Criptografia Pós-Quântica com Códigos Corretores de Erros

Rafael Misoczki[1], **Paulo S. L. M. Barreto**[2*]

[1]Departamento de Ciência da Computação – Instituto de Matemática e Estatística
Universidade de São Paulo (USP)
São Paulo – SP – Brazil

[2]Depto. de Eng. de Comp. e Sist. Digitais – Escola Politécnica
Universidade de São Paulo (USP)
São Paulo – SP – Brazil

misoczki@linux.ime.usp.br, pbarreto@larc.usp.br

Abstract. *This graduation work describes the concepts relevant to the Post-Quantum Cryptography, i.e., to secure cryptography with respect to Quantum Computers, presenting 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.*

Resumo. *Este trabalho de conclusão de curso descreve os conceitos pertinentes à Criptografia Pós-Quântica, i.e., à criptografia segura perante Computadores Quânticos, apresentando 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, é apresentado o 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.*

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]. Porém, sabe-se que esse tipo de solução é vulnerável a ataques provenientes de computadores quânticos [Shor 1994]. Nesse contexto, é sugerida a utilização do Sistema Criptográfico McEliece [McEliece 1978] que, ao apoiar-se em problemas pertinentes à teoria da codificação (mais especificamente de decodificação de mensagens com erros aleatórios), tem se mostrado seguro

*Orientador do trabalho. Bolsista de Produtividade em Pesquisa CNPq, processo 312005/2006-7.

inclusive neste cenário (não é conhecido algoritmo, seja quântico ou clássico, que resolve tal problema em tempo polinomial), recebendo assim a classificação de "pós-quântico". Seguindo neste sentido, para assinaturas digitais pode ser utilizado o Sistema CFS [Courtois et al. 2001].

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.

Este texto é dividido em: conceituações da Teoria da Codificação, caracterização dos Códigos de Goppa, algoritmos do Sistema McEliece, algoritmos do Sistema CFS e descrição da implementação da Biblioteca Criptográfica Pós-Quântica desenvolvida como aplicação da pesquisa realizada.

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:

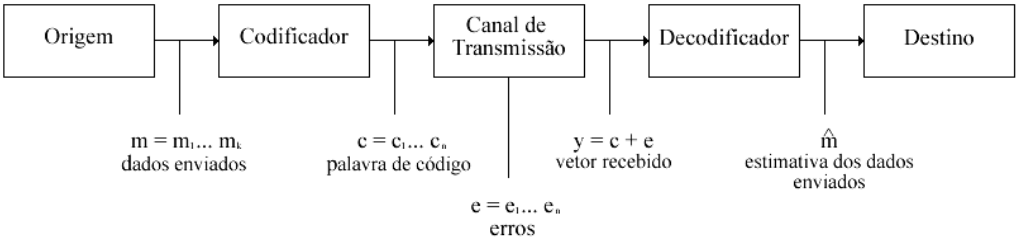

Figura 1. Canal de Comunicação.

2.1. 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, a matriz de dimensões $k \times n$, deverá ter suas k linhas linearmente independentes entre si e seus elementos pertencentes a $\mathbb{F}_2$, gerando assim o código C-(n, k) sobre $\mathbb{F}_2$. Descrevendo o código C em função de uma *matriz geradora* G, teríamos: $C = \{ uG \mid u \in \mathbb{F}^k \}$. A fim de ilustrar os conceitos, utilizaremos a matriz geradora do Código de Hamming-(7,4,3) [Huffman and Pless 2003]:

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

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 $m = [m_1\ m_2\ m_3\ m_4]$ e o código de destino o Código de Hamming-(7,4,3). Assim, a mensagem codificada m' seria da forma: $m' = mG = [\ (m_1 + m_2 + m_3)\ (m_1 + m_2 + m_4)\ m_2\ (m_1 + m_3 + m_4)\ m_3\ m_4\ m_1\ ]$. É fácil perceber que ao ler as componentes: 7ª, 3ª, 5ª e por último, 6ª, determinaremos mensagem original m, ilustrando assim, um método de decodificação [Au et al. 2003] para o Código de Hamming-(7,4,3).

2.2. Métodos de Correção de Erros

Para a tarefa de corrigir códigos com erros, precisaremos das seguintes definições, as quais nos possibilitarão explorar o fato enunciado no Teorema 1:

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.

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.

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, então, das características que as mensagens codificadas têm. Por exemplo, para o método de Máxima Probabilidade, é mantida uma tabela com todas as palavras que podem ser escritas nesse código, associando cada uma delas a todas as possíveis mensagens formadas pela ocorrência de um ou mais erros (até t erros). O Sistema McEliece faz uso do método de Decodificação

de Síndromes, o qual é mais eficiente do que o Máxima Probabilidade, e é apresentado 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.

Exemplo 1. A matriz de verificação de paridade para o Código de Hamming (7, 4, 3) seria:

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

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:

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

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

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

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 $(mG)H = 0$.

4. Sistema Criptográfico McEliece

O Sistema Criptográfico McEliece 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.

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:

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

4.1. 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).

4.2. 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, gerar 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 $c = m' + e$.

4.3. 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 $m = (mS)S^{-1}$.

5. CFS

O *CFS* [Courtois et al. 2001] foi proposto como um Sistema de Assinaturas Digitais baseado no Sistema Criptográfico 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. Para essas tarefas, deve ser escolhido um código linear, ilustrado na explicação como C. Então, o *CFS* usa uma função de hash pública para resumir o documento m a ser assinado no vetor $h(m)$. Decodificando esse resumo com o algoritmo de correção de erros do código escolhido, obtemos 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.

5.1. 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 e P uma matriz de permutação aleatória $n \times n$.

Assim, as chaves seriam: Chave Privada = G, Chave Pública = (V, t).

5.2. 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).

5.3. Verificação de Assinatura

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

6. Implementação

Como aplicação desta teoria, desenvolvemos em linguagem Java uma Biblioteca Criptográfica Pós-Quântica, nos possibilitando uma maneira prática de operar criptograficamente sob os preceitos enunciados no texto. Para esta tarefa, foi necessário desenvolver abstrações de diversos conceitos algébricos, de forma que o computador pudesse 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, as 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 $q = n/m$. Levando-se em conta 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 [Engelbert et al. 2007], 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*.

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 (caso necessário, polinômios de grau quatro também são testados). 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: $0 = 0000$, $x^1 = 0010$, $x^2 = 0100$, $x^3 = 1000$, $x^4 = 0011$, etc. Com esta construção, podemos nos aproveitar de uma interessante propriedade: a soma de elementos desses corpos pode ser perfeitamente executada como um XOR entre os bits que os representam. 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 $\mathbb{F}_{2^m}$. Para generalizá-las em $\mathbb{F}_2^n$ 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. 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 $i + j$. Para a operação quadrática, nos aproveitamos de uma abordagem mais eficiente do que multiplicar o polinômio por si mesmo. Como o quadrado de somas é igual a soma dos quadrados, dado o módulo 2 descartar os termos cruzados, implementamos esse cálculo eficientemente como sugerido na formulação:

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

Como estamos operando sobre corpos finitos, uma implementação de redução modular polinomial se fez necessária. Para isso, implementamos a idéia de que a redução de A mod B atua cancelando a mais alta potência de A, subtraindo um múltiplo de B, até que o grau de A seja menor do que o de B. Para o 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 é simples e recai sobre chamadas a *gcd* (maior divisor comum), *mod* (resto da divisão de dois polinômios), e o que se chama de *Endomorfismo de Frobenius* (elevar um polinômio sobre $\mathbb{F}_{2^m}$ a uma potência de 2^m).

Além destas operações, o restante da codificação se traduz 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.

7. Conclusão

Neste trabalho, descrevemos e implementamos o Sistema Criptográfico McEliece. Este sistema tem diversas particularidades que o tornam merecedor de estudos mais aprofundados. Sua aplicabilidade, questionada pelo tamanho de suas chaves, deve ser considerada, pois, mesmo sem a evolução da computação quântica, seu uso já é interessante por sua eficiência algorítmica. O sistema de assinatura CFS também se apresenta bastante eficiente, dada sua próxima relação com o Sistema McEliece. A implementação destes sistemas se mostrou razoavelmente simples quanto à estrutura de dados e complexidade.

Referências

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

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.

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

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

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

McEliece, R. J. (1978). A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116.

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

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

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

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.

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.

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

Sugiyama, Y., Kasahara, M., Hirasawa, S., and Namekawa, T. (1975). A method for solving key equation for decoding goppa codes. *Information and Control*, 27.

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

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.