**MINISTÉRIO DA DEFESA**
**EXÉRCITO BRASILEIRO**
**DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA**
**INSTITUTO MILITAR DE ENGENHARIA**
**SEÇÃO DE ENGENHARIA DE COMPUTAÇÃO / SE8**

**JONATHAN CORREIA CARVALHOSA**

# APLICAÇÃO DE RETICULADOS EM CRIPTOGRAFIA

**Rio de Janeiro**
**2012**

**INSTITUTO MILITAR DE ENGENHARIA**

**JONATHAN CORREIA CARVALHOSA**

# APLICAÇÃO DE RETICULADOS EM CRIPTOGRAFIA

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Prof. José Antonio Moreira Xexéo, D. Sc.

Rio de Janeiro

2012

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270



CARVALHOSA, Jonathan Correia.

Aplicação de reticulados em criptografia. Jonathan Correia Carvalhosa. - Rio de Janeiro: Instituto Militar de Engenharia, 2012.

Iniciação à Pesquisa – Instituto Militar de Engenharia, 2012.

1. Criptografia
2. Reticulados
3. Computação Quântica
4. RSA
5. ElGamal

**INSTITUTO MILITAR DE ENGENHARIA**

**JONATHAN CORREIA CARVALHOSA**

# APLICAÇÃO DE RETICULADOS EM CRIPTOGRAFIA

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Prof. José Antonio Moreira Xexéo.

Aprovada em 28 de junho de 2012 pela seguinte Banca Examinadora:

_____

Prof. José Antonio Moreira Xexéo – D. C., do IME

_____

Prof. Anderson Fernandes Pereira dos Santos – D. Sc., do IME

_____

Prof. Julio Cesar Duarte – D. C., do IME

Rio de Janeiro

2012

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

# RESUMO

Os principais sistemas criptográficos utilizados na atualidade estão ameaçados pelo desenvolvimento de computadores quânticos, conforme foi demonstrado por Shor em 1993. Este trabalho desenvolve uma pesquisa sobre um criptossistema baseado em reticulado, NTRU, que surgiu como uma excelente alternativa resistente a ataques quânticos. Além disso, apresenta um breve resumo sobre os conhecimentos básicos de criptografia e sistemas antigos vulneráveis a ataques quânticos.

ABSTRACT

The main cryptographic systems used today are threatened by the development of quantum computers, as demonstrated by Shor in 1993. This paper develops a research on a lattice-based cryptosystem, NTRU, which emerged as an excellent alternative, resistant to quantum attacks. It also presents a brief summary of cryptography's basic knowledge and legacy systems vulnerable to quantum attacks.

# 1 INTRODUÇÃO

## 1.1 CONTEXTUALIZAÇÃO

A criptografia é a técnica de modificar uma mensagem a fim de que ela seja inteligível apenas para aqueles que conhecem seu protocolo. Seu desenvolvimento se deve à necessidade de governos e pessoas manterem suas comunicações em sigilo. A criptografia possui duas técnicas básicas: substituição e transposição. A transposição consiste na permutação dos símbolos de um texto, conservando cada um seu significado. Na substituição cada símbolo do texto é trocado por um ou mais símbolos, segundo uma determinada regra pré-estabelecida.

Um reticulado, da mesma forma que um espaço vetorial, é um conjunto de combinações de vetores de coordenadas reais. A partir de uma base de vetores, todos os demais são obtidos através de adições e multiplicações por escalares inteiros. A necessidade dos coeficientes serem inteiros é o que diferencia um reticulado de um espaço vetorial, proporcionando peculiaridades que possibilitam sua aplicação à criptografia.

## 1.2 OBJETIVO

O objetivo principal desse trabalho é realizar uma pesquisa abordando a aplicação de reticulado em criptografia, que possa ser utilizada como fonte de consulta para o aprendizado, tanto de reticulado, como de NTRU – um eficiente criptossistema baseado em reticulado e resistente a ataques quânticos.

Como um complemento à pesquisa, esse trabalho tem como objetivo secundário realizar um estudo mais genérico na área de criptografia, incluindo criptografia quântica, abordando seus conceitos básicos e resumindo alguns criptossistemas baseados em outros tipos de áreas matemáticas: como RSA e ElGamal, que são vulneráveis a ataques quânticos.

## 1.3 MOTIVAÇÃO

A criptografia tem um papel muito importante nas guerras. Na Primeira Guerra, por exemplo, a Alemanha saiu atrás por não possuir um departamento de criptografia e criptoanálise forte como a França. Suas comunicações a rádio eram interceptadas e decifradas, fazendo com que perdessem o elemento surpresa em seus ataques.

De forma a garantir a segurança de suas comunicações na Segunda Guerra, a Alemanha criou a máquina eletro-mecânica “Enigma”, que gerava cifras polialfabéticas através do movimento contínuo de seus rotores. As peculiaridades da Enigma geravam uma segurança tal que desafiou os aliados, dando ampla vantagem aos alemães. Somente quando os aliados conseguiram driblar sua segurança eles foram capazes de vencer a guerra. Isso demonstra como a criptografia foi crucial para definir os rumos dos acontecimentos.

A criptografia, atualmente, tem como principal importância garantir a segurança eletrônica. As empresas precisam esconder suas informações confidenciais, os bancos precisam garantir a segurança das contas de seus clientes, grupos de telefonia e TV a cabo precisam que seus serviços não sejam visíveis para pessoas desautorizadas. Os usuários da internet querem que suas senhas sejam protegidas para que suas contas em e-mails, redes sociais e diversos outros sites não sejam invadidas. A informação tornou-se o bem mais valioso no contexto mundial e qualquer falha em sua segurança pode gerar a perda de milhões de dólares.

O avanço na criptografia permitiu que a internet, que é um meio de comunicação inseguro, seja utilizada para movimentação de contas bancárias, compras através de cartão de crédito e armazenamento de informações extremamente valiosas de países e empresas, uma dinâmica necessária para um mundo onde o tempo é cada vez mais valorizado.

## 1.4 METODOLOGIA

Este trabalho consiste em uma pesquisa sobre a aplicação de reticulados em criptografia e tem como base a busca de informações em livros (principalmente livros relacionados à Criptografia e à Matemática), artigos publicados nessa área e

sites com conteúdo confiável na internet. Nesta seção descreveremos a metodologia seguida para sua realização.

Inicialmente, estudamos os conceitos básicos de criptografia, fundamentais para o desenvolvimento do resto do trabalho. Para reforçar os conceitos, vimos o algoritmo de Diffie-Hellman, revisando alguns conhecimentos de matemática discreta para um melhor entendimento do algoritmo.

Para conhecer mais sobre a criptografia, estudamos dois criptossistemas baseados em problemas parecidos com o do algoritmo de Diffie-Hellman: ElGamal e RSA, revisando as demonstrações de alguns teoremas da álgebra modular, necessários para melhor entendimento dos criptossistemas.

Para perceber a necessidade da aplicação de reticulados na criptografia, estudamos o computador quântico, buscando as informações básicas para a introdução ao tema. Em seguida, lemos sobre a abordagem matemática dos *qubits* e procuramos conhecer melhor os algoritmos quânticos. A parte relativa à Mecânica Quântica e aos circuitos quânticos do computador está presente nas fontes utilizadas, mas foi omitida por não fazer parte do objetivo do trabalho.

Por fim, fizemos uma pesquisa detalhada sobre a aplicação de reticulados em criptografia. Vimos o principal problema difícil baseado em reticulado, o problema de encontrar o vetor mais próximo (CVP - *Closest Vector Problem*), e dois criptossistemas o que usam, o GGH e NTRU. Para um maior embasamento teórico buscando entender suas equações matemáticas, e vimos alguns algoritmos utilizados para tentar quebrá-los.

## 1.5 ESTRUTURA DA MONOGRAFIA

Este trabalho está estruturado em 7 capítulos. O capítulo dois apresenta os conceitos básicos sobre a criptografia, que são definições e principais componentes e uma introdução ao computador quântico.

No terceiro capítulo está a fundamentação matemática necessária para a compreensão dos criptossistemas que serão abordados no trabalho.

O quarto capítulo discorre sobre os primeiros criptossistemas de chave pública: o Algoritmo de Diffie Hellman, o criptossistema de ElGamal e o criptossistema RSA.

O quinto capítulo discorre sobre os criptossistemas baseados em reticulado: o

GGH e o NTRU.

Finalmente, o sexto capítulo apresenta as conclusões e sugestões de trabalhos futuros.

# 2 CONCEITOS BÁSICOS

## 2.1 CRIPTOGRAFIA

Criptografia é a ciência e o estudo da escrita secreta [1]. Seu objetivo é garantir a autenticidade e o sigilo das mensagens trocadas, evitando assim que uma mensagem falsa seja enviada em nome de outra pessoa (autenticidade), ou que seu conteúdo seja lido por pessoas desautorizadas (sigilo). Para isso, as mensagens são transformadas em cifras através de um algoritmo de criptografia, que é controlado por uma ou mais chaves. Para essa cifra ser transformada na mensagem original ela tem que passar por um algoritmo de decodificação, que por sua vez, também é controlado por uma ou mais chaves. O esquema é mostrado na figura abaixo (FIG 0.1).

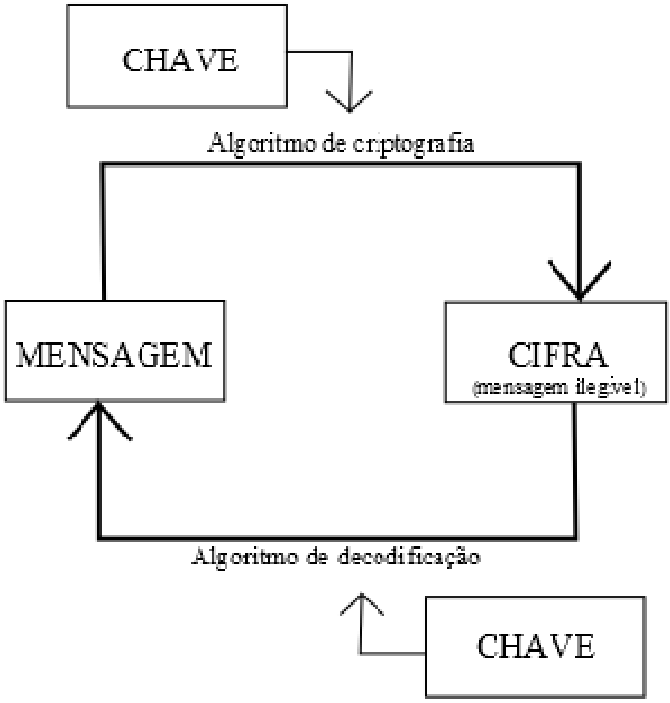


**FIG 0.1 –** Esquema básico de criptografia

Uma chave é uma senha trocada pelos usuários através de uma comunicação segura. Diferentes chaves podem gerar diferentes cifras para uma mesma mensagem e um mesmo algoritmo de criptografia. Para que a criptografia seja

eficiente, um inimigo que intercepte uma cifra não pode ser capaz de decifrá-la sem o conhecimento de todas as chaves, independente da quantidade de cifra que for interceptada e do conhecimento dos algoritmos utilizados. A ciência responsável por tentar decifrar as cifras é a criptoanálise.

Um algoritmo de criptografia é uma função matemática que recebe como entrada a mensagem e as chaves e retorna uma cifra, que é gerada através de transposições e substituições de elementos da mensagem original. O algoritmo de decodificação é a sua função inversa.

## 2.2 SISTEMA CRIPTOGRÁFICO

Um Sistema Criptográfico, ou criptossistema, é o espaço matemático que possibilita o uso da criptografia. Para isso, ele deve ser composto de cinco componentes [1]:

- **O espaço das mensagens:** São as possíveis mensagens que podem ser trocadas. Se uma mensagem for uma conversa entre amigos que só dominam o português, o espaço das mensagens são as palavras da língua portuguesa; se a mensagem indica o valor de um salário, ela tem que ser um número com duas casas decimais. Exemplo: R$ 625, 00;

- **O espaço das mensagens cifradas:** Conjunto gerado por todas as possíveis mensagens, associadas a todos os possíveis algoritmos de criptografia e suas possíveis chaves;

- **O espaço das chaves:** São as possíveis chaves que podem ser associadas aos algoritmos de criptografia ou decodificação. Em um bom sistema criptográfico, os algoritmos devem ser eficientes para todas as chaves, de forma que a escolha da chave não prejudique a eficiência do sistema.

- **Um conjunto de algoritmos de criptografia:** São funções matemáticas

que recebem como entrada a mensagem original e as chaves e retornam a mensagem cifrada. Em um bom criptossistema, é praticamente impossível determinar sua inversa (algoritmo de decodificação), a menos que se conheça o valor das suas chaves.

Essas funções são conhecidas como *trapdoor functions*, um conceito introduzido por Diffie e Hellman [2]. Uma *trapdoor function* é uma função matemática na qual, a partir de certo conhecimento extra, chamado *trapdoor information,* sua função inversa pode ser facilmente calculada. Porém, sem o seu conhecimento seu cálculo é computacionalmente inviável.

Uma *trapdoor function* pode ser comparada a um cadeado, e sua *trapdoor information* à posse de sua chave. O cadeado pode ser fechado facilmente com ou sem a posse da chave, mas sem ela é praticamente impossível abrir o cadeado.

O cálculo da função inversa sem o conhecimento das chaves deve ser um problema matemático difícil, isto é, não deve existir nenhuma solução conhecida em tempo polinomial. Por isso, a segurança do sistema deve estar unicamente no segredo das chaves, e não no segredo do algoritmo. É importante que o algoritmo, associado à suas chaves, possa calcular de forma rápida e fácil as cifras para todas as possíveis mensagens [3].

- **Um conjunto de algoritmos de decodificação:** É a função inversa do algoritmo de criptografia. Dadas a mensagem cifrada e as chaves, o algoritmo de decodificação deve ser capaz de retornar a mensagem original. Para que o sistema seja fácil de usar, é importante que essa função seja facilmente calculada a partir do algoritmo e da chave. Além disso, o conhecimento de qualquer quantidade de pares (mensagem, cifra) não pode ser suficiente para determinar o algoritmo de decodificação [3].

### 2.2.1 SISTEMA SIMÉTRICO

Um sistema criptográfico é dito simétrico, ou de chave privada, quando a chave utilizada para decifrar é a mesma, ou pode ser facilmente deduzida a partir da chave utilizada para cifrar [1]. Com o conhecimento dessa chave, todos os usuários têm o mesmo poder para criptografar e descriptografar as mensagens [3]. Um exemplo de uso de criptografia simétrica é a Cifra de César, em que a chave combinada por César para se comunicar com seus generais era três, o que significava que as letras das palavras da mensagem cifrada eram três letras à frente no alfabeto das letras das palavras da mensagem original. Na figura abaixo (FIG. 2.2) é ilustrado um exemplo do uso da Cifra de César.

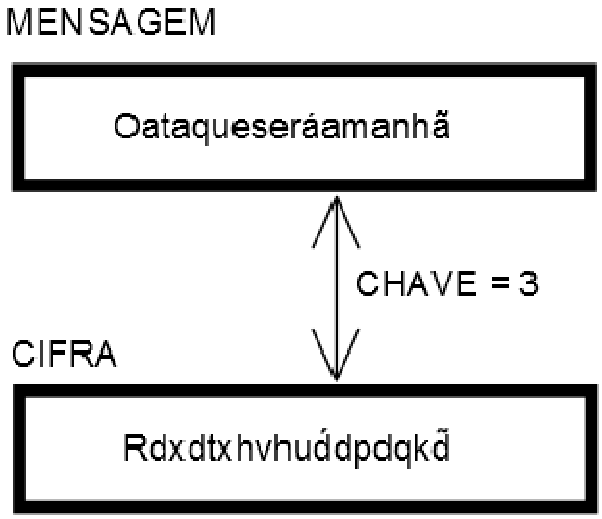


**FIG 2.2 –** Cifra de César

No sistema simétrico, a autenticidade e o sigilo são garantidos pelo segredo da chave, desde que ela seja única para cada par de usuários. Nesse caso, para *n* usuários se comunicarem com privacidade seriam necessárias pelo menos $(n^2 - n)/2$ chaves, o que, para sistemas muito grandes, poderia causar problemas na entrega de todas as chaves através de meios seguros [2]. Caso o grupo optasse por possuir apenas uma chave, as mensagens não poderiam ser lidas por nenhuma pessoa de fora, mas dentro do grupo a autenticidade e o sigilo não poderiam ser garantidos.

### 2.2.2 SISTEMA ASSIMÉTRICO

O conceito de criptografia assimétrica, ou de chave pública, foi introduzido em 1976 por Diffie e Hellman. Nele, a chave deixou de pertencer a um par de usuários para pertencer a cada indivíduo. Além disso, cada pessoa possui duas chaves, uma pública e uma privada. A segurança do sistema criptográfico está unicamente no segredo da chave privada.

A autenticidade e o sigilo são garantidos com a escolha da chave que será usada para criptografar a mensagem. Uma mensagem criptografada com uma chave pública de um usuário só pode ser descriptografada com a chave privada do mesmo usuário, e vice versa.

Na figura abaixo (FIG. 2.3) é apresentado o esquema usado para garantir o sigilo e autenticidade de uma mensagem *M* enviada da pessoa *A* para *B.* $C_A$ e $C_B$ são os algoritmos de criptografia usando, respectivamente, a chave privada de *A* e a chave pública de *B*. $D_A$ e $D_B$ são os algoritmos de decodificação usando, respectivamente, a chave pública de *A* e a chave privada de *B*.

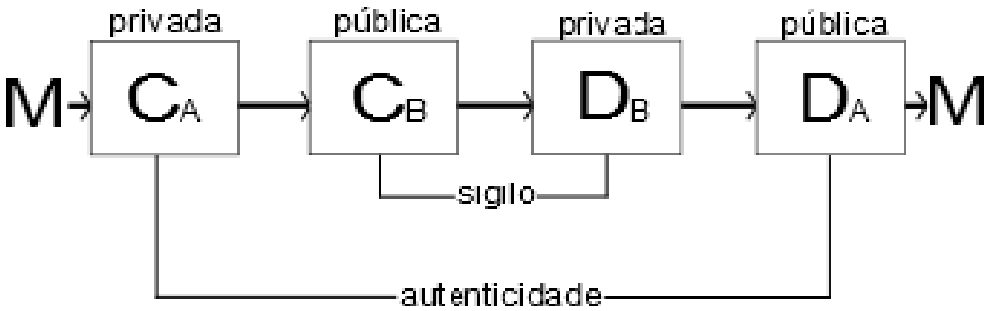


**FIG 2.3 –** Criptografia Assimétrica

Para que sejam garantidos, ao mesmo tempo, a autenticidade e o sigilo de uma mensagem, o esquema da figura FIG. 2.3 deve ser seguido por completo. Caso contrário, só há garantia de um dos dois.

Nesse novo modelo, para *n* usuários se comunicarem com segurança são necessárias apenas *2n* chaves ao todo, o que, para sistemas muito grandes, constitui uma vantagem sobre o método antigo [2].

## 2.3 COMPUTADOR QUÂNTICO

A primeira descrição de um aparato computacional utilizando fenômenos quânticos foi realizada por Paul Benioff em 1980. Mais tarde, David Deutsch criou o primeiro algoritmo quântico e apresentou uma generalização da Máquina de Church-Turing para modelos quânticos, o Computador Quântico Universal, uma prova de que, pelo ponto de vista teórico, a construção de um computador seria possível [4]. Isso impulsionou o crescimento das pesquisas e o maior desenvolvimento da teoria da computação quântica.

O computador quântico é um dispositivo que usa as leis da Mecânica Quântica para processar informação [5]. Sua unidade fundamental é o qubit, ou *quantum bit*, que possui a propriedade conhecida como superposição coerente de estados distintos, que é a capacidade de estar em dois estados distintos ao mesmo tempo. Os qubits, diferentemente dos bits clássicos, podem representar simultaneamente os valores 0 e 1, capacidade que confere ao computador quântico todo o seu poder computacional. Essa propriedade pode ser descrita matematicamente pelo uso de números complexos, conforme veremos mais adiante.

A principal dificuldade encontrada na construção de um computador quântico é a alta incidência de erros, já que em um sistema quântico, qualquer tentativa de medição pode alterar a sua configuração, invalidando os dados armazenados e perdendo, assim, todo o processamento realizado. Essa característica pode ser observada através da experiência conhecida como divisão de raio [6]. Outra dificuldade enfrentada na construção do computador quântico é a forte influência que o meio externo pode ter no computador, alterando assim os seus dados.

A computação quântica ainda é um campo muito novo da ciência. Os computadores quânticos mais potentes atualmente conseguem lidar com poucos qubits de informação e operam em maquinários gigantescos e extremamente caros, exatamente como eram os primeiros computadores da década de 40, que acabaram evoluindo até os computadores dos dias atuais. A principal aplicação do computador quântico seria solucionar problemas que os computadores clássicos não são capazes de resolver de forma eficiente, como Inteligência Artificial e problemas relacionados a sistemas criptográficos bastante utilizados, como a fatoração de números inteiros e o problema do logaritmo discreto em corpos finitos.

### 2.3.1 ANÁLISE MATEMÁTICA DOS QUBITS

Em computação quântica toda a informação é representada através de estados quânticos. O bit clássico é então substituído pelo *quantum bit*, ou qubit, e os valores de 0 e 1 de um bit são substituídos pelos vetores $|0\rangle$ e $|1\rangle$ e representados por:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{e} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Diferentemente de um bit, que nunca pode representar simultaneamente os valores 0 e 1, um qubit genérico $|\psi\rangle$ pode também ser uma combinação linear dos vetores $|0\rangle$ e $|1\rangle$, ou seja:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ onde } \alpha \text{ e } \beta \in C$$

e $C$ é o conjunto dos números complexos

Os vetores $|0\rangle$ e $|1\rangle$ constituem uma base ortonormal do espaço vetorial $C^2$. Essa base é chamada base computacional e o vetor $|\psi\rangle$ é chamado de superposição dos vetores $|0\rangle$ e $|1\rangle$, com amplitudes $\alpha$ e $\beta$ [7]. Em Mecânica Quântica, vetor é também chamado de estado.

A interpretação física do qubit é que ele está simultaneamente nos estados $|0\rangle$ e $|1\rangle$. Toda a informação nele contida está no nível quântico, e para torná-la acessível no nível clássico é necessária a realização de uma medida. A mecânica quântica diz que o processo de medida altera o estado do qubit, fazendo-o assumir o estado $|0\rangle$, com probabilidade $|\alpha|^2$ ou o estado $|1\rangle$, com probabilidade $|\beta|^2$. Essa é uma das grandes dificuldades da construção de um computador quântico, já que a realização de uma medida altera o estado do qubit e não revela os valores de $\alpha$ e $\beta$.

A partir da interpretação física, é possível extrair uma importante equação matemática para o qubit. Sabendo que a soma de todas as probabilidades tem que ser unitária, podemos concluir que:

$$|\alpha|^2 + |\beta|^2 = 1$$

Portanto, $|\psi\rangle$ pode ser interpretado matematicamente como um vetor de $C^2$ cuja norma é unitária.

### 2.3.2 PRINCIPAIS ALGORITMOS QUÂNTICOS

Até hoje, existem apenas três algoritmos quânticos [8]: os algoritmos de Deutsch, Shor e Grover. O algoritmo de Deutsch introduziu um conceito totalmente novo, o parelelismo quântico, que é a capacidade de analisar vários valores de uma função de uma só vez. Portanto, utilizando o método quântico de Deutsch, é possível verificar se uma função é crescente, decrescente ou constante utilizando apenas uma operação.

Shor, em 1993, desenvolveu o mais famoso algoritmo da computação quântica, um algoritmo para fatoração em tempo polinomial. Um número de 1024 bits é fatorado classicamente em 100 mil anos, enquanto que, com o algoritmo de Shor, ele pode ser fatorado em menos de 5 minutos [8]. O trabalho de Shor foi o grande responsável pelo gigantesco aumento recente das pesquisas na área da computação quântica.

Outro importante algoritmo quântico foi desenvolvido por Grover em 1994. Ele possibilita a realização de uma busca em uma lista não ordenada em um número de operações proporcional à raiz quadrada do tamanho da lista. Ele não chega a ser tão impressionante como o algoritmo de Shor, mais ainda assim constitui um grande aumento de velocidade em relação ao seu análogo clássico, que busca em um número de operações proporcional ao tamanho da lista.

### 2.3.3 ALGORITMO DE SHOR

O algoritmo de Shor (AS) é o melhor algoritmo para fatoração existente. Isso acontece porque ele utiliza em sua etapa mais lenta a Transformada de Fourier Quântica (TFQ), que necessita de $o(n^2)$ operações, enquanto que sua análoga, a Transformada de Fourier Rápida (TFR), requer $o(n2^n)$. O processo completo para implementação da TFQ pode ser encontrado em [8].

O AS é constituído de várias etapas clássicas e apenas uma etapa quântica, a da Transformada de Fourier, e retorna os divisores de um número. O número N que será utilizado no AS deve ser ímpar e não pode ser uma potência de um primo. Um algoritmo eficiente para testar se um número é uma potência de um primo pode ser encontrado em [9].

As etapas do AS serão descritas a seguir:

1. Se N=1 ou é potência de um primo, termina o algoritmo. Se N for potência de um primo, retorne-o.
2. Escolha um número qualquer a < N.
3. Calcule MDC(a, N).
4. Se MDC(a, N) ≠ 1, retorne a, faça N ← N/a e volte para o passo 1.
5. Utilize a TFQ para calcular o período r da função $f(x) = a^x \pmod N$.
6. Se r é ímpar ou $a^{r/2} = -1 \pmod N$, volte para o passo 1.
7. Retorne $d_1 = MDC(a^{r/2} + 1, N)$ e $d_2 = MDC(a^{r/2} - 1, N)$.
8. Faça $N \leftarrow N/d_1d_2$ e volte para o passo 1.

Como exemplo de utilização do Algoritmo de Shor, pode-se calcular os divisores de N=15:

- Escolho a = 7, MDC(a, N) = 1.
- O período r da função $f(x) = 7^x \pmod{15}$ é 4.
- r não é ímpar e $a^{r/2} = 4 \pmod{15}$. OK!
- Retorno $d_1 = MDC(50, 15) = 5$ e $d_2 = MDC(48, 15) = 3$.
- Faço $N \leftarrow 15/15_2 = 1$ e volto para o passo 1.
- N = 1, fim do algoritmo: os divisores são 3 e 5.

# 3 FUNDAMENTAÇÃO MATEMÁTICA

Para melhor compreensão do estudo dos métodos de criptografia, será apresentada uma breve fundamentação matemática, com definições, algoritmos e teoremas relevantes.

## 3.1 CONHECIMENTOS DE MATEMÁTICA DISCRETA

### 3.1.1 OPERAÇÃO BINÁRIA

Uma operação binária é uma função fechada em um conjunto, que associa dois de seus elementos a um elemento qualquer do conjunto. Uma operação binária $\Theta$ em um conjunto A pode ser definida como $\Theta$: A x A $\rightarrow$ A e podemos ter

$$a_i \,\Theta\, a_j = a_k \qquad \forall\, a_i, a_j, a_k \in \text{A}.$$

Uma operação binária em um conjunto A pode ter as seguintes propriedades [10]:

- **Associatividade:**

$$(a \,\Theta\, b) \,\Theta\, c = a \,\Theta\, (b \,\Theta\, c) \qquad \forall\, a, b, c \in \text{A}.$$

- **Comutatividade:**

$$a \,\Theta\, b = b \,\Theta\, a \qquad \forall\, a, b \in \text{A}.$$

- **Existência de elementos neutros laterais:** Um elemento $e \in$ A é chamado elemento neutro à esquerda se

$$e \,\Theta\, a = \text{a} \qquad \forall\, a \in \text{A}.$$

Analogamente, o elemento *e* é chamado elemento neutro à direita se

$$a \,\Theta\, e = \text{a} \qquad \forall\, a \in \text{A}.$$

Se A possui identidade à esquerda e à direita, então ambos coincidem em uma única identidade, chamada de elemento neutro do conjunto A.

- **Existência de elementos inversos laterais:** Para conjuntos que possuam elemento neutro, definem-se também os inversos laterais de um elemento. O elemento *b* é dito inverso à esquerda de *a* se

  $b \ominus a = e$ onde $a \in A$ e *e* é o elemento neutro.

  Da mesma forma, *b* é dito inverso à direita se

  $a \ominus b = e$ onde $a \in A$ e *e* é o elemento neutro.

  Se *a* possui inverso à esquerda e à direita, então ambos coincidem em um único elemento chamado de inverso de *a*.

### 3.1.2 ESTRUTURAS ALGÉBRICAS

Listamos abaixo as definições das principais estruturas algébricas [10]:

- **Semigrupo:** Um semigrupo G consiste em um conjunto não vazio A e uma operação binária associativa $\ominus$ definida em A. G é representado por [A, $\ominus$].

- **Monóide:** Um monóide é um semigrupo que tem elemento neutro.

- **Grupo:** Um grupo G = [A, $\ominus$] é um monóide tal que cada elemento do conjunto A possui um único elemento inverso. Se $\ominus$ for comutativa em A, G é chamado grupo comutativo. Se A for um conjunto finito, G é chamado de grupo finito e define-se como ordem de G o número de elementos de A, representado por #(A). Exemplo de grupo: [R, +], onde R é o conjunto dos números reais.

- **Subgrupo:** G` = [A`, $\ominus$] é chamado subgrupo do grupo G = [A, $\ominus$] se G` é um grupo e A` $\subset$ A. Exemplo: [Z, +] é um subgrupo de [R, +], onde Z e R são respectivamente, o conjuntos dos números inteiros e o dos números reais.

- **Anel:** Um anel é um sistema [A, Θ, Θ], onde [A, Θ] é um grupo comutativo e [A, Θ] um semigrupo tal que, para quaisquer elementos *a*, *b*, *c* $\in$ A, valham as leis da distributividade:

  $a \Theta (b \Theta c) = (a \Theta b) \Theta (a \Theta c)$ e $(b \Theta c) \Theta a = (b \Theta a) \Theta (c \Theta a)$.

  A operação Θ é chamada de adição e seu elemento neutro chamado de 0, ou elemento nulo. A operação Θ é chamada de multiplicação e seu elemento neutro, se existir, é chamado de 1, ou elemento unitário. Exemplo: [Z, +, .] é um anel, onde Z é o conjunto dos inteiros com as operações soma e multiplicação.

- **Sub-anel:** R` = [A`, Θ, Θ] é chamado sub-anel de R = [A, Θ, Θ] se satisfaz $A` \subset A$ , $a \Theta b \in A`$ e $a \Theta b \in A`$ $\forall a, b \in A`$.
- **Ideal:** O sub-anel R` é um ideal do anel R se satisfaz às condições $a \Theta a` \in A`$ e $a` \Theta a \in A`$ $\forall a \in A$ e $a` \in A`$.

- **Corpo:** Um anel C = [A, Θ, Θ] é chamado de corpo quando [A*, Θ] é um grupo comutativo. A* é o conjunto formado pelos elementos não-nulos de C. Um exemplo de corpo é o anel [R, +, .] composto pelo conjunto dos números reais com as operações soma e multiplicação.

## 3.2 ÁLGEBRA MODULAR

A álgebra modular ocorre em um corpo finito $F_p = [Z_p, +_p, ._p]$, onde *p* é um número natural, $Z_p$ é o conjunto formado pelos números naturais menores que *p* e suas operações binárias $+_p$ e $._p$ são as operações de soma e multiplicação dentro do corpo $F_p$. As operações $+_p$ e $._p$ são chamadas soma modulo *p* e produto modulo *p*, pois oferecem como resultado o resto da divisão da soma ou produto por *p*. A figura abaixo (FIG. 3.1) mostra as operações soma e produto dentro do corpo $F_5$:

| + | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 4 | 0 |
| 2 | 2 | 3 | 4 | 0 | 1 |
| 3 | 3 | 4 | 0 | 1 | 2 |
| 4 | 4 | 0 | 1 | 2 | 3 |

| • | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 |
| 2 | 0 | 2 | 4 | 1 | 3 |
| 3 | 0 | 3 | 1 | 4 | 2 |
| 4 | 0 | 4 | 3 | 2 | 1 |

**FIG 3.1 –** Tabela de Soma e Produto

A partir da operação produto, define-se em $F_p$ a operação potência. O valor de $a^n$ em $F_p$ é obtido a partir da execução de n operações de produto

$(a \cdot_p a \cdot_p a \cdot_p \ldots \cdot_p a)$ para $a \in Z_p$

A partir da operação potência, define-se em $F_p$ o conceito de raiz primitiva. Um elemento $g \in F_p$ é chamado raiz primitiva se suas potências geram todos os elementos não nulos de $F_p$. Para exemplificar, analisaremos os conjuntos gerados pelas potências dos elementos não nulos de $F_5$:

$1^0 = 1$ $1^1 = 1$ $1^2 = 1$ $1^3 = 1$ $1^4 = 1$ Conjunto gerado: { 1 }

$2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^3 = 3$ $2^4 = 1$ Conjunto gerado: { 1, 2, 3, 4 }

$3^0 = 1$ $3^1 = 3$ $3^2 = 4$ $3^3 = 2$ $3^4 = 1$ Conjunto gerado: { 1, 2, 3, 4 }

$4^0 = 1$ $4^1 = 4$ $4^2 = 1$ $4^3 = 4$ $4^4 = 1$ Conjunto gerado: { 1, 4 }

Os elementos 2 e 3 são as raízes primitivas de $F_5$, pois suas potências geram todos os seus elementos não nulos. Já os elementos 1 e 4 não são raízes primitivas, pois suas potências não geram todos os elementos não nulos de $F_5$.

Como continuações do estudo da álgebra modular serão apresentados alguns teoremas fundamentais para o estudo dos criptossistemas de ElGamal e RSA [11]: o Algoritmo de Euclides, o Teorema de Fermat e o Teorema de Euler.

### 3.2.1 INVERSO MULTIPLICATIVO

Teorema: A condição necessária e suficiente para que um elemento *a* tenha inverso multiplicativo em $F_p$ é que MDC(*a, p*) = 1.

Prova:

Sejam os inversos multiplicativos $a, a^{-1} \in F_p$.

$$a.a^{-1} = 1 \pmod{p} \qquad \text{(EQ. 3.1)}$$

Como $a.a^{-1}$ deixa resto 1 na divisão por *p*, existe um inteiro *k* tal que

$$a.a^{-1} - k.p = 1 \qquad \text{(EQ. 3.2)}$$

Seja *d* um inteiro tal que *d* divide *a* e *p* ao mesmo tempo. Pela EQ. 3.2:

$$0.a^{-1} - k.0 = 1 \pmod{d} \qquad \text{(EQ. 3.3)}$$

Pela EQ. 3.3, conclui-se que *d* = 1 e MDC(*a, p*) = 1.

Sejam agora *a* e *p* inteiros tais que MDC(*a, p*) = 1. Logo existem $k_1$ e $k_2$ tais que

$$a.\ k_1 + p.\ k_2 = 1 \qquad \text{(EQ. 3.4)}$$

Analisando o resto na divisão por *p* na equação EQ. 3.4, obtemos

$$a.k_1 = 1 \pmod{p} \qquad \text{(EQ. 3.5)}$$

Pela EQ. 3.5, vemos que *a* possui inverso multiplicativo em $F_p$, o que conclui a prova.

### 3.2.2 ALGORITMO DE EUCLIDES

O Algoritmo de Euclides (AE) é um procedimento que retorna o MDC(a, b) sem a necessidade de fatorar os números:

1 – Se b > a inverta os valores de b e a;

2 – Divida a por b e encontre o quociente q e o resto r;

3 – Se r = 0, retorne b.

4 – se r ≠ 0, faça:

4.1 – a ← b;

4.2 – b ← q;

4.3 – volte para o passo 2;

### 3.2.3 ALGORITMO DE EUCLIDES ESTENDIDO

O Algoritmo de Euclides Estendido (AEE) é uma aplicação dos quocientes obtidos no Algoritmo de Euclides para obter os coeficientes da equação:

$$ax + by = MDC(a, b)$$

Ele é uma maneira eficiente de determinar o inverso multiplicativo e é muito utilizado no RSA para a solução na equação EQ 4.8.

Os coeficientes x e y são obtidos a partir dos valores de $x_n$ e $y_n$ na seguinte recursão [11]:

$$\begin{cases} x_{-1} = 1, x_0 = 0, x_k = x_{k-2} - q_k x_{k-1} \\ y_{-1} = 0, y_0 = 1, y_k = y_{k-2} - q_k y_{k-1} \end{cases}$$ , onde $q_i$ são os coeficientes do AE

Será calculado o inverso multiplicativo de 17 mod 46 para exemplificar a utilização do AEE. A partir das divisões sucessivas do AE, obtemos:

$46 = 17 \times 2 + 12 \rightarrow q_1 = 2$;

$17 = 12 \times 1 + 5 \rightarrow q_2 = 1$;

$12 = 5 \times 2 + 2 \rightarrow q_3 = 2$;

$5 = 2 \times 2 + 1 \rightarrow q_4 = 2$;

$2 = 2 \times 1 + 0$;

Fim do Algoritmo de Euclides.

$y_1 = y_{-1} - q_1 y_0 = -2$;

$y_2 = y_0 - q_2 y_1 = 3$;

$y_3 = y_1 - q_3 y_2 = -8$;

$y_4 = y_2 - q_4 y_3 = 19$;

Fim do Algoritmo de Euclides Estendido: $17^{-1} \bmod 46 = 19$

### 3.2.4 TEOREMA DE FERMAT

O Teorema de Fermat é um resultado muito importante para a álgebra modular, e fundamental para a prova de outros teoremas importantes que serão utilizados. Ele diz que, para todo primo *p*:

$$a^{p-1} = 1 \pmod{p} \quad \forall\, a \neq 0\,,\ a \in F_p$$

Prova:

*Seja o* conjunto $A_p = \{ 0, a, 2^a, 3^a, \ldots, (p-1)a \}\,,\ a \in F_p$.

Como todos os elementos de $A_p$ também são elementos de $F_p$, temos $A_p \subset F_p$. Seja $x \in F_p$. Como MDC($a$, $p$) = 1, podemos mostrar que $x \in A_p$ :

$$x = x.1 = x.\ a^{-1}.a = y.a\ ,\quad onde\ y = x.a^{-1}$$

Dessa forma, concluímos que os conjuntos $A_p$ e $F_p$ são iguais, o que garante a seguinte igualdade:

$$a.2^a.3^a.\ ...\ .\ (p\text{-}1)a = 1.2.3\ ...\ (p\text{-}1) \qquad \text{(EQ. 3.6)}$$

Do lado esquerdo da EQ. 3.6 temos o produto de todos os elementos não-nulos de $A_p$, e do lado direito o produto dos elementos não-nulos de $F_p$. Como os conjuntos são iguais, a equação é válida. Desenvolvendo a equação, obtemos:

$$a^{p-1}\ .\ (p\text{-}1)! = (p\text{-}1)!$$

$$(a^{p-1} - 1)\ ((p - 1)!) = 0$$

$$a^{p-1} - 1 = 0$$

$$a^{p-1} = 1\ (\text{mod}\ p)$$

O que conclui a demonstração do Teorema de Fermat.

### 3.2.5 TEOREMA DE EULER

O Teorema de Euler é uma generalização do Teorema de Fermat. Ele diz que, para todo inteiro $n$:

$$a^{\phi(n)} = 1\ (\text{mod}\ n) \qquad \forall\ a \text{ tal que MDC}(a, n) = 1$$

A função $\phi(n)$, conhecida como Função de Euler, é definida como a quantidade de elementos $k$ de $F_n$ que são tais que MDC($k$, $n$) = 1. Ela possui 3 propriedades básicas [11]:

- $\phi(p) = p\text{-}1$ ($p$ primo)
- $\phi(p^k) = p^k\text{-}p^{k-1}$ ($p$ primo)
- $\phi(p.q) = \phi(p).\ \phi(q)$ ($p$ e $q$ primos entre si)

Prova do Teorema:

*Seja o* conjunto $\Phi_n = \{ x_1, x_2, x_3, ... , x_{\phi(n)} \}$ , composto por todos os elementos $x$ de $F_n$ tais que MDC($x$, $n$) = 1. Por definição, o conjunto $\Phi_n$ tem $\phi(n)$ elementos. Seja também o conjunto $Na = \{ ax_1, ax_2, ax_3, ... , ax_{\phi(n)} \}$, onde $a$ é tal que MDC($a$,$n$) = 1.

Como todos os elementos de $Na$ são primos com $n$, temos $Na \subset \Phi_n$

Seja $x \in \Phi_n$. Como MDC($a$, $n$) = 1, podemos mostrar que $x \in Na$ :

$$x = 1.x = a.a^{-1}.x = y.a \ , \quad onde\ y = x.a^{-1} \in \Phi_n$$

Dessa forma, concluímos que os conjuntos *Na* e $\Phi_n$ são iguais, o que garante a seguinte igualdade:

$$ax_1.ax_2.ax_3. \ ... \ .ax_{\phi(n)} = x_1.x_2.x_3. \ ... \ . \ x_{\phi(n)} \qquad \text{(EQ. 3.7)}$$

Do lado esquerdo da EQ. 3.7 temos o produto de todos os elementos de $A_n$, e do lado direito o produto dos elementos de $\Phi_n$. Como os conjuntos são iguais, a equação é válida. Já que os elementos de $\Phi_n$ são inversíveis, podemos cancelá-los para obter:

$$a^{\phi(n)} = 1 \ (\text{mod } n)$$

O que conclui a demonstração do Teorema de Euler.

## 3.3 ÁLGEBRA EM ANÉIS POLINOMIAIS

A álgebra em anéis polinomiais ocorre em anéis da forma $R = [P_n, +, *]$, onde $P_n$ é um conjunto de polinômios de grau menor que *n e* suas operações binárias + e $*$ são obtidas a partir das operações de soma e multiplicação de polinômios. As operações + e $*$ são chamadas soma modulo *n* e produto modulo *n*, pois oferecem como resultado o resto da divisão da soma ou produto pelo polinômio $x^n-1$.

Os anéis polinomiais são fundamentais para o estudo do criptossistema NTRU, que utiliza basicamente dois tipos de anéis: *R* e $R_q$, que podem ser definidos como [3]:

$$R = \frac{Z[x]}{(x^n-1)}, \qquad R_q = \frac{Z_q[x]}{(x^n-1)}$$

Desta forma, os elementos de *R* e $R_q$ são representados de forma geral por:

$$a(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_{n-1}x^{n-1}$$

onde os coeficientes $a_i$ pertencem respectivamente a *Z* ou $Z_q$. Além da representação polinomial, os elementos também podem ser representados de forma vetorial:

$$a = (a_0, a_1, a_2, a_3, ... , a_{n-1})$$

No anel *R*, a operação de soma é definida de forma idêntica à soma de polinômios, enquanto a operação produto é definida como o resto do produto dos polinômios na divisão pelo polinômio $(x^n - 1)$. Já no anel $R_q$, se um dos coeficientes do polinômio resultado das operações de soma ou produto estiver fora do intervalo

[0, q-1] ele é substituído por seu resto na divisão por q, de forma que todos os coeficientes sempre fiquem nesse intervalo. O procedimento será exemplificado a seguir, onde serão utilizados os polinômios a(x) e b(x) e os valores de $n = 3$ e $q = 7$.

$$a(x) = x^2 + 3x + 5 \text{ , } b(x) = 2x^2 + x + 10$$

Operação soma, respectivamente, nos anéis $R$ e $R_q$:

$$a(x) + b(x) = 3x^2 + 4x + 15$$

$$a(x) + b(x) = 3(\text{mod } 7)x^2 + 4(\text{mod } 7)x + 15(\text{mod } 7) = 3x^2 + 4x + 1$$

Operação produto, respectivamente, nos anéis $R$ e $R_q$:

- Produto comum: $(x^2 + 3x + 5)(2x^2 + x + 10) = 2x^4 + 7x^3 + 23x^2 + 35x + 50$
- Resto por $(x^3 - 1)$: $32x^2 + 35x + 50$

$$a(x) * b(x) = 32x^2 + 35x + 50$$

$$a(x) * b(x) = 32(\text{mod } 7)x^2 + 35(\text{mod } 7)x + 50(\text{mod } 7) = 4x^2 + 1$$

Um divisor comum entre dois elementos *a* e *b* de um anel polinomial é um terceiro elemento que divide *a* e *b* ao mesmo tempo. O elemento c é MDC(a,b) se qualquer divisor comum entre a e b também divide c (desconsiderando as constantes multiplicativas). Logo, o grau do MDC entre dois polinômios é único, mas seu valor pode variar de uma constante multiplicativa.

Como continuação do estudo da álgebra em anéis polinomiais, serão apresentadas algumas definições e teoremas fundamentais para o estudo do criptossistema NTRU: o polinômio ternário, a transposição simétrica e o inverso multiplicativo.

### 3.3.1 POLINÔMIO TERNÁRIO

Um polinômio ternário é aquele que possui apenas coeficientes -1, 0 e 1. Um polinômio ternário da forma $T(d_1, d_2)$ é tal que [3]:

- $d_1$ coeficientes são iguais a 1.
- $d_2$ coeficientes são iguais a -1.
- Os outros $N - d_1 - d_2$ coeficientes são iguais a 0.

### 3.3.2 TRANSPOSIÇÃO SIMÉTRICA

Seja a(x) um polinômio pertencente a $R_q$. A transposição simétrica (*centered lift*)

de a(x) para R é o único polinômio a`(x) que satisfaz:

$$a`_i \pmod q = a_i \pmod q \ , \ -\frac{q}{2} < a`_i \leq \frac{q}{2}$$

Exemplo: seja o polinômio $a(x) = 4x^4 + 7x^3 + x + 6$ pertencente a $R_{11}$. A transposição simétrica a`(x) é obtida substituindo por $a_i - 11$ cada elemento $a_i$ maior que 5: $a`(x) = 4x^4 - 4x^3 + x - 5$.

### 3.3.3 INVERSO MULTIPLICATIVO

Teorema: Seja p um número primo. A condição necessária e suficiente para que um elemento a(x) tenha inverso multiplicativo em $R_p$ é que MDC(a(x), $x^N - 1$) = 1.

Prova:

Sejam os inversos multiplicativos a(x), $a^{-1}(x) \in F_p$.

$$a(x) * a^{-1}(x) = 1 \pmod{x^N - 1} \qquad \text{(EQ. 3.8)}$$

Como $a(x) * a^{-1}(x)$ deixa resto 1 na divisão por $x^N - 1$, existe um polinômio k tal que

$$a(x) * a^{-1}(x) - k(x) * (x^N - 1) = 1 \qquad \text{(EQ. 3.9)}$$

Seja d(x) um polinômio tal que d(x) divide a(x) e $x^N - 1$ ao mesmo tempo. Pela EQ. 3.9:

$$0 * a^{-1}(x) - k(x) * 0 = k \pmod{d(x)} \qquad \text{(EQ. 3.10)}$$

Pela EQ. 3.10, conclui-se que d(x) = 1 e MDC(a(x), $x^N - 1$) = 1.

Seja agora a(x) um polinômio tal que MDC(a(x), $x^N - 1$) = 1. Logo existem os polinômios $k_1(x)$ e $k_2(x) \in R_p$ tais que

$$a(x) * k_1(x) + (x^N - 1) * k_2(x) = 1 \qquad \text{(EQ. 3.11)}$$

Analisando o resto na divisão por $x^N - 1$ na equação EQ. 3.11, obtemos

$$a(x) * k_1(x) = 1 \pmod{x^N - 1} \qquad \text{(EQ. 3.12)}$$

Pela EQ. 3.12, vemos que a(x) possui inverso multiplicativo em $R_p$, o que conclui a prova. O procedimento para calcular o inverso multiplicativo em $R_p$ é semelhante ao Algoritmo de Euclides Estendido, descrito na seção 3.2.3.

No entanto, como o MDC pode variar de uma constante multiplicativa, se o MDC encontrado for um inteiro $k \neq 1$, o produto $a(x) * a^{-1}(x)$, onde $a^{-1}(x)$ é o resultado da aplicação do AES também será diferente de 1. Esse problema pode ser resolvido sempre que p for primo, já que qualquer que seja o valor de k, sempre vai existir seu

inverso multiplicativo em $R_p$. Desta forma, o valor do inverso multiplicativo de a(x) pode ser obtido dividindo o polinômio encontrado utilizando o AES pelo valor do MDC encontrado utilizando o AE.

Será mostrado no exemplo a seguir, para calcular o inverso de

$$a(x) = (x^6+x^5+x^4+x+1) \in R_2, \text{ para n=7:}$$

A partir das divisões sucessivas do Algoritmo de Euclides, calcular o MDC entre a(x) e $(x^7-1) \equiv (x^7+1)$ em $R_2$:

$x^7+1 = (x^6+x^5+x^4+x+1)(x+1) + (x^4+x^2) \rightarrow q_1 = x+1;$

$x^6+x^5+x^4+x+1= (x^4+x^2)(x^2+x) + (x^3+x+1) \rightarrow q_2 = x^2+x;$

$x^4+x^2 = (x^3+x+1)(x) + (x) \rightarrow q_3 = x;$

$(x^3+x+1) = (x)(x^2+1) + 1 \rightarrow q_4 = x^2+1;$

$x = (1)(x) + 0;$

Fim do Algoritmo de Euclides: MDC = 1. Agora aplicaremos a recursão do AES:

$$\begin{cases} x_{-1} = 1, x_0 = 0, x_k = x_{k-2} - q_k x_{k-1} \\ y_{-1} = 0, y_0 = 1, y_k = y_{k-2} - q_k y_{k-1} \end{cases}$$ , onde $q_i$ são os coeficientes do AE

$y_1 = y_{-1} - q_1 y_0 = x+1;$

$y_2 = y_0 - q_2 y_1 = 1 -(x^2+x)(x+1) = x^3+x+1;$

$y_3 = y_1 - q_3 y_2 = x+1 -x(x^3+x+1) = x^4+x^2+1;$

$y_4 = y_2 - q_4 y_3 = x^3+x+1 -(x^2+1)(x^4+x^2+1) = x^6+x^3+x;$

Fim do Algoritmo de Euclides Estendido: $(x^6+x^5+x^4+x+1)^{-1} \bmod (x^7-1) = (x^6+x^3+x)$

## 3.4 CONHECIMENTOS DE ÁLGEBRA LINEAR

### 3.4.1 VETOR

Um vetor é uma grandeza física que possui magnitude, direção e sentido [12]. Ele pode ser representado geometricamente por um segmento de reta no plano ou no espaço. A ponta da seta é a extremidade, ou ponto final, e a outra ponta é a origem do vetor, como pode ser visto na figura abaixo (FIG 3.2), onde A é a origem do vetor, B sua extremidade e r sua magnitude.

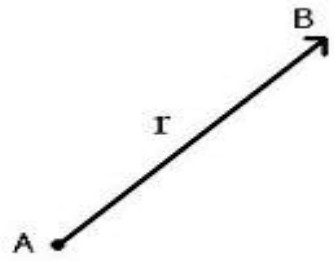


**FIG 3.2 –** Representação Geométrica do Vetor

Matematicamente, um vetor é representado por uma elemento de $R^n$, que são uma seqüência ordenada de *n* números reais. Seja *v* um vetor pertencente a $R^n$ *:*

$$v = ( v_1, v_2, \ldots , v_n) \text{ , onde } v_k \in R \; \forall \; k$$

Definimos o vetor *–v* como simétrico do vetor *v:*

$$-v = ( -v_1, -v_2, \ldots , -v_n) \text{ , onde } v_k \text{ é a coordenada } k \text{ do vetor } v$$

Definimos também o vetor 36V como multiplicação do vetor *v* pelo escalar k*:*

$$kv = ( kv_1, kv_2, \ldots , kv_n) \text{ , onde } k \in R$$

Para dois vetores *a* e *b*, definimos *a* + *b* como a operação soma vetorial*:*

$$a + b = ( a_1 + b_1 , a_2 + b_2, \ldots , a_n + b_n)$$

Definimos também o número real $a \circ b$ como produto escalar entre *a* e *b:*

$$a \circ b = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

Dado um vetor *v*, sua norma, ou magnitude, pode ser calculada a partir da equação:

$$| v |^2 = v \circ v \qquad \text{(EQ. 3.8)}$$

### 3.4.2 ESPAÇO VETORIAL

Um espaço vetorial *V* é um subconjunto de $R^n$ que é fechado em relação à soma e à multiplicação por escalares de *R*. Um espaço vetorial é representado por:

$$\alpha_1 v_1 + \alpha_2 v_2 \in V, \; \forall \; v_1, v_2 \in V \text{ e } \forall \; \alpha_1, \alpha_2 \in R$$

### 3.4.3 COMBINAÇÃO LINEAR

Seja *B* um conjunto composto pelos vetores $v_1, v_2, \ldots , v_k \in R^n$*.* Uma combinação linear de *B* é qualquer vetor que possa ser escrito da forma:

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k, \text{ com } \alpha_i \in R \text{ para todo } i$$

O conjunto *V* composto por todas as combinações lineares de *B* é chamado de

espaço vetorial gerado por *B.*

### 3.4.4 INDEPENDÊNCIA LINEAR

Um conjunto de vetores $B = v_1, v_2, \ldots, v_k \in R^n$ é dito linearmente independente, ou LI, se, e somente se, a única maneira de se representar o vetor nulo como uma combinação linear dos vetores de *B* for fazendo todos os coeficientes serem nulos

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k = \mathbf{0} \leftrightarrow \alpha_1 = \alpha_2 = \ldots = \alpha_k = 0$$

Caso exista alguma outra forma de representar o vetor nulo que contenha pelo menos um coeficiente não nulo, o conjunto é dito linearmente dependente, ou LD.

### 3.4.5 BASE DO ESPAÇO VETORIAL

Uma base de um espaço vetorial *V* é um conjunto de vetores linearmente independentes que geram *V.* Se *B* é uma base do espaço vetorial *V*, qualquer vetor de *V* pode ser representado de maneira única como uma combinação linear de v de vetores de *V*:

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k, \quad w \in V$$

para apenas uma escolha de $\alpha_1, \alpha_2, \ldots, \alpha_k$

Uma base *B* é dita ortogonal quando possui a seguinte propriedade:

$$x \circ y = 0, \text{ se } x \neq y, \forall x, y \in B$$

Uma base ortogonal *B* é dita ortonormal quando possui a seguinte propriedade:

$$|| x || = 1, \forall x \in B$$

### 3.4.6 ALGORITMO DE GRAM-SCHMIDT

Seja $B = \{ v_1, v_2, \ldots, v_n \}$ uma base do espaço vetorial *V.* O algoritmo de Gram-Schmidt (AGS) gera uma base ortogonal $B^* = \{ v_1^*, v_2^*, \ldots, v_n^* \}$ para *V* [13]:

1 – Faça $v_1^* = v_1$;

2 – Para *i* de 2 até *n*:

2.1 – Para *j* de 1 até *i-1*:

2.1.1 – calcule $\mu_{ij} = v_i \circ v_j^* / ||v_j^*||^2$;

2.2 – Faça $v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{ij} \cdot v_j^*$

3 – Retorne $B^* = \{ v_1^*, v_2^*, ..., v_n^* \}$;

Prova por indução de que $B^*$ é ortogonal:

Primeiro vamos utilizar o algoritmo para calcular $v_2^*$:

$$v_2^* = v_2 - \mu_{21} v_1^* = v_2 - v_2 \circ v_1^* / ||v_1^*|| \, . \, v_1^* \qquad \text{(EQ 3.9)}$$

Pela equação EQ. 3.9, podemos calcular $v_2^* \circ v_1^*$:

$$v_2^* \circ v_1^* = v_2 \circ v_1^* - v_2 \circ v_1^* / ||v_1^*||^2 \, . \, v_1^* \circ v_1^* = v_2 \circ v_1^* - v_2 \circ v_1^* / ||v_1^*||^2 \, . \, ||v_1^*||^2$$

$$v_2^* \circ v_1^* = 0 \qquad \text{(EQ. 3.10)}$$

Isso implica que o conjunto $\{ v_1^*, v_2^* \}$ é ortogonal.

Suponho que o conjunto $\{ v_1^*, ..., v_i^* \}$ é ortogonal. Desejamos provar que o conjunto $\{ v_1^*, ..., v_{i+1}^* \}$ também é ortogonal.

Primeiro vamos utilizar o algoritmo para calcular $v_{i+1}^* \circ v_k^*$, onde $k < i+1$:

$$v_{i+1}^* \circ v_k^* = (v_{i+1} - \sum_{j=1}^{i} \mu_{i+1j} \cdot v_j^*) \circ v_k^* \qquad \text{(EQ. 3.11)}$$

Como o conjunto $\{ v_1^*, ..., v_i^* \}$ é ortogonal, sabemos que $v_j^* \circ v_k^* = 0$ para $j \neq k$. Desta forma, podemos melhorar a EQ. 3.11:

$$v_{i+1}^* \circ v_k^* = v_{i+1} \circ v_k^* - \mu_{i+1k} \cdot v_k^* \circ v_k^* \qquad \text{(EQ. 3.12)}$$

Substituindo o valor de $\mu_{i+1k}$:

$$v_{i+1}^* \circ v_k^* = v_{i+1} \circ v_k^* - v_{i+1} \circ v_k^* / ||v_k^*||^2 \, . \, v_k^* \circ v_k^* = v_{i+1} \circ v_k^* - v_{i+1} \circ v_k^* / ||v_k^*||^2 . \, ||v_k^*||^2$$

$$v_{i+1}^* \circ v_k^* = 0 \qquad \text{(EQ. 3.13)}$$

Isso implica que o conjunto $\{ v_1^*, ..., v_{i+1}^* \}$ também é ortogonal. Pelo princípio da indução finita, podemos afirmar que o conjunto $B^* = \{ v_1^*, v_2^*, ..., v_n^* \}$ é ortogonal, o que conclui a prova.

## 3.5 CONHECIMENTOS DE RETICULADO

### 3.5.1 DEFINIÇÃO DE RETICULADO

Seja $B = v_1, v_2, ..., v_n \in R^m$ um conjunto de vetores linearmente independentes. O Reticulado L gerado por $B$ é um conjunto de combinações lineares de $v_1, v_2, ..., v_n$ com coeficientes inteiros. B é chamada a base do reticulado L e n sua dimensão [3].

$$L = \{ a_1 v_1 + a_2 v_2 + ... + a_n v_n \}, \text{ onde } a_1, a_2, ..., a_n \in Z$$

O domínio fundamental de um reticulado é o conjunto

$$F(L) = \{ t_1 v_1 + t_2 v_2 + ... + t_n v_n \}, \text{ onde } 0 \leq t_i \leq 1$$

O discriminante, ou volume de um reticulado é a medida da região interior ao domínio fundamental e é obtido por

$$V(L) = \; | \det(v_1 | \, v_2 | ... | \, v_n) \, |$$

Na figura abaixo (FIG. 3.3) será apresentado um exemplo de reticulado de dimensão dois e o domínio fundamental *F* associado a sua base ($v_1$, $v_2$). A base de um reticulado pode ser qualquer conjunto de vetores LI que gerem todos os seus pontos.

Como será visto adiante, bases muito longas e pouco ortogonais, como ($v_3$, $v_4$) são consideradas ruins, enquanto que bases pequenas e bem ortogonais, como ($v_1$, $v_2$) são consideradas bases boas.

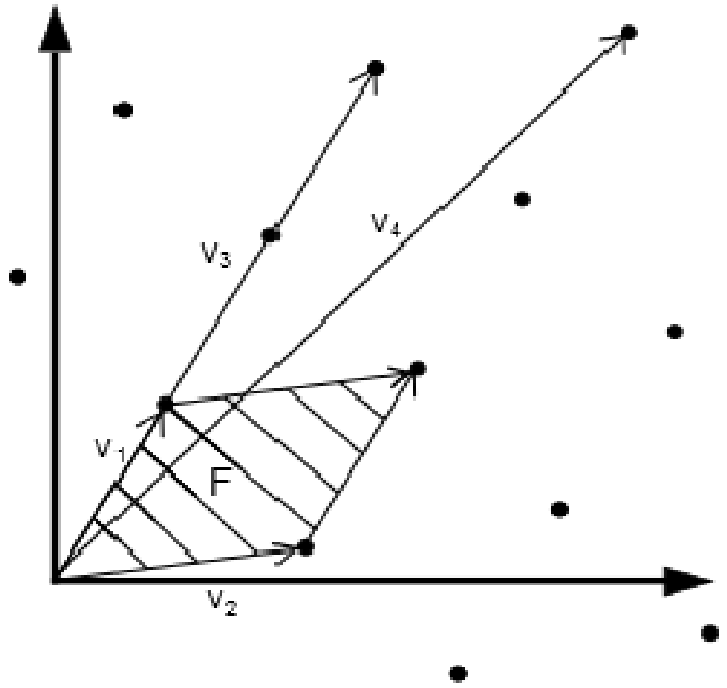


**FIG 3.3 –** Exemplo de Reticulado

### 3.5.2 RELAÇÃO ENTRE AS BASES DE UM RETICULADO

Sejam $V = \{ v_1, v_2, ..., v_n \}$ e $W = \{ w_1, w_2, ..., w_n \}$ duas bases de $L$. Podemos representar cada vetor de $W$ como uma combinação linear dos vetores de $V$:

$$w_1 = a_{11}v_1 + a_{12}v_2 + ...+ a_{1n}v_n$$

$$w_2 = a_{21}v_1 + a_{22}v_2 + ...+ a_{2n}v_n$$

...

$$w_n = a_{n1}v_1 + a_2v_2 + ...+ a_{nn}v_n$$

Fazendo $W = (w_1| w_2|...|w_n)$, $V = (v_1|v_2|...|v_n)$ e denotando a matriz de elementos $a_{ij}$ por A, podemos representar esse sistema de equações na forma matricial:

$$W = AV \qquad \text{(EQ. 3.14)}$$

Analogamente, podemos representar V em função de W:

$$V = A^{-1}W \qquad \text{(EQ. 3.15)}$$

Pela definição de reticulado, todos os elementos de A e de $A^{-1}$ tem que ser inteiros. Logo, temos que $\det(A), \det(A^{-1}) \in Z$.

Para qualquer matriz inversível, sabemos que:

$$\det(A).\ \det(A^{-1}) = 1$$

Só existem duas formas de um produto de inteiros ser 1: 1x1=1 ou (-1) x (-1) = 1.

Isso prova o seguinte resultado [3]:

Duas bases de um reticulado estão relacionadas por uma matriz de coeficientes inteiros e cujo determinante tem módulo unitário.

Além disso, como | det(V) | = | det(W) |, o volume V($L$) é invariante em relação à base.

### 3.5.3 BASE BOA E BASE RUIM

A qualificação de uma base em boa ou ruim está intimamente ligada ao problema do vetor mais próximo em um reticulado $L$ (*Closest Vector Problem* - CVP).

O CVP consiste em, dado um vetor $v \in R^m$, encontrar o vetor $w \in L$ cuja distância até $v$ é mínima. O custo para resolver o CVP é exponencial em relação ao valor de $n$ e isso se torna inviável para valores muito grandes. No entanto, é possível obter uma aproximação para a solução do CVP a partir do domínio fundamental relacionado a uma base do reticulado.

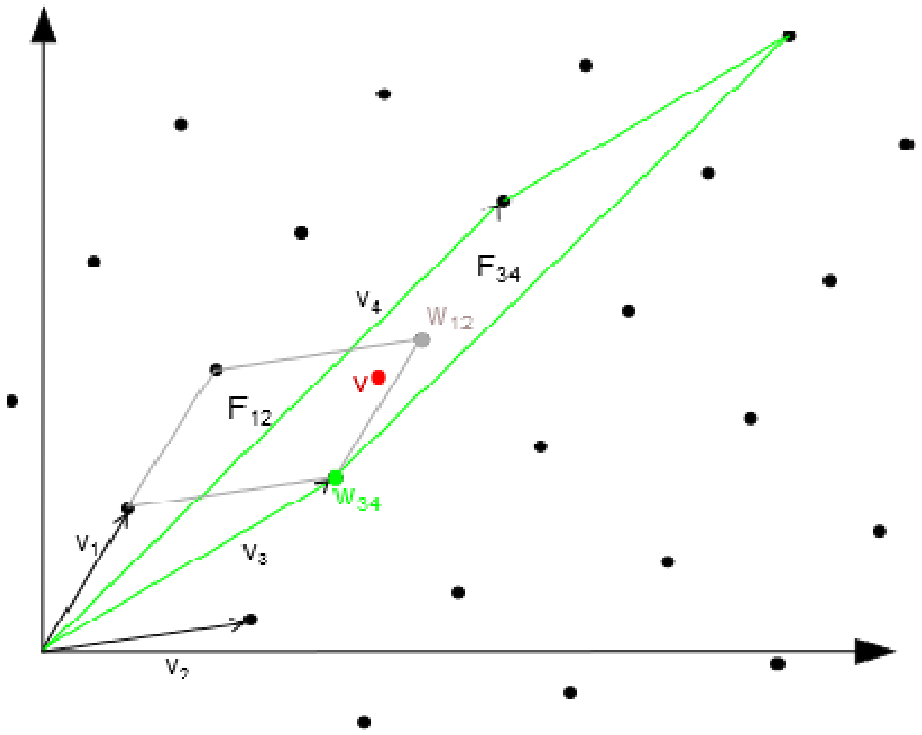


**FIG 3.4 –** Aproximação do CVP

Para isso, basta transladar o domínio fundamental de forma a conter o vetor *v.* O vértice do domínio fundamental mais próximo de *v* é a solução aproximada do CVP. A figura abaixo (FIG. 3.4) mostra a aproximação sendo feita com uma base boa {$v_1$, $v_2$} e com uma base ruim {$v_3$, $v_4$}.

A base {$v_1$, $v_2$} é uma base boa e o vetor $w_{12}$ obtido pela aproximação é a solução correta do CVP, pois é o vetor do reticulado que é mais próximo de *v*. Já a base {$v_3$, $v_4$} é uma base ruim e a aproximação obtida através dela foi o vetor $w_{34}$, que não é a solução do CVP.

De um modo geral, bases boas são constituídas de vetores razoavelmente ortogonais de módulo pequeno. Bases ruins são constituídas de vetores muito pouco ortogonais e bastante alongados.

Isso pode ser quantificado através de um valor conhecido como a Taxa de Hadamard relativa à base B:

$$\mathrm{H(B)} = \left(\frac{\mathrm{V(L)}}{||\mathrm{v_1}||\,||\mathrm{v_2}||\,\ldots\,||\mathrm{v_n}||}\right)^{1/\mathrm{n}},\ 0 \leq \mathrm{H(B)} \leq 1. \qquad \text{(EQ. 3.16)}$$

Quanto mais próximo de um for a Taxa de Hadamard, mais ortogonais são os vetores da base. Quanto mais próxima de zero ela for, menos ortogonais são os vetores da base. O princípio geométrico associado é do volume do paralelepípedo e

a medida dos seus lados. Quando os lados do paralelepípedo são ortogonais, seu volume é igual ao produto da medida de seus lados e *H(B)* = 1. Quando os lados do paralelepípedo formam ângulos muito agudos, seu volume é pequeno e H(B) ≈ 0.

O problema do vetor não nulo (*Shortest Vector Problem* - SVP), que consiste em encontrar o menor vetor não nulo do reticulado, é outro problema relevante na área de reticulados. Ele pode ser visto como um caso específico de CVP em que se deseja encontrar o vetor mais próximo do vetor nulo. Portanto, técnicas para resolver ou obter uma boa aproximação da solução do CVP também resolvem o SVP.

O método conhecido como Heurística de Gauss estima a norma do menor vetor não nulo como:

$$\lambda_{GAUSS}(\mathrm{L}) = \sqrt{\frac{n}{2\pi e}}\,\mathrm{V}(\mathrm{L})^{1/\mathrm{n}} \qquad \text{(EQ. 3.17)}$$

### 3.5.4 ALGORITMO DE BABAI

O Algoritmo de Babai [3] retorna a solução do CVP com o auxílio de uma base boa:

1 – Escreva $v$ como uma combinação linear dos vetores da base: $v = \sum t_i v_i$;

2 – Para cada $t_i$ faça $a_i$ ser o inteiro mais próximo de $t_i$;

3 – Retorne w = $a_1v_1 + a_2v_2 + ... + a_nv_n$.

A seguir será ilustrado um exemplo simples de utilização do Algoritmo de Babai:

Seja $\{v_1, v_2\}$ uma base boa do reticulado $L$ de dimensão 2:

$$v_1 = (1, -1) \text{ e } v_2 = (2, 2)$$

Deseja-se encontrar o vetor mais próximo de $v$ = (0.6, 2.2):

$v = -0.8v_1 + 0.7v_2 \rightarrow t_1 = -0.8$ e $t_2 = 0.7$

$a_1 = -1$ e $a_2 = 1$

$w = a_1v_1 + a_2 v_2 = (1, 3)$

Fim do Algoritmo de Babai.

### 3.5.5 TRANSFORMANDO UMA BASE RUIM EM UMA BASE BOA

Como vimos no Algoritmo de Babai, resolver o CVP com o auxílio de uma base

boa é tão fácil quanto resolver um sistema de equações lineares. Dessa forma, torna-se interessante poder transformar uma base qualquer de um reticulado em uma base boa, o que equivale a encontrar uma nova base com vetores pequenos e bastante ortogonais.

A solução do CVP é conhecidamente um problema difícil em reticulados, o que quer dizer que não existe nenhum algoritmo que possa calcular uma base boa com total certeza e em tempo polinomial. No entanto, existe um algoritmo para, a partir de uma base ruim do reticulado, obter uma base melhor que ela em tempo polinomial.

Essa base é conhecida como LLL reduzida (Lenstra, Lenstra e Lovász) e é obtida a partir de um algoritmo que utiliza o processo de ortogonalização de Gram-Schmidt descrito na seção 3.3.6. A seguir iremos recapitular os passos do AGS:

1 – Faça $v_1^* = v_1$;

2 – Para $i$ de 2 até $n$:

2.1 – Para $j$ de 1 até $i-1$:

2.1.1 – calcule $\mu_{ij} = v_i \circ v_j^* / ||v_j^*||^2$;

2.2 – Faça $v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{ij} . v_j^*$

3 – Retorne B* = { $v_1^*, v_2^*, ..., v_n^*$ };

Os coeficientes $\mu_{ij}$ utilizados no passo 2.2 não são inteiros. Desta forma, os vetores $v_i^*$ retornados pelo algoritmo não são pertencentes ao reticulado, pois não são combinações lineares dos vetores da base utilizando coeficientes inteiros (definição de reticulado).

A base descrita como LLL reduzida é aquela que atende a duas condições:

$$|\mu_{ij}| \leq \frac{1}{2}, \text{ para } j < i \qquad \text{(EQ. 3.18)}$$

$$||v_{i-1}^*||^2 \leq 2||v_i^*||^2 \qquad \text{(EQ. 3.19)}$$

De um modo geral, uma base LLL reduzida é uma excelente aproximação de uma base boa para valores de $n < 200$ e é considerada uma aproximação ruim para $n > 500$. [14]

O algoritmo descrito a seguir é a principal ameaça aos sistemas criptográficos que utilizam reticulado e é largamente utilizado para criptoanálise. Dada uma base qualquer $B = \{ v_1, v_2, ..., v_n \}$, o Algoritmo LLL retorna uma base LLL reduzida: [13]

1 – Utilizando o AGS, calcule os valores de $B^*$ e $\mu_{ij}$;

2 – $i = 2$;

2 – Enquanto $i \leq n$:

2.1 – Para $j$ de $i-1$ até 1:

2.1.1 – Faça $v_i = v_i - \lceil \mu_{ij} \rfloor v_j$, onde $\lceil \ \rfloor$ representa inteiro mais próximo;

2.1.2 – Utilizando o AGS, atualize os valores de $B^*$ e $\mu_{ij}$;

2.2 – Se $||v_{i-1}^*||^2 > 2||v_i^*||^2$ :

2.2.1 – troque de lugar $v_{i-1}$ e $v_i$;

2.2.2 – Utilizando o AGS, atualize os valores de $B^*$ e $\mu_{ij}$;

2.2.3 – Faça $i = \max(i-1, 2)$;

2.3 – Caso contrário faça $i = i+1$;

3 – Retorne { $v_1, v_2, ..., v_n$ };

A seguir será calculada uma base LLL reduzida a partir da base B = {v1, v2} para ilustrar a utilização do Algoritmo LLL:

$$v_1 = (1,2),\ v_2 = (1,3),\ \det(B) = 1$$

$$H(B) = \sqrt[4]{50} = 0.376$$

A base B é ruim, já que sua Taxa de Hadamard é baixa.

Utilizando o AGS, calculamos:

$$\mu_{11} = 1,\ v_1^* = (1,2) \text{ e } \mu_{21} = \frac{7}{5},\ v_2^* = (-\frac{2}{5}, \frac{1}{5})$$

No passo, 2.1.1:

$$v_2 = v_2 - \lceil \mu_{21} \rfloor v_1 = v_2 - v_1 = (0, 1)$$

Novos valores do AGS:

$$\mu_{11} = 1,\ v_1^* = (1,2) \text{ e } \mu_{21} = \frac{2}{5},\ v_2^* = (-\frac{2}{5}, \frac{1}{5})$$

No passo 2.2, como $||v_1^*||^2 > 2||v_2^*||^2$, trocamos $v_1$ e $v_2$ de lugar e i = 2

Utilizando o AGS, calculamos:

$$\mu_{11} = 1,\ v_1^* = (0,1) \text{ e } \mu_{21} = 2,\ v_2^* = (1,0)$$

No passo, 2.1.1:

$$v_2 = v_2 - \lceil \mu_{21} \rfloor v_1 = v_2 - 2v_1 = (1, 0)$$

Nesse caso, a variável i é incrementada e o algoritmo retorna {(0,1), (1,0)}.

A Taxa de Hadamard relativa à nova base LLL reduzida é 1, o que indica que ela é uma base boa (bem melhor que a base anterior).

# 4 PRIMEIROS CRIPTOSSISTEMAS DE CHAVE PÚBLICA

O conceito de criptografia de chave pública foi publicado pela primeira vez em 1976, por Diffie e Hellman, em seu artigo “New Directions in Cryptography”. Diffie e Hellman apresentaram ainda um método seguro para troca de chaves baseado em um problema difícil da álgebra modular, o Problema do Logaritmo Discreto.

O trabalho desenvolvido por esses dois criptógrafos foi um grande marco na história da criptografia e motivou vários outros pesquisadores a desenvolverem criptossistemas de chave pública e a descobrirem outros problemas matemáticos complexos.

Nesta seção serão vistos o Algoritmo de Diffie Hellman e dois criptossistemas também baseados na álgebra modular, o criptossistema de ElGamal e o RSA. Os criptossistemas que serão estudados baseiam-se em dois problemas difíceis, o Problema de Logaritmo Discreto (Discrete Logarithm Problem - DLP) e o Problema da Raiz em um Corpo Finito (Discrete Root Problem - DRP). As definições do DLP e do DRP se encontram abaixo: [3]

Sejam $p$ um número primo, g uma raiz primitiva de $F_p$ e $h$ um elemento não nulo de $F_p$. O DLP consiste em encontrar o expoente $x$ tal que $g^x = h$, onde o número $x$ é chamado de logaritmo discreto de $h$ na base $g$ e é denotado por $\log_g(h)$. Já o DRP consiste em encontrar a base x tal que $x^n = c$, onde x é chamado raiz *n-ésima* de $c$.

Até hoje não foi encontrada nenhuma solução em tempo polinomial para o DLP e o DRP, a fatoração de números inteiros em computadores convencionais. No entanto, Shor demonstrou que a construção de um computador quântico viabilizaria a solução desses problemas em tempo polinomial, o que quebraria todos os sistemas criptográficos que se baseiam neles.

## 4.1 ALGORITMO DE DIFFIE HELLMAN

O algoritmo de Diffie-Hellman tem como objetivo compartilhar uma chave secreta entre dois usuários através de um canal de comunicação inseguro. Ele é utilizado como gerador de chaves aleatórias para sistemas de criptografia simétrica.

Para isso, os valores de $p$ e $g$ precisam ser estabelecidos de forma que $p$ seja

um número primo grande e $g$ seja uma raiz primitiva de $F_p$. Métodos para facilitar a obtenção desses números podem ser obtidos em [3].

O processo de execução do algoritmo será exemplificado por dois usuários, Ana e Bruno, que desejam compartilhar uma chave secreta. Após a determinação dos valores de $p$ e $g$, cada um dos usuários deve escolher um inteiro menor que $p$ e mantê-lo em sigilo. Sejam $a$ e $b$ os valores escolhidos, respectivamente, por Ana e por Bruno.

Em seguida, Ana e Bruno trocam os valores

$$A = g^a \qquad \text{e} \qquad B = g^b \qquad \text{(EQ. 4.1)}$$

Qualquer pessoa pode ter acesso aos valores de $p$, $g$, $A$ e $B$, uma vez que eles foram todos trocados através de um meio de comunicação inseguro.

Finalmente, Ana e Bruno utilizam suas chaves secretas $a$ e $b$ para obter os valores $A` = A^b$ e $B` = B^a$, que são iguais, uma vez que, pela EQ. 4.1:

$$A` = (g^a)^b = g^{ab}$$

$$B` = (g^b)^a = g^{ba} = g^{ab}$$

Esse valor comum é a chave secreta que é transmitida através de um meio de comunicação inseguro sem que nenhuma pessoa além de Ana e Bruno possa descobri-la. Essa chave pode ser utilizada por eles em um sistema criptográfico de chave privada para criptografar suas mensagens.

Para que outra pessoa descubra o valor de A` = B`, é necessário que se conheça o valor de $a$ ou $b$. Como Ana e Bruno mantêm esses números em segredo, a única forma de encontrá-los seria resolvendo uma das equações: $A = g^a$ ou $B = g^b$, que são a mesma equação do DLP. Portanto, a segurança do algoritmo de Diffie Hellman está na dificuldade da solução do DLP.

## 4.2 CRIPTOSSISTEMA DE ELGAMAL

O criptossistema de ElGamal está diretamente relacionado ao algoritmo de Diffie Hellman e ao Problema do Logaritmo Discreto em um Corpo Finito. Uma vez que ele possui um algoritmo de criptografia e um algoritmo de decodificação, é possível a troca segura de qualquer mensagem numérica $m \in F_p$.

Para sua utilização, assim como no algoritmo de Diffie Hellman, também são necessários o estabelecimento dos valores de $p$ e $g$, de forma que $p$ seja um número

primo grande e $g$ seja uma raiz primitiva de $F_p$, e a escolha de uma chave privada por parte de cada usuário.

Usaremos novamente Ana e Bruno para ilustrar a utilização do criptossistema de ElGamal. Sejam $a$ e $b$ as chaves privadas, respectivamente, de Ana e Bruno. Em seguida, cada um deles deve anunciar os valores $A = g^a$ e $B = g^b$, que serão suas chaves públicas.

### 4.2.1 SIGILO

Vamos supor que Bruno deseja enviar uma mensagem $m$ para Ana, de forma que ninguém além dela possa ler a mensagem. Para isso, Bruno deve escolher um número $k \in F_p$, de forma que $k$ e $p$ sejam primos entre si. O número $k$, conhecido também como *ephemeral key* [3], é usado apenas com o intuito de criptografar a mensagem e depois deve ser descartado.

Em seguida, Bruno deve calcular os valores $c_1$ e $c_2$:

$$c_1 = g^k \qquad \text{(EQ. 4.2)}$$

$$c_2 = mA^k \qquad \text{(EQ. 4.3)}$$

A mensagem cifrada é o par ($c_1$, $c_2$), que é enviado à Ana. Para que ela possa ler a mensagem original, ela precisa usar sua chave privada $a$ para calcular $x$ a partir da EQ. 4.2:

$$x = c_1^a = g^{ka} = (g^a)^k = A^k \qquad \text{(EQ. 4.4)}$$

Ana pode recuperar a mensagem $m$ multiplicando $c_2$ pelo inverso multiplicativo de $x$, uma vez que, através das equações EQ. 4.3 e EQ. 4.4:

$$c_2 x^{-1} = mA^k x^{-1} = mA^k A^{-k} = m$$

Para que outra pessoa possa recuperar $m$ a partir de $c_1$ e $c_2$, é necessário descobrir o valor de $k$ a partir da equação EQ. 4.2, que é a mesma equação do DLP. Portanto, o sigilo das mensagens no criptossistema de ElGamal depende unicamente da dificuldade da solução do DLP.

### 4.2.2 AUTENTICIDADE

Vamos supor agora que Bruno deseja assinar a mensagem, de forma que Ana possa ter certeza de que foi ele quem enviou. Para isso, ele precisa usar sua chave

privada *b* para criar uma assinatura de forma que Ana, conhecendo a chave pública *B* de Bruno, possa confirmá-la.

A assinatura para uma mensagem *m* é um par (*r*, *s*), escolhido de maneira que seja válida a equação de verificação [15]:

$$g^m = B^r r^s \qquad \text{(EQ. 4.5)}$$

Para isso, Bruno precisa escolher um número aleatório $k \in F_p$ tal que *k* e p sejam primos entre si, para poder calcular *r :* (*k* é um *ephemeral key*)

$$r = g^k \qquad \text{(EQ. 4.6)}$$

A partir de EQ. 4.6 e EQ. 4.1 é possível desenvolver EQ 4.5 para obter o valor de s:

$$g^m = g^{br} g^{ks} = g^{br + ks}$$

$$m = br + ks$$

$$s = (m - br)\, k^{-1} \qquad \text{(EQ. 4.7)}$$

Para que outra pessoa tente se passar por Bruno, forjando sua assinatura, ela precisa determinar valores para *r* e *s* que atendam à equação EQ. 4.5. Para isso, ele pode tentar resolver a equação EQ. 4.5 fixando um valor para *r* ou fixando um valor para *s*. Em ambas as formas, a dificuldade na solução da equação é a mesma do DLP. Portanto, a autenticidade das mensagens no criptossistema de ElGamal depende da dificuldade da solução do DLP.

## 4.3 CRIPTOSSISTEMA RSA

O criptossistema RSA está relacionado com o Problema das Raízes Discretas em um Corpo Finito e à dificuldade de fatoração de números grandes. Suas equações permitem a troca segura de qualquer mensagem numérica $m \in F_p$.

Para isso, cada usuário precisa estabelecer dois números primos secretos *p* e *q*, e publicar o produto *n = pq.* Apesar de *n* ser público, os números *p* e *q* ficam protegidos devido à enorme dificuldade em fatorá-lo [16], uma vez que *n* deve ser um número com mais de 200 algarismos.

Em seguida, cada usuário deve publicar um expoente *e* para ser usado para criptografar as mensagens, de tal forma que *e* seja primo com $\phi$(n), onde $\phi$ é a função de Euler, que para *n = pq* vale

$$\phi(p.q) = (p-1).(q-1)$$

A chave privada será o inteiro *d,* tal que

$$ed = 1 \pmod{\phi(n)} \qquad \text{(EQ. 4.8)}$$

A partir dos valores de *e* e $\phi$(n), pode-se calcular *d* em tempo polinomial com o algoritmo do cálculo do inverso multiplicativo, descrito no capítulo 3.2.1

As equações utilizadas para criptografar e descriptografar as mensagens são:

$$C(M) = M^e \qquad \text{(EQ. 4.9)}$$

$$D(C) = C^d \qquad \text{(EQ. 4.10)}$$

Desejamos provar que as funções C e D são funções inversas, ou seja:

$$D(C(M)) = C(D(M)) = M$$

Da forma que foram definidas as equações EQ. 4.8 e EQ. 4.9, temos

$$D(C(M)) = C(D(M)) = M^{ed} \qquad \text{(EQ. 4.11)}$$

Por EQ 4.8, podemos afirmar que existe um inteiro *k* tal que

$$ed = k.\phi(n) + 1 \qquad \text{(EQ. 4.12)}$$

A partir de EQ. 4.12, podemos desenvolver $M^{ed}$ para obter

$$M^{ed} = M^{k.\phi(n)\,+\,1} = M.M^{k.\phi(n)} = M.\,(M^{\phi(n)})^k \qquad \text{(EQ. 4.13)}$$

Pelo Teorema de Euler, $M^{\phi(n)} = 1 \pmod{n}$. Ao substituir em EQ. 4.13, vemos que

$$M^{ed} = M.\,(M^{\phi(n)})^k = M.\,(1)^k = M$$

Isso garante que as equações do criptossistema RSA para criptografar e descriptografar as mensagens são funções inversas. Além disso, o conhecimento da fatoração do número *n* e da função de Euler permite a solução do DRP em um Corpo Finito $F_n$. Para que a fatoração de *n* seja computacionalmente inviável e o criptossistema seja seguro, os valores de (*p, q, e)* devem ser escolhidos conforme [16].

Para ilustrar a utilização do Criptossistema RSA para garantir o sigilo e a autenticidade das mensagens, utilizaremos novamente Ana e Bruno. As chaves públicas de Ana e Bruno são, respectivamente, os pares ($n_A$, $e_A$) e ($n_B$, $e_B$), enquanto que suas chaves privadas são, respectivamente, os pares ($n_A$, $d_A$) e ($n_B$, $d_B$).

Na tabela TAB. 4.1 são apresentadas todas as possíveis maneiras de Bruno enviar uma mensagem *m* para Ana, de forma a garantir, respectivamente, sigilo, autenticidade, e ambos.

**TAB 4.1 –** Troca de Mensagens RSA

| | | |
|---|---|---|
| SIGILO | Bruno envia | $c = m^{e_A}$ |
| | Ana lê | $m = c^{d_A}$ |
| AUTENTICIDADE | Bruno envia | $c = m^{d_B}$ |
| | Ana lê | $m = c^{e_B}$ |
| AMBOS | Bruno envia | $c = m^{e_A d_B}$ |
| | Ana lê | $m = c^{d_A e_B}$ |

O RSA é um dos criptossistemas mais utilizados atualmente. Sua principal desvantagem, no entanto, é a demora para criptografar e decriptografar as mensagens. Estudos comparativos apontam uma diferença de cerca de 10 vezes no tempo de execução do RSA em relação ao ECC, um criptossistema baseado em curvas elípticas [17] e de até 100 vezes em relação aos criptossistemas baseados em reticulado [14].

# 5 CRIPTOSSISTEMAS BASEADOS EM RETICULADO

A utilização de reticulados em criptografia foi proposta pela primeira vez por Ajtai em 1995 [18]. O trabalho de Ajtai, apesar de não ter produzido um bom resultado prático, foi extremamente importante sob o ponto de vista teórico e serviu de motivação para o estudo e maior desenvolvimento da área.

Nesta seção serão apresentados dois criptossistemas que se baseiam em problemas difíceis relacionados aos reticulados, o GGH e o NTRU. As definições e os conceitos matemáticos relacionados aos reticulados podem ser encontrados no capítulo 3 e sua leitura é fundamental para o entendimento dos criptossistemas.

## 5.1 CRIPTOSSISTEMA GGH

O criptossistema GGH está diretamente relacionado ao CVP e à sua solução aproximada pelo algoritmo de Babai. Apesar de ser muito mais rápido para criptografar e descriptografar as mensagens, sua principal desvantagem, e a causa de sua ineficiência, é o enorme tamanho que suas chaves precisam ter para que sejam seguras.

Utilizaremos Ana e Bruno para ilustrar a utilização do GGH. Cada um deles, a partir de um reticulado $L$ de dimensão $n$, deve escolher uma base boa $V$ para sua chave privada e uma base ruim $W$ para chave pública. Desta forma, a segurança do GGH está na dificuldade de se obter uma base boa a partir de uma base ruim do reticulado.

Bruno deseja enviar uma mensagem $M = (m_1, m_2, ..., m_n)$ para Ana, $M \in Z^n$. Para criptografar a mensagem, ele deve gerar um pequeno vetor aleatório $r$ e escrever $M$ em função da chave pública $W_A$ de Ana conforme a equação abaixo:

$$C = m_1w_{A1} + m_2w_{A2} + ... + m_nw_{An} + r \qquad \text{(EQ. 6.1)}$$

Se o vetor $r$ for pequeno, o vetor $c$ será próximo ao ponto do reticulado

$$P = (m_1w_{A1}, m_2w_{A2}, ..., m_nw_{An})$$

Para recuperar a mensagem, Ana utiliza o algoritmo de Babai para obter P a partir de C e de sua chave privada, a base boa $V_A$. O esquema é representado na

figura FIG. 6.1 para o valor de $n$ = 2:

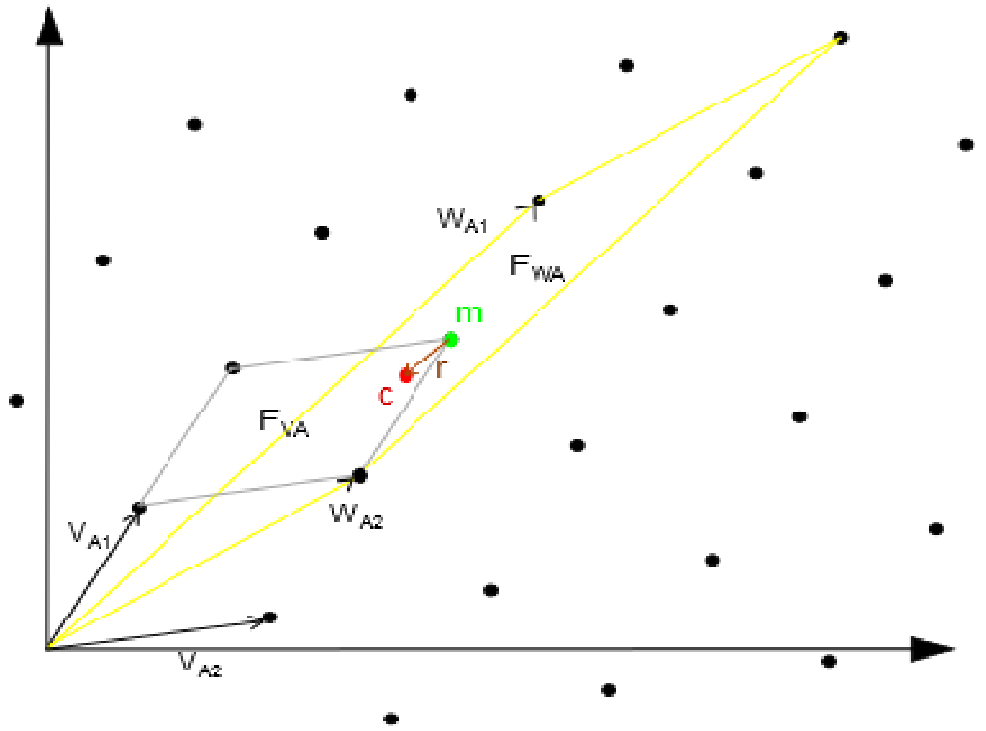


**FIG 6.1 –** GGH em Duas Dimensões

No GGH de dimensão $n$, a mensagem cifrada $c$ é um vetor de $R^n$ que possui uma distância pequena da mensagem original $m$. Para recuperar essa mensagem, é necessário se obter uma base boa do reticulado. Portanto, qualquer ataque ao criptossistema consiste na tentativa de se obter uma base boa (chave privada) a partir de uma base ruim (chave pública), o que pode ser feito utilizando o algoritmo LLL.

Para que o criptossistema GGH seja seguro, o algoritmo LLL não pode ser capaz de recuperar uma base boa do reticulado. Para isso acontecer, a dimensão $n$ deve ser superior a 500. Computacionalmente, isso é equivalente a uma matriz de 250000 números inteiros, que na maioria dos computadores ocupa um espaço de um megabyte. Esse tamanho enorme das chaves torna o criptossistema GGH impraticável.

## 5.2 CRIPTOSSISTEMA NTRU

O criptossistema NTRU está relacionado à álgebra em anéis polinomiais, mas seu problema difícil pode ser relacionado ao CVP em um reticulado. Ele é composto

por duas partes independentes: o NTRU-*encript*, utilizado para garantir o sigilo das mensagens, e o NTRU-*sign*, utilizado para garantir a autenticidade.

Nesta seção veremos o NTRU-encript de forma bem detalhada, analizando matematicamente suas equações e técnicas para tentar quebrá-lo. Como será visto no capítulo 5.2.3, o problema de recuperar sua chave privada a partir da chave pública pode ser transformado no problema de obter o menor vetor de um tipo de reticulado obtido com essas chaves, denominado NTRU-*lattice* [3].

O esquema de autenticação do NTRU foi desenvolvido de forma complementar e apresentado pela primeira vez em 2001, cinco anos depois da introdução da primeira versão do criptossistema NTRU em 1996, que previa apenas o NTRU-*encript*. Em sua primeira forma, chamada NSS (NTRU Signature Scheme [19]), a assinatura das mensagens dependia de parâmetros probabilísticos e podia falhar mesmo tendo sido feita a partir da chave privada. Com auxílio de outros dois criptógrafos, ela foi melhorada em 2002 e passou a ser chamada NTRU-*sign*, que será vista no capítulo 5.2.4.

### 5.2.1 NTRU-ENCRIPT

Para utilização do NTRU-*encript*, devem ser fixados um primo N e os módulos p e q, de forma que MDC(p, N) = MDC(p, q) = 1, para definir os anéis polinomiais *R*, $R_p$ e $R_q$:

$$R = \frac{Z[x]}{(x^n\text{-}1)}, \qquad R_p = \frac{Z_p[x]}{(x^n\text{-}1)}, \qquad R_q = \frac{Z_q[x]}{(x^n\text{-}1)}$$

Fixados os anéis polinomiais, um último inteiro d deve ser escolhido de forma que os parâmetros satisfaçam a seguinte equação:

$$q > p.(6d + 1) \qquad \text{(EQ. 6.2)}$$

Ana e Bruno desejam utilizar o NTRU-*encript*. Para isso, cada um deles precisa escolher dois polinômios ternários para serem suas chaves privadas:

$$f(x) \in T(d+1, d) \text{ e } g(x) \in T(d, d)$$

O polinômio f(x) deve ter inverso multiplicativo tanto em $R_p$ como em $R_q$, denotados respectivamente por $f^{-1}{}_p(x)$ e $f^{-1}{}_q(x)$. Enquanto um desses inversos não existir, f(x) deve ser descartado e substituído até que ambos os inversos existam. Como nenhum polinômio pertencente a T(d, d) possui inverso multiplicativo em $R_p$,

não é possível escolher g(x) de forma a ele ter inverso. Isso acontece porque MDC(g(x), $x^n$-1) ≠ 1, já que o polinômio x -1 é um fator comum: ambos tem a soma de seus coeficientes iguais a zero (ver seção 3.3.1).

Atendidas as restrições, cada um deles revela sua chave pública:

$$h(x) = f^{-1}_q(x) * g(x) \pmod q \qquad \text{(EQ. 6.3)}$$

### 5.2.2 CRIPTOANÁLISE DO NTRU-ENCRIPT

Para quebrar o NTRU-*encript*, assim como para quebrar qualquer criptossistema, é preciso recuperar a chave privada a partir da chave pública. Nesse caso, isso é equivalente a encontrar os polinômios f(x) ∈ T(d+1, d) e g(x) ∈ T(d, d) a partir da equação EQ. 6.4:

$$f(x) * h(x) = g(x) \pmod q \qquad \text{(EQ. 6.4)}$$

Diversas técnicas podem ser utilizadas para tentar recuperar os polinômios f e g da equação EQ. 6.12, como força bruta e algoritmos de tratamento de colisão. Segundo [3], a melhor complexidade já encontrada é de $o(3^{n/2}/\sqrt{n})$, que ainda sim é uma complexidade muito alta.

No entanto, é possível resolver o problema da EQ 6.4 a partir de certo tipo de reticulado denominado $L_{NTRU}$, cuja base pode ser disposta por linhas na matriz *M:*

$$M = \begin{pmatrix} I & H \\ 0 & qI \end{pmatrix} \qquad \text{(EQ. 6.5)}$$

A base do reticulado $L_{NTRU}$ é uma matriz 2n x 2n organizada em quatro blocos: os cantos superior esquerdo, superior direito, inferior esquerdo e inferior direito, onde estão, respectivamente:

- A matriz identidade
- A matriz H, composta por todas as rotações da base pública h
- A matriz nula
- A matriz identidade multiplicada por q.

A matriz H será representada abaixo:

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_{n-1} & h_0 & \cdots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_0 \end{pmatrix} \qquad \text{(EQ. 6.6)}$$

Dado o reticulado $L_{NTRU}$ definido pela EQ. 6.5, será demonstrado que ele

contém o vetor $(f \quad g)$, onde f e g são os polinômios da equação EQ. 6.4. Para isso reescreveremos a equação da seguinte maneira:

$$f(x) * h(x) = g(x) + qu(x), \text{ onde } u(x) \in R$$

$$g(x) = f(x) * h(x) - qu(x) \qquad \text{(EQ. 6.7)}$$

Expressando g(x) na forma de vetor, obtemos:

$$g(x) = \begin{pmatrix} h_0f_0+h_{n-1}f_1+\ldots+h_1f_{n-1} - qu_0 \\ h_1f_0+h_0f_1+\ldots+h_2f_{n-1} - qu_1 \\ \vdots \\ h_{n-1}f_0+h_{n-2}f_1+\ldots+h_0f_{n-1} - qu_{n-1} \end{pmatrix} \qquad \text{(EQ. 6.8)}$$

Executando o produto da base *M* de $L_{NTRU}$ pelo vetor $\begin{pmatrix} f \\ -u \end{pmatrix}$, obtemos:

$$(f \quad -u)\begin{pmatrix} I & H \\ 0 & qI \end{pmatrix} = (f \quad fH - qu) \qquad \text{(EQ. 6.9)}$$

Executando o produto do vetor f pela matriz H, obtemos:

$$(f_0 \quad f_1 \quad \cdots \quad f_{n-1})\begin{pmatrix} h_0 & h_{n-1} & \cdots & h_1 \\ h_1 & h_0 & \cdots & h_2 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_{n-2} & \ldots & h_0 \end{pmatrix} = \begin{pmatrix} h_0f_0+h_{n-1}f_1+\ldots+h_1f_{n-1} \\ h_1f_0+h_0f_1+\ldots+h_2f_{n-1} \\ \vdots \\ h_{n-1}f_0+h_{n-2}f_1+\ldots+h_0f_{n-1} \end{pmatrix} \qquad \text{(EQ. 6.10)}$$

Pelas equações EQ. 6.8, EQ. 6.9 e EQ. 6.10, verifica-se uma nova igualdade:

$$(f \quad -u)\begin{pmatrix} I & H \\ 0 & qI \end{pmatrix} = (f \quad g) \qquad \text{(EQ. 6.11)}$$

Pela equação EQ. 6.11, fica claro que o vetor v = $(f \quad g)$ pode ser escrito como uma combinação linear dos vetores da base do reticulado $L_{NTRU}$, o que prova que ele pertence ao reticulado. Além disso, como os coeficientes de f e g estão entre -1, 0 e 1, provavelmente v é o vetor de menor módulo em $L_{NTRU}$. Dessa forma, a melhor maneira de tentar quebrar o NTRU-*encript* é utilizar o algoritmo LLL na matriz M para encontrar uma base boa e depois o algoritmo de Babai para encontrar o vetor mais próximo de zero (seções 3.5.4 e 3.5.5).

O NTRU-*encript* já foi quebrado algumas vezes pelo algoritmo LLL, porém sua versão mais atual é bastante segura. Em [14], o tempo estimado para quebrá-lo com n=250 é de $10^{10}$ anos, e com n=250 de $10^{20}$ anos, o que indica uma segurança aproximadamente igual à do RSA.

### 5.2.3 FUNCIONAMENTO DO NTRU-ENCRIPT

Bruno deseja enviar uma mensagem $M = (m_1, m_2, ..., m_n)$ para Ana, onde $M$ é a transposição simétrica de um polinômio de $R_p$, ou seja, um polinômio cujos coeficientes estão entre $^{p}/_{2}$ e $-\,^{p}/_{2}$ e são inteiros. Para criptografar a mensagem, ele deve gerar um vetor aleatório $r \in T(d, d)$ e escrever C em função de $M$ e da chave pública $h_A$ de Ana conforme a equação abaixo:

$$C(x) = p(h(x) * r(x)) + M(x) \pmod q \qquad \text{(EQ. 6.12)}$$

Para recuperar a mensagem original, Ana precisa usar sua chave privada f(x) e seu inverso $f^{-1}_{p}(x)$. Primeiramente, ela deve calcular um valor auxiliar a(x):

$$a(x) = f(x) * C(x) \pmod q \qquad \text{(EQ. 6.13)}$$

Finalmente, ela utiliza $f^{-1}_{p}(x)$ e faz a transposição simétrica para recuperar $M`(x)$:

$$M`(x) = f^{-1}_{p}(x) * a(x)_{t} \pmod p \qquad \text{(EQ. 6.14)}$$

A seguir será feita a análise matemática das equações, a fim de verificar se a mensagem $M`(x)$ recuperada por Ana realmente é igual à mensagem original $M(x)$.

### 5.2.4 ANÁLISE MATEMÁTICA DAS EQUAÇÕES

Multiplicando ambos os lados de EQ. 6.12 por f(x) e utilizando EQ. 6.13 e EQ.6.14, obtemos uma nova equação:

$$f(x) * C(x) = p(f(x) * f^{-1}_{q}(x) * g(x) * r(x)) + f(x) * M(x) \pmod q$$

$$a(x) = p(g(x) * r(x)) + f(x) * M(x) \pmod q \qquad \text{(EQ. 6.15)}$$

Consideraremos a igualdade descrita na equação EQ. 6.15. Como g(x) e r(x) pertencem a T(d, d), os coeficientes de $g(x) * r(x)$ estão entre 2d, no caso extremo em que os d coeficientes 1 de g(x) multiplicam os d coeficientes 1 de r(x) e os d coeficientes -1 de g(x) multiplicam os d coeficientes -1 de f(x), e -2d, no caso inverso. Desta forma, os coeficientes de $p(g(x) * r(x))$ podem variar de -2dp a 2dp.

Como f(x) pertence a T(d+1, d) e os coeficientes de M(x) estão entre $^{p}/_{2}$ e $-\,^{p}/_{2}$, os coeficientes do produto $f(x) * M(x)$ estão entre $(2d+1)\,^{p}/_{2}$, no caso extremo em que todos os d+1 coeficientes 1 de f(x) multiplicam coeficientes $^{p}/_{2}$ de M(x) e os d coeficientes -1 de f(x) multiplicam coeficientes $-\,^{p}/_{2}$ de M(x), e $-\,(2d+1)\,^{p}/_{2}$, no caso inverso. Desta forma, os coeficientes de a(x) podem variar entre $(6d+1)\,^{p}/_{2}$, no caso

em que o maior coeficiente possível de $p(g(x) * r(x))$ e o maior coeficiente possível de $f(x) * M(x)$ coincidem, e $-(6d+1)^{p}/_{2}$, no caso inverso.

Pela equação EQ. 6.2, podemos afirmar a diferença entre o maior coeficiente de a(x) e seu menor coeficiente é sempre menor que q. Isso garante que a igualdade da equação EQ 6.15 com todos os coeficientes mod q implica que a mesma igualdade também vale para os coeficientes originais. Por isso, a EQ. 6.2 é fundamental para que o funcionamento correto do criptossistema NTRU [3], já que possibilita uma nova equação:

$$a(x)_t = p(g(x) * r(x)) + f(x) * M(x) \qquad \text{(EQ. 6.16)}$$

Aplicando mod p na equação EQ. 6.16, podemos cancelar $p(g(x) * r(x))$:

$$a(x)_t = f(x) * M(x) \pmod{p} \qquad \text{(EQ. 6.17)}$$

Multiplicando por $f^{-1}_p(x)$ e aplicando a transposição simétrica, podemos utilizar a equação EQ. 6.14 para chegar em:

$$M`(x) = M(x) \qquad \text{(EQ. 6.18)}$$

O que conclui a análise matemática das equações.

### 5.2.5 NTRU-SIGN

O NTRU-*sign* é baseado no problema do CVP em um reticulado. A assinatura da mensagem, com o auxílio da chave pública, deve permitir a obtenção da solução do CVP, enquanto que uma assinatura inválida gera uma solução incorreta. Desta forma, a chave privada deve ser capaz de gerar uma base boa para o reticulado e a chave pública uma base ruim, exatamente a mesma idéia do criptossistema GGH.

No entanto, sabemos que o algoritmo LLL é capaz de transformar uma base ruim em uma base boa, o que torna o NTRU-*sign* inseguro para $n < 500$. Além disso, sabemos que essa desvantagem faz com o tamanho da matriz seja grande demais para que ela possa ser utilizada como chave pública ou chave privada, motivo pelo qual o criptossistema GGH não é prático.

A solução encontrada para esse problema foi utilizar como chave pública e chave privada polinômios que possam ser posteriormente transformados em matrizes. Desta forma, o tamanho das chaves seria pequeno, já que 500 números inteiros não ocupam mais de 2 kB de espaço em um computador. Para isso definiremos a matriz associada ao polinômio p como:

$$m(p) = \begin{pmatrix} p_0 & p_1 & \cdots & p_{n-1} \\ p_{n-1} & p_0 & \cdots & p_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ p_1 & p_2 & \cdots & p_0 \end{pmatrix} \quad \text{(EQ. 6.19)}$$

A matriz m(p) possui uma propriedade como mostrado nas equações EQ. 6.8 e EQ. 6.10:

$$f(x) * p(x) = f(x) * m(p) \quad \text{(EQ. 6.20)}$$

Para utilização do NTRU-*sign*, primeiro devem ser fixados: um primo N, um módulo q, de forma que MDC(q, N) = 1, e um inteiro d.

Bruno deseja utilizar o NTRU-*sign* para assinar seus documentos. Para isso, ele precisa escolher dois polinômios ternários para serem suas chaves privadas:

$$f(x) \in T(d+1, d) \text{ e } g(x) \in T(d+1, d)$$

A partir de sua chave privada, Bruno é capaz de obter as matrizes m(f) e m(g), ambas de dimensão n. A partir delas, ele deve ser capaz de obter outros dois vetores F e G tais a matriz B seja uma base boa do reticulado tal que det(B) = q:

$$B = \begin{pmatrix} m(f) & m(g) \\ m(F) & m(G) \end{pmatrix} \quad \text{(EQ. 6.21)}$$

O processo para obtenção dos vetores F e G pode ser visto em [20].

Em seguida ele revela sua chave pública:

$$h(x) = f^{-1}_q(x) * g(x) \pmod q \quad \text{(EQ. 6.22)}$$

A base ruim M associada à chave pública é a mesma da equação EQ. 6.5:

$$H = \begin{pmatrix} I & m(h) \\ 0 & qI \end{pmatrix} \quad \text{(EQ. 6.23)}$$

### 5.2.6 FUNCIONAMENTO DO NTRU-SIGN

Bruno deseja enviar uma assinatura para a mensagem $M = (M_1, M_2)$ para Ana, onde $M_1$ *e* $M_2$ são polinômios pertencentes a $R_q$. Para isso, ele deve utilizar o algoritmo de Babai para encontrar o vetor mais próximo de *M.* Os três passos do algoritmo de Babai são:

1 – Escreva *M* como uma combinação linear dos vetores da base: $M = \sum t_i B_i$;

2 – Para cada $t_i$ faça $a_i$ ser o inteiro mais próximo de $t_i$;

3 – Retorne $w = a_1B_1 + a_2B_2 + ... + a_nB_n$.

Do passo 1, obtemos a seguinte equação:

$$M = TB$$

$$T = MB^{-1} \quad \text{(EQ. 6.24)}$$

Como B é uma matriz de determinante q, a matriz inversa de B pode ser calculada pelo método dos cofatores [12]:

$$B^{-1} = \frac{1}{q}\begin{pmatrix} m(G) & -m(g) \\ -m(F) & m(f) \end{pmatrix} \quad \text{(EQ. 6.25)}$$

A partir da equação EQ. 6.24 e da equação EQ. 6.20, podemos calcular o vetor T:

$$T = \frac{1}{q}(M_1 \quad M_2)\begin{pmatrix} m(G) & -m(g) \\ -m(F) & m(f) \end{pmatrix}$$

$$T = \frac{1}{q}\left(M_1 * G - M_2 * F \quad -M_1 * g + M_2 * f\right) \quad \text{(EQ. 6.26)}$$

No passo 2, devemos substituir cada coeficiente de T pelo seu inteiro mais próximo. Seja $A = (A_1 \quad A_2)$ o vetor com os novos coeficientes. O vetor W, que é a solução do CVP, pode ser obtido a partir de A através do passo 3:

$$W = (A_1 \quad A_2)\begin{pmatrix} m(f) & m(g) \\ m(F) & m(G) \end{pmatrix}$$

$$W = (A_1 * f + A_2 * F \quad A_1 * g + A_2 * G) \quad \text{(EQ. 6.27)}$$

A assinatura enviada por Bruno são os n primeiros coeficientes de W:

$$s = A_1 * f + A_2 * F \quad \text{(EQ. 6.28)}$$

Para Ana conferir a validade da assinatura, ela usa a chave pública h de Bruno para recuperar os últimos n coeficientes de $W = (s \quad t)$ que ele não enviou, conforme a equação abaixo:

$$t = s * h \quad \text{(EQ. 6.29)}$$

Por definição, como o ponto $(s \quad s * h)$ está no reticulado, pois ele pode ser obtido a partir da base pública fazendo:

$$(s \quad 0)\begin{pmatrix} I & m(h) \\ 0 & qI \end{pmatrix} = (s \quad s * h)$$

A verificação da assinatura de Bruno é feita através de sua chave pública h, recuperando a possível solução do CVP. Pela Heurística de Gauss (EQ. 3.17), a distância da solução enviada por Bruno até a mensagem *M* não pode ser maior do que $\lambda_{GAUSS}$ multiplicado por um fator de $\sqrt{n}$, onde *n* é a dimensão do reticulado. Desta forma, se alguém tentar forjar a assinatura de Bruno, enviando uma falsa solução do CVP, Ana irá rejeitá-la através da Heurística de Gauss.

Outra técnica utilizada para tentar forjar uma assinatura é o chamado

*Parallelepiped Atack*, que consiste recuperar a chave privada a partir de vários pares de mensagens e assinaturas conhecidas. Segundo [21], para N = 250, com apenas 400 pares é possível quebrar o NTRU-*sign,* o que o tornaria totalmente inseguro. No entanto, esse problema pode ser contornado com o auxílio de um algoritmo que modifica, a cada assinatura, os polinômios F e G obtidos a partir da chave privada, como descrito em [22].

A seguir será apresentado um pequeno exemplo de utilização do criptossistema NTRU (NTRU-*encript* + NTRU-*sign*).

### 5.2.7 PEQUENO EXEMPLO DE UTILIZAÇÃO DO NTRU

Bruno e Ana estão usando o criptossistema NTRU para trocar mensagens. Para o NTRU-*encript,* foram estabelecidas as seguintes constantes:

$$N = 5,\ p = 2,\ q = 17,\ d = 1$$

As chaves privadas de Ana no NTRU-encript são:

$$f_A(x) = x^3 - x + 1 \in T(2, 1),\ g_A(x) = x - 1 \in T(1, 1)$$

Em seguida, ela calcula as inversas $f_A^{-1}{}_p(x)$ e $f_A^{-1}{}_q(x)$. A partir das divisões sucessivas do Algoritmo de Euclides, o MDC entre $f_A(x)$ e $(x^5-1) \equiv (x^5+1)$ em $R_2$:

$x^5+1 = (x^3 - x + 1)(x^2+1) + (x^2+x) \rightarrow q_1 = x^2+1;$

$x^3 - x + 1 = (x^2+x)(x+1) + 1 \rightarrow q_2 = x+1;$

$x^2+x = (1)(x^2+x) + 0;$

Fim do Algoritmo de Euclides: MDC = 1. Agora aplicaremos a recursão do AES:

$$\begin{cases} x_{-1} = 1,\ x_0 = 0,\ x_k = x_{k-2} - q_k x_{k-1} \\ y_{-1} = 0,\ y_0 = 1,\ y_k = y_{k-2} - q_k y_{k-1} \end{cases}$$ , onde $q_i$ são os coeficientes do AE

$y_1 = y_{-1} - q_1 y_0 = -(x^2+1);$

$y_2 = y_0 - q_2 y_1 = 1 + (x+1)(x^2+1) = x^3+x^2+x;$

Fim do Algoritmo de Euclides Estendido:

$$(x^3 - x + 1)^{-1} \bmod (x^5-1) = (x^3+x^2+x) \text{ em } R_2$$

Calcularemos agora o MDC entre $f_A(x)$ e $(x^3-1) \equiv (x^5+16)$ em $R_{17}$:

$x^5+16 = (x^3 - x + 1)(x^2+1) + (16x^2+x+15) \rightarrow q_1 = x^2+1;$

$x^3 - x + 1 = (16x^2+x+15)(16x+16) + (15x+16) \rightarrow q_2 = 16x+16;$

$16x^2+x+15 = (15x+16)(9x+12) + 10 \rightarrow q_3 = 9x+12;$

$15x+16 = (10)(10x+5) + 0;$

Fim do Algoritmo de Euclides: MDC = 10, $10^{-1}$ (mod 17) = 12. Agora aplicaremos a recursão do AES:

$$\begin{cases} x_{-1} = 1, x_0 = 0, x_k = x_{k-2} - q_k x_{k-1} \\ y_{-1} = 0, y_0 = 1, y_k = y_{k-2} - q_k y_{k-1} \end{cases}$$ , onde $q_i$ são os coeficientes do AE

$y_1 = y_{-1} - q_1 y_0 = -(x^2+1)$;

$y_2 = y_0 - q_2 y_1 = 1 + (16x+16)(x^2+1) = 16x^3+16x^2+16x$;

$y_3 = y_1 - q_3 y_2 = -(x^2+1) - (16x^3+16x^2+16x)(9x+12) = x^4+x^3+16$;

Fim do Algoritmo de Euclides Estendido:

$$(x^3 - x + 1)^{-1} \bmod (x^5-1) = (6x^4+14x^3+2x^2+8x+5) \text{ em } R_{17}$$

As inversas calculadas a partir das chaves privadas de Ana são:

$$f_A^{-1}{}_p(x) = x^3+x^2+x$$

$$f_A^{-1}{}_q(x) = 6x^4+14x^3+2x^2+8x+5$$

Em seguida Ana divulga sua chave pública:

$$h_A(x) = f^{-1}{}_q(x) * g(x) \pmod q = 8x^4+5x^3+6x^2+14x+1$$

Para o NTRU-*sign* foram estabelecidas as seguintes constantes [3]:

$$N = 11, q = 23, d = 3$$

As chaves privadas de Bruno no NTRU-*sign* são:

$$f_B(x) = -x^{10}+x^9-x^8+x^4+x^3-x+1 \in T(4, 3),$$

$$g_B(x) = x^9-x^6-x^5+x^3-x^2+x+1 \in T(4, 3)$$

Utilizando o Algoritmo de Euclides, Bruno calcula a inversa de f(x) e divulga sua chave pública:

$$f_B^{-1}{}_q(x) = 11x^{10}+12x^8+11x^7+9x^6+5x^5+x^4+20x^3+21x^2+8x+18$$

$$h_B(x) = f_B^{-1}{}_q(x) * g_B(x) = 7x^{10}+12x^9+10x^8+19x^7+12x^6+6x^5+21x^4+15x^2+2x+12$$

Os vetores $F_B$ e $G_B$ obtidos a partir de $f_B(x)$ e $g_B(x)$ são:

$$F_B(x) = -x^9-x^8+x^7-5x^6-x^5+x^3-2x^2-x-3$$

$$G_B(x) = 2x^9+4x^6+5x^5+x^4+x^2-x-1$$

Bruno deseja enviar a mensagem $M(x) = x^3+x$ para Ana e usar o NTRU-*encript* para garantir o sigilo. Para isso, ele gera um vetor aleatório $r(x)=x^2-x \in T(1,1)$ e criptografa a mensagem usando a chave pública dela e o vetor r(x):

$$C(x) = p(h(x) * r(x)) + M(x) \pmod{17} = 2((8x^4+5x^3+6x^2+14x+1) * (x^2-x))+x^3+x$$

$$C(x) = 2x^4+8x^2+15x+11 \pmod{17}$$

Ana então utiliza suas chaves privadas para recuperar a mensagem original:

$$a(x) = f_A(x) * C(x) = (x^3-x+1) * (2x^4+8x^2+15x+11) = 3x^3+12x^2+4x \pmod{17}$$

$$M(x) = f_A^{-1}{}_p(x) * a(x)_t = (x^3+x^2+x) * (3x^3-5x^2+4x) = x^3+x \pmod 2$$

Para garantir a autenticidade, ele vai assinar uma mensagem usando o NTRU-*sign*. A mensagem que será assinada por Bruno é $M = (M_1, M_2)$:

$$M_1(x) = 4x^{10}+7x^9+10x^8-6x^7+11x^6+11x^5+6x^4-6x^3+9x^2-9x+3$$

$$M_2(x) = -7x^{10}+11x^9+11x^8+x^7-3x^6-7x^5-2x^4-3x^2+9x+5$$

Para assinar a mensagem, primeiro ele calcula os vetores $A_1$ e $A_2$ a partir da equação EQ. 6.26:

$$A_1(x) = 3x^{10}+2x^9+2x^8+x^7-3x^6+4x^5+7x^4+3x^3+x^2+4x+3$$

$$A_2(x) = -2x^{10}-x^9-x^7-x^6+x^5+2x^2+x+1$$

Por fim, ele usa suas chaves privadas $f_B(x)$ e $F_B(x)$ para calcular a assinatura s(x) e envia à Ana:

$$s(x) = x^{10}+6x^9+10x^8-9x^7+12x^6+9x^5+5x^4-6x^3+6x^2-8x+1$$

Para conferir a validade da assinatura, Ana utiliza a chave pública de Bruno para recuperar sua possível solução do CVP e calcular a distância até a mensagem:

$$t(x) = -4x^{10}+13x^9+13x^8+3x^7-x^6-8x^5-3x^4+2x^3-2x^2+8x+6$$

$$D = \| (s \quad t) - (M_1 \quad M_1) \| \approx 8.544$$

Finalmente, ela calcula a distância sugerida pela Heurística de Gauss e compara com a distância obtida pela solução de Bruno:

$$\lambda_{GAUSS}(\mathrm{L}) = \sqrt{\frac{n}{2\pi e}}\mathrm{V}(\mathrm{L})^{1/\mathrm{n}} = \sqrt{\frac{22*23}{2\pi e}} = 5.44$$

$$8.544 < \lambda_{GAUSS}(\mathrm{L}) * \sqrt{n} = 18.052$$

A solução enviada por Bruno é aceita, pois a distância é menor que a distância limite calculada a partir da Heurística de Gauss.

# 6 CONCLUSÃO

Este trabalho tem sua importância no estudo da criptografia. O estudo e o conhecimento dessa área são fundamentais para a garantia da troca de mensagens em meios inseguros mantendo sigilo e autenticidade da comunicação.

Numa primeira etapa do trabalho, buscou-se a compreensão dos conceitos básicos da criptografia, alguns dos principais sistemas criptográficos atualmente utilizados e a ameaça que tem surgido a esses sistemas: o computador quântico.

Numa etapa posterior, buscou-se a compreensão de dois criptossistemas baseados em reticulado. Dentre eles, o NTRU destacou-se como uma excelente opção que é segura contra algoritmos clássicos de criptoanálise e imune a todos os algoritmos quânticos até hoje desenvolvidos. Além disso, suas equações apresentam um tempo de execução muito menor do que as de sistemas criptográficos mais antigos e ainda bastante utilizados, como RSA, ElGamal e ECC.

Como contribuição do trabalho realizado, destaca-se a criação de uma fonte de consulta para o aprendizado de criptografia com reticulado, que pode ser utilizada como ponto de partida para um maior aprofundamento no tema.

Como sugestões para trabalhos futuros, podem-se destacar o aprofundamento no estudo do NTRU, buscando métodos para otimizá-lo, tornando-o mais rápido e mais seguro, visando sua implementação e utilização para troca de mensagens com criptografia no IME.

Outras sugestões para trabalhos futuros seriam o aprofundamento no estudo do LLL, buscando métodos para otimizá-lo e o aprofundamento no estudo da Computação Quântica, buscando conhecer melhor seus algoritmos e procurando desenvolver outros.

# 7 REFERÊNCIAS BIBLIOGRÁFICAS


[1] DENNING, D. E. **Cryptography and Data Security**. Purdue, 1982.

[2] DIFFIE, W.; HELLMAN, M. **New Directions in Criptography**. IEEE Transactions on Information Theory: 644-654, 1976.

[3] HOFFSTEIN, J.; PIPHER, J.; SILVERMAN, J. H. **An Introduction to Mathematical Cryptography**. Springer, 2008.

[4] DEUTSCH, D. **Quantum Teory, the Church-Turing Principle, and the Universal Quantum Computer**. Proc. Roy. Soc. Lond.: 97-117, 1985.

[5] ALVES, F. L. **Computação Quântica: Fundamentos Físicos e Perspectivas**. Monografia – Apresentada no curso de Ciências da Computação, da Universidade Federal de Lavras, para obtenção do grau de Bacharel.

[6] ALEGRETTI, F. J. P. **Computação Quântica**. Trabalho – Apresentado no curso de Pós-Graduação em Computação, da Universidade Federal do Rio Grande do Sul.

[7] PORTUGAL, R. **Uma Introdução à Computação Quântica** São Carlos, 2004.

[8] OLIVEIRA, I. S.; SARTHOUR, R. S. **Computação Quântica e Informação Quântica**. Centro Brasileiro de Pesquisas Físicas, 2004

[9] BACH, E.; SORENSON.; J. **Sieve Algorithms for Perfect Power Testing**. Buttler, 1993.

[10] PREPARATA, F. P.; YEH, R. T. **Introduction to Discrete Structures for Computer Science and Engineering** (Addison-Wesley series in computer science and information processing), 1973.

[11] FIGUEIREDO, L. M. **Números Primos e Criptografia de Chave Pública**. UFF, 2006.

[12] SANTOS, R., J. **Álgebra Linear e Aplicações.** UFMG, 2010.

[13] HINEK, M. J. **Lattice Atacks in Cryptography: A Partial Overview.** University of Waterloo, Canada, 2004.

[14] SILVERMAN, J. H. **An Introduction to the Theory of Lattices and Applications to Cryptography**. Wyoming, 2006.

[15] ELGAMAL, T. **A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.** IEEE Transactions on Information Theory:

469-472, 1985.

[16] RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.** Communications of ACM: 120-126, 1978.

[17] MENDES, A. V. **Estudo de Criptografia com Chave Pública Baseada em Curvas Elípticas**. Monografia – Apresentada no curso de Ciência da Computação, da Universidade Estadual de Montes Claros, para obtenção do grau de Bacharel.

[18] AJTAI, M.; DWORK, C.; **A Public-Key Cryptosystem with Worst-Case/ Average-Case Equivalence**. Proc. 30th ACM STOC: 284-293, 1997.

[19] HOFFSTEIN, J.; PIPHER, J.; SILVERMAN, J. H. **NSS: An NTRU Lattice-Based Signature Scheme**. Burlington, 2001.

[20] HOFFSTEIN, J.; PIPHER, J.; SILVERMAN, J. H.; GRAHAM, N. H.; WHYTE, W. **NTRU-Sign: Digital Signatures Using the NTRU Lattice**. Burlington, 2002.

[21] NGUYEN, P. Q.; **A Note on the Security of NTRU-Sign**. École normale supérieure & CNRS, 2006.

[22] HASEGAWA, S.; ISOBE, S.; MAMBO, M.; SHIZUYA, H.; FUTA, Y.; OHMORI, M.; **A Countermeasure for Protecting NTRU-Sign Against the Transcript Attack**. Department of Computer and Mathematical Sciences, Tohoku, 2007.