e$, com peso $w$ e dimensão $n$

**Saída**: $c$ tal que $\mathrm{H}c^{\mathrm{T}} = 0$, ou FALHA.

**while** $\delta \in \mathbb{N} \geq 0$

    $s \leftarrow \mathrm{H}c^{\mathrm{T}}$

    **for** i = 1 **to** dim($s$) **do**

        **if** $s$[i] = 1 **then**

            **for** j = 1 **to** $w$ **do**

                contador[j] ← #upc para cada bit  //número de equações de //verificação de paridade não //satisfeitas

            **end for**

        **end if**

    **end for**

    Maxupc ← o maior #upc

    **for** i = 1 **to** $n$ **do**

        **if** contador[i] ≥ (Maxupc − δ) **then**

            **Flip** $c_{\mathrm{i}}$

        **end if**

    **end for**

    **if** $\mathrm{H}c^{\mathrm{T}} = \mathbf{0}$ **then**

        **return** $c$

    **end if**

    δ ← δ – 1  //No caso de falha na decodificação.

**end while**

**return** FALHA

---

Von Maurich e Güneysu (2014, p. 2) mencionam o trabalho de Heyse, von Maurich e

Güneysu (2013) no qual vários algoritmos de decodificadores foram analisados quanto a sua adequação e desempenho ao trabalhar com códigos QC-MDPC em dispositivos embarcados.

Heyse, von Maurich e Güneysu (2013, p. 5) descrevem versões de algoritmos de decodificadores que usam a técnica de inversão de bits (*bit-flipping*), as quais tiveram avaliadas a capacidade de correção de erros e contabilizados os números de iterações necessárias, em média, para que cada versão decodificasse uma palavra código. O ponto de partida para a investigação foram os decodificadores:

> *Decodificador* $\mathcal{A}$*:* proposto por Misoczki, Tillich, Sendrier e Barreto (2013), o qual computa a síndrome, verifica o número de equações de verificação de paridade (upc) não satisfeitas uma vez para determinar o valor $Max_{upc}$ e uma segunda vez para inverter todos os bits da mensagem cifrada que satisfazem $b \geq Max_{upc} - \delta$ equações. A seguir a síndrome é recomputada e comparada com zero.
>
> *Decodificador* $\mathcal{B}$*:* apresentado por Gallager (1962), que computa a síndrome, a seguir verifica o número de equações de verificação de paridade (upc) não satisfeitas uma vez por iteração $i$ e a seguir inverte o bit da mensagem cifrada se *#upc* for maior que um limite pré-calculado $b_i$. A seguir a síndrome é recalculada e comparada com zero.

A partir da observação que os decodificadores $\mathcal{A}$ e $\mathcal{B}$ recomputam a síndrome após cada iteração, o que é caro em termos computacionais, Heyse, von Maurich e Güneysu (2013, p. 5) propõem uma otimização baseada no seguinte fato: se a quantidade de equações de verificação de paridade não satisfeitas excede um limite $b$, o bit correspondente na mensagem cifrada é invertido e a síndrome modificada. Portanto, afirmam Heyse, von Maurich e Güneysu (2013, p. 5) *"gostaríamos de ressaltar que a síndrome não muda aleatoriamente; a nova síndrome é igual à síndrome antiga acumulada com a linha* $h_j$ *da matriz de verificação de paridade que corresponde ao bit invertido j da mensagem cifrada. Logo, a necessidade de recálculo da síndrome é eliminada atualizando-se a mesma de acordo com os bits da mensagem cifrada correspondentes que são invertidos".* Com esta ideia, são propostos e avaliados os novos decodificadores:

> *Decodificador* $\mathcal{C}_i$: computa a síndrome, verifica o número de equações de

verificação de paridade (upc) não satisfeitas uma vez para determinar o valor $Max_{upc}$ e uma segunda vez para inverter todos os bits da mensagem cifrada que satisfazem $b \geq Max_{upc} - \delta$ equações. Se um bit $j$ da mensagem cifrada é invertido, a linha correspondente $h_j$ da matriz de verificação de paridade é adicionada à síndrome, resultando numa síndrome relativa a uma nova palavra código sem necessidade de recálculo.

*Decodificador* $\mathcal{C}_2$: computa a síndrome, verifica o número de equações de verificação de paridade (upc) não satisfeitas uma vez para determinar o valor $Max_{upc}$ e uma segunda vez para inverter todos os bits da mensagem cifrada que satisfazem $b \geq Max_{upc} - \delta$ equações. Se um bit $j$ da mensagem cifrada é invertido, a linha correspondente $h_j$ da matriz de verificação de paridade é adicionada a uma síndrome temporária. Ao final de cada iteração esta síndrome temporária é adicionada à síndrome resultando numa síndrome relativa a uma nova palavra-código sem necessidade de recálculo.

*Decodificador* $\mathcal{D}$: similar ao Decodificador $\mathcal{B}$ com o limite pré-computado $b_i$, porém utilizando a atualização direta da síndrome como feito no Decodificador $\mathcal{C}_2$.

*Decodificador* $\mathcal{E}$: similar ao Decodificador $\mathcal{C}_2$, porém ele compara a síndrome com zero após cada inversão de bit e encerra a iteração corrente de imediato caso a síndrome tenha sido zerada.

*Decodificador* $\mathcal{F}$: similar ao Decodificador $\mathcal{D}$ utilizando a mesma ideia de antecipação de encerramento de execução do Decodificador $\mathcal{E}$.

A quantidade média de iterações necessárias para a decodificação de uma palavra código e a correspondente taxa de falha de decodificação encontradas por Heyse, von Maurich e Güneysu (2013) estão mostradas na Tabela 2.1.

Tabela 2.1–Avaliação do desempenho e capacidade de correção de erro de diferentes decodificadores para um criptossistema QC-MDPC McEliece com parâmetros n0 = 2, n = 9600, r = 4800, w = 90. Fonte Heyse, von Maurich e Güneysu (2013, Apêndice).

| **Decodificador** | **Número de erros** | **Taxa de falha** | **Número médio de iterações** |
|---|---|---|---|
| *Decodificador* $\mathcal{A}$ | 84 | 0,00041 | 5,2964 |
| | 85 | 0,00089 | 5,3857 |

| | | | |
|---|---|---|---|
| | 86 | 0,00221 | 5,4975 |
| | 87 | 0,00434 | 5,6261 |
| | 88 | 0,00891 | 5,7679 |
| | 89 | 0,01802 | 5,9134 |
| | 90 | 0,03264 | 6,0677 |
| *Decodificador* $\mathcal{B}$ | 84 | 0,00051 | 3,1425 |
| | 85 | 0,00163 | 3,1460 |
| | 86 | 0,00631 | 3,1607 |
| | 87 | 0,01952 | 3,2022 |
| | 88 | 0,05195 | 3,4040 |
| | 89 | 0,11462 | 3,5009 |
| | 90 | 0,24080 | 3,8972 |
| *Decodificador* $C_1$ | 84 | 0,00044 | 5,2862 |
| | 85 | 0,00106 | 5,3924 |
| | 86 | 0,00172 | 5,4924 |
| | 87 | 0,00480 | 5,6260 |
| | 88 | 0,00928 | 5,7595 |
| | 89 | 0,01762 | 5,9078 |
| | 90 | 0,03315 | 6,0685 |
| *Decodificador* $C_2$ | 84 | 0,00018 | 3,3791 |
| | 85 | 0,00068 | 3,4180 |
| | 86 | 0,00148 | 3,4643 |
| | 87 | 0,00378 | 3,5279 |
| | 88 | 0,00750 | 3,5942 |
| | 89 | 0,01500 | 3,6542 |
| | 90 | 0,02877 | 3,7435 |
| *Decodificador* $\mathcal{D}$ | 84 | 0,00001 | 2,4002 |
| | 85 | 0,00003 | 2,4980 |
| | 86 | 0,00004 | 2,5979 |
| | 87 | 0,00031 | 2,6958 |
| | 88 | 0,00093 | 2,7875 |
| | 89 | 0,00234 | 2,8749 |
| | 90 | 0,00552 | 2,9670 |
| *Decodificador* $\mathcal{E}$ | 84 | 0,00019 | 3,3754 |
| | 85 | 0,00073 | 3,4218 |

| | | | |
|---|---|---|---|
| | 86 | 0,00153 | 3,4673 |
| | 87 | 0,00375 | 3,5314 |
| | 88 | 0,00728 | 3,5886 |
| | 89 | 0,01529 | 3,6563 |
| | 90 | 0,02840 | 3,7343 |
| *Decodificador* $\mathcal{F}$ | 84 | 0,00000 | 2,4047 |
| | 85 | 0,00002 | 2,5000 |
| | 86 | 0,00008 | 2,5983 |
| | 87 | 0,00039 | 2,6939 |
| | 88 | 0,00094 | 2,7912 |
| | 89 | 0,00209 | 2,8793 |
| | 90 | 0,00506 | 2,9630 |

Os dados mostram claramente a superioridade dos decodificadores $\mathcal{D}$ e $\mathcal{F}$ na capacidade de correção de erros, bem como na necessidade de um menor número de iterações para a decodificação. Os decodificadores $\mathcal{A}$ e $C_1$ são os menos eficientes com uma média de mais de 5 iterações de inversão de bits. Os decodificadores $\mathcal{D}$ e $\mathcal{F}$ propostos por Heyse, von Maurich e Güneysu (2013) economizam em média 2,9 iterações se comparados com o decodificador $\mathcal{A}$ e 0,7 iterações comparados com $\mathcal{B}$ o que está diretamente relacionado com o tempo necessário para a decodificação, que acontece quatro vezes mais rápido.

A vantagem em tempo de processamento do decodificador $\mathcal{F}$ sobre o decodificador $\mathcal{D}$ é devida ao cancelamento imediato da sua execução caso a síndrome torne-se zero. Outra observação interessante feita por Heyse, von Maurich e Güneysu (2013) e que vale para todos os decodificadores é que, se uma palavra código é decodificada, esta decodificação ocorre depois de um pequeno número de iterações. Eles perceberam que se uma palavra-chave não é decodificada dentro de 4 a 6 iterações, um maior número de iterações não ocasiona uma decodificação bem-sucedida. Portanto, uma precoce detecção de uma falha de decodificação é possível de ser realizada.

O decodificador $\mathcal{F}$ que superou todos os outros tanto em tempo de processamento bem como na taxa de falha de decodificação corresponde ao Algoritmo 2.4 abaixo.

Este decodificador pré-calcula seus limiares $b_i$ baseado nos parâmetros do código como proposto por Gallager (1962) e, ao contrário de outras soluções que recalculam a síndrome

após cada iteração, apenas a atualiza enquanto processa a decodificação da mensagem cifrada.

**Algoritmo 2.4** Variante eficiente do algoritmo de bit-flipping para decodificação de códigos (QC)-MDPC. Fonte: Von Maurich e Güneysu (2014, p.3).

**Entrada**: $y = m\mathrm{G} + e$, com peso $w$ e dimensão $n$

**Saída**: $c$ tal que $\mathrm{H}c^{\mathrm{T}} = 0$, ou FALHA.

```
while δ ∈ ℕ ≥ 0
    s ← Hc^T
    for i = 1 to dim(s) do
        if s[i] = 1 then
            for j = 1 to w do
                contador[j] ← #upc para cada bit      //número de equações de
                                                      //verificação de paridade não
                                                      //satisfeitas
            end for
        end if
    end for
    Maxupc ← o maior #upc
    for i = 1 to n do
        if contador[i] ≥ b_i then
            Flip c_i
            s← s XOR h_i
        end if
        if s = 0 then
            return c
        end if
    end for
    δ ← δ – 1                  //No caso de falha na decodificação.
end while
return FALHA
```

#### 2.4.4.4 Segurança do QC-MDPC McEliece

Seja uma instância do criptossistema de McEliece definido por um código $(n,r,w)$-QC-MDPC, com capacidade de correção de $t$ erros. Seja $C$ o código QC-MDPC definido pela

chave pública (a matriz geradora de $C$). Misoczki, Tillich, Sendrier e Barreto (2013, p. 13) afirmam que os melhores ataques para cada cenário são:

- ataque de distinção de chave: exibir uma palavra código de $C^{\perp}$ com peso *w*.
- ataque de recuperação de chave: exibir *r* palavras código de $C^{\perp}$ com peso *w*.
- ataque de decodificação: decodificar *t* erros num código (*n*, *n-r*)-linear.

Ataques de chave têm por objetivo a recuperação do decodificador, que é secreto, ou apenas possibilitar a distinção da chave pública de uma matriz aleatória.

Ataques de mensagem tentam a decodificação de uma mensagem particular considerada como uma palavra código com adição de ruído.

Para realizar os ataques citados é necessário resolver ou o problema de ter que encontrar a palavra código ou o problema de decodificação computacional da síndrome. Para ambos os problemas a melhor técnica conhecida é a denominada decodificação por conjunto de informação (ISD) descrita por Prange (1962).

Seja $WF_{isd}(n, r, t)$ a representação do custo de decodificação de *t* erros, ou seja, o custo de se encontrar uma palavra código de peso *t*, num código binário (*n*, *t*)-linear onde existe uma única solução para o problema. Os algoritmos ISD assumem um padrão para buscar o vetor de erro procurado e procedem à análise de um determinado conjunto de candidatos até que uma solução seja encontrada. Este conjunto de candidatos é usualmente armazenado em lista de um determinado tamanho *L* e cada candidato apresenta probabilidade *P* de produzir a solução desejada. Quando os parâmetros do algoritmo são ótimos, o esforço necessário $WF_{isd}(n, r, t)$ para a quebra do criptossistema corresponde à razão *L*/*P* corrigida por um pequeno fator.

Sendrier (2011), por outro lado, analisa a situação quando o problema de decodificação tem múltiplas soluções e o adversário se satisfaz com uma única delas, o que é mencionado como "*decodificando um de muitos*" (DOOM). Quando um problema tem $N_S$ soluções, a probabilidade de sucesso *P* aumenta por um fator $N_S$ (desde que $N_S \mathrm{P} \ll 1$) e quando $N_i$ instâncias são tratadas simultaneamente o tamanho da lista *L* aumenta no máximo por um fator $\sqrt{N_i}$. Portanto, a técnica DOOM representa um ganho de $Ns/\sqrt{N_i}$.

Misoczki, Tillich, Sendrier e Barreto (2013, p.14) avaliam o emprego das técnicas ISD e

DOOM com respeito a cada tipo de ataque ao criptossistema QC-MDPC McEliece proposto por eles.

*Ataque de distinção de chave*. Para se distinguir uma chave pública de uma matriz aleatória é suficiente se produzir uma palavra código de peso $w$ no código dual $C^{\perp}$. Neste cenário, um adversário utilizando ISD para a síndrome zerada irá se deparar com um problema de $r$ soluções (as $r$ linhas da matriz de verificação de paridade esparsa). Portanto, $N_S = r$ e $N_i = 1$ e o ataque de distinção de chave é dado por:

$$WF_{dist}(n,r,w) = \frac{WF_{isd}(n,n-r,w)}{r} \quad .$$

*Ataque de recuperação de chave*. Para a recuperação de uma chave privada é suficiente que se recupere todas as equações de verificação de paridade. As variantes ISD são todas parametrizadas e pode-se, portanto, fazer $r$ chamadas independentes para um determinado algoritmo. Cada chamada custa em média $\frac{WF_{isd}(n,n-r,w)}{r}$ porque existem $r$ palavras código de peso $w$. Logo, a recuperação de todas as equações será:

$$WF_{reco}(n,r,w) = r\frac{WF_{isd}(n,n-r,w)}{r} = WF_{isd}(n,n-r,w) \quad .$$

*Ataque de decodificação*. Para códigos MDPC a segurança da mensagem está relacionada à dificuldade de decodificação de $t$ erros num código binário linear aparentemente aleatório de tamanho $n$ e codimensão $r$:

$$WF_{dec}(n,r,t) = WF_{isd}(n,r,t) \quad .$$

No caso quase-cíclico, qualquer deslocamento cíclico da síndrome $s \in \boldsymbol{F}_2^r$ limitado à extensão de um bloco resulta numa nova instância cuja solução é idêntica à da síndrome original. O número de instâncias e o de soluções são dados por $N_i = N_S = r$. Há, portanto, um fator $\sqrt{r}$ a ser levado em consideração. Logo,

$$WF_{dec}^{QC}(n,r,t) \geq \frac{WF_{isd}(n,r,t)}{\sqrt{r}} \quad .$$

Mediante a análise de segurança realizada, Misoczki, Tillich, Sendrier e Barreto (2013, p. 17) propõem os parâmetros para sua variante QC-MDPC McEliece, os quais estão

mostrados na Tabela 2.2. Com estes parâmetros, os códigos QC-MDPC apresentam taxas de falha de decodificação inferiores a $10^{-7}$ com o emprego de algoritmos de inversão de bits. Misoczki, Tillich, Sendrier e Barreto (2013, p.16) ainda acrescentam que "*com respeito à eficiência da nossa proposta, a etapa de criação das chaves depende apenas da geração de palavras aleatórias, para a chave privada, e do produto de blocos de bits quase-cíclicos, para a chave pública. A criptografia é reduzida a um produto matriz-vetor e a adição de vetores. E para a decodificação, uma implementação em C++ não-otimizada rodando em um processador Intel Xeon CPU@3.20GHz decifra uma mensagem em menos de 3 milissegundos com os parâmetros de 80-bits de segurança*".

Tabela 2.2–Parâmetros sugeridos para variante QC-MDPC McEliece. Tamanhos em bits. Fonte Misoczki, Tillich, Sendrier e Barreto (2013, p. 17).

| Nível de segurança | $n_0$ | $n$ | $r$ | $w$ | $t$ | Tamanho da chave |
|---|---|---|---|---|---|---|
| 80 | 2 | 9600 | 4800 | 90 | 84 | 4800 |
| 80 | 3 | 10752 | 3584 | 153 | 53 | 7168 |
| 80 | 4 | 12288 | 3072 | 220 | 42 | 9216 |
| 128 | 2 | 19712 | 9856 | 142 | 134 | 9856 |
| 128 | 3 | 22272 | 7424 | 243 | 85 | 14848 |
| 128 | 4 | 27200 | 6800 | 340 | 68 | 20400 |
| 256 | 2 | 65536 | 32768 | 274 | 264 | 32768 |
| 256 | 3 | 67584 | 22528 | 465 | 167 | 45056 |
| 256 | 4 | 81920 | 20480 | 644 | 137 | 61440 |

Na última edição de seu artigo, Misoczki, Tillich, Sendrier e Barreto (2013) propõem que seu criptossistema trabalhe com os seguintes parâmetros para um nível de segurança de 80-bits:

$$n0 = 2;\ n = 9602;\ r = 4801;w = 90;\ t = 84$$

Observa-se que a codimensão $r$ foi incrementada em um bit ($n$ por consequência em dois bits) comparando-se com os parâmetros propostos anteriormente. De acordo com os autores isto é necessário devido a um ataque viável e não publicado caso $r$ não seja primo e, portanto, seja fatorável.

A Tabela 2.3 proporciona uma comparação entre tamanhos de chaves da variante QC-

MDPC McEliece de Misoczki, Tillich, Sendrier e Barreto (2013), de um variante QC-LDPC de Baldi, Bodrato e Chiaraluce (2008), de uma variante McElice com códigos quase-diádicos de Goppa de Misoczki e Barreto. (2009) e da proposta original de McEliece com parâmetros atualizados descrita por Bernstein, Lange e Peters (2008).

Tabela 2.3–Comparação de tamanhos de chaves. Tamanhos em bits. Fonte Misoczki, Tillich, Sendrier e Barreto (2013, p. 17).

| Nível de segurança | QC-MDPC | QC-LDPC | QD-Goppa | Goppa |
|---|---|---|---|---|
| 80 | 4800 | 12096 | 20480 | 460647 |
| 128 | 9856 | - | 32768 | 1537536 |
| 256 | 32768 | - | 65536 | 7667855 |

Sejam os parâmetros $n_0 = 2$, $n = 9600$, $r = 4800$, $w = 90$, $t = 84$ sugeridos por Misoczki, Tillich, Sendrier e Barreto (2013) para sua variante QC-MDPC McEliece. Neste caso há um custo de $2^{92,70}$ para recuperação de chave e $2^{87,16}$ para ataques de decodificação. Considerando o ganho DOOM, os valores são corrigidos para $2^{80,47}$ e $2^{81,04}$.

# 3 IMPLEMENTAÇÃO

Neste capítulo são detalhadas as técnicas utilizadas na implementação do criptossistema com a descrição do trabalho e a delimitação do estudo.

## 3.1 DESCRIÇÃO DO TRABALHO

A proposta é a construção de uma implementação eficiente do criptossistema McEliece baseado em códigos QC-MDPC publicado por Misoczki, Tillich, Sendrier e Barreto (2013).

O desenvolvimento do criptossistema foi feito em linguagem C na plataforma Intel e desconsiderou qualquer limitação de memória ou capacidade de processamento e armazenamento do hardware, desta maneira sendo possível, por exemplo, armazenar as estruturas de dados todas e por completo em memória RAM, o que possibilitou ganho em algumas rotinas, pois todos os dados necessários para o processamento estavam disponíveis.

Para sua realização teve como ponto de partida o artigo de Misoczki, Tillich, Sendrier e Barreto (2013) com a construção do gerador das chaves, do codificador e do decodificador descritos, adotando os seguintes parâmetros apresentados pelos autores (em bits), para um nível de segurança de 80 bits:

$$n_0 = 2,\ n = 9600,\ r = 4800,\ w = 90,\ t = 84,\ \text{tamanho da chave} = 4800.$$

Portanto, uma mensagem de 4800 bits ($r$) é codificada em uma mensagem cifrada de 9600 ($n$) bits à qual 84 bits de erro ($t$) são adicionados. A matriz de paridade H possui linhas com peso constante 90 ($w$) e consiste de 2 ($n_0$) blocos $H_0$ e $H_1$.

Uma versão inicial considerada como "*versão didática*" foi construída, cujo objetivo era apenas o entendimento do funcionamento do criptossistema. Esta versão serviu como prova de conceito e permitiu a assimilação de muitas das ideias envolvidas. Na versão inicial o tipo de dados adotado para os vetores e matrizes foi o inteiro sem sinal (*unsigned int*) e cada elemento dessas estruturas de dados armazenou apenas 1 bit de informação.

Após a construção da versão inicial do criptossistema, a sedimentação de ideias e a tomada de tempo de execução das rotinas de encriptação e decriptação, o esforço todo passou a ser

direcionado para a otimização e melhora do desempenho da decriptação, pois como mencionado, esta etapa é a que consome maior tempo de processamento. Adotou-se como uma referência para medição da evolução do trabalho o desempenho da respectiva rotina de decriptação mencionado no artigo de Misoczki, Tillich, Sendrier e Barreto (2013, p. 16), que é de 3 ms.

Seguindo a versão inicial, várias versões subsequentes do criptossistema foram desenvolvidas nas quais novas ideias eram testadas incrementalmente para melhorar o tempo de decriptação. As mal sucedidas obviamente eram descartadas. A seguir encontra-se detalhado o trabalho de implementação:

**Geração das chaves:** como não havia nenhuma restrição de projeto, os vetores e matrizes binários, mesmo sendo esparsos, ficaram todos armazenados integralmente em memória durante toda a execução do programa, o que possibilitou ganho em algumas operações tais como as atualizações de contadores na decriptação, por exemplo. Para a manipulação dessas estruturas de dados foi utilizada a biblioteca numérica de domínio público para representação de vetores e matrizes *Meschach*, a qual provê os tipos de dados básicos matriz e vetor bem como algumas operações primárias tais como a cópia de blocos de dados entre ambos. A biblioteca teve que ser devidamente customizada para que os tipos de dados armazenados pelas matrizes e vetores binários fossem representados como números inteiros sem sinal, uma vez que ela vem configurada para o tipo real (*float*).

Durante a tarefa de geração das chaves do criptossistema existe a necessidade de inversão de matrizes binárias. Para a codificação desta tarefa selecionou-se o trabalho de Jasinski, Pedroni, Gortan e Godoy Jr. (2010), o qual traz um algoritmo eficiente que implementa uma versão melhorada do método de Gauss-Jordan para a inversão de matrizes binárias.

**Cifrador:** para a construção do cifrador, com relação a aspectos de desempenho, o maior cuidado tomado nas versões que sucederam a versão inicial didática foi com a representação de conjuntos de bits por números inteiros sem sinal de 32 bits inicialmente e 64 bits numa segunda etapa. Ou seja, o tipo de dado de cada elemento dos vetores e matrizes foi determinado como inteiro sem sinal de 32 bits e 64 bits em versões sucessivas do criptossistema de tal maneira que cada elemento passou a armazenar 32 ou 64 bits, dependendo da versão, em vez de 1 bit do sistema inicial. Desta maneira os bits puderam passar a ser processados em blocos, o que garante melhora no desempenho de tarefas que

necessitem ser aplicadas em linhas de matrizes ou vetores, tais como a multiplicação de matrizes por matrizes ou de vetores por matrizes, nas quais são executadas operações lógicas (XOR's e AND's) nas linhas binárias. Como desvantagem, o acesso a bits específicos tornou-se mais complexo, uma vez que cada bit passou a ser definido por dois índices: o índice do vetor ou matriz em que reside o bloco do qual faz parte e sua posição na palavra de 32 ou 64 bits. Porém esse ônus dos dois índices foi superado em muito pelo ganho representado pela possibilidade de processamento dos bits em blocos.

Tanto a rotina de geração de chaves quanto a de encriptação não eram objetivos da otimização e, portanto, ficaram estagnadas na versão na qual são considerados blocos de 64 bits sem sinal.

**Decriptador:** acompanhando a geração de chaves e a encriptação, as matrizes e vetores foram definidos como elementos de tipos de dados inteiros sem sinal de 32 bits e 64 bits em versões sucessivas do decriptador, até por questão de compatibilidade com as rotinas de geração de chaves e a cifradora. Na decriptação procurou-se fazer uso ao máximo de recursos e técnicas que permitissem a economia no número de operações executadas pelo processador e a consequente melhora no desempenho do processamento.

Como pode ser observado no Algoritmo 2.3, numa tarefa de decodificação existe o cálculo inicial da síndrome, que corresponde à multiplicação da matriz de paridade H pelo vetor transposto que representa a mensagem cifrada *c*. Medindo-se o tempo de processamento de todos os procedimentos que compõem a rotina de decodificação, percebe-se que esta multiplicação corresponde à etapa de maior custo no algoritmo. Um truque bastante simples e eficiente e que reduz em muito o tempo gasto para se decodificar uma mensagem é aplicado exatamente nesta multiplicação: na computação da síndrome, calcula-se o produto do vetor transposto pela matriz transposta. E na realização deste cálculo percorre-se cada elemento individual binário do vetor e as posições dos bits não nulos selecionam as linhas da matriz transposta às quais deve ser aplicado o XOR. O resultado desta operação é a síndrome. Como o vetor mensagem cifrada tem cerca de apenas um terço de suas posições diferentes de zero, isto significa que somente aproximadamente um terço dos XOR's das linhas da matriz H precisa efetivamente ser computado para se chegar ao valor da síndrome. Este truque foi implementado e mostrou-se bastante útil devido à configuração da matriz e do vetor que deviam ser multiplicados.

A Figura 3.1 ilustra o processo. Desejando-se multiplicar um vetor binário por uma matriz binária inicialmente calcula-se a transposta da matriz. O vetor resultado da multiplicação é dado pelo XOR das linhas da matriz transposta que correspondem às posições do vetor que são iguais a 1.

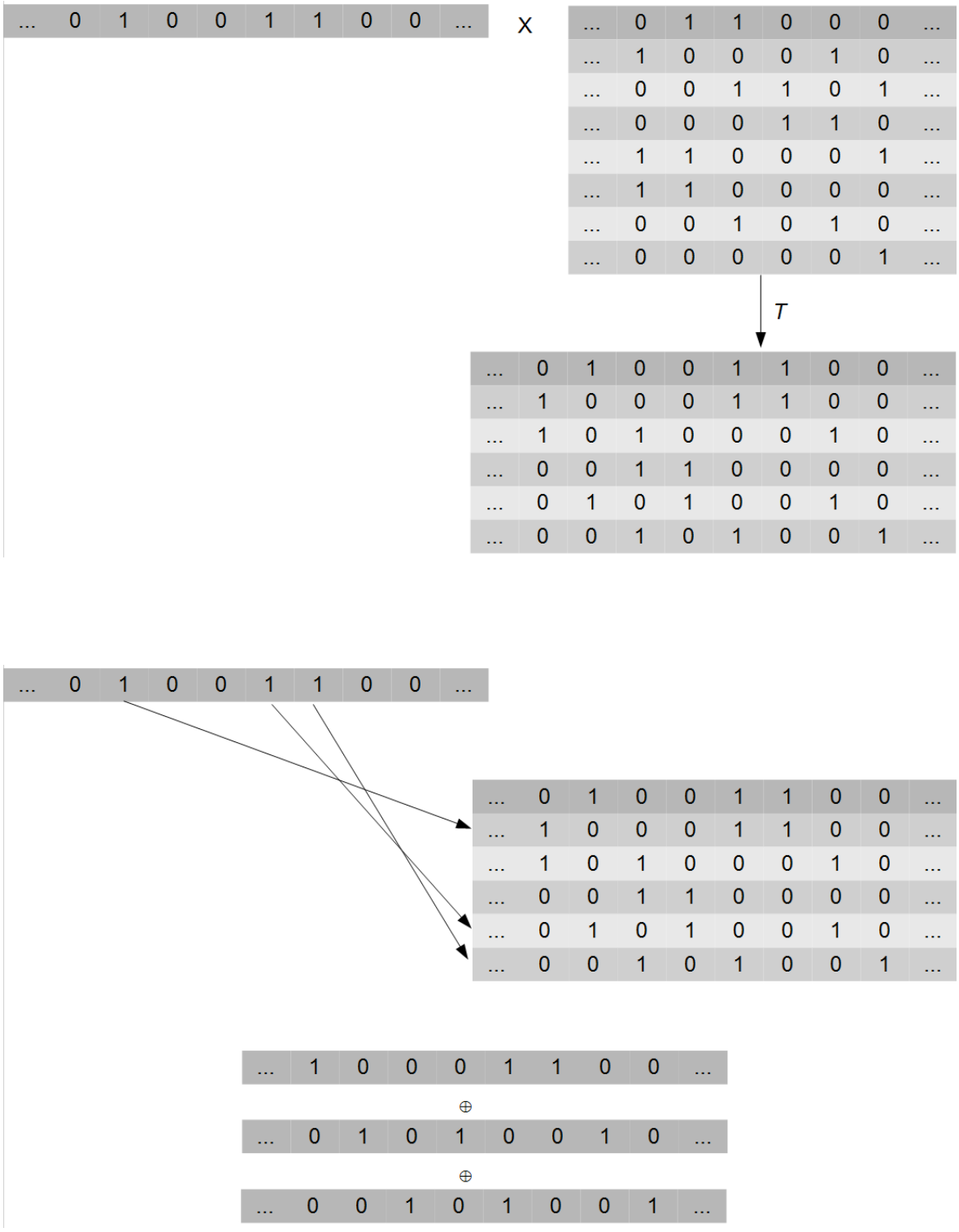

Figura 3.1–Multiplicação matriz x vetor.

No algoritmo da decodificação, Algoritmo 2.3, após o cálculo da síndrome há um passo no qual é executado o cálculo dos contadores que registram a quantidade de equações de verificação de paridade não satisfeitas associadas a cada bit da mensagem. Uma vez que esta contagem é repetida a cada atualização da síndrome, foi testada alternativa na qual os contadores eram apenas atualizados em vez de zerados e recalculados. Nos testes armazenou-se a síndrome anterior e comparou-se a mesma com a síndrome atual e apenas os contadores relacionados às posições dos bits que mudaram entre ambas as versões da síndrome eram incrementados ou decrementados, dependendo da mudança do bit de zero para um ou de um para zero. A medição de tempo dos testes desaprovou esta solução alternativa ao mostrar que a recontagem de todos os contadores da maneira como está proposto no algoritmo é a estratégia mais rápida.

Levando-se em consideração que a operação de multiplicação de matriz por vetor para o cálculo da síndrome é a mais onerosa da decodificação e que a mesma é processada em blocos de 64 bits, uma técnica de otimização que representou impacto positivo no tempo de processamento foi a adoção de instruções vetoriais intrínsecas nas operações de XOR que existem tanto no cálculo quanto no processamento da síndrome.

Instruções intrínsecas são aquelas que o compilador mapeia em uma sequência de uma ou mais instruções em linguagem de máquina com o grande benefício de possibilitar o acesso a novos registradores específicos do hardware assim como a novos tipos de dados que não estão disponíveis através dos métodos padrão das linguagens de programação.

As funções intrínsecas são inerentemente mais rápidas que as convencionais providas pela linguagem, O código para uma função intrínseca geralmente é inserido em linha, evitando a sobrecarga de uma chamada de função e permitindo instruções de máquina altamente eficientes, pois o compilador tem o conhecimento prévio e específico de como as instruções intrínsecas se comportam.

Mediante o uso de instruções vetoriais intrínsecas – instruções SIMD (*single instruction multiple data*) - é possível a paralelização do código com a realização de operações lógicas ou matemáticas em múltiplos pares de operandos simultaneamente.

Para o processamento das matrizes e vetores foram utilizadas instruções vetoriais intrínsecas Intel do conjunto de extensões AVX e AVX2. Com o conjunto de instruções AVX é possível o acesso a registradores de 256 bits da família de processadores Intel com

a microarquitetura *Sandy Bridge*. O conjunto de instruções AVX2 torna disponíveis registradores de 512 bits da família de processadores com microarquitetura *Haswell*. O ganho de tempo resultante do uso das instruções vetoriais nestes casos deve-se à paralelização: como afirmado, após serem carregados conjuntos de elementos de 32 ou 64 bits múltiplos inteiros do tamanho dos registradores, é executado um XOR simultâneo em vários pares de operandos.

Uma última otimização experimentada diz respeito à substituição do decodificador da proposta de Misoczki, Tillich, Sendrier e Barreto (2013) descrito no Algoritmo 2.3 pelo decodificador eficiente estudado por Von Maurich e Güneysu (2014, p.3) e apresentado no Algoritmo 2.4, o que significa uma redução no número de operações de cálculo de síndromes, que correspondem a multiplicações de vetores mensagem por matrizes de paridade e, como já foi dito, é o maior gargalo de desempenho no algoritmo de decodificação. Comparando os Algoritmos 2.3 e 2.4 observa-se que em 2.4 a síndrome é calculada uma única vez e a partir de então ela passa a ser atualizada apenas através de XOR's. As duas propostas foram implementadas, testadas e comprovou-se ganho significativo no tempo de processamento da rotina de decritação que faz economia na quantidade de multiplicações realizadas, correspondente ao decodificador mais eficiente. Há que ser ressaltado que o tempo de execução é bastante melhorado com a redução do número de multiplicações necessariamente em conjunto com a utilização dos limiares $b_i$ pré-calculados em cima dos parâmetros do código. São estes limiares que diminuem significativamente a quantidade de iterações executadas pela rotina de decodificação e os testes realizados com e sem os limiares comprovaram esta afirmativa.

## 3.2 DELIMITAÇÃO DO ESTUDO

É importante ressaltar que tanto na encriptação quanto na decriptação a aleatoriedade para a geração dos vetores binários (vetor *h* e vetor *e*) foi simulada, uma vez que a utilização de um gerador de números aleatórios verdadeiro está fora do escopo deste trabalho.

# 4 RESULTADOS

Este capítulo apresenta uma descrição geral dos resultados obtidos e procurou-se analisá-los e associá-los à evolução do trabalho ao longo das modificações e sucessivas versões implementadas.

## 4.1 VISÃO GERAL

Os resultados apresentados abaixo foram obtidos a partir de execuções das várias versões implementadas em linguagem C do criptossistema de McEliece baseado em códigos QC-MDPC. Para as implementações e testes foi utilizada uma estação de trabalho com CPU Intel Core I7-4770 trabalhando na frequência de 3,40 GHz e rodando sistema operacional Linux Ubuntu 12.04 LTS.

Em cada versão implementada do criptossistema, os tempos medidos foram calculados a partir da média dos tempos de execução de mil decodificações de uma mesma mensagem cifrada. Procedeu-se da mesma maneira para a obtenção do número médio de iterações para cada decodificação. Deve ser ressaltado que apesar de ter sido considerada uma mesma mensagem nas mil decodificações, a cada cifração é adicionado um vetor erro aleatório à mensagem, de tal maneira que em cada execução é submetido ao algoritmo decodificador um vetor mensagem diferente.

Inicialmente foi construída uma versão do criptossistema que corresponde literalmente ao que está proposto no artigo de Misoczki, Tillich, Sendrier e Barreto (2013). Não foi utilizada nenhuma estratégia de otimização do código fonte, apenas alterou-se o tipo de dados dos elementos armazenados por vetores e matrizes para inteiros de 32 bits sem sinal e 64 bits sem sinal em versões subsequentes e mediu-se os tempos respectivos, apresentados na Tabela 4.1. A ideia é estabelecer um marco inicial comparativo e indicativo da evolução dos resultados do trabalho. No artigo os autores mencionam que o tempo de decodificação por eles alcançado é de cerca de 3 ms, o que corresponde a 9,6 milhões de ciclos, tendo em vista que a CPU por eles utilizada opera em 3,2 GHz.

Tabela 4.1–Versão original do criptossistema proposto por Misoczki, Tillich, Sendrier e Barreto (2013).

| Tipo de dados (vetores/matrizes) | Tempo de decodificação (ms) | Ciclos de decodificação |
|---|---|---|
| Inteiros de 32 bits sem sinal | 8,230 | 27,982 x $10^6$ |
| Inteiros de 64 bits sem sinal | 6,740 | 22,916 x $10^6$ |

A seguir o trabalho concentrou-se na construção e teste de versões alternativas otimizadas. O passo inicial da decodificação corresponde ao cálculo da síndrome, que nada mais é que o resultado do produto da mensagem cifrada pela matriz de paridade. Analisando-se o desempenho da rotina de decodificação, observa-se que esta multiplicação é uma etapa de alto custo de processamento. Portanto, uma primeira otimização significou a aplicação de um truque na rotina de multiplicação de matriz por vetor, a qual passou a selecionar para a operação de XOR apenas as linhas da matriz transposta associada cujas posições correspondem a valores não nulos do vetor. Como o vetor possui apenas um terço de seus elementos diferentes de zero, esta estratégia economizou dois terços de operações XOR desnecessários para a obtenção da síndrome.

Além da redução do número de XOR's para o cálculo da síndrome, o uso das instruções vetoriais intrínsecas na rotina de multiplicação possibilitou o paralelismo com a execução de XOR's simultâneos em múltiplos pares de operandos carregados nos registradores específicos de 256 bits (AVX) e 512 bits (AVX2).

Todas estas ações refletem-se nos tempos mostrados na Tabela 4.2, onde a decodificação passou a acontecer em menos de 3 ms (9,6 x $10^6$ ciclos), tempo citado por Misoczki, Tillich, Sendrier e Barreto (2013) e estabelecido como referência para a pesquisa.

Tabela 4.2–Versão otimizada do criptossistema proposto por Misoczki, Tillich, Sendrier e Barreto (2013).

| Instruções vetoriais | Tempo de decodificação (ms) | Ciclos de decodificação |
|---|---|---|
| AVX | 2,595 | 8,823 x $10^6$ |
| AVX2 | 2,362 | 8,0308 x $10^6$ |

Conforme afirmado na descrição do trabalho de implementação, a multiplicação para o cálculo da síndrome é a etapa que mais demanda esforço computacional na tarefa de decodificação de uma mensagem cifrada. No Algoritmo 2.2 esta multiplicação é executada

em cada uma das iterações que fazem parte das tentativas de decifração da mensagem.

O Algoritmo 2.3 apresenta um decodificador eficiente que por sua vez elimina esses cálculos repetidos da síndrome, assim como reduz a quantidade de iterações necessárias para a decifração da mensagem. Uma versão alternativa do decodificador foi implementada contemplando o Algoritmo 2.3 e os tempos medidos, como esperado, espelharam a melhora significativa da decodificação como resultado da eliminação das multiplicações sucessivas do vetor mensagem cifrada pela matriz de paridade e a substituição desta tarefa pela simples atualização da síndrome calculada no passo inicial da decodificação, assim como a redução do número de iterações do algoritmo com a substituição dos seus limiares por valores pré-calculados baseados em parâmetros do código.

A Tabela 4.3 mostra que esta ação de substituição do algoritmo de decodificação original pelo decodificador eficiente permitiu que o objetivo do trabalho proposto fosse alcançado com a obtenção de uma implementação eficiente do criptossistema McEliece baseado em códigos QC-MDPC.

Tabela 4.3–Comparação entre versões alternativas do criptossistema QC-MDCP McEliece.

| **Instruções vetoriais** | **Algoritmo** | **Tempo de decodificação (ms)** | **Ciclos de decodificação** | **Iterações** |
|---|---|---|---|---|
| AVX | 2.2 | 2,595 | $8{,}823 \times 10^6$ | 5 |
| AVX | 2.3 | 0,945 | $3{,}213 \times 10^6$ | 2 |
| AVX2 | 2.2 | 2,362 | $8{,}0308 \times 10^6$ | 5 |
| AVX2 | 2.3 | 1,173 | $3{,}9882 \times 10^6$ | 2 |

# 5 DISCUSSÃO E CONCLUSÃO

Neste trabalho construiu-se uma versão otimizada eficiente de um criptossistema de McEliece baseado em códigos QC-MDPC. Este criptossistema acena como um possível substituto dos criptossistemas assimétricos hoje mais utilizados imune ao computador quântico. Na sua concepção original o criptossistema de McEliece baseado em códigos de Goppa apresenta como inconveniente o tamanho das suas chaves bastante superior aos hoje utilizados. A proposta de Misoczki, Tillich, Sendrier e Barreto (2013) apresenta uma versão na qual níveis de segurança atuais são alcançados com chaves de dimensões compatíveis com as dos criptossistemas de chave pública atualmente difundidos. E a implementação de uma versão com tamanhos de chaves compatíveis e tempos de processamento eficientes é mais uma evidência que reforça estes criptossistemas como candidatos a alternativas viáveis caso a computação quântica se estabeleça além dos modelos teóricos e dos laboratórios.

A proposta do criptossistema de McElice usando códigos Goppa trabalha com chaves de 460647 bits, como visto, enquanto o RSA, segundo o CA/Browser Forum (consórcio de autoridades certificadoras - CAs - e empresas fabricantes de navegadores de internet) e a NIST (agência norte-americana que publica normas técnicas para os departamentos governamentais dos Estados Unidos) deve trabalhar com chaves de no mínimo 2048 bits para certificados que expirem após 31 de dezembro de 2013.

Com a substituição dos códigos Goppa por códigos QC-MDPC, vimos que os 80 bits de segurança são alcançados com chaves de 4800 bits.

Segundo Khurana, Koleva e Basney (2007, p. 5), os tempos de decifração do RSA medidos por eles numa estação AMD Opteron 2.2Ghz com processador 248 rodando o sistema operacional Linux Debian foram:

Tabela 5.1–Tempo de decifração do RSA como função do tamanho da chave. Fonte: Khurana, Koleva e Basney (2007, p. 5).

| **Tamanho da chave (bits)** | 512 | 1024 | 2048 |
|---|---|---|---|
| **Tempo de decifração (ms)** | 0,5 | 2,4 | 13,1 |
| **Ciclos de decifração** | $1{,}1 \times 10^6$ | $5{,}28 \times 10^6$ | $28{,}82 \times 10^6$ |

Já Boneh e Shacham (2002, p. 4) apresentam os seguintes tempos para a decifração do RSA usando OpenSSL 0.9.5 numa estação Pentium III 750 MHz com 256 MB RAM e rodando o sistema operacional Debian “Potato”.

Tabela 5.2–Tempo de decifração do RSA como função do tamanho da chave. Fonte: Boneh e Shacham (2002, p. 4).

| | **Tamanho da chave (bits)** | | |
|---|---|---|---|
| | **768** | **1024** | **2048** |
| **Tempo de decifração (ms)** | 4,67 ms | 8,38 ms | 52,96 ms |
| **Ciclos de decifração** | $3{,}5025 \times 10^6$ | $6{,}285 \times 10^6$ | $39{,}72 \times 10^6$ |

Neste trabalho a decifração foi conseguida em 0.945 ms ($3{,}213 \times 10^6$ ciclos) para uma chave de 4800 bits, o que demonstra que o criptossistema de McEliece implementado com códigos QC-MDPC pode ser considerado como um perfeito substituto para o RSA em aplicações como o IPsec, o SSL ou PGP, já que os tamanhos de chave e tempos de processamento de ambos são próximos. E se considerarmos a menor complexidade dos algoritmos de encriptação e decriptação do McEliece, além de sua imunidade a ataques desde sua concepção há mais de 30 anos e também aos futuros computadores quânticos capazes de executar o algoritmo de Shor (1997), ele de fato se torna um forte candidato que deve ser apreciado como solução para implementações que requeiram a criptografia assimétrica.

Para trabalhos futuros sugere-se estudar novas alternativas para evitar a recontagem dos contadores a cada iteração do algoritmo de inversão de bits (*bit-flipping*). Esta recontagem representa um custo de processamento que possivelmente pode ser reduzido.

Outra proposta é a implementação do criptossistema proposto por Misoczki, Tillich, Sendrier e Barreto (2013) em outras plataformas, por exemplo num ambiente de computação distribuída.

As estruturas de dados que apoiam a representação do criptossistema, matrizes e vetores, têm características esparsas devido às próprias características dos códigos utilizados. Logo, uma possível nova implementação do criptossistema poderia abordar a representação alternativa destas estruturas de dados em memória, o que de alguma maneira possibilitaria a tentativa de ganho na eficiência do processamento.

Na última edição de seu artigo, Misoczki, Tillich, Sendrier e Barreto (2013) propõem que seu criptossistema trabalhe com os seguintes parâmetros para um nível de segurança de 80-bits:

$$n0 = 2;\ n = 9602;\ r = 4801;w = 90;\ t = 84$$

Assim, um bloco de mensagem de 4801 bits (*r*) é codificado em uma mensagem cifrada com 9602 bits (*n*) a qual 84 erros (*t*) são adicionados. A matriz de verificação de paridade H tem peso de linha constante $w = 90$ e consiste de 2 (*n0*) blocos circulantes.

Observa-se que a codimensão *r* foi incrementada em um bit (*n* por consequência em dois bits) comparando-se com os parâmetros propostos anteriormente. De acordo com os autores isto é necessário devido a um ataque viável e não publicado caso *r* não seja primo e, portanto, seja fatorável.

Um possível trabalho futuro pode contemplar a implementação de uma versão do criptossistema com os novos parâmetros sugeridos por Misoczki, Tillich, Sendrier e Barreto (2013) onde deve ser ressaltado que a necessidade de *r* ser primo também faz com que ele deixe de ser múltiplo de 64 bits, o que se reflete na obrigatória reescrita das rotinas que trabalham com as matrizes e vetores e que processam os blocos de dados através de repetições controladas por contadores. O esforço de tratamento das exceções referentes aos bits que ficam além dos blocos tratados nos laços de repetições e o custo de processamento destas operações provavelmente será uma outra razão para a alteração da representação das estruturas de dados em memória.

# REFERÊNCIAS BIBLIOGRÁFICAS


BALDI, M.; CHIARALUCE, F. **Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes**. In: IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY - ISIT 2007, Junho 2007. p. 2591-2595.

BALDI, M.; BODRATO, M.; CHIARALUCE, F. **A new analysis of the McEliece cryptosystem based on QC-LDPC codes**. Proceedings of the 6th international conference on Security and Cryptography for Networks (SCN '08), 2008. Berlin,Heidelberg:Springer-Verlag, p. 246-262.

BERLEKAMP E.; MCELIECE, R.; VAN TILBORG, H. **On the inherent intractability of certain coding problems (corresp.).** IEEE Transactions on Information Theory, v. 24, n. 3, p. 384-386. Maio 1978.

BERNSTEIN, D. J. **Introduction to post-quantum cryptography**. In: BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E. (Ed.). Post-Quantum Cryptography. Springer, 2009. p. 1-14.

BERNSTEIN, D. J.; LANGE, T.; PETERS, C. **Attacking and defending the McEliece cryptosystem**. Proceedings of the 2nd International Workshop on Post-Quantum Cryptography (PQCrypto '08), 2008. Berlin, Heidelberg:Springer-Verlag, p. 31-46.

BONEH, D.; SHACHAM, H. **Fast Variants of RSA**. Cryptobytes Journal, v. 5, n. 1, p. 1-9. 2002.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**, 3. ed. Cambridge, Massachusetts: The MIT Press, 2009.

DIFFIE, W.; HELLMAN, M. **Multiuser Cryptographic Techniques**. IEEE Transactions on Information Theory, Novembro 1976.

ELGAMAL, T. **A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms**. Proceedings, Crypto 84, 1984.

FAUGÈRE, J.-C.; OTMANI, A.; PERRET, L.; TILLICH, J.-P. **Algebraic cryptanalysis of McEliece variants with compact keys**. In: GILBERT, H., (Ed.). Advances in Cryptology – Eurocrypt'2010, Lecture Notes in Computer Science. Springer, 2010. v. 6110, p. 279-298.

GABORIT, P. **Shorter keys for code based cryptography**. In: INTERNATIONAL WORKSHOP ON CODING AND CRYPTOGRAPHY – WCC'2005, 2005, Bergen, Norway. ACM Press, 2005. p. 81-91.

GALLAGER, R. G. **Low-Density Parity-Check Codes**. IRE Trans. on Info. Theory, v. IT-8, p. 21-28, Janeiro 1962.

HALLGREN, S.; VOLLMER, U. **Quantum Computing**. In: BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E. (Ed.). Post-Quantum Cryptography. Springer, 2009. p. 15-34.

HEYSE, S.; VON MAURICH, I.; GÜNEYSU, T. **Smaller Keys for Code-based Cryptography: QC-MDPC McEliece Implementations on Embedded Devices**. In: BERTONI, G.; CORON, J.-S., (Ed.). CHES, Lecture Notes in Computer Science. Springer, 2013. v. 8086, p. 273-292.

HUFFMAN, W.; PLESS, V. **Fundamentals of Error-Correcting Codes**. Cambridge University Press, 2003.

JASINSKI, R. P.; PEDRONI V. A.; GORTAN, A.; GODOY JR, W. **An Improved GF(2) Matrix Inverter with Linear Time Complexity**. Procedings of the IEE International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2010. p. 322-327.

KHURANA, H.; KOLEVA, R.; BASNEY, J. **Performance of Cryptographic Protocols for High-Performance, High-Bandwidth and High-Latency Grid Systems**. Procedings of the IEEE International Conference on e-Science and Grid Computing, 2007. p. 431-439.

MCELIECE, R. J. **A Public-Key Cryptosystem Based on Algebraic Coding Theory**. DSN Progress Report 42-44, p. 114-116. Janeiro e Fevereiro 1978.

MCELIECE, R. J. **The Theory of Information and Coding**. Cambridge University Press. Julho 2004.

MESCHACH: **Matrix computations in C**. Disponível em: <http://homepage.math.uiowa.edu/~dstewart/meschach/html_manual/manual.html>. Acesso em 20 de maio de 2014.

MILNE, J. S. **Group Theory (v3.13)**. 2013. Disponível em: <www.jmilne.org/math/>.

Acesso em 18 de agosto de 2014.

MISOCZKI, R.; BARRETO, P. S. L. M. **Compact McEliece keys from Goppa codes.** Selected Areas in Cryptography, p. 376-392, 2009.

MISOCZKI, R.; TILLICH, J. P.; SENDRIER, N.; BARRETO, P. S. L. M. **MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes**. In: IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY. 2013. v. 2013.

MONICO, C; ROSENTHAL, J.; SHOKROLLAHI, A. **Using low density parity check codes in the McEliece cryptosystem**. In: IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY – ISIT'2000, 2000, Sorrento, Itália. IEEE, 2000. p. 215.

MOON, T. K. **Error Correction Coding Mathematical Methods and Algorithms**. John Wiley & Sons, 2005.

OTMANI, A., TILLICH, J.; DALLOT, L. **Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes**. Special Issues of Mathematics in Computer Science, v. 3, n. 2, p. 129-140, Janeiro 2010.

OVERBECK, R.; SENDRIER, N. **Code-Based Cryptography**. In: BERNSTEIN, D. J.; BUCHMANN, J.; DAHMEN, E., (Ed.). Post-Quantum Cryptography. Springer, 2009. p. 94-145.

PRANGE, E. **The use of information sets in decoding cyclic codes**. IRE Transactions on Information Theory, v. 8, n. 5, p. 5-9, Setembro 1962.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. **A Method for Obtaining Digital Signatures and Public Key Cryptosystems**. Communications of the ACM, Fevereiro 1978.

SENDRIER, N. **Decoding one out of many**. In: YANG, B.-Y., (Ed.), *Post-Quantum Cryptography*, Lecture Notes in Computer Science, Berlin/Heidelberg: Springer, 2011. v. 7071. 10.1007/978-3-642-25405-5-4.

SHOR, P. W. **Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer**. SIAM Journal on Computing, v. 26, n. 5, p. 1484-1509, 1997.

SIPSER, M. **Introduction to the Theory of Computation**. 2. ed. Estados Unidos: Thompson Course Technology, 2006.

STALLINGS, W. **Cryptography and Network Security**. 6. ed. Prentice Hall, 2013.

STINSON, D. R. **Cryptography Theory and Practice**. 3. ed. Chapman & Hall/CRC, 2006.

TERR, David. **Polynomial Time**. De MathWorld A Wolfram Web Resource, criado por Eric W. Weisstein. Disponível em: <http://mathworld.wolfram.com/PolynomialTime.html>. Acesso em: 18 de agosto de 2014.

VON MAURICH, I.; GÜNEYSU, T. **Lightweight Code-based Cryptography: QC-MDPC McEliece Encryption on Reconfigurable Devices**, Procedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE). Dresden: IEEE, 2014. p. 1-6.