Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

Criptografia Pós-Quântica com Códigos Corretores de Erros

Rafael Misoczki

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Paulo S. L. M. Barreto

São Paulo
2008

Universidade de São Paulo — USP
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Carlos Eduardo Ferreira

CIP — Catalogação Internacional na Publicação

, Rafael Misoczki.

Criptografia Pós-Quântica com Códigos Corretores de Erros / Rafael Misoczki . São Paulo : USP, 2008.
41 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de São Paulo, São Paulo, 2008.

1. Criptografia Pós-Quântica, 2. Criptografia, 3. Computação Quântica,
4. Segurança

CDU 004

Endereço: Universidade de São Paulo
Instituto de Matemática e Estatística
Rua do Matão, 1010 - Cidade Universitária
CEP 05508-090
São Paulo–SP — Brasil

Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

Criptografia Pós-Quântica com Códigos Corretores de Erros

Rafael Misoczki

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Paulo S. L. M. Barreto (Orientador)
Poli/USP

Prof. Dr. Carlos Eduardo Ferreira
Coordenador do Bacharelado em Ciência da Computação

São Paulo, 1 de Dezembro de 2008

Resumo

Este trabalho de conclusão de curso descreve, analisa e implementa o Sistema Criptográfico McEliece, o qual é classificado como Pós-Quântico por se tratar de um sistema seguro perante as abordagens atuais de ataques utilizando Computação Quântica. Também é apresentada uma introdução à teoria da codificação, métodos de correção de erros, emprego da decodificação de síndromes e os Códigos de Goppa. A partir destes pré-requisitos, temos a descrição do Sistema de Criptografia Pós-Quântica McEliece, expondo seus métodos de geração de chaves, codificação e decodificação de mensagens, além de um esquema de assinatura digital baseado nesses moldes, o CFS. Por fim, considerações quanto à implementação em linguagem Java da Biblioteca Criptográfica Pós-Quântica, desenvolvida como aplicação destes conceitos e que utiliza as otimizações pesquisadas para a implementação eficiente de operações sobre Corpos Finitos.

Palavras-chave: Criptografia Pós-Quântica, Criptografia, Computação Quântica, Segurança

Abstract. *This graduation work describes, examines and implements the McEliece Post-Quantum Cryptosystem, which is classified as Post-Quantum because it is a secure system to the current attacks approaches using Quantum Computing. It's also presented an introduction to the Coding Theory, methods of errors correction, syndrome decoding employment and the Goppa Codes. From these preconditions, the McEliece Post-Quantum Cryptosystem is presented, with an explantation of its methods to generate keys, encoding and decoding of messages, in addition to a digital signature scheme based on those patterns, the CFS. Finally, considerations about the implementation in Java of the post-quantum cryptographic library, developed as application of these concepts and uses the optimizations researched to efficient implementation of operations over finite fields.*

Sumário

I Parte Objetiva **1**

1 Introdução **2**

1. Motivação e Objetivos 2
2. Organização do texto 3

2 Teoria da Codificação **4**

1. Códigos Lineares 4
2. Métodos de Correção de Erros 5
3. Códigos de Goppa 8

3 Sistema Criptográfico McEliece **11**

1. Geração de chaves 12
2. Encriptação 13
3. Decriptação 13
4. Sistema de Assinatura Digital (CFS) 13
   - 4.1. Geração de Chave 14
   - 4.2. Assinatura 14
   - 4.3. Verificação de Assinatura 14

4 Implementação **15**

1. Organização do Projeto 15
2. Representação computacional de estruturas algébricas 18
3. Algoritmos implementados 19
   - 3.1. Algoritmo Estendido de Euclides 20
   - 3.2. Teste de Irredutibilidade de Ben-Hor 21
4. Otimizações 22

4.1. Soma de elementos no corpo básico como operação de Ou-Exclusivo . 22
4.2. Multiplicação e inverso multiplicativo no corpo base 22
4.3. Operações no corpo estendido 23
4.4. Quadrado das somas como soma dos quadrados 23
5. Utilizando a Biblioteca Criptográfica Pós-Quântica 23
5.1. Interface Gráfica 23
5.2. Linha de comando (Testes) 27

5 Conclusão **29**

II Parte Subjetiva **30**

6 Desafios, relacionamento com o curso e trabalhos futuros **31**
1. Desenvolvimento da pesquisa 31
2. Desafios e Frustrações 32
3. Relação do projeto com as disciplinas do curso 32
4. Trabalhos futuros 33

Referências **34**

Lista de Figuras

2.1 Canal de Comunicação. 4

4.1 Organização do Projeto. 17
4.2 Exemplo de Painel do Swing. 24
4.3 Tela inicial da Biblioteca Criptográfica Pós-Quântica. 25
4.4 Tela de Geração de Par de Chaves da Biblioteca Criptográfica Pós-Quântica. . . 26
4.5 Tela de Decriptação de mensagens da Biblioteca Criptográfica Pós-Quântica. . 28

Parte I

Parte Objetiva

Capítulo 1

Introdução

A maioria das soluções criptográficas utilizadas atualmente é embasada em problemas relacionados à teoria dos números, tal como o de fatoração de números inteiros em primos, no caso do Algoritmo RSA [Rivest et al. 1978], e o de logaritmos discretos, para criptografia baseada em curvas elípticas [Miller 1985] (abreviada por ECC). Porém, em 1994, Peter W. Shor apresentou algoritmos quânticos [Shor 1994] que resolvem estes problemas em tempo polinomial, tornando estas soluções vulneráveis a ataques provenientes de computadores quânticos. Neste cenário, inúmeras pesquisas foram realizadas a fim de buscar novas metodologias criptográficas que fossem seguras perante estas abordagens de ataques. Enquanto isso, outras linhas de pesquisa se preocuparam em validar os sistemas criptográficos já existentes, mas não empregados em larga escala, quanto a sua capacidade em resistir a esses ataques. Como exemplo deste último grupo, temos o Sistema Criptográfico McEliece, que vem se mostrando como uma interessante alternativa.

A perda da segurança dos sistemas criptográficos usuais depende da viabilidade da computação quântica, a qual ainda não é uma realidade. Porém, sabemos que a mesma vem recebendo notória atenção tanto da comunidade científica quanto da indústria e governos. Como exemplo deste interesse, recentemente, setores da indústria anunciaram a construção de computadores com processadores que utilizam aspectos da mecânica quântica para o seu processamento. Mesmo com um poderio computacional reduzido, estes dispositivos tem um valor simbólico ao mostrar a possível viabilidade na construção de computadores quânticos de grande capacidade.

1. Motivação e Objetivos

Levando em consideração a possibilidade da computação quântica apresentar-se viável, acarretando na invalidez da segurança dos sistemas atualmente classificados como seguros (os quais utilizam as abordagens tradicionais de criptografia, tais como RSA ou Criptografia baseada em Curvas Elípticas), é evidente a necessidade de estudar maneiras alternativas de segurança neste contexto. Sendo assim, este trabalho tem três objetivos principais: primariamente, o entendimento de uma das alternativas pós-quânticas mais promissoras, o Sistema Criptográfico McEliece; secundariamente, a pesquisa de como implementar eficientemente este sistema, considerando as operações e representações necessárias do campo da álgebra abstrata e, por fim, o desenvolvimento de uma Biblioteca Criptográfica que opere segundo esses preceitos.

2. Organização do texto

Este texto é dividido em duas seções principais: Parte Objetiva e Parte Subjetiva. A primeira contém o texto técnico referente ao trabalho realizado. Para a correta compreensão do funcionamento do sistema estudado, alguns conceitos da teoria da Codificação são necessários e, por isto, são apresentados primariamente. A seguir, a descrição dos algoritmos do McEliece, juntamente com uma análise de suas características. Por fim, os detalhes sobre a implementação da Biblioteca Criptográfica Pós-Quântica, desenvolvida como aplicação da pesquisa realizada. A Parte Subjetiva, requisitada pela disciplina MAC0499 - Trabalho de Formatura Supervisionado, contém as considerações sobre os desafios e frustrações encontrados durante o desenvolvimento do trabalho, além da relação entre a teoria aprendida durante o curso com os conceitos necessário à pesquisa, finalizando com a sugestão dos possíveis trabalhos futuros.

Capítulo 2

Teoria da Codificação

A Teoria da Codificação tem como objetivo assegurar que, ao transmitir uma coleção de dados através de um canal sujeito a ruídos (ou seja, à perturbações nos dados enviados), o destinatário dessa transação possa recuperar a mensagem original. Para isso, deve-se encontrar maneiras eficientes de adicionar informação redundante à mensagem original de tal forma que, caso a mensagem chegue ao destinatário contendo erros (existindo inversão em certos bits, para o caso de mensagens binárias), o recebedor possa corrigi-la. A Figura 1, baseada em [Huffman and Pless 2003], ilustra bem esta situação:

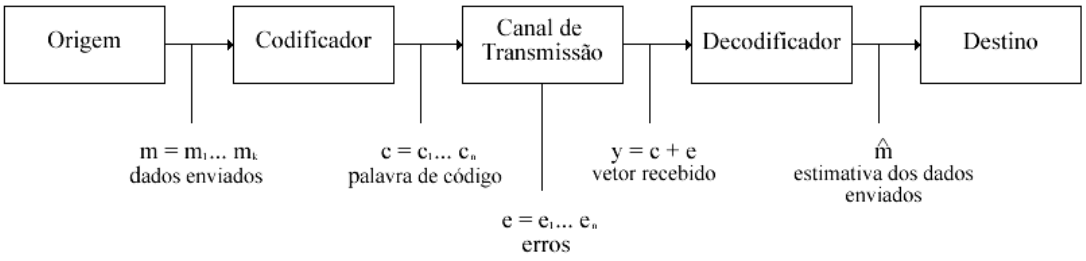

Figura 2.1. Canal de Comunicação.

Como será explicado mais adiante, no contexto da criptografia, esta capacidade de correção de ruídos é utilizada com o objetivo de dificultar a ação de possíveis ataques, adicionando estes erros propositadamente e aumentando assim a complexidade para a quebra do sistema.

1. Códigos Lineares

Inicialmente, vamos introduzir alguns conceitos úteis à tarefa de codificar mensagens. O primeiro deles se refere a *Código Linear Binário*, que pode ser definido como:

Definição 1 Um *código linear binário* C-(n,k) , de tamanho n e posto k, é um subespaço linear C com dimensão k do espaço vetorial $\mathbb{F}_2^n$, onde $\mathbb{F}_2$ é um corpo finito com 2 elementos. Os parâmetros n e k também são conhecidos respectivamente como *comprimento*

e *dimensão* do código. Um vetor que represente um elemento deste código é conhecido como *palavra de código*.

Uma maneira eficiente de representar um código linear C-(n, k) é através de uma *matriz geradora*. Tal matriz poderá gerar qualquer palavra pertencente ao código por meio da combinação linear de suas linhas. Para isso, ela deverá ter dimensões $k \times n$, suas k linhas deverão ser linearmente independentes entre si e seus elementos pertencentes a $\mathbb{F}_q$. Descrevendo o código C em função de uma *matriz geradora* G, teríamos: $C = \{ u\mathrm{G} \mid u \in \mathbb{F}^k \}$. A fim de ilustrar os conceitos, utilizaremos como exemplo, a seguinte matriz geradora:

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

Esta matriz gera o código conhecido como Código de Hamming (7, 4, 3) [Huffman and Pless 2003], o qual será empregado para ilustrar alguns conceitos a seguir. Para codificar uma mensagem m em um determinado código, basta multiplicá-la pela matriz geradora deste código. Exemplificando, suponha que a mensagem a ser codificada seja m = [1 1 0 0]. Para codificá-la, basta multiplicar o vetor m por G, obtendo assim a palavra de código m' = [0 0 1 1 0 0 1]. Observe que o vetor m' é maior do que o vetor m, ilustrando o fato de termos adicionado informação redundante aos dados originais, o que nos ajudará posteriormente na tarefa de corrigir mensagens com erros. Analisando a matriz geradora G, note que para uma mensagem original $m = [m1, m2, m3, m4]$, sua mensagem codificada m' seria da forma:

$$m' = m * G = [(m1 + m2 + m3), (m1 + m2 + m4), m2, (m1 + m3 + m4), m3, m4, m1]$$

Expressando a palavra codificada desta maneira, fica claro que se lermos as suas componentes na ordem: 7ª, 3ª, 5ª e por último, 6ª, determinaremos diretamente a seqüência $[m1, m2, m3, m4]$, correspondente a mensagem original m e ilustrando assim, um método de decodificação [Au et al. 2003] para o Código de Hamming-(7,4,3).

2. Métodos de Correção de Erros

Atualmente, existem diversas abordagens para o problema da correção de erros em mensagens binárias. Algumas delas são: Máxima Probabilidade, Mínima Distância e Decodificação de Síndromes. A idéia central destes algoritmos é tentar se aproveitar das informações redundantes geradas na codificação ou das características que as mensagens codificadas têm. Antes de começar a analisar alguns dos métodos de correção de erros, as definições a seguir serão necessárias:

Definição 2 O peso de um vetor é o número de componentes não-nulas.

Definição 3 O peso mínimo d de um código C é o menor peso existente em alguma palavra não totalmente nula (diferente de zero em alguma componente) desse código.

Desta forma, o vetor [1 0 1] tem peso 2, já o vetor [0 0 0] tem peso 0. Tendo estas definições sido caracterizadas, é possível explorar o fato enunciado no Teorema 1:

Teorema 1 ([Voloch 1987]) Um código linear binário, representado por (n, k, d), sendo n seu comprimento, k sua dimensão e d seu peso mínimo, pode corrigir até t = $\lfloor \frac{d-1}{2} \rfloor$ erros.

Assim, para o Código de Hamming (7, 4 3), temos que $n = 7$, $k = 4$ e $d = 3$. Desses parâmetros, podemos concluir que a mensagem codificada terá tamanho 7, as mensagens originais deverão ter tamanho 4 e que este código é capaz de corrigir até $t = \lfloor (d - 1)/2 \rfloor = 1$ erro.

No contexto do Sistema McEliece, estamos interessados em estudar o método conhecido por Decodificação de Síndromes, porém, para melhor ilustrar essa tarefa, iremos começar com uma explanação sobre o método de Máxima Probabilidade. Imagine que desejamos transmitir a mensagem [1 1 0 0], que ao ser codificada no Código de Hamming (7, 4, 3) se transforma em [0 0 1 1 0 0 1]. Porém, durante a transmissão dos dados, uma perturbação ocorreu em um dos bits, acarretando no fato de que a mensagem recebida foi [0 0 1 1 1 0 1]. Como este código tem $t = 1$, podemos nos assegurar de que ele é capaz de corrigir esse bit alternado. Para isso, o método de Máxima Probabilidade utiliza-se de uma tabela contendo cada palavra de código associada a todos as 7 possíveis variações (correspondentes ao erro em cada um dos bits):

Tabela 2.1. Tabela de Correção de Erros do Método Máxima Probabilidade para o exemplo

Palavra	0000000	1101001	1110000	0011001	$\cdots$
	0000001	1101000	1110001	0011000	$\cdots$
	0000010	1101011	1110001	0011000	$\cdots$
	0000001	1101000	1110001	0011000	$\cdots$
	0000010	1101011	1110001	0011000	$\cdots$
	0000001	1101000	1110001	0011000	$\cdots$
	0000010	1101011	1110001	0011000	$\cdots$
	0000001	1101000	1110001	0011000	$\cdots$

Para corrigir a mensagem recebida [0 0 1 1 1 0 1] o algoritmo encontra sua referência na tabela e a substitui pela palavra da primeira linha de sua coluna. Note que esse tipo de abordagem sempre retornará o código que mais se assemelha à mensagem contendo os erros, ou seja, o de maior probabilidade de acerto. Vale ressaltar que este é considerado um dos melhores métodos de correção de erros, porém, a construção de tabelas para códigos maiores do que o exemplificado é dispendiosa e ineficiente. A fim de melhorar esse aspecto, apresentaremos o

método de Decodificação de Síndromes, posteriormente às seguintes definições .

Definição 4 Seja C um código (n, k, d). Definimos $C^{\perp}$:={ $v \in V \mid v \cdot c = 0, \forall\, c \in C$ }, onde $\cdot$ representa o produto escalar. $C^{\perp}$ é chamado de *dual do código C* e tem dimensão $n - k$.

Definição 5 Uma *matriz de verificação de paridade* de um código C é uma matriz de dimensões $(n - k) \times n$, onde as linhas são linearmente independentes entre si e tais que o produto (módulo 2) de cada uma delas por qualquer palavra do código C resulte em 0. Essa matriz gera o *código dual* de C.

A matriz de verificação de paridade para o Código de Hamming (7, 4, 3) seria:

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

Selecionando a primeira linha [0 0 0 1 1 1 1] e a palavra de código [0 0 1 1 0 0 1], verificamos:

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

Definição 6 Seja $v \in \mathbb{F}_q^n$. Multiplicando a *matriz de verificação de paridade* de um código C por v^t (a transposição do vetor v), o vetor resultante de dimensão $(n - k)$ é chamado de *Síndrome* de v em C.

Fato 1 Seja m uma mensagem enviada pela origem e $\overline{m}$ a mensagem recebida pelo destino. Caso a síndrome de $\overline{m}$ for não nula, a mensagem contém erros e sua síndrome é igual à síndrome do vetor de erros e, tal que $\overline{m} = m + e$. (Lembrando que Hm^t corresponde a um vetor nulo, temos que: $H\overline{m}^t = H(m + e)^t = Hm^t + He^t = He^t$.)

Assim, a maneira para corrigir uma mensagem codificada $\overline{m}$ recebida pode ser:

1. Calcular e armazenar as síndromes de todos os possíveis vetores de erros.
2. Calcular a síndrome da mensagem recebida.

3. Identificar o vetor de erro através de sua síndrome, que é igual à síndrome da mensagem recebida.
4. Somar o vetor de erros identificado pelo passo anterior à mensagem recebida, obtendo a mensagem original.

Observe que novamente temos a utilização de uma estrutura armazenando dados pré-calculados a fim de determinarmos qual palavra de código a mensagem recebida representa. Porém, neste caso, apenas calculamos e armazenamos a *síndrome* de cada possível erro, ao invés de todas as variações de cada uma das palavras códigos. Essa diferença acarreta em um grande ganho de desempenho, visto que a matriz contendo as possíveis palavras e suas perturbações ocupa 2^n entradas, enquanto a matriz contendo a síndrome dos erros tem tamanho 2^{n-k+1}. O Código de Hamming (7, 4, 3), utilizado até o momento, não é o utilizado pelo Sistema Criptográfico McEliece, que, por razões de maior adaptabilidade às práticas criptográficas, utiliza os códigos da família conhecida por Códigos de Goppa.

3. Códigos de Goppa

Os códigos de Goppa, propostos por V. D. Goppa em 1970, têm características bastante interessantes para seu emprego junto ao Sistema McEliece. Por exemplo, a determinação de seu peso mínimo é uma tarefa de fácil execução, como pode ser verificado em [Engelbert et al. 2007]. Outros fatores que o tornam propício a este fim se referem à eficiente maneira de corrigir erros baseada em seu polinômio gerador e ao fato de que, desconhecendo seu polinômio gerador, não existe algoritmo eficiente para essa correção. Abaixo, temos as definições necessárias para a construção e caracterização das estruturas e operações para o emprego dos Códigos de Goppa junto ao McEliece.

Definição 7 O polinômio mônico $g \in \mathbb{F}_{2^m}[X]$ de grau t, definido por: $g(X) = \sum_{i=0}^{t} g_i X^i$ é chamado de *Polinômio de Goppa*.

Definição 8 Seja $L = (\gamma_0, ..., \gamma_{n-1}) \in \mathbb{F}_{2^m}^n$ tal que nenhum dos γ_i seja raiz do Polinômio de Goppa g, ou seja, $g(\gamma_i) \neq 0$, $\forall 0 <= i < n$. Então, este conjunto é chamado *Suporte de Código*.

Definição 9 Para qualquer vetor $c = (c_0, ..., c_{n-1}) \in \mathbb{F}_2^n$, definimos a *síndrome* de c como:

$$S_c(X) = -\sum_{i=0}^{n-1} \frac{c_i}{g(\gamma_i)} \frac{g(x) - g(\gamma_i)}{X - \gamma_i} \mod g(X)$$

Definição 10 O *Código Binário de Goppa*, denotado por $G(L, g(X))$ sobre $\mathbb{F}_2$ é o conjunto de todos os vetores $c = (c_0,...,c_{n-1}) \in \mathbb{F}_2^n$ tais que $S_c(X) = 0$ seja assegurado em $\mathbb{F}_{2^m}[X]$. Se g for irredutível sobre $\mathbb{F}_{2^m}$, então G é chamado *Código de Goppa Binário Irredutível*, e como $g(\gamma) \neq 0$, $\forall \gamma \in \mathbb{F}_{2^m}$, então L contém todos os elementos de $\mathbb{F}_{2^m}$.

De [Engelbert et al. 2007], a *matriz de verificação de paridade* para os Códigos de Goppa, pode ser construída a partir de:

$$\frac{g(X) - g(\gamma_i)}{X - \gamma_i} = \sum_{j=0}^{t} g_j \frac{X^i - \gamma_i^j}{X - \gamma_i} = \sum_{s=0}^{t-1} X^s \sum_{j=s+1}^{t} g_j \gamma_i^{j-1-s},\ 0 \leq i < n$$

Assim, para $c \in G(L, g(X))$, devemos ter $\forall s = 0, \cdots, t-1$:

$$\sum_{i=0}^{n-1} \left(\frac{1}{g(\gamma_i)} \sum_{j=s+1}^{t} g_j \gamma_i^{j-1-s} \right) c_i = 0$$

Dessa forma, a *matriz de verificação de paridade* de $G(L, g(X))$ pode ser escrita como:

$$H = \begin{pmatrix} g_t g(\gamma_0)^{-1} & \cdots & g_t g(\gamma_{n-1})^{-1} \\ (g_{t-1} + g_t\gamma_0)g(\gamma_0)^{-1} & \cdots & (g_{t-1} + g_t\gamma_{n-1})g(\gamma_{n-1})^{-1} \\ \vdots & \ddots & \vdots \\ (\sum_{j=1}^{t} g_j \gamma_0^{j-1})g(\gamma_0)^{-1} & \cdots & (\sum_{j=1}^{t} g_j \gamma_{n-1}^{j-1})g(\gamma_{n-1})^{-1} \end{pmatrix} = XYZ$$

$$X = \begin{pmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix}, Y = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \gamma_0 & \gamma_1 & \cdots & \gamma_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_0^{t-1} & \gamma_1^{t-1} & \cdots & \gamma_{n-1}^{t-1} \end{pmatrix},$$

$$e \quad Z = \begin{pmatrix} \frac{1}{g(\gamma_0)} & & & \\ & \frac{1}{g(\gamma_1)} & & \\ & & \ddots & \\ & & & \frac{1}{g(\gamma_{n-1})} \end{pmatrix}$$

Definição 11 Seja C um Código de Goppa, com polinômio de Goppa g e suporte $L = (\gamma_0, \cdots, \gamma_{n-1})$, então o polinômio $\sigma(X) \equiv \prod_{e_i=1}(X - \gamma_i) \in \mathbb{F}_{2^m}[X]/g(x)$ é conhecido como *Polinômio Localizador de Erros* de C e tem a propriedade de $\sigma(\gamma_i) = 0 \Leftrightarrow e_i = 1$.

Assim, uma maneira para corrigir uma mensagem recebida $c = mG + e$ seria a partir da construção do Polinômio Localizador de Erros, como enunciado a seguir:

1. Calcular a síndrome de e, aproveitando-se do fato de que $S_e = S_c$.
2. Construir o Polinômio Localizador de Erros a partir da Síndrome de e (ver [Engelbert et al. 2007, seção 2.2]).

3. Determinar o conjunto de i tais que $\sigma(\gamma_i) = 0$, podendo assim, determinar e a partir da propriedade existente na definição de Polinômio Localizador de Erros.
4. Calcular $mG = c + e$, obtendo m a partir do sistema linear $(mG)H = 0$.

Capítulo 3

Sistema Criptográfico McEliece

O Sistema Criptográfico McEliece é um sistema de chaves assimétricas que representa uma interessante alternativa às soluções empregadas atualmente. Utilizando-se da habilidade de códigos lineares em conseguir corrigir mensagens contendo um número estimado de erros, ele provê um sistema resistente às abordagens atuais dos ataques originados em computadores quânticos. Para isso, o sistema parte do pressuposto de que o problema da decodificação de um código linear genérico é NP-difícil. Assim, a idéia central seria selecionar um código específico que tenha um algoritmo de decodificação eficiente (para a versão original, Códigos de Goppa), e, então, disfarçar este código como sendo um código linear genérico, implicando em um eficiente esquema de chaves assimétricas, que grosseiramente pode ser descrito como: a chave pública seria o código disfarçado e a chave privada, o código original.

Tal sistema é dotado de diversas particularidades que ressaltam seus pontos positivos e negativos. Uma das características mais relevantes deste sistema recai sobre a sua eficiência algorítmica. Seus algoritmos de encriptação e decriptação de mensagens consomem muito menos tempo do que os relacionados ao RSA [Engelbert et al. 2007].

Porém, nem todas as particularidades deste sistema são benéficas. O tamanho de suas chaves, representadas por grandes matrizes, por exemplo, é consideravelmente maior do que as do RSA para um mesmo nível de segurança, sendo o fator preponderante para sua não empregabilidade em larga escala. Para se ter uma idéia dessa magnitude, [Engelbert et al. 2007] afirma que para encriptação segura, a chave pública deve ser de no mínimo 88KB e para assinaturas seguras, de no mínimo 597KB, números bem acima dos tradicionais 1024–2048 bits do RSA. Atualmente, existem alguns estudos objetivando a redução destes valores.

Na Tabela 3.1, temos um comparativo da eficiência algorítmica, custo para a quebra e tamanho de chaves do Sistema McEliece perante o RSA e Criptografia baseada em Curvas Elípticas. Vale ressaltar alguns pontos importantes sobre este comparativo. A ordem de complexidade para as operações do sistema criptográfico McEliece, quadrática, é de fácil entendimento, uma vez que os passos computacionais que efetuam as operações se resumem a multiplicações matriz por vetor. Com relação ao custo de quebra, os algoritmos de ordem cúbica para o RSA e ECC são os algoritmos quânticos propostos por Peter W. Shor, citados no início desta monografia. Para algoritmos clássicos, esta ordem de complexidade permanece exponencial. Por fim, o tamanho das chaves do McEliece e de ECC está adequado ao nível de segurança do RSA 1024 bits.

Tabela 3.1. Eficiência Algorítmica: McEliece x RSA x ECC

	RSA	ECC	McEliece
Custo das operações	$O(n^3)$	$O(n^3)$	$O(n^2)$
Custo para quebra	$O(n^3)$	$O(n^3)$	$O(2^n)$
Tamanho das Chaves	1024 bits	160 bits	88 KB

Outro fator que influencia diretamente na eficiência e no nível de segurança do Sistema McEliece se refere a escolha do Código utilizado. A versão original, proposta por R.J. McElice em 1978, sugere os Códigos de Goppa e se mantém inquebrada até o momento. Apesar desta eficiente versão original, foram propostas inúmeras modificações no sistema McEliece, principalmente relacionadas à família de código utilizado. Niederreiter propôs em 1986 um sistema ([Niederreiter 1986]) que utilizasse, ao invés de Códigos de Goppa, códigos GRS. Porém, em [Sidelnikov and Shestakov 1992] foi mostrado que esta escolha torna o sistema inseguro. Neste contexto, é preciso saber identificar o que seria um código classificado como *bom*. Como dito anteriormente, os Códigos de Goppa são bastante indicados para fins criptográficos, porém, ainda restaria decidir por seus parâmetros n, k, e t, que têm as escolhas limitadas pelo seguinte teorema:

Teorema 2 ([Voloch 1987]) Seja $C - (n, k)$ um código linear de peso mínimo d sobre $\mathbb{F}_2$. Então $d + k \leq n + 1$.

Isso implica que, de certa forma, deve-se optar entre eficiência e capacidade de correção de erros. Segundo [Menezes et al. 1996], uma boa escolha para os parâmetros do código do Sistema McEliece é $n = 1024$, $t = 38$ e $k \geq 644$. A Tabela 3.2 sugere alguns parâmetros relacionados ao nível de segurança.

Tabela 3.2. Parâmetros x Nível de Segurança

m	t	Passos Computacionais para a Quebra
11	64	2^{101}
12	128	2^{186}

A seguir, temos a definição dos algoritmos do Sistema McEliece, em sua versão original.

1. Geração de chaves

O algoritmo de geração de chaves pode ser dividido em duas etapas: A primeira se refere à escolha do código a ser utilizado e à representação desse código através de uma matriz geradora, já a segunda, à tarefa de disfarçar este código, multiplicando sua matriz geradora por uma matriz aleatória inversível e por uma matriz aleatória de permutação.

1. Escolha do Código:
   (a) Escolher um Código de Goppa $G = (L, g(X))$ capaz de corrigir até t erros.
   (b) Obter a matriz geradora $k \times n$ para este código G.
2. Disfarce do Código:
   (a) Gerar uma matriz S binária inversível $k \times k$ aleatória.

(b) Gerar uma matriz de permutação P $n \times n$ aleatória.
(c) Calcular: $E = SGP$.

Assim, as chaves seriam:

- Chave Privada: (P^{-1}, G, S^{-1});
- Chave Pública: (E, t).

2. Encriptação

Para a encriptação de uma mensagem m, de tamanho k, em um vetor c, de tamanho n, basta multiplicar m pela matriz geradora do código e, então, se aproveitando do fato de estarmos lidando com um Código capaz de corrigir até t erros, adicionar t erros aleatórios nessa palavra de código:

1. Obter m', de tamanho n, como sendo $m' = mE$.
2. Gerar um vetor aleatório e, de peso t e tamanho n, e somá-lo a m', obtendo assim o vetor $c = m' + e$.

3. Decriptação

Para a decriptação de um código c de tamanho n, temos os seguintes passos:

1. Obter $c' = cP^{-1} = (mSGP + e)P^{-1} = mSG + eP^{-1}$.
2. Tomando $m' = mS$ e $e' = eP^{-1}$, corrigir o erro e' em $m'G + e'$, obtendo mSG.
3. Decodificar mSG em mS.
4. Obter $m = (mS)S^{-1}$.

4. Sistema de Assinatura Digital (CFS)

Outra demanda criptográfica que até recentemente estava em aberto, refere-se a prática de assinatura digital, uma das aplicações mais comuns no ramo da segurança da informação, e que o Sistema Criptográfico McEliece não dispunha. Porém, em 2001, Courtois, Finiasz, e Sandrier apresentaram o Sistema de Assinatura Digital CFS [Courtois et al. 2001], que por ser baseado no Sistema Criptográfico McEliece também é classificado como Pós-Quântico e que preenche está lacuna que representava outro fator significativo para a não empregabilidade do Sistema McEliece.

Por definição, um sistema de assinatura digital deve prover uma maneira de assinar qualquer documento de maneira que identifique unicamente seu autor e que disponha de um algoritmo público eficiente de verificação de assinatura. No âmbito do *CFS*, para estas tarefas e assim como no McEliece, deve ser escolhido um código linear, que será chamado de C na explicação. Então, o *CFS* usa uma função de hash pública para resumir o documento m a ser assinado no vetor $h(m)$. O próximo passo é tratar esse resumo como uma palavra de código, ou seja, aplicar a ele o algoritmo de decriptação do código escolhido, obtendo um vetor c', correspondendo a assinatura da mensagem m. Para a verificação da assinatura, basta encriptar c', recebido junto da mensagem m, e verificar se corresponde ao cálculo do hash da mensagem m.

4.1. Geração de Chave

1. Escolher um Código de Goppa $G(L, g(X))$.
2. Obter sua matriz $(n-k) \times n$ de verificação de paridade H.
3. Calcular $V = SHP$, onde S é uma matriz binária inversível $(n-k) \times (n-k)$ aleatória.
4. Calcular P uma matriz $n \times n$ aleatória de permutação.

Assim, as chaves seriam:

- Chave Privada = G.
- Chave Pública = (V, t).

4.2. Assinatura

1. Encontrar o menor $i \in \mathbb{N}$ tal que, para $c = h(m, i)$ e $c' = S^{-1}c$, c' seja uma síndrome decodificável de G.
2. Usando o algoritmo de decodificação de G, obter o vetor de erros e', cuja síndrome seja c', ou seja $c' = H(e')^t$.
3. Obter $e^t = P^{-1}(e')^t$. Assim, a assinatura é o par: (e, i).

4.3. Verificação de Assinatura

1. Obter $c = Ve^t$.
2. Aceite somente se $c = h(m, i)$.

Capítulo 4

Implementação

Como aplicação desta teoria, desenvolvemos uma Biblioteca Criptográfica Pós-Quântica, ou seja, um software capaz de gerar, importar e exportar pares de chaves, além de encriptar e decriptar mensagens, possibilitando assim uma maneira prática de operar criptograficamente sob os preceitos enunciados no texto. Para estas tarefas, foram disponibilizadas diferentes configurações de parâmetros do Sistema Criptográfico McEliece, visando uma adequabilidade a diferentes situações que o usuário possa desejar. Outro aspecto considerado nesta implementação, refere-se a disponibilização de testes para a própria Biblioteca, objetivando a prova de que o conceito foi implementado com sucesso.

Como dito anteriormente, a biblioteca foi codificada em linguagem de programação Java. Esta escolha, apesar de soar um pouco contraditória para um sistema que tem como uma de suas principais qualidades a eficiência algorítmica, se mostrou como um ótimo otimizador de produtividade. Por exemplo, ao utilizar o Java, foi possível a utilização de uma API (de Application Programming Interface - ou Interface de Programação de Aplicativos) para a codificação da interface visual. A API em questão é a Swing [Sun Microsystems 2008b], e fornece meios bastante práticos no desenvolvimento de janelas, menus, entre outros componentes visuais. Outra facilidade advinda da escolha do Java como linguagem de codificação, refere-se à ferramenta de documentação chamada Javadoc [Sun Microsystems 2008a]. Com esta ferramenta foi possível gerar uma documentação sobre as classes e métodos do projeto de maneira simples e organizada. Como resultado, obtivemos a documentação em HTML, disponível no diretório *doc* do projeto.

1. Organização do Projeto

O projeto que compõe a Biblioteca Criptográfica Pós-Quântica está disposto seguindo a hierarquia ilustrada pela Figura 4.1. Os pacotes e classes principais estão descritas a seguir.

- Pacote *gui*: Este pacote contém as classes utilizadas pelo framework *Swing* do Java e que compõem a interface gráfica ao usuário. As classes terminadas em *Frame* representam as janelas da aplicação (Exemplo: *UsingLibFrame* - janela de instruções para a utilização da Biblioteca). Já as classes terminadas em *Panel* representam aos painéis internos à janela principal.
- Pacote *math*: Este pacote contém as classes que implementam as estruturas algébricas necessárias ao sistema, tais como Anel Polinmonial (*PolynomialRing*), Corpo Finito

(*FiniteField*), Código de Goppa (*GoppaCodes*), além de estruturas de dados básicas como matriz e vetor.
- Pacote *mceliece*: Este pacote contém as classes relativas ao Sistema McEliece, como as que representam as chaves, e as destinadas à encriptação e decriptação de mensagens.
- Pacote *tests*: Pacote que contém as classes de testes (as quais são acessadas na interação por linha de comando).

A seguir, apresentamos uma breve descrição de cada uma das classes do projeto, sugerindo as suas funcionalidades e o que representam.

1. Pacote *gui*:
    - *AboutFrame*: Janela de informações sobre a Biblioteca
    - *ConfigContentPanel*: Painel de configurações
    - *ContentPanel*: Painel de conteúdo principal
    - *DecryptContentPanel*: Painel de decriptação
    - *EncryptContentPanel*: Painel de encriptação
    - *ExportContentPanel*: Painel de exportação de par de chaves
    - *FootPanel*: Painel de rodapé
    - *GenerateKeysContentPanel*: Painel de geração de par de chaves
    - *ImportContentPanel*: Painel de importação
    - *LibInterface*: Janela principal
    - *MenuKeysPanel*: Painel de menu de par de chaves
    - *MenuPanel*: Painel de menu principal
    - *TopPanel*: Painel de título
    - *UsingLibFrame*: Janela de de instruções para utilização da Biblioteca
    - *UtilsGui*: Classe que provê métodos úteis a toda a interface gráfica
2. Pacote *math*:
    - *BasicPolynomial*: Representação do polinômio redutor do corpo finito básico. Provê estaticamente a obtenção do polinômio redutor a partir do parâmetro *m*.
    - *FiniteField*: Representação de Corpo Finito.
    - *Goppa*: Representação de um Código de Goppa
    - *Matrix*: Classe que representa uma Matriz com elementos em um corpo finito
    - *Polynomial*: Representação de um Polinômio com coeficientes no corpo base (ou seja, representa um elemento do corpo estendido)
    - *PolynomialRing*: Representação de anel polinomial
    - *Vector*: Classe que representa um Vetor com elementos em um corpo finito
3. Pacote *mceliece*:
    - *Decrypt*: Classe responsável pela tarefa de Decriptação
    - *Encrypt*: Classe responsável pela tarefa de Encriptação
    - *KeyPair*: Representação do par de chaves
    - *PrivateKey*: Representação da chave privada
    - *PublicKey*: Representação da chave pública
4. Pacote *tests*:
    - *Test*: Classe responsável pela execução da bateria de testes
    - *UtilsTests*: Classe que provê métodos úteis aos testes

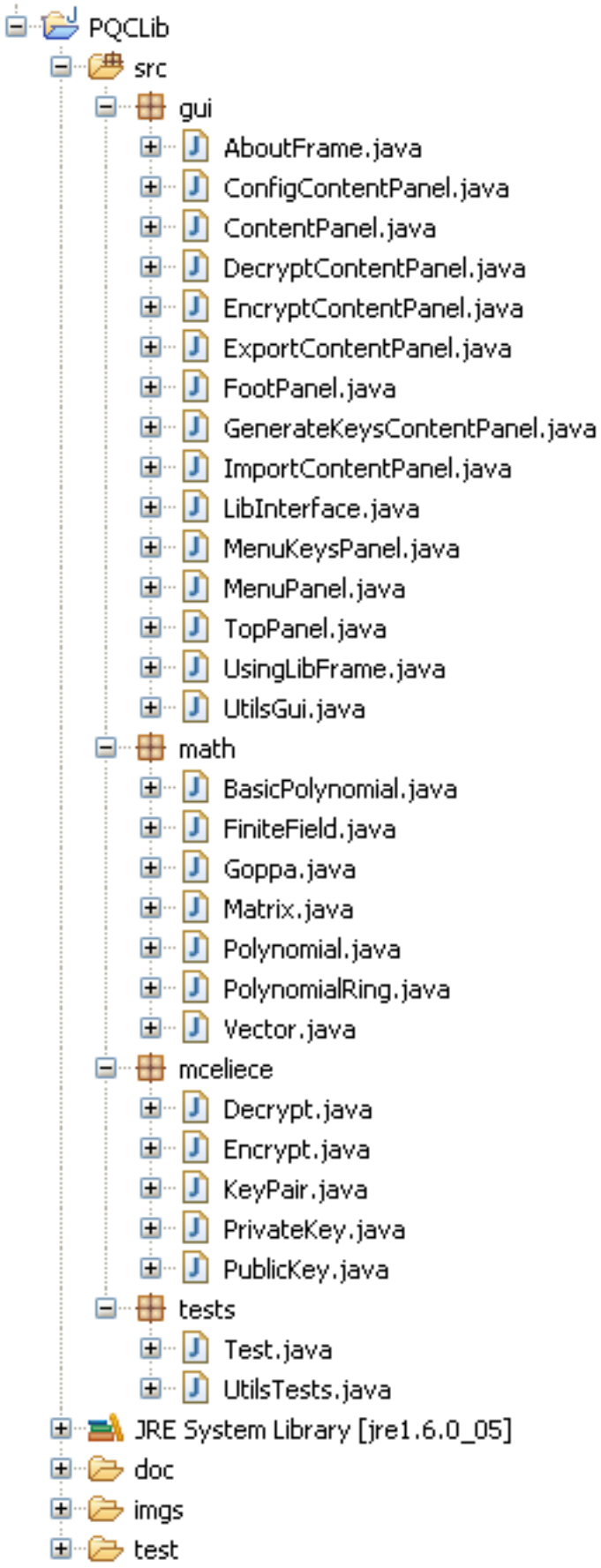

Figura 4.1. Organização do Projeto.

2. Representação computacional de estruturas algébricas

Para a codificação do Sistema Criptográfico McEliece, foi necessário desenvolver abstrações de diversos conceitos algébricos, de forma que o computador pudesse representar as estruturas e simular as operações necessárias. Como ponto de partida, tivemos a implementação do conceito de Corpo Finito e a definição de como representar seus elementos. Dada a característica 2 dos corpos finitos utilizados, poderíamos representar seus elementos como sendo sequências de bits. Porém, como veremos a seguir, algumas operações sobre esses elementos recaem sobre simples operações binárias, tais como a de ou-exclusivo (i.e., *XOR*), o que nos fez considerar o fato de as operações de lógica binária sobre conjuntos de bits serem otimizadas em relação às sobre bit-a-bit. Desta forma, optamos por representar elementos como sendo um vetor de words. Algebricamente, como uma coleção de elementos do subcorpo $\mathbb{F}_{2^m}$ (o qual chamaremos de *corpo base*) de $\mathbb{F}_{2^n}$ com $m \mid n$ e que pode ser representada polinomialmente como:

$$\alpha_0 + \alpha_1 x + ... + \alpha_{q-1} x^{q-1}$$

, com $\alpha_i \in \mathbb{F}_{2^m}$ e $q = n/m$. Como desejávamos deixar o sistema flexível com relação aos seus parâmetros originais (m e t), ou seja, possibilitar ao usuário escolher esses valores, era necessário determinarmos uma faixa de valores que conferissem segurança e/ou praticidade à aplicação. Na Tabela 3.2, temos uma análise entre os valores m e t, considerando o nível de segurança. Com esta análise, e levando em conta as considerações existentes em [Engelbert et al. 2007], onde é apontado que para o Sistema McEliece, corpos base de extensão 12 e, para assinatura digital (CFS), de extensão 16, já nos dão um nível de segurança satisfatório, estipulamos um m máximo igual a 16. Desta forma, foi possível determinar uma estrutura de dados que representasse elementos do corpo base, sem o desperdício excessivo de espaço em memória, nem particularidades para as diferentes opções de parâmetros. Desta forma, um tipo de dado do Java se mostrou bastante pertinente para a representação dos elementos do corpo base: o *short*, que dispõe de exatos 16 *bits*. A Tabela 4.1 mostra os possíveis valores para os parâmetros m e t, disponíveis na Biblioteca Criptográfica Pós-Quântica. Tais parâmetros, acabam por definir implicitamente o tamanho das mensagens codificadas e o tamanho que as mensagens devem ter antes da codificação. Esta relação é dada pelas equações a seguir, onde n representa o tamanho da mensagem codificada e k o tamanho que a mensagem deve ter antes da codificação:

$$n = 2^m$$

$$k = n - m * t$$

Tabela 4.1. Parâmetros disponíveis na Biblioteca

m	t	k	n
4	3	4	16
4	2	8	16
5	4	12	32
8	16	128	256

É importante ressaltar que estes parâmetros não foram implementados visando uma utilização comercial, visto que, com valores tão reduzidos, a abordagem de ataque por busca exaustiva pode ser viável. Entretanto, utilizando estes valores é possível validar facilmente que o sistema funciona. Outro fator que nos desmotivou a empregar parâmetros maiores vem da

condição a seguir. Pensando em um emprego comercial, uma idéia razoável seria utilizar em conjunto com o Sistema McEliece, algum outro sistema de criptografia simétrica, trabalhando da seguinte forma: Ao invés de utilizar o McEliece para encriptar as mensagens (o que não seria recomendável na prática, já que o fator de expansão da palavra de código com relação à mensagem original é considerável), o usuário as encriptaria com o sistema simétrico e utilizaria o McEliece apenas para encriptar a chave simétrica. Desta forma, a mensagem codificada não seria muito grande (em relação à mensagem original) e ainda assim desfrutaria da segurança contra ataques quânticos, visto que a chave estaria protegida. Como julgamos que esta funcionalidade fugiria ao escopo da nossa prova de conceitos e demandaria tempo considerável, mantivemos o Sistema Criptográfico McEliece de maneira pura, mas sem suporte a parâmetros comerciais.

Com a estrutura de dados para armazenar os elementos do corpo base definida, precisávamos determinar qual representação algébrica nos basearíamos. Optamos por utilizar a abstração da representação polinomial, fato este que demanda a determinação de um polinômio redutor para que as operações no corpo base pudessem ser simuladas. Como o corpo base não deve ter extensão maior do que 16, esta determinação é rápida, testando a primitividade dos possíveis trinômios mônicos com termos independentes não nulos, ou seja, polinômios da forma:

$$x^p + \cdots + x^r + \cdots 1, para 16 < q < r$$

Dado o subconjunto reduzido de possibilidades, optamos por não efetuar este cálculo na própria biblioteca, deixando 15 possibilidades (para $m = 1$, $m = 2$, $\cdots$, $m = 15$) no código. Desta forma, testamos a primitividade dos polinômios com o auxílio da linguagem de álgebra abstrata Magma [Wieb Bosma and Playoust 1997], a qual se mostrou útil em todos os momentos que desejávamos validar os resultados obtidos na Biblioteca. Assim, definido o polinômio redutor $g(x)$, construímos os elementos do corpo base a partir da condição $g(x) = 0$. Exemplificando, para um corpo base de extensão 4, um polinômio redutor seria $x^4 + x + 1$, o que nos dá a informação: $x^4 + x + 1 = 0 \Rightarrow x^4 = x + 1$ e nos possibilita determinar os demais elementos:

$0 = 0$	$x^4 = x + 1$	$x^8 = x^2 + 1$	$x^{12} = x^3 + x^2 + x + 1$
$x = x^1$	$x^5 = x^2 + x$	$x^9 = x^3 + x$	$x^{13} = x^3 + x^2 + 1$
$x^2 = x^2$	$x^6 = x^3 + x^2$	$x^{10} = x^2 + x + 1$	$x^{14} = x^3 + 1$
$x^3 = x^3$	$x^7 = x^3 + x + 1$	$x^{11} = x^3 + x^2 + x$	$x^{15} = 1$

Assim, a representação binária destes elementos é:

$0 = 0000$	$x^4 = 0011$	$x^8 = 0101$	$x^{12} = 1111$
$x = 0010$	$x^5 = 0110$	$x^9 = 1010$	$x^{13} = 1101$
$x^2 = 0100$	$x^6 = 1100$	$x^{10} = 0111$	$x^{14} = 1001$
$x^3 = 1000$	$x^7 = 1011$	$x^{11} = 1110$	$x^{15} = 0001$

3. Algoritmos implementados

Na implementação deste sistema, foi necessária a utilização de alguns algoritmos alheios ao McEliece, mas que eram necessários para o computo de passos intermediários. Alguns deles são bem conhecidos, tal como o Algoritmo Estendido de Euclides, outros nem tão célebres, como o Teste de Irredutibilidade de Ben-Or. A seguir, apresento a descrição desses algoritmos e a indicação de onde foram necessários, além da razão para seu emprego.

3.1. Algoritmo Estendido de Euclides

A idéia deste algoritmo foi utilizada, com algumas variações, em três processos da Biblioteca. No primeiro caso, para o cômputo do inverso polinomial módulo um outro polinômio (presente na classe *Polynomial*, no método *getInverseModPolynomial()*). A seguir temos o pseudo-código implementado:

Algorithm 1 Algoritmo Estendido de Euclides ($f(x)^{-1} \bmod g(x), f, g \in K[X]$)

INPUT: f ▷ Polinômio $f(x)$.
INPUT: g ▷ Polinômio $g(x)$.
OUTPUT: $f(x)^{-1} \bmod g(x)$

1: $F \leftarrow f, F \leftarrow f, G \leftarrow g, B \leftarrow 1, C \leftarrow 0$
2: **while** $degree(F) < 0$ **do**
3: **if** $degree(F) < degree(G)$ **then**
4: $F \leftrightarrow G, B \leftrightarrow C$
5: $j \leftarrow degree(F) - degree(G), h \leftarrow F_{deg(F)}/G_{deg(G)}$
6: $F \leftarrow F - hx^jG, B \leftarrow B - hx^jC$
7: **end if**
8: **end while**
9: **if** $F \neq 0$ **then**
10: **return** B/F_0
11: **else**
12: **return** "Não inversível"
13: **end if**

O segundo caso de uso, refere-se a verificação se dois polinômios são coprimos, o que é equivalente a testar se o MDC (máximo divisor comum) entre eles é diferente de um. A seguir temos o pseudo-código da implementação presente em *Polynomial*, no método *Coprime()*.

Algorithm 2 Algoritmo Estendido de Euclides (se dois polinômios são coprimos)

INPUT: f ▷ Polinômio $f(x)$.
INPUT: g ▷ Polinômio $g(x)$.
OUTPUT: $f(x)coprimodeg(x)$

1: $F \leftarrow f, G \leftarrow g$
2: **while** $degree(F) < 0$ **do**
3: **if** $degree(F) < degree(G)$ **then**
4: $F \leftrightarrow G$
5: **end if**
6: $j \leftarrow degree(F) - degree(G)$
7: $\lambda \leftarrow F_{degree(F)}/G_{degree(G)}$
8: $F \leftarrow F + \lambda * x^j * G$
9: **end while**
10: **return** $F \neq 0$

O terceiro caso referente à utilização do Algoritmo Estendido de Euclides, aplica-se à determinação do Polinômio Localizador de Erros. Este passo está presente na classe *Decrypt*, no método *ErrorLocator()*. A seguir, apresentamos seu pseudo-código.

Algorithm 3 Algoritmo Estendido de Euclides (Polinômio Localizador de Erros)

INPUT: s ▷ Polinômio $s(x)$.
INPUT: g ▷ Polinômio $g(x)$.
OUTPUT: *polinomiolocalizadordeerros*

1: $v(x) \leftarrow \sqrt{x_1/s(x)} \bmod g(x)$
2: $F \leftarrow v, B \leftarrow 1, C \leftarrow 0, t \leftarrow degree(g)$
3: **while** $degree(F) < \lfloor t \rfloor$ **do**
4: **if** $degree(F) < degree(G)$ **then**
5: $F \leftrightarrow G, B \leftrightarrow C$
6: **end if**
7: $j \leftarrow degree(F) - degree(G), h \leftarrow F_{degree(F)}/G_{degree(G)}$
8: $F \leftarrow F - /G_{degree(G)}$
9: $F \leftarrow F - hx^jG, B \leftarrow B - hx^jC$
10: **end while**
11: $\sigma(x) \leftarrow G(x)^2 + xC(X)^2$
12: **return** σ

3.2. Teste de Irredutibilidade de Ben-Hor

Para a geração do código de Goppa, é preciso obter um polinômio que, entre outras características, seja irredutível. Para realizar este teste de irredutibilidade, implementamos o algoritmo de Ben-Or, que pode ser analisado em [Gao and Panario 1997], onde é apontado como o mais eficiente para esta tarefa. Tal algoritmo, apesar de complexo em um primeiro momento, recai apenas a chamadas de *MCD* (maior divisor comum), *mod* (resto da divisão de dois polinômios), e o que é conhecido como *Endomorfismo de Frobenius* (elevar um polinômio sobre $\mathbb{F}_{2^m}$ a uma potência de 2^m). Este teste está implementado na classe *Polynomial*, no método *isIrreducible()*, e tem o seu pseudo-código apresentado a seguir.

Algorithm 4 teste de Irredutibilidade de Ben-Or

INPUT: g ▷ Polinômio $g(x)$.
OUTPUT: *polinomioeirredutivel*

1: $K \leftarrow CoefficientRing(g)$
2: $m \leftarrow degree(K), q \leftarrow \#K, t \leftarrow degree(g)$
3: $y \leftarrow x^{2^m} \bmod g$ ▷ $EndomorfismodeFrobenius$
4: $z[] \leftarrow y^i mod g, 0 <= i < t$
5: $v \leftarrow x$
6: **for** $j \leftarrow 1$ **to** $tdiv2$ **do**
7: **for** $i \leftarrow 1$ **to** $\#v$ **do**
8: $w \leftarrow w + v[i] * z[i]$
9: **end for**
10: $v \leftarrow w$
11: **if** $notCoprime((v - x) mod g, g)$ **then** ▷ $UtilizaoAlgoritmo2$
12: **return** $false$
13: **end if**
14: **end for**
15: **return** *true*

4. Otimizações

Como uma das maiores preocupações deste projeto, tivemos a busca por implementações e abordagens eficientes das operações necessárias ao funcionamento do Sistema Criptográfico McEliece. Desta forma, idealizamos a representação das estruturas algébricas de maneira que as operações fossem as mais otimizadas possíveis. A seguir, apresentamos as otimizações implementadas.

4.1. Soma de elementos no corpo básico como operação de Ou-Exclusivo

Como exemplo desta preocupação, temos que, dada a representação dos elementos do corpo base ser da maneira explicada em 2., é possível nos aproveitarmos de uma interessante propriedade: a soma de elementos desses corpos pode ser perfeitamente executada como um XOR entre os bits que os representam. Isto é devido à a característica 2 de seus coeficientes. A seguir, alguns exemplos retirados do corpo finito calculado na seção 2 e que demonstram esta propriedade:

$$x^4 + x^{14} = (x + 1) + (x^3 + 1) = x^3 + x + 2 =$$
$$= x^3 + x + 0 = x^3 + x = x^9$$

Realizando o mesmo cálculo, porém com a representação binária e o XOR no lugar da soma:

$$0011 \oplus 1001 = 1010$$

Que corresponde a $x^3 + x = x^9$. Outro exemplo:

$$x^6 + x^10 = (x^3 + x^2) + (x^2 + x + 1) = x^3 + 2x^2 + x + 1 =$$
$$= x^3 + 0x^2 + x + 1 = x^3 + x + 1 = x^7$$

Realizando o mesmo cálculo, porém com a representação binária e o XOR no lugar da soma:

$$1100 \oplus 0111 = 1011$$

Que corresponde a $x^3 + x + 1 = x^7$.

4.2. Multiplicação e inverso multiplicativo no corpo base

Para as outras operações aritméticas, o cenário é um pouco mais complexo, mas, ainda assim, foi possível empregar uma solução simples e eficiente. Para o caso da multiplicação e da determinação do inverso multiplicativo, utilizamos a abordagem apresentada por [Win et al. 1996], o qual sugere que as operações possam ser resumidas a simples consultas a tabelas pré-calculadas. Para isso, a partir de um gerador γ, deve-se determinar todos os pares (α, i) tais que $\alpha = \gamma^i$ ($\alpha \in \mathbb{F}_{2^m} \setminus \{0\}$; $0 \leq i < 2^m - 1$). Estes pares devem ser armazenados em duas tabelas: uma denominada de log[], ordenada em α, e uma de exp[], ordenada em i. Assim, o produto dos elementos $\alpha, \beta \in \mathbb{F}_{2^m} \setminus \{0\}$ e a inversão de α seria obtido a partir de:

$$\alpha\beta = \exp[(\log[\alpha] + \log[\beta]) \bmod (2^m - 1)]$$

$$\alpha^{-1} = \exp[-\log[\alpha] \bmod (2^m - 1)]$$

Porém, seguindo esta idéia, ainda nos resta o problema de determinar o gerador γ utilizado. Para essa tarefa, tomamos proveito da seguinte propriedade algébrica: caso tenhamos o polinômio

redutor como sendo um polinômio primitivo, podemos determinar um gerador diretamente, o g(x) = x. Vale ressaltar que estas operações estão definidas sobre o corpo base ($\mathbb{F}_{2^m}$).

4.3. Operações no corpo estendido

Para as operações no corpo estendido ($\mathbb{F}_{2^n}$), deve-se levar em conta o grau do termo no polinômio representativo dos elementos de $\mathbb{F}_{2^n}$. Por exemplo, para o caso da soma de $\alpha_0+\alpha_1 x+...+\alpha_{q-1}x^{q-1}$ com $\beta_0 + \beta_1 x + ... + \beta_{q-1}x^{q-1}$, deve-se somar apenas os termos (pertencentes a $\mathbb{F}_{2^m}$) que tenham mesmo grau em $\mathbb{F}_{2^n}$, tal como α_1 e β_1, assim por diante.

$$\alpha_0 + \alpha_1 x + ... + \alpha_{q-1}x^{q-1} + \beta_0 + \beta_1 x + ... + \beta_{q-1}x^{q-1} =$$
$$(\alpha_0 + \beta_0) + (\alpha_1 + \beta_1)x + ... + (\alpha_{q-1} + \beta_{q-1})x^{q-1}$$

Para o caso da multiplicação, a idéia é análoga a de multiplicação convencional de polinômios: em $\left(\alpha_0 + \alpha_1 x + ... + \alpha_{q-1}x^{q-1}\right) * \left(\beta_0 + \beta_1 x + ... + \beta_{q-1}x^{q-1}\right)$, o resultado de $\alpha_i * \beta_j$ deve ser armazenado no coeficiente do termo de grau $i + j$.

4.4. Quadrado das somas como soma dos quadrados

Outra abordagem eficiente trata da operação quadrática de um polinômio. Ao invés de multiplicar o polinômio por si mesmo, levamos em conta o fato de que "o quadrado de somas é igual a soma dos quadrados", dado o fato das operações serem realizadas módulo 2. Com isso, os termos cruzados múltiplos de 2 são descartados, como ilustrado no exemplo a seguir.

$$(g(x) + z(x))^2 = g(x)^2 + 2g(x)y(x) + y(x)^2 =$$
$$= g(x)^2 + 0g(x)y(x) + y(x)^2 = g(x)^2 + y(x)^2$$

Formalizando, temos:

$$\left(\sum_{i=0}^{q-1} \alpha_i x^i\right)^2 = \sum_{i=0}^{q-1} \alpha_i^2 x^{2i}$$

Além destas operações, o restante da codificação se traduziu em manipulações (geração aleatória, adição, multiplicação e inversão) de matrizes e vetores binários, que podem ser aplicadas diretamente das definições dos algoritmos enunciados no texto.

5. Utilizando a Biblioteca Criptográfica Pós-Quântica

A Biblioteca Cirptográfica Pós-Quântica dispõe de duas interfaces: uma gráfica e outra por linha de comando. A primeira tem o objetivo de prover um acesso intuitivo às funcionalidades do sistema. Já a segunda é destinada a validação de suas operações básicas, tais como geração de par de chaves, encriptação e decriptação.

5.1. Interface Gráfica

A interface gráfica da Biblioteca foi desenvolvida com o auxílio da API Swing do Java. Por meio dela, a tarefa de criar janelas, painéis (janelas internas a outra janela, porém sem barras de ferramenta), menus, botões, entre outros componentes visuais e controles de interação com o usuário foi bastante facilitada. Para a criação de painéis, por exemplo, bastava estender a

classe *JPanel*, criar um construtor que chamasse o construtor de *JPanel* e um método *initialize* para a definição dos atributos inicias do mesmo. A seguir, temos a codificação resumida de um dos painéis utilizados na Biblioteca (o que disponibiliza a informação sobre o *status* do par de chaves), ilustrando a simplicidade nessa construção.

```
package gui;

import java.awt.Rectangle;

/**
 * Painel de rodapé da Biblioteca.
 *
 * @author Rafael Misoczki
 *
 */
public class FootPanel extends JPanel {

    private static final long serialVersionUID = 1L;

    private JTextField jTextField = null;

    /**
     * Construtor.
     */
    public FootPanel() {

    /**
     * Método de inicialização do painel.
     */
    private void initialize() {

    /**
     * Atualiza o campo de informações sobre o par de chaves.
     */
    public void refreshStatusKeys() {

    /**
     * Obtém campo de informações sobre o par de chaves carregado.
     *
     * @return javax.swing.JTextField
     */
    private JTextField getJTextField() {

}
```

Figura 4.2. Exemplo de Painel do Swing.

A seguir, apresentamos a simulação de interação entre um usuário e a Biblioteca, por intermédio da interface gráfica criada.

Tela Inicial

A tela inicial da Biblioteca, disponibiliza em seu menu principal, apenas o acesso à seção de Par de Chaves, pois tanto a encriptação quanto a decriptação não são possíveis sem ter carregado em memória algum par de chaves. Porém, a partir dos menus superiores, são fornecidos atalhos rápidos para as funcionalidades de exportação/importação de chaves (os quais, no primeiro momento, estão inativos pela mesma razão que os botões de encriptação e decriptação), além de acesso à tela de configuração da Biblioteca, janela de ajuda e de informações sobre a Biblioteca.

Figura 4.3. Tela inicial da Biblioteca Criptográfica Pós-Quântica.

Configuração do Sistema

Antes da geração de um par de chaves, é interessante atentar às configurações do sistema, acessadas através do menu *Ferramentas / Opções*. Nesta tela, estão presentes as opções de parâmetros do Sistema McElice, tais como os valores de m e t. Como explicado em 2., estes parâmetros devem seguir algumas relações invariantes que definem o tamanho do bloco a ser encriptado (mensagem original) e do bloco codificado. Como parâmetros iniciais, temos a seguinte configuração:

Tabela 4.2. Parâmetros inciais da Biblioteca

m	t	k	n
4	3	4	16

É importante ressaltar que configurações distintas não podem atuar em um mesmo processo de criptografia. Por exemplo, utilizar uma chave gerada com os parâmetros $m = 4$, $t = 3$ para a encriptação de uma mensagem com tamanho de bloco maior do que 4. Por este fato, ao ter uma configuração nova salva na memória da Biblioteca, a mesma descarta qualquer par de chaves que estava anteriormente carregado.

Geração de Par de Chaves

Para a geração de chaves, é necessário informar uma descrição para o par de chaves, a fim de poder identificá-lo na importação/exportação de par de chaves, ou até mesmo durante o seu próprio uso, já que no rodapé da Biblioteca é informada a descrição do par de chaves carregado em memória. Para gerá-las basta então, clicar em *Gerar Par de Chaves* e caso o processamento seja concluído com êxito, a mesma é carregada na memória, viabilizando as demais funcionalidades da Biblioteca.

Figura 4.4. Tela de Geração de Par de Chaves da Biblioteca Criptográfica Pós-Quântica.

Importação e Exportação de par de chaves

A funcionalidade de exportação e importação de chaves provê uma maneira de salvar um par de chaves utilizado, para utilização em algum outro momento. Esta é uma funcionalidade exclusiva da interface gráfica (visto que ainterface por linha de comando visa os testes das operações básicas da biblioteca - geração de par de chaves, encriptação e decriptação) e demandou a criação de um simples protocolo para armazenamento das chaves em arquivo, que pode ser descrito como:

- Descrição do par de chaves
- m,t
- Matriz E, da chave pública

- Matriz H, da chave privada
- Polinômio g, da chave privada
- Matriz S, da chave privada
- Matriz P, da chave privada

Vale ressaltar que as matrizes são impressas binariamente, ou seja, como sequências de 0's e 1's, separados por espaços duplos, e quebra de linha em caso de mudança da linha da matriz. Já o polinômio g imprime os seus coeficientes em forma numérico-decimal, ou seja, pega a sequência binária e calcula o número referente a ela. Estes termos são separados por vírgula.

Encriptação de mensagens

Para encriptar uma mensagem é necessário ter um par de chaves carregado em memória. Uma vez de posse de um par de chaves, para encriptar uma mensagem basta acessar *Encriptar* na tela inicial. Então, será informado qual par de chaves está carregado em memória e que será utilizado para a encriptação. Deve-se informar a mensagem a ser codificada, de tamanho informado na tela e que respeite as configurações selecionadas na tela Opções. Ao clicar em *Encriptar*, será gerado um log do processo de encriptação, além da mensagem encriptada, em caso de sucesso.

Decriptação de mensagens

Para decriptar uma mensagem é necessário ter um par de chaves carregado em memória. Uma vez de posse de um par de chaves, para decriptar uma mensagem basta acessar *Decriptar* na tela inicial. Então, será informado qual par de chaves está carregado em memória e que será utilizado para a decriptação. Deve-se informar a mensagem codificada, de tamanho informado na tela e que respeite as configurações selecionadas na tela Opções. Ao clicar em *Decriptar*, será gerado um log do processo de decriptação, além da mensagem decriptada, em caso de sucesso. É importante ressaltar que, para o sucesso do decriptação, é necessário utilizar o mesmo par de chaves empregado na encriptação da mensagem. No caso da decriptação com sucesso, a tela exibida é análoga a Figura 4.5.

5.2. Linha de comando (Testes)

A execução da Biblioteca Criptográfica Pós-Quântica por meio de linha de comando visa a realização de testes de suas funcionalidades básicas (geração de chaves, encriptação e decriptação). Esta bateria de testes consiste na verificação de mensagens decriptadas, comparando-as com as mensagens originais. Para isso, o sistema recebe via parâmetro de linha de comando (opção *-c número*) ou assume um valor padrão para o número de pares de chaves a serem testados. Para cada par, o sistema testa todas as possibilidades de mensagens a serem encriptadas. Por exemplo, para os parâmetros $m = 4$, $t = 3$ existem 16 possíveis mensagens a serem codificadas, já que o tamanho do bloco para encriptação é de $k = 4$. Através dessa interface também é possível informar outras configurações para o teste, como indica o menu de ajuda exibido ao usuário caso o mesmo informe *-h*. Alguns cuidados devem ser tomados na passagem desses parâmetros, tais como informar valores consistentes de m e t, ou não solicitar impressão de log do processamento interno para testes extremamente exaustivos (a saída resultante poderia ser muito grande).

- -h : Mostra as opçõess de parâmetros
- -l : Loga o processamento interno dos testes
- -f nomeDoArquivo (nome é opcional) : Indica arquivo de saída dos testes
- -c número : Indica o número de testes (número de pares de chaves a serem gerados)
- -m número : Indica o parmetro *m* do Sistema McEliece
- -t número : Indica o parâmetro *t* do Sistema McEliece

Ao fim da execução da bateria de testes, o usuário é informado se foram detectadas falhas no sistema. Além disso, algumas informações são logados naturalmente, tais como o Polinômio de Goppa de cada uma das chaves, além do resultado, para cada uma das mensagens codificadas se coincidiu a mensagem decriptada.

Figura 4.5. Tela de Decriptação de mensagens da Biblioteca Criptográfica Pós-Quântica.

Capítulo 5

Conclusão

Neste trabalho, descrevemos e implementamos o Sistema Criptográfico McEliece. Tal sistema tem diversas particularidades que o tornam merecedor de estudos mais aprofundados, principalmente quando consideramos a possível viabilidade da computação quântica. Sua aplicabilidade, questionada pelo tamanho de suas chaves, deve ser considerada por muitos motivos. O primeiro, refere-se ao fato de que mesmo sem a evolução da computação quântica, seu uso já é interessante por sua eficiência algorítmica. Outro fator a ser atentado são as recentes pesquisas no intuito de reduzir o tamanho destas chaves. Porém, mesmo melhorando esse aspecto, muito provavelmente nunca serão reduzidas aos níveis do RSA ou ECC, mas isto pode não significar muito nos dias atuais em que, com o crescente barateamento dos dispositivos de armazenamento de dados aliados à melhoria sensível nas velocidades de transmissão de dados entre os sistemas computacionais, mesmo com tamanhos de chaves consideradas atualmente grandes, o sistema poderia ser empregado em diversas situações. Por exemplo, em um ambiente de dispositivos móveis conectado à rede de alta velocidade de transmissão, este sistema poderia se encaixar perfeitamente, trazendo até mesmo, um outro fator positivo. Tais dispositivos geralmente têm recursos de processamento escassos, o que seria perfeitamente indicado para sistemas mais eficientes algoritmicamente. Quanto à implementação deste sistema, concluímos que, apesar do embasamento teórico ser bastante forte para a sua codificação, a implementação em si, apresentou-se não tão complexa. Principalmente quando consideramos as estrutura de dados empregadas, que em suma, não passavam de matrizes e vetores binários.

Parte II

Parte Subjetiva

Capítulo 6

Desafios, relacionamento com o curso e trabalhos futuros

1. Desenvolvimento da pesquisa

Esta pesquisa teve início em Abril de 2008, quando, motivado pela idéia de desenvolver um trabalho na área de criptografia, procurei o Prof. Dr. Paulo S. L. M. Barreto, o qual veio a orientar o trabalho. Sugerindo um trabalho na área de pesquisa de criptografia pós-quântica, o Paulo me apresentou diversos conceitos inovadores, que me atraíram bastante à proposta. Desde o início do projeto, a curva de aprendizado se manteve bastante íngreme, fato este que acredito ter sido determinante para a manutenção de meu interesse durante todo o ano.

Nos primeiros dois meses, trabalhei para adquirir ou (principalmente) relembrar os conceitos necessários para o entendimento do sistema criptográfico McEliece e relacionados à teoria da codificação. Passados esses primeiros meses, comecei a lidar com a linguagem de programação Magma, a fim de esboçar e validar os primeiros traços do sistema McEliece. Neste período também tomei contato com uma implementação do Paulo para este sistema feita em Magma. Apesar de mais parecer um pseudo-código, de tão alto nível que é esta linguagem, isto foi bastante útil no sentido de verificar se estava indo no sentido certo do desenvolvimento.

Passado este período, o Paulo comentou comigo sobre um evento, que ocorreria em Gramado, RS, no mês de Setembro. Tratava-se do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, organizado pela Sociedade Brasileira de Computação, e que teria uma sessão destinada a trabalhos de graduação. Prontamente nos comprometemos a mandar um resumo para os avaliadores. O único problema era que até aquele momento, eu dispunha de poucas coisas documentadas. Na verdade, apenas um esboço da estrutura do texto, indicando quais assuntos seria interessante pesquisar. Logo, aumentei consideravelmente o tempo dedicado ao projeto, focando o *deadline* para a entrega deste texto. Felizmente, conseguimos produzir um resumo de qualidade considerável, já que tivemos o trabalho aceito para publicação e apresentação. Por meio de uma bolsa de produtividade fornecida pelo CNPq ao Paulo, pude ir ao Rio Grande do Sul apresentar o trabalho, juntamente com outros dois alunos da Escola Politécnica, que também são orientandos do Paulo e que tiveram os seus trabalhos aceitos.

Entre a entrega do resumo e a entrega da versão final para o simpósio, passaram cerca de trinta dias e, neste período, trabalhamos na correção de erros e em uma melhor divisão do resumo com relação aos tópicos abordados. Tendo submetido a versão final para o simpósio, já

nos encontrávamos no início de Agosto e a Biblioteca começava a tomar forma. Neste período, já tinha conseguido implementar toda a álgebra do corpo finito básico, além do teste de irredutibilidade. Entretanto, como a viagem se aproximava, também voltei minhas atenções à produção da apresentação para o SBSeg. Além disso, o Simpósio Internacional de IC da USP se aproximava e decidimos tentar submeter um outro resumo. Este era mais enxuto, portanto demandou bem menos tempo do que o para o SBSeg. Tendo preparado a apresentação para o SBSeg e submetido o resumo ao SIICUSP, fui para Gramado. A experiência de apresentar o trabalho no Simpósio foi bastante valiosa, principalmente pelo contato com outros pesquisadores da área de criptografia que trabalham em outras universidades como da UFRGS, da Unicamp, entre outras. Outro ponto positivo para esta viagem foi a experiência de apresentar o trabalho, mesmo que em uma versão preliminar. Isto ajudou muito na apresentação final do trabalho na disciplina de Mac0499 - Trabalho de Formatura Supervisionado.

De volta à São Paulo, retomei a codificação da Biblioteca. Passados cerca de seis semanas, estava com a codificação básica praticamente pronta. Ainda restava fazer uma interface gráfica, classes de testes, documentação, entre outras funcionalidades adicionais, porém, ela já funcionava. Então, a monografia começou a receber mais atenção. Como a apresentação do trabalho à disciplina já se aproximava, comecei a trabalhar adequando a apresentação do SBSeg aos avanços do projeto. Por fim, o apresentei para a sala e conclui a monografia.

2. Desafios e Frustrações

Como dito na seção 1., a apresentação no SBSeg foi bastante valiosa, trazendo a experiência de ter apresentado um trabalho em um simpósio de âmbito nacional, além de conhecimentos e contatos da área. A apresentação no SIICUSP também foi de grande valia. Outra experiência extremamente gratificante durante o projeto, fica por conta de ter trabalhado em conjunto com o professor Paulo. Sempre atencioso, não me deixou sem suporte em nenhum momento. Muito pelo contrário, durante todo este período, mantivemos um contato bastante regular, seja em conversas presenciais ou por mensagens eletrônicas. Muitas das vezes me procurando para me informar e discutir sobre as evoluções da área, acabou por me motivar ainda mais a seguir nesta área. Acredito, sinceramente, que não poderia ter tido uma orientação melhor.

Quanto às frustrações, creio que, principalmente no início do projeto, senti a dificuldade pela enorme quantidade de conhecimento a ser assimilado em pouco tempo. Porém, como dito anteriormente, este foi também um dos fatores que acabaram por me manter motivado. Também tive problemas com relação à busca de referências nesta área. Por mais que procurasse fontes, acabava, muitas das vezes, recaindo sobre os mesmos autores e títulos.

Com relação ao curso do BCC, creio ter aproveitado muito do que ele tem a oferecer. Em oito semestres, sem nenhuma reprovação, aprendi bastante sobre a difícil arte de escalonar muitas tarefas, para prazos curtos. Ainda mais, por ter dividido, em muitos períodos, o meu tempo com outros compromissos, tais como estágio. Não julgo ter sido prejudicado por essa escolha, aliás, acho que me proporcionei uma visão diferente da maioria dos colegas, já tendo uma noção razoável dos dois ambientes possíveis para o futuro: acadêmico e profissional. Porém, não tenho como objetivo me distanciar de nenhum deles, possivelmente, tentar aliar estas duas frentes de maneira produtiva.

3. Relação do projeto com as disciplinas do curso

Para o desenvolvimento deste projeto foram necessários conhecimentos de diversas disciplinas do BCC, sendo, na minha opinião, as de maior relevância:

- *Álgebra I e II*: Estas matérias me apresentaram todo o conteúdo algébrico necessário a compreensão da teoria matemática encontrada na criptografia. Conceitos como corpos finitos, anéis polinomiais, entre outros tão necessários a essa implementação, foram vistos pela primeira vez durante este curso.
- *Lab. prog I e II*: Durante esta disciplina aprendi diversas ferramentas que me foram úteis na construção desse sistema. Muitas delas aumentaram significativamente a minha produtividade como desenvolvedor.
- *Programação orientada a Objetos*: Esta disciplina me fez conhecer de uma maneira mais abrangente o paradigma de orientação a objetos. O projeto desenvolvido utiliza inúmeros conceitos ensinados neste curso.
- *Estruturas de Dados*: Esta disciplina me apresentou algumas das estruturas de dados que mais utilizo. E, durante a codificação do projeto, não foi diferente.
- *Princípios de Desenvolvimento de Algoritmos*: Toda a lógica da computação começa a ser ensinada nesta matéria, portanto, é direta a relação de que ela teve uma influência significativa no processo de desenvolvimento da Biblioteca.

4. Trabalhos futuros

Tenho sérias pretensões em seguir nesta área, desenvolvendo pesquisas mais detalhadas. A área de criptografia pós-quântica ainda é pouco explorada (principalmente, no Brasil) e acredito ser bastante promissora. Quanto à biblioteca em si, penso que seria interessante estendê-la, possivelmente acoplando um sistema criptográfico simétrico, a fim de conferir uma aplicabilidade mais comercial. Outra abordagem seria implementar outras soluções criptográficas pós-quânticas a fim de ter uma base de comparação com relação a eficiência das mesmas.

Referências

Au, S., Eubanks-Turner, C., and Everson, J. (2003). The mceliece cryptosystem. Unpublished manuscript. 5

Courtois, N., Finiasz, M., and Sandrier, N. (2001). How to achieve a mceliece-based digital signature scheme. In *Lecture Notes in Computer Science*, pages 157–174. Springer Berlin / Heidelberg. 13

Engelbert, D., Overbeck, R., and Schmidt, A. (2007). A summary of mceliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1:151–199. 8, 9, 11, 18

Gao, S. and Panario, D. (1997). Tests and constructions of irreducible polynomials over finite fields. 21

Huffman, W. C. and Pless, V. (2003). *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 1st edition. 4, 5

Menezes, A., Oorschot, P., and Vanstone, S. (1996). *Handbook of Applied Cryptography*. CRC Press, 1st edition. 12

Miller, V. (1985). Uses of elliptic curves in cryptography. In *Advances in Cryptology, Crypto 85*, Lecture Notes in Computer Science, pages 417–426. Springer. 2

Niederreiter, H. (1986). Knapsack-type crytosystems and algebraic coding theory. *Prob. Contr. Inform. Theory,*, pages 157–166. 12

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

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In Shor, P. W., editor, *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press. 2

Sidelnikov, V. and Shestakov, S. (1992). On insecurity of cryptosystems based on generalized reed-solomon codes. *Diskretnaya Mat.*, 4(3). 12

Sun Microsystems, I. (2008a). Javadoc tool home page. 15

Sun Microsystems, I. (2008b). Trail: Creating a gui with jfc/swing. 15

Voloch, J. F. (1987). *Códigos Corretores de Erros*. CNPQ, 1st edition. 6, 12

Wieb Bosma, J. C. and Playoust, C. (1997). *The Magma algebra system. I. The user language.* J. Symbolic Comput., 1st edition. 19

Win, E., Bosselaers, A., Vandenberghe, S., Gersem, P., and Vandewalle, J. (1996). A fast software implementation for arithmetic operations in gf(2n). In *Lecture Notes in Computer Science*, pages 65–76. Springer Berlin / Heidelberg. 22