Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Gustavo Henrique Muriel Zanon

Criptografia Homomórfica

São Paulo
Novembro de 2016

Criptografia Homomórfica

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Marcos Antonio Simplicio Junior
Cosupervisor: Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto

São Paulo
Novembro de 2016

Resumo

A computação na nuvem processa grandes volumes de dados, e muitos desses dados precisam ser manipulados sem perda de confidencialidade. Uma proposta de solução para esse problema consiste no uso da criptografia homomórfica.

Apesar dos esquemas totalmente homomórficos apresentados até o presente mostrarem-se inviáveis para fins práticos, podemos utilizar criptossistemas parcialmente e ligeiramente homomórficos para atender a necessidades de aplicações específicas.

Este trabalho aborda teoria e implementação dos criptossistemas parcialmente homomórfico de Paillier e ligeiramente homomórfico NTRU. O primeiro mostra-se útil para aplicações simples como cálculo de médias, enquanto o segundo, para um número pequeno de somas e multiplicações.

Palavras-chave: segurança de dados, criptografia homomórfica, Paillier, NTRU.

Abstract

Cloud computing processes large volumes of data, and many of these data need to be handled without loss of confidentiality. A proposed solution to this problem is the use of homomorphic cryptography.

Although the fully homomorphic schemes presented so far prove to be infeasible for practical purposes, we can use partially and somewhat homomorphic cryptosystems to meet specific applications needs.

This work addresses the theory and implementation of partially homomorphic Paillier cryptosystems and somewhat homomorphic NTRU. The former is useful for simple applications such as averaging, while the latter for a small number of sums and multiplications.

Keywords: data security, homomorphic cryptography, Paillier, NTRU.

Sumário

Capítulo 1

Introdução

1.1 Motivações

Sistemas contemporâneos de grande escala, envolvendo volumes de dados muito elevados, tornaram-se acessíveis a empresas de porte inferior ao que tipicamente permitiria estabelecer e manter os recursos computacionais necessários ao processamento daqueles dados. Por exemplo, clínicas de todos os tamanhos têm a mesma necessidade de acesso a bases de dados de pacientes, diagnósticos, tratamentos, redes de laboratórios, provedores de planos de saúde e outras clínicas, hospitais e centros de pesquisa médica. Da mesma forma, companhias aéreas e escolas também possuem a mesma necessidade de acesso a bases de passageiros e alunos, respectivamente. As necessidades de armazenamento e processamento desses sistemas podem ser atendidas pela terceirização de recursos, transferindo para o ambiente de rede a totalidade dos dados pertinentes a esses sistemas. Esse fenômeno, denominado pelo termo genérico *computação em nuvem*, transcende a escala original da Internet, que em princípio compartilharia somente recursos próprios do ambiente de rede, para um vasto ambiente integrado de natureza essencialmente virtual: o negócio como um todo é transferido para o âmbito da rede (tipicamente representada com um ícone de nuvem, daí o nome).

Figura 1.1: Em um primeiro momento, servidores particulares de pequeno e médio porte comunicam-se através da internet.

Figura 1.2: Em um segundo momento, os servidores particulares passaram a se comunicar com sensores, notebooks e smartphones.

Figura 1.3: Atualmente, servidores e dados confidenciais estão distribuídos pela nuvem.

Em contrapartida, a computação em nuvem acarreta uma fragilidade potencial a esses mesmos negócios, que podem ser suprimidos irreversivelmente seja por acidente ou seja por malícia, com uma facilidade muito maior do que se teria em um ambiente proprietário fechado.

Além da evitar perda ou supressão dos dados, os maiores desafios da computação em nuvem envolvem o processamento em si: deseja-se, por um lado, manter a confidencialidade dos dados a despeito do ambiente potencialmente hostil em que possam se encontrar (equipamento alheio), e por outro, aproveitar a capacidade de processamento desse mesmo ambiente.

Assim, clínicas poderiam realizar buscas de diagnósticos relevantes sem revelar as pessoas diagnosticadas, ou ainda, cruzar dados de pacientes para prever um possível surto de uma doença. Companhias aéreas poderiam buscar por passageiros suspeitos de serem terroristas sem revelar quais os possíveis suspeitos em um voo. Escolas poderiam fazer uma análise estatística comparando as notas de seus estudantes com as notas de alunos de demais escolas, mantendo nota de cada aluno confidencial.

Para atender ao requisito de confidencialidade, define-se a noção de *criptografia homomórfica*, pela qual todas as operações algébricas possíveis entre dados legíveis podem ser aplicadas igualmente a dados cifrados, produzindo as formas cifradas dos resultados dessas operações. Idealmente, as próprias operações algébricas deveriam ser confidenciais, constituindo o conceito de *ofuscação*, que nesse sentido combina a criptografia homomórfica com o modelo de von Neumann, e torna confidencial até mesmo a natureza do processamento aplicado aos dados cifrados.

Sistemas parcialmente homomórficos, em que apenas um determinado tipo de operação algébrica é permitida sobre criptogramas, são conhecidos há algum tempo. Um dos esquemas pioneiros é o criptossistema de Paillier [Paillier(1999)], que oferece apenas a adição homomórfica de criptogramas. O sistema Boneh-Goh-Nissim [Boneh *et al.*(2005)Boneh, Goh, e Nissim] amplia o escopo e permite, além de um número arbitrário de termos numa soma, também uma multiplicação homomórfica por termo (mas apenas uma). O trabalho de Gentry [Gentry(2009)], envolvendo reticulados, mostrou pela primeira vez a viabilidade

de cifração totalmente homomórfica, onde um número qualquer de termos e um número qualquer de fatores por termo são permitidos.

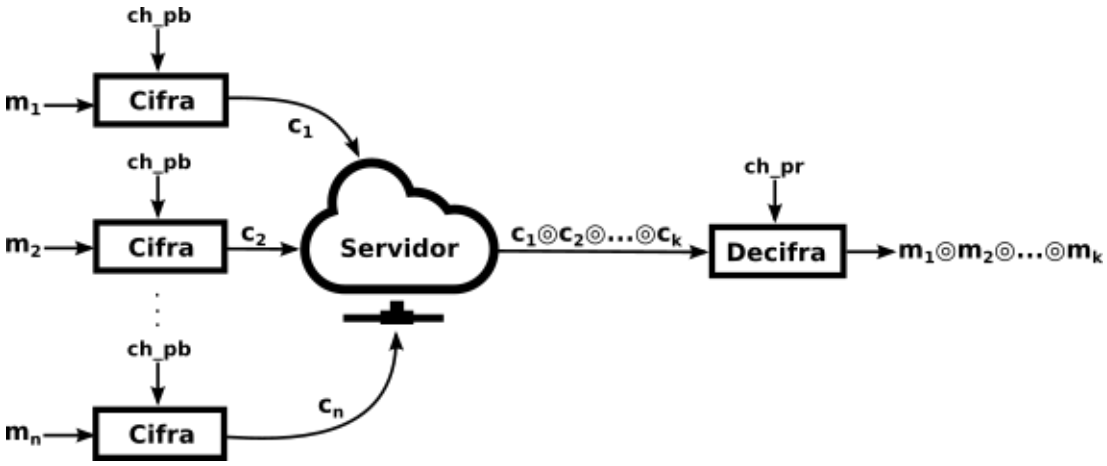

Figura 1.4: Sistema totalmente homomórfico onde mensagens cifradas por uma mesma chave pública são enviadas para um servidor capaz realizar operações sob esses dados. Um usuário sob posse da chave privada poderá acessar os dados operados após a decifração.

Contudo, o sistema de Gentry e muitos refinamentos derivados são excessivamente pesados para uso prático, constituindo até hoje mais demonstrações de princípio que esquemas práticos. Isso, porém, não impede que suas ideias centrais encontrem utilidade prática: a noção de criptografia ligeiramente homomórfica (em inglês, *somewhat homomorphic encryption*) ou homomórfica por camadas (em inglês, *leveled homomorphic encryption*) contém a funcionalidade realmente necessária na prática, a saber, a capacidade de obter um resultado cifrado consistente não pela aplicação de *qualquer* expressão algébrica a um conjunto de dados cifrados, mas pela aplicação das expressões *relevantes* ao negócio. Estas são tipicamente restritas pelo escopo do negócio, e parâmetros adequados podem ser escolhidos para permitir o cálculo homomórfico dessas expressões (mas não necessariamente de outras).

O estado da arte em criptografia homomórfica permite obter sistemas ligeiramente homomórficos, em que a complexidade das operações sobre dados cifrados (efetuadas na nuvem) é especificada de antemão, com desempenho razoável, embora grandes desafios persistam nesse sentido. Uma exposição dos esquemas ligeiramente homomórficos básicos seria, portanto, de grande valia potencial para sistemas do mundo real conforme os exemplos acima.

1.2 Objetivos

Este trabalho visa a descrever não somente o criptossistema de Paillier (Capítulo 3), um dos mais simples esquemas homomórficos, mas também o criptossistema NTRU (Capítulo 4), cujas propriedades homomórficas ganharam destaque recente.

O primeiro, embora contemple apenas adições homomórficas, baseia-se no mesmo arcabouço de uma das formas mais tradicionais de criptografia de chave pública, o sistema RSA [Rivest *et al.*(1978)Rivest, Shamir, e Adleman]. Isso o torna implementável de maneira substancialmente mais simples, exigindo somente recursos comumente disponíveis em plataformas computacionais modernas (por exemplo, ambientes Java ou Python). A segurança do sistema de Paillier deriva da dificuldade de resolver o problema da fatoração inteira.

O segundo esquema, menos tradicional, é contudo um dos mais antigos baseados em reticulados. Sua segurança origina-se na dificuldade de um problema computacional inteiramente distinto do RSA: encontrar vetores curtos em certos reticulados modulares.

1.3 Organização do texto

O restante desta monografia está organizado da seguinte forma. O capítulo 2 apresenta e classifica os diferentes esquemas homomórficos. O capítulo 3 descreve o criptossistema de Paillier. O capítulo 4 introduz o criptossistema básico NTRU e indica como ele é, por natureza, homomórfico. O capítulo 5 relata sobre as implementações obtidas desses esquemas e avalia sua viabilidade prática. O capítulo 6 traz as conclusões acerca do trabalho realizado.

Capítulo 2

Conceitos preliminares

Esquemas criptográficos homomórficos nos permitem realizar operações sob mensagens cifradas. Veremos duas definições necessárias para podermos classificar os criptossistemas homomórficos.

Definição 2.1 (Homomorfismo aditivo)**.** *Sejam* $\langle \mathcal{A}, \circ \rangle$ *e* $\langle \mathcal{B}, \bullet \rangle$ *grupos sob a operação de adição.*

Seja uma função bijetora de encriptação $e : \mathcal{A} \to \mathcal{B}$ *e sua função (inversa) de decriptação* $d : \mathcal{B} \to \mathcal{A}$*.*

Dizemos que um esquema possui homomorfismo aditivo se satisfaz:

$$d(e(a_1) \bullet e(a_2)) = d(e(a_1)) \circ d(e(a_2)))$$

onde $a_1, a_2 \in \mathcal{A}$

Definição 2.2 (Homomorfismo multiplicativo)**.** *Sejam* $\langle \mathcal{A}, \Diamond \rangle$ *e* $\langle \mathcal{B}, \blacklozenge \rangle$ *grupos sob a operação de multiplicação.*

Seja uma função bijetora de encriptação $e : \mathcal{A} \to \mathcal{B}$ *e sua função (inversa) de decriptação* $d : \mathcal{B} \to \mathcal{A}$*.*

Dizemos que um esquema possui homomorfismo multiplicativo se satisfaz:

$$d(e(a_1) \blacklozenge e(a_2)) = d(e(a_1)) \Diamond d(e(a_2)))$$

onde $a_1, a_2 \in \mathcal{A}$

Observe que as operações de adição e multiplicação de grupos não são necessariamente as de soma e multiplicação usuais. Observe também que a operação definida em um grupo

$\mathcal{A}$ pode ser diferente da mesma operação definida no grupo $\mathcal{B}$.

Criptossistemas homomórficos podem ser classificados como:

- **Parcialmente Homomórficos** *(PHE – Partially Homomorphic Encryption)*: Satisfazem apenas uma das definições vistas enteriormente, isso é, ou são aditivamente ou multiplicativamente homomórficos.
- **Ligeiramente Homomórficos** *(SHE – Somewhat Homomorphic Encryption)*: Satisfazem ambas as definições 2.1 e 2.2, porém o número de operações é limitado.
- **Totalmente Homomórficos** *(FHE – Fully Homomorphic Encryption)*: Satisfazem ambas as definições 2.1 e 2.2, com número ilimitado de operações.

Note que, por possuir ambas operações definidas, os criptossistemas ligeiramente e totalmente homomórficos são construídos sobre estruturas de anéis.

O trabalho de Gentry [Gentry(2009)] mostra que um esquema ligeiramente homomórfico pode se tornar totalmente homomórfico através de *bootstrapping*. De modo geral, essa técnica consiste em atualizar uma cifra de tempos em tempos com a finalidade de reduzir o fator limitante do número de operações homomórficas.

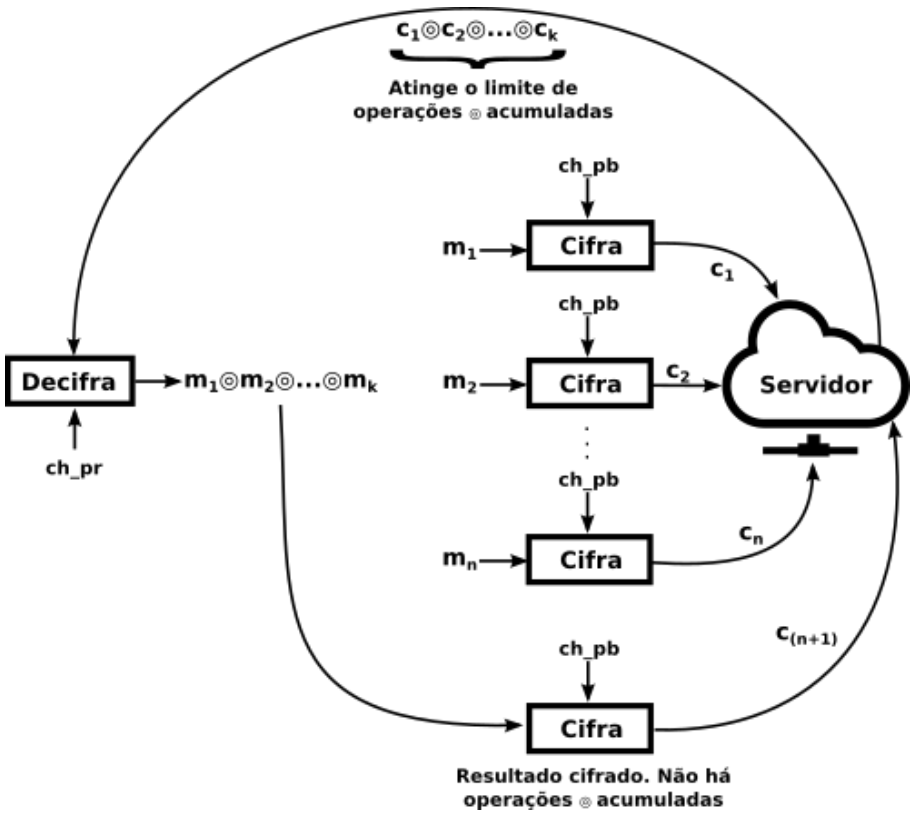

Figura 2.1: Exemplo de *bootstrapping*: A mensagem cifrada atinge o limite de operações permitidas e é enviada para um usuário sob posse da chave privada. Esse usuário decifra e cifra a mensagem novamente, obtendo conteúdo equivalente e sem operações acumuladas.

Capítulo 3

Criptossistema de Paillier

É um esquema de criptografia de chave pública e parcialmente homomórfico. Sua segurança é baseada na intratabilidade do problema dos resíduos quadráticos, mostrado não mais difícil que resolver o RSA [Paillier(1999)]. O RSA, por sua vez, não é mais difícil que determinar a fatoração de um número inteiro [Rivest *et al.*(1978)Rivest, Shamir, e Adleman]. Sabemos que a fatoração de um inteiro pode ser feita em tempo polinomial em um computador quântico através do algoritmo de Shor [Shor(1997)], motivo pelo qual buscam-se alternativas resistentes a ataques com computadores quânticos. Tais alternativas são conhecidas como pós-quânticas.

3.1 Fundamentação teórica

Nesta seção, fundamentaremos a teoria necessária para compreender o funcionamento do criptossistema de Paillier. As definições, lemas, teoremas e demonstrações utilizadas baseiam-se tanto no artigo original [Paillier(1999)], quanto em estudos mais detalhados desse artigo [O'Keeffe(2008), Volkhausen(2006)].

Apresentaremos inicialmente a função matemática utilizada para cifrar uma mensagem.

$$e_g : \mathbb{Z}_n \times \mathbb{Z}_n^* \to \mathbb{Z}_{n^2}^*$$
$$e_g(x, y) \to g^x \cdot y^n \pmod{n^2}$$

Ao longo deste capítulo, consideraremos $n = p \cdot q$. Os números p e q são primos grandes e possuem número de bits proporcional à segurança desejada.

Definição 3.1. *A ordem de um elemento a é o menor valor positivo m tal que $a^m = 1$.*

Seja um elemento não totalmente aleatório [1] $g \in \mathbb{Z}_{n^2}^*$, onde $\mathbb{Z}_{n^2}^*$ é o grupo das unidades de $\mathbb{Z}_{n^2}$. O elemento g deve ser escolhido de tal forma que sua ordem seja múltipla de n em $\mathbb{Z}_{n^2}^*$. Veremos que tal escolha de g faz-se necessária para que a função de encriptação seja bijetora.

Definição 3.2. *Seja a fatoração de um inteiro x em primos dada por $x = \prod_{i=1}^{k} p_i^{l_i}$, a função de Carmichael $\lambda(x)$ é dada por $\lambda(x) = \text{MMC}(\lambda(p_1^{l_1}), ..., \lambda(p_k^{l_k}))$ onde:*

$$\lambda(p_i^{l_i}) = \begin{cases} 2^{l_i-2} & \text{se } p_i = 2 \text{ e } l_i > 2 \\ p_i^{l_i-1}(p_i - 1) & \text{caso contrário} \end{cases}$$

Note que para $n = p \cdot q$, p e q primos, temos $\lambda(n) = \text{MMC}((p-1),(q-1))$

Lema 1. $\lambda(n^2) = n\lambda(n)$

Demonstração.

$$\lambda(n^2) = \lambda(p^2q^2) = \text{MMC}(p(p-1), q(q-1)) = pq \cdot \text{MMC}(p-1, q-1) = n\lambda(n)$$

□

Teorema 1 (Carmichael)**.** *Sejam a e x relativamente primos, isso é,* $\text{MDC}(a,x) = 1$*, então:*

$$a^{\lambda(x)} \equiv \ (mod\ n)$$

onde $\lambda(x)$ é a função de Carmichael.

Lema 2. $a^{\lambda(n)} \equiv 1\ (mod\ n)$

Demonstração. Como $a \in \mathbb{Z}_{n^2}^*$, então $\text{MDC}(a, n^2) = 1$. Logo, $\text{MDC}(a,n) = 1$ e o teorema de Carmichael se aplica. □

Lema 3. $a^{n\lambda(n)} \equiv 1\ (mod\ n^2),\ a \in \mathbb{Z}_{n^2}^*$

Demonstração. Pelo Lema 1, $n\lambda(n) = \lambda(n^2)$. Temos então $a^{\lambda(n^2)}\ (\text{mod}\ n^2)$ com $\text{MDC}(a, n^2) = 1$ e também podemos aplicar o teorema de Carmichael. □

[1] É dito não totalmente aleatório pois alguns valores de g não garantem o funcionamento do algoritmo, conforme abordaremos mais adiante

O Lema 2 nos diz que $g^{\lambda(n)} \equiv 1 \pmod{n},\ g^{\lambda(n)} \in \mathbb{Z}^*_{n^2}$. A partir disso, $g^{\lambda(n)} \pmod{n^2} \equiv 1 \pmod{n}$ implica que $k = (g^{\lambda(n)} \pmod{n^2} - 1)/n$ para algum $k \in \mathbb{Z}_n$. Se k não é congruente a nenhum múltiplo de p ou q, então possui inverso e é uma unidade em $\mathbb{Z}_n$, isso é, $k \in \mathbb{Z}^*_n$. Por conveniência, definimos uma função $L(u) = \frac{u-1}{n}$. Assumindo que k possui inverso, podemos definir

$$\mu \equiv k^{-1} \pmod{n} \equiv L(g^{\lambda(n)} \pmod{n^2})^{-1}$$

Lema 4. *Dados dois conjuntos finitos $\mathcal{A}$ e $\mathcal{B}$ com o mesmo número de elementos $|\mathcal{A}| = |\mathcal{B}| = s$, uma função $f : \mathcal{A} \to \mathcal{B}$ é injetora se e somente se é sobrejetora.*

Demonstração. Sejam $\mathcal{A} = \{a_1, a_2, ..., a_s\}$ e $\mathcal{B} = \{b_1, b_2, ..., b_s\}$.

Suponha que f é injetora. Como $\mathcal{B}$ possui s elementos, eles devem ser uma permutação de $f(a_1), f(a_2), ..., f(a_s)$, distintos em $\mathcal{B}$ (pois supomos f injetora). Portanto todo elemento de $\mathcal{B}$ é correspondido por um elemento em $\mathcal{A}$ através de f, e f é sobrejetora.

Suponha que f é sobrejetora; Todo $b_i \in \mathcal{B}$ possui correspondência através de $f(a_j)$ para algum $a_j \in \mathcal{A}$. Como todos elementos de $\mathcal{B}$ são correspondidos através de f, há pelo menos s elementos em $\mathcal{A}$. Como $|\mathcal{A}| = s$, há s elementos distintos em $\mathcal{A}$ que são mapeados a um elemento distinto de $\mathcal{B}$, e portanto f é injetora. □

Definição 3.3 (Função ϕ de Euler)**.** *Seja a fatoração de um inteiro x em primos dada por $x = \prod_{i=1}^{k} p_i^{l_i}$, a função totiente de Euler é definida por:*

$$\phi(x) = \prod_{i=1}^{k} p_i^{l_i - 1}(p_i - 1)$$

A função $\phi(x)$ também pode ser definida como a quantidade de inteiros $a, 1 < a < x$ tal que $\text{MDC}(a, x) = 1$. Como os elementos de $\mathbb{Z}^*_n$ são unidades, então $\text{MDC}(a, n) = 1\ \forall a \in \mathbb{Z}^*_n$ e portanto $|\mathbb{Z}^*_n| = \phi(n)$. Analogamente, e $|\mathbb{Z}^*_{n^2}| = \phi(n^2)$

Lema 5. $\phi(x^2) = x\phi(x)$

Demonstração.

$$\phi(x^2) = \prod_{i=1}^{k} p_i^{2l_i - 1}(p_i - 1) = \prod_{i=1}^{k} p_i^{l_i} p_i^{l_i - 1}(p_i - 1) = (\prod_{i=1}^{k} p_i^{l_i})(\prod_{i=1}^{k} p_i^{l_i - 1}(p_i - 1)) = x\phi(x)$$

□

Lema 6. $(x + yn)^n \equiv x\ (mod\ n^2)$

Demonstração. Pelo Binômio de Newton,
$(x + yn)^n = \sum_{k=0}^{n} \binom{n}{k} x^{m-k}(yn)^k = x + [\text{potências de } n^m, m \geq 2] \equiv x \pmod{n^2}$ □

Lema 7. $(1+n)^x \equiv 1 + x \cdot n \ (mod\ n^2)$

Demonstração. Novamente, pelo Binômio de Newton temos
$(1+n)^x = \sum_{k=0}^{n} \binom{n}{k} n^k = 1 + xn + [\text{potências de } n^m, m \geq 2] \equiv 1 + xn \pmod{n^2}$ □

Teorema 2. *Se a ordem de g é não-nula e múltipla de n, então e_g é bijetora para todo $e_g(x, y) = g^x \cdot y^n \ (mod\ n^2)$*

Demonstração. A função e_g mapeia elementos de $\mathbb{Z}_n \times \mathbb{Z}_n^*$ para $\mathbb{Z}_{n^2}^*$. Sabemos que a ordem de $\mathbb{Z}_{n^2}^* = \phi(n^2) = n\phi(n) = |\mathbb{Z}_n \times \mathbb{Z}_n^*|$. Ambos conjuntos possuem o mesmo número de elementos, e pelo lema 4, se a função e_g for injetora, então também será sobrejetora e portanto e_g será bijetora.

Suponha que $g^{x_1} \cdot y_1^n \equiv g^{x_2} \cdot y_2^n \pmod{n^2}$. Então

$$
\begin{aligned}
(g^{x_1} \cdot y_1^n)/(g^{x_2} \cdot y_2^n) &\equiv 1 \pmod{n^2} && \\
g^{x_1 - x_2} \cdot (y_1/y_2)^n &\equiv 1 \pmod{n^2} && \\
g^{\lambda(n)(x_1-x_2)} \cdot (y_1/y_2)^{n\lambda(n)} &\equiv 1 \pmod{n^2} && \text{elevando ambos os lados por } \lambda(n) \\
g^{\lambda(n)(x_1-x_2)} &\equiv 1 \pmod{n^2} && \text{pelo Lema 3}
\end{aligned}
$$

Por definição, $\lambda(n)(x_1 - x_2)$ é um múltiplo da ordem de g. Como assumimos que a ordem de g é um múltiplo não-nulo de n e sabemos que $\text{MDC}(\lambda(n), n) = 1$, então $n|(x_1 - x_2)$ ou equivalentemente, $x_1 - x_2 \equiv 0 \pmod{n}$. Portanto $x_1 = x_2$ em $\mathbb{Z}$ já que ambos são elementos mapeados de $\mathbb{Z}_n$, onde a equivalência é válida. Assim:

$$
\begin{aligned}
g^{x_1 - x_2} \cdot (y_1/y_2)^n &\equiv 1 \pmod{n^2} \\
(y_1/y_2)^n &\equiv 1 \pmod{n^2}
\end{aligned}
$$

Considere $y_1 \equiv y_2 + \alpha n \pmod{n^2}$ para algum $\alpha \in \mathbb{Z}_{n^2}^*$. Isso implica que $y_1 \equiv y_2 \pmod{n}$.

$$
\begin{aligned}
(y_1)^n &\equiv (y_2 + \alpha n)^n \pmod{n^2} && \text{elevando ambos os lados por } n \\
(y_1)^n &\equiv (y_2)^n \pmod{n^2} && \text{pelo Lema 6}
\end{aligned}
$$

Portanto, $x_1 = x_2$ e como $y_1, y_2 \in \mathbb{Z}_n^*$, então $y_1 = y_2$. □

Esse teorema garante que um mapeamento feito pela função de encriptação é único.

Dada a fundamentação teórica necessária para entender o funcionamento do algoritmo, podemos então descrevê-lo.

3.2 O algoritmo

3.2.1 Geração de chaves

Algorithm 1 Gera o par de chaves privada (n, g) e pública (λ, μ)

function L(u, n)
  return $\frac{(u-1)}{n}$
end function

function PAILLIER_KEYGEN
  $p \xleftarrow{\$}$ primo de k bits
  $q \xleftarrow{\$}$ primo de k bits
  $n \leftarrow p \cdot q$
  $\lambda \leftarrow \text{MMC}((p-1), (q-1))$
  repeat
    $g \xleftarrow{\$} g' \in \mathbb{Z}^*_{n^2}$
    $\mu \leftarrow \text{L}(g^\lambda \pmod{n^2}, n)$
  until $(\exists \mu^{-1} \in \mathbb{Z}_n)$
  $\mu \leftarrow \mu^{-1} \in \mathbb{Z}_n$
  return n, g, λ, μ
end function

3.2.2 Encriptação

Algorithm 2 Cifra uma mensagem m a partir da chave pública (n, g) e de um ruído r

function PAILLIER_ENCRYPT($m \in \mathbb{Z}_n, g, n$)
  $r \xleftarrow{\$} r' \in \mathbb{Z}^*_n$
  $c \leftarrow g^m \cdot r^n \pmod{n^2}$
  return c
end function

3.2.3 Decriptação

Algorithm 3 Decifra uma mensagem a partir da cifra c e da chave privada (λ, μ)

function PAILLIER_DECRYPT($c \in \mathbb{Z}^*_{n^2}, \lambda, \mu$)
  $m \leftarrow \text{L}(c^\lambda \pmod{n^2}, n) \cdot \mu \pmod{n}$
  return m
end function

3.2.4 Corretude

Começaremos definindo uma notação que nos acompanhará ao longo desta seção:

Para $e_g(x, y) \equiv g^x \cdot y^n \equiv w \pmod{n^2}$, denotaremos $[[w]]_g = x$.

Vimos que a função e_g é bijetora, e portanto mapeia pares (x, y) para todos elementos de $\mathbb{Z}^*_{n^2}$. Como $g \in \mathbb{Z}^*_{n^2}$, então $\exists t$ tal que $e_t(x, y) \equiv t^x y^n \equiv g \pmod{n^2}$, e pela definição da função, t deve possuir ordem não-nula múltipla de n. Considere o elemento $(1+n) \in \mathbb{Z}^*_{n^2}$. Pelo lema 7, $(1+n)^x \equiv 1 + xn \pmod{n^2}$. Para $x = n$, temos que $(1+n)^x \equiv 1 \pmod{n^2}$ e portanto $(1+n)$ possui ordem múltipla de n em $\mathbb{Z}^*_{n^2}$. Com isso, podemos substituir $t = (1+n)$ e $x = [[g]]_{(1+n)}$ tal que

$$g \equiv e_{(1+n)}([[g]]_{1+n}, y) \equiv (1+n)^{[[g]]_{1+n}} y^n \pmod{n^2} \tag{3.1}$$

Ao criptografarmos uma mensagem m, calculamos

$$\begin{aligned} c &\equiv e_g(m, r) \equiv g^m r^n \pmod{n^2} \\ &\equiv g^m r^n \equiv [(1+n)^{[[g]]_{(1+n)}} y^n]^m r^n \pmod{n^2} && \text{pela relação 3.1} \\ &\equiv (1+n)^{m[[g]]_{(1+n)}} y^n \pmod{n^2} \end{aligned}$$

Por definição $[[c]]_{(1+n)} \equiv m[[g]]_{(1+n)}$. Note que $[[c]]_{(1+n)} \in \mathbb{Z}_n$, pois $c = e_{(1+n)}([[c]]_{(1+n)}, u)$. Daí, segue que:

$$[[c]]_{(1+n)} \equiv m[[g]]_{(1+n)} \Rightarrow m \equiv \frac{[[c]]_{(1+n)}}{[[g]]_{(1+n)}} y^n \pmod{n} \tag{3.2}$$

$$\begin{aligned} g^{\lambda(n)} &\equiv [(1+n)^{[[g]]_{(1+n)}} y^n]^{\lambda(n)} \pmod{n^2} \\ &\equiv (1+n)^{\lambda(n)[[g]]_{(1+n)}} y^{n\lambda(n)} \pmod{n^2} \\ &\equiv (1+n)^{\lambda(n)[[g]]_{(1+n)}} \pmod{n^2} && \text{pelo Lema 3} \\ &\equiv 1 + n\lambda(n)[[g]]_{(1+n)} \pmod{n^2} && \text{pelo Lema 7} \end{aligned}$$

Aplicando a função $L(u) = \frac{u-1}{n}$ em $g^{\lambda(n)} \pmod{n^2}$, temos

$$\begin{aligned} L(g^{\lambda(n)} \pmod{n^2}) &\equiv L(1 + n\lambda(n)[[g]]_{(1+n)}) \pmod{n} \\ &\equiv \frac{1 + n\lambda(n)[[g]]_{(1+n)} - 1}{n} \pmod{n} \\ &\equiv \frac{n\lambda(n)[[g]]_{(1+n)}}{n} \pmod{n} \\ &\equiv \lambda(n)[[g]]_{(1+n)} \pmod{n} \end{aligned}$$

Portanto, $\mu \equiv L(g^{\lambda(n)} \pmod{n^2})^{-1} \equiv (\lambda(n)[[g]]_{(1+n)})^{-1} \pmod{n}$

Similarmente, para $c^{\lambda(n)} \pmod{n^2}$

$$\begin{aligned} c^{\lambda(n)} &\equiv [(1+n)^{[[c]]_{(1+n)}} d^n]^{\lambda(n)} \pmod{n^2} \\ &\equiv (1+n)^{\lambda(n)[[c]]_{(1+n)}} d^{n\lambda(n)} \pmod{n^2} \\ &\equiv (1+n)^{\lambda(n)[[c]]_{(1+n)}} \pmod{n^2} && \text{pelo Lema 3} \\ &\equiv 1 + n\lambda(n)[[c]]_{(1+n)} \pmod{n^2} && \text{pelo Lema 7} \end{aligned}$$

Aplicando a função $L(u)$ em $c^{\lambda(n)} \pmod{n^2}$, temos

$$\begin{aligned} L(c^{\lambda(n)} \pmod{n^2}) &\equiv L(1 + n\lambda(n)[[c]]_{(1+n)}) \pmod{n} \\ &\equiv \frac{1 + n\lambda(n)[[c]]_{(1+n)} - 1}{n} \pmod{n} \\ &\equiv \frac{n\lambda(n)[[c]]_{(1+n)}}{n} \pmod{n} \\ &\equiv \lambda(n)[[c]]_{(1+n)} \pmod{n} \end{aligned}$$

Podemos agora verificar qual o resultado obtido ao realizarmos $PAILLIER\_DECRYPT(c,\lambda,\mu)$

$$\begin{aligned} L(c^{\lambda(n)} \pmod{n^2})\mu &\equiv \frac{\lambda(n)[[c]]_{(1+n)}}{\lambda(n)[[g]]_{(1+n)}} \\ &\equiv \frac{[[c]]_{(1+n)}}{[[g]]_{(1+n)}} \pmod{n} \\ &\equiv m \pmod{n} && \text{pelo relação obtida em 3.2} \end{aligned}$$

Repare que para uma mesma chave pública (n, g), o valor μ é constante e deve ser calculado apenas uma vez. A decifração envolve uma exponenciação módulo n^2, o cálculo de $L(u)$ e uma multiplicação módulo n por um valor pré-calculado.

3.3 Homomorfismo aditivo

Sejam M mensagens $m_1, m_2, ..., m_M$. Suponha que cada uma dessas mensagens seja cifrada pelo algoritmo de Paillier. Teremos então M criptogramas $c_1 \equiv g^{m_1} r_1^n \pmod{n^2}$, $c_2 \equiv g^{m_2} r_2^n \pmod{n^2}$, ..., $c_M \equiv g^{m_M} r_2^n \pmod{n^2}$.

Considere agora o produto desses M criptogramas $\prod_{i=1}^{M} c_i$:

$$\prod_{i=1}^{M} c_i \equiv \prod_{i=1}^{M} c_i (g^{m_i} r_i^n) \equiv g^{\sum_{i=1}^{M} m_i} (\prod_{i=1}^{M} r_i)^n \pmod{n^2}$$

Podemos observar que o resultado obtido é equivalente a cifrar uma mensagem $\sum_{i=1}^{M} m_i \in \mathbb{Z}_n$ com um número aleatório $\prod_{i=1}^{M} r_i \in \mathbb{Z}_n^*$, isso é, $e_g(\sum_{i=1}^{M} m_i, \prod_{i=1}^{M} r_i)$. Vamos aplicar a

função de decriptografia e conferir se o resultado é de fato o esperado. Para isso, precisamos calcular $L((\prod_{i=1}^{M} c_i)^{\lambda(n)} \pmod{n^2})\mu \pmod{n}$.

Dos cálculos obtidos na seção anterior, temos que $\mu \equiv (\lambda(n)[[g]]_{(1+n)})^{-1} \pmod{n}$ e $g^{\lambda} \equiv (1+n)^{\lambda(n)[[g]]_{(1+n)}} \pmod{n^2}$. Ambas equivalências aplicam-se nesse caso uma vez que dependem apenas da chave pública (n, g).

$$\begin{aligned}
(\prod_{i=1}^{M} c_i)^{\lambda(n)} &\equiv g^{\lambda(n)\sum_{i=1}^{M} m_i} (\prod_{i=1}^{M} c_i)^{n\lambda(n)} \pmod{n^2} \\
&\equiv g^{\lambda(n)\sum_{i=1}^{M} m_i} \pmod{n^2} && \text{pelo Lema 3} \\
&\equiv (1+n)^{\lambda(n)[[g]]_{(1+n)}\sum_{i=1}^{M} m_i} \pmod{n^2} \\
&\equiv 1 + n\lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i \pmod{n^2} && \text{pelo Lema 7}
\end{aligned}$$

Aplicando a função $L(u)$ em $(\prod_{i=1}^{M} c_i)^{\lambda(n)} \pmod{n^2}$, temos

$$\begin{aligned}
L((\prod_{i=1}^{M} c_i)^{\lambda(n)} \pmod{n^2}) &\equiv L(1 + n\lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i) \pmod{n} \\
&\equiv \frac{1 + n\lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i - 1}{n} \pmod{n} \\
&\equiv \frac{n\lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i}{n} \pmod{n} \\
&\equiv \lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i \pmod{n}
\end{aligned}$$

Finalmente, obtemos:

$$\begin{aligned}
L((\prod_{i=1}^{M} c_i)^{\lambda(n)} \pmod{n^2})\mu &\equiv \frac{\lambda(n)[[g]]_{(1+n)} \sum_{i=1}^{M} m_i}{\lambda(n)[[g]]_{(1+n)}} \pmod{n} \\
&\equiv \sum_{i=1}^{M} m_i \pmod{n}
\end{aligned}$$

Então

$$\begin{aligned}
&PAILLIER_DECRYPT(PAILLIER_ENCRYPT(m_1,n,g) \cdot PAILLIER_ENCRYPT(m_2,n,g),\lambda,\mu) = \\
&= PAILLIER_DECRYPT(PAILLIER_ENCRYPT(m_1,n,g),\lambda,\mu) + \\
&+ PAILLIER_DECRYPT(PAILLIER_ENCRYPT(m_2,n,g),\lambda,\mu)
\end{aligned}$$

e segue que a propriedade de homomorfismo aditivo 2.1 é válida.

Note que se essas mensagens fossem números inteiros, então $\sum_{i=1}^{M} m_i$ seria limitado pelo

valor n. Se $\sum_{i=1}^{M} m_i \geq n$, então $\sum_{i=1}^{M} m_i$ sofrerá uma redução $\pmod n$, o que implica em perda de informação. Na prática, n é um valor grande já que o número de bits de n é igual à soma do número de bits de p e q.

3.4 Outras propriedadeas

Além da adição homomórfica, o criptossistema ainda nos permite somar uma mensagem a um texto cifrado.

Suponha duas mensagens m_1, m_2 e o criptograma da primeira mensagem $c = g_1^m r^n \pmod{n^2}$. O produto cg^{m_2} é dado por

$$cg^{m_2} \equiv g^{m_1} r^n g^{m_2} \equiv g^{(m_1+m_2)} r^n \pmod{n^2}$$

Repare que, similar ao caso anterior, a expressão obtida equivale a $e_g(m_1 + m_2, r)$.

Ao aplicarmos a função *PAILLIER_DECRYPT()*, obtemos

$$\begin{aligned}
L((cg^{m_2})^{\lambda(n)} \pmod{n^2})\mu \pmod n &\equiv L((g^{(m_1+m_2)}r^n)^{\lambda(n)} \pmod{n^2})\mu \pmod n \\
&\equiv L(g^{\lambda(n)(m_1+m_2)} r^{n\lambda(n)} \pmod{n^2})\mu \pmod n \\
&\equiv L(g^{\lambda(n)(m_1+m_2)} \pmod{n^2})\mu \pmod n \\
&\equiv L((1+n)^{\lambda(n)[[g]]_{(1+n)}(m_1+m_2)} \pmod{n^2})\mu \pmod n \\
&\equiv L(1 + n\lambda(n)[[g]]_{(1+n)}(m_1 + m_2) \pmod{n^2})\mu \pmod n \\
&\equiv \frac{\lambda(n)[[g]]_{(1+n)}(m_1 + m_2)}{\lambda(n)[[g]]_{(1+n)}} \pmod n \\
&\equiv (m_1 + m_2) \pmod n
\end{aligned}$$

Também, o Paillier nos permite multiplicar um texto cifrado por uma mensagem.

Considere uma mensagem m_1, seu criptograma $c \equiv g^{m_1} r^n \pmod{n^2}$ e outra mensagem m_2. Ao realizarmos $c^{m_2} \pmod{n^2}$, obtemos o criptograma correspondente à $m_1 m_2$.

$$\begin{aligned}
L((c^{m_2})^{\lambda(n)} \pmod{n^2})\mu \pmod n &\equiv L((g^{m_1 m_2} r^n)^{\lambda(n)} \pmod{n^2})\mu \pmod n \\
&\equiv L(g^{\lambda(n)(m_1 m_2)} r^{n\lambda(n)} \pmod{n^2})\mu \pmod n \\
&\equiv L(g^{\lambda(n)(m_1 m_2)} \pmod{n^2})\mu \pmod n \\
&\equiv L((1+n)^{\lambda(n)[[g]]_{(1+n)}(m_1 m_2)} \pmod{n^2})\mu \pmod n \\
&\equiv L(1 + n\lambda(n)[[g]]_{(1+n)}(m_1 m_2) \pmod{n^2})\mu \pmod n \\
&\equiv \frac{\lambda(n)[[g]]_{(1+n)}(m_1 m_2)}{\lambda(n)[[g]]_{(1+n)}} \pmod n \\
&\equiv (m_1 m_2) \pmod n
\end{aligned}$$

Repare que multiplicar um texto cifrado por uma mensagem equivale a realizar somas homomórficas consecutivas, já que a mensagem é número inteiro em $\mathbb{Z}_n$.

$$\prod_{i=1}^{m_2} e_g(m_1, r_1) = e_g(\sum_{i=1}^{m_2}, \prod_{i=1}^{m_2} r_1) \tag{3.3}$$

$$e_g(m_1, r_1)^{m_2} = e_g(m_2 m_1, r_1^{m_2}) \tag{3.4}$$

Exponenciações modulares podem ser feitas de forma eficiente com o algoritmo *double-and-add*. Portanto é preferível usar 3.4 à 3.3.

Capítulo 4

Criptossistema NTRU

O NTRU [Hoffstein *et al.*(1998)Hoffstein, Pipher, e Silverman] é um criptossistema ligeiramente homomórfico de chave pública. É conjecturado pós-quântico por ser construído sobre reticulados, estruturas algébricas que permitem explorar problemas computacionalmente difíceis como os problemas do vetor mais curto (*SVP – Shortest Vector Problem*) e do vetor mais próximo (*CVP – Closest Vector Problem*).

4.1 Fundamentação teórica: Anéis polinomiais quociente

Tanto nesta seção quanto na seção 4.5, fundamentaremos a teoria necessária para o entendimento do criptossistema NTRU. As definições utilizadas baseiam-se tanto no artigo original [Hoffstein *et al.*(1998)Hoffstein, Pipher, e Silverman] e trabalhos derivados [Hoffstein *et al.*(2015)Hoffstein, Pipher, Schanck, Silverman, Whyte, e Zhang], quanto em outros estudos sobre criptografia em reticulados [Bollauf(2014), Pipher(2002), Barreto(2015)].

Vamos abordar aqui conceitos iniciais sob a perspectiva de polinômios.

Definição 4.1. *O criptossistema NTRU utiliza-se de dois anéis polinomiais quociente:*

$$(\mathbb{Z}/q\mathbb{Z})[x]/\langle x^N - 1\rangle \text{ e } (\mathbb{Z}[x]/p\mathbb{Z})/\langle x^N - 1\rangle$$

com N, p primos e $q \gg p$

O anel $(\mathbb{Z}/p\mathbb{Z})[x]/\langle x^N - 1\rangle$ possui polinômios com coeficientes módulo p e grau máximo $N-1$, já que formam uma classe de resíduos módulo o ideal $\langle x^N - 1\rangle$. De forma análoga, o anel $(\mathbb{Z}/q\mathbb{Z})[x]/\langle x^N - 1\rangle$ possui polinômios com coeficientes módulo q e grau máximo $N-1$. Por simplicidade, serão denotados por $\mathcal{R}_{N,q}$ e $\mathcal{R}_{N,p}$, respectivamente.

Um elemento $f \in \mathcal{R}_{N,q}$ ou $f \in \mathcal{R}_{N,p}$ pode ser escrito não somente como um polinômio,

mas também como vetor [1]:

$$f = \sum_{i=0}^{N-1} a_i x^i = [a_0, a_1, ..., a_{N-1}]$$

Em ambos os anéis a multiplicação entre dois elementos é dada pela convolução cíclica, denotada por $\circledast$.

Definição 4.2. *Sejam* $f = \sum_{i=0}^{N-1} a_i x^i$ *e* $g = \sum_{j=0}^{N-1} b_j x^j$. *A convolução cíclica entre* f *e* g *é definida como:*

$$f \circledast g = \sum_{k=0}^{N-1} (a * b)_k x^k$$

onde $(a * b)_k = \sum_{l=0}^{N-1} a_l g_{(k-l) \pmod N}$

Note que calcular o valor $f \circledast g$ leva tempo $O(N^2)$. Porém, veremos que f ou g possuem coeficientes pequenos, e portanto seu cálculo será rápido. Por outro lado, se N é considerado grande, então o cálculo pode ser otimizado através da Transformada Rápida de Fourier em tempo $O(N \log N)$. Para isso, tomamos um número N' que representa a primeira potência de dois tal que $N' > N$. É comum escolher previamente um N próximo de N', como por exemplo $N = 509$ e $N' = 512 = 2^9$. Como N' é potência de dois, podemos calcular a convolução entre os dois polinômios em tempo $O(N \log N)$ através da NNT (*Number Theoretic Transform* [2]), que é uma generalização da Transformada Rápida de Fourier sob anéis quociente. O polinômio resultante terá grau máximo $2N$ no ideal $\langle x^{2N'} - 1 \rangle$. Por fim, reduzimos esse produto pelo ideal $\langle x^N - 1 \rangle$ em tempo $O(N)$.

Ainda, o algoritmo leva em conta polinômios ternários com coeficientes $-1, 0, 1$. Denotaremos tais polinômios por $\mathcal{T}_N(d_1, d_2)$, onde d_1 e d_2 correspondem à quantidade de coeficientes iguais a -1 e 1, respectivamente. O número de coeficientes iguais a 0 será $N - 1 - d_1 - d_2$.

Podemos seguir com a descrição do algoritmo.

[1] a equivalência entre polinômio e vetor será importante na seção 4.5

[2] Implementações eficientes da NTT fogem do escopo deste trabalho, contudo podem ser lidas em [Alkim *et al.*(2016)Alkim, Ducas, Pöppelmann, e Schwabe]

4.2 O algoritmo

4.2.1 Geração de chaves

Algorithm 4 Gera o par de chaves privada f e pública h

function NTRU_KEYGEN
  repeat
    $f' \xleftarrow{\$} \mathcal{T}_N(d_f, d_f)$
    $f \leftarrow p \cdot f' + 1$
  until $(\exists f^{-1} \in \mathcal{R}_{N,q})$
  repeat
    $g' \xleftarrow{\$} \mathcal{T}_N(d_g, d_g)$
    $g \leftarrow p \cdot g'$
  until $(\exists g^{-1} \in \mathcal{R}_{N,q})$
  $h = f^{-1} \circledast g \in \mathcal{R}_{N,q}$
  return h, f
end function

4.2.2 Encriptação

Algorithm 5 Cifra uma mensagem m a partir da chave pública h e de um ruído r

function NTRU_ENCRYPT(m, h)
  $r \xleftarrow{\$} \mathcal{T}_N(d_r, d_r)$
  $c \leftarrow h \circledast r + m \in \mathcal{R}_{N,q}$
  return c
end function

4.2.3 Decriptação

Algorithm 6 Decifra uma mensagem a partir da cifra c e da chave privada f

function NTRU_DECRYPT(c, f)
  $m \leftarrow f \circledast c \in \mathcal{R}_{N,q}$
  $m \leftarrow$ CENTER(m, q)
  $m \leftarrow m \in \mathcal{R}_{N,p}$
  return m
end function

4.2.4 Corretude

Seja m um polinômio ternário $\mathcal{T}_(d_m, d_m)$ obtido através mapeamento de uma mensagem original. Do algoritmo, temos a chave pública:

$$h = f_q \circledast g \in \mathcal{R}_{N,q} \tag{4.1}$$

onde f_q é garantidamente o inverso da chave privada f no anel $\mathcal{R}_{N,q}$, isto é:

$$f \circledast f_q = 1 \in \mathcal{R}_{N,q} \tag{4.2}$$

Após aplicarmos a função $NTRU\_ENCRYPT(m,h)$, obtemos a mensagem cifrada:

$$c = h \circledast r + m \in \mathcal{R}_{N,q} \tag{4.3}$$

A função $NTRU\_DECRYPT(c,f)$, reliza a convolução entre c e f no anel $\mathcal{R}_{N,q}$:

$$\begin{aligned}
f \circledast c &= f \circledast (h \circledast r + m) && \text{pela equação 4.3} \\
&= f \circledast h \circledast r + f \circledast m && \text{pela propriedade distributiva do anel} \\
&= f \circledast (f_q \circledast g) \circledast r + f \circledast m && \text{pela equação 4.1} \\
&= (f \circledast f_q) \circledast g \circledast r + f \circledast m && \text{pela propriedade associativa do anel} \\
&= g \circledast r + f \circledast m && \text{pela equação 4.2}
\end{aligned}$$

Em seguida, centralizamos o polinômio resultante de modo que seus coeficientes fiquem no intervalo $[-q/2, q/2]$. Esse passo é necessário pois o algoritmo assume que o polinômio representante da mensagem original m possui coeficientes centralizados.

Por fim, passamos o polinômio para o anel $\mathcal{R}_{N,p}$, o que equivale a reduzir cada um de seus coeficientes módulo p. Observe que a função de geração de chaves $NTRU\_KEYGEN()$ escolhe propositalmente $g = p \cdot g'$ e $f = p \cdot f' + 1$ em $\mathcal{R}_{N,q}$ de modo $g = 0$ e $f = 1$ em $\mathcal{R}_{N,p}$. Com isso, temos:

$$g \circledast r + f \circledast m = m \in \mathcal{R}_{N,p}$$

e segue que $NTRU\_DECRYPT(NTRU\_ENCRYPT(m,h),f)=m$, conforme esperado.

4.3 Homomorfismo aditivo

Sejam duas mensagens [3] m_1 e m_2. Para um mesmo par de chaves, suas mensagens cifradas serão, respectivamente, $c_1 = h \circledast r_1 + m_1$ e $c_2 = h \circledast r_2 + m_2$ no anel $\mathcal{R}_{N,q}$.
Tomemos a soma das mensagens cifradas $c_1 + c_2$ e apliquemos a função de decriptação:

$$\begin{aligned}
f \circledast (c_1 + c_2) &= f \circledast (h \circledast r_1 + m_1 + h \circledast r_2 + m_2) \\
&= f \circledast h \circledast (r_1 + r_2) + f \circledast (m_1 + m_2) \\
&= g \circledast (r_1 + r_2) + f \circledast (m_1 + m_2) \in \mathcal{R}_{N,q} \\
&= m_1 + m_2 \in \mathcal{R}_{N,p}
\end{aligned}$$

[3] Por simplicidade, polinômios ternários que representam mensagens mapeadas serão chamados apenas de mensagens

Então

$$\begin{aligned}&NTRU\_DECRYPT(NTRU\_ENCRYPT(m_1,h)+NTRU\_ENCRYPT(m_2,h),f)=\\&=NTRU\_DECRYPT(NTRU\_ENCRYPT(m_1,h),f)+\\&+NTRU\_DECRYPT(NTRU\_ENCRYPT(m_2,h),f)\end{aligned}$$

e segue que a propriedade de homomorfismo aditivo 2.1 é válida.

Considere agora um número inteiro $\alpha \geq 0$ com no máximo N bits. Podemos mapeá-lo para um polinômio $m(x) = \sum_{i=0}^{N-1} a_i x^i \in \mathbb{Z}[x]$, onde o i-ésimo coeficiente de $m(x)$ assume o mesmo valor do i-ésimo bit de α, e com isso temos que $m(2) = \alpha$.

Suponha que queiramos somar M números inteiros cifrados $c_i, i = 1...M$ a partir de uma mesma chave pública h. Ao generalizar o resultado anterior, temos:

$$\begin{aligned}f \circledast \sum_{i=1}^{M} &= f \circledast \sum_{i=1}^{M}(h \circledast r_i + m_i) = f \circledast h \circledast \sum_{i=1}^{M} r_i + f \circledast \sum_{i=1}^{M} m_i \\ &= g \circledast \sum_{i=1}^{M} r_i + f \circledast \sum_{i=1}^{M} m_i \in \mathcal{R}_{N,q} \\ &= \sum_{i=1}^{M} m_i \in \mathcal{R}_{N,p}\end{aligned}$$

Contudo, tal generalização não é válida por si só.

Sejam $m_j(x) = \sum_{i=0}^{N-1} a_{i,j} x^i$, onde $a_{i,j}$ representa o i-ésimo coeficiente do j-ésimo inteiro e $r(x) = \sum_{j=1}^{M} m_j(x) = \sum_{j=1}^{M} \sum_{i=0}^{N-1} a_{i,j} x^i = \sum_{i=0}^{N-1} \sum_{j=1}^{M} a_{i,j} x^i = \sum_{i=0}^{N-1} c_i x^i$ a soma de M inteiros. No último passo de $NTRU\_DECRYPT()$, ao passarmos $r(x)$ do anel $\mathcal{R}_{N,q}$ para o $\mathcal{R}_{N,p}$, realizamos uma redução módulo p em cada um de seus coeficientes. Caso $c_i \geq p$ para algum i, teremos perda de informação e o valor da soma não será o esperado. Como o valor máximo que um coeficiente c_i pode ter é $\sum_{j=1}^{M} max(a_{i,j}) = \sum_{j=1}^{M} 1 = M$, então para que a generalização acima seja válida, devemos garantir que $p > M$ (e consequentemente $q \gg M$).

4.4 Homomorfismo multiplicativo

A propriedade multiplicativamente homomórfica não é tão direta quanto a aditiva. Para a decifração do produto entre duas mensagens cifradas, devemos realizar uma convolução

com a chave privada f^2 em vez de f:

$$\begin{aligned} f^2 \circledast (c_1 \circledast c_2) &= f^2 \circledast ((h \circledast r_1 + m_1) \circledast (h \circledast r_2 + m_2)) \\ &= f^2 \circledast (h^2 \circledast r_1 \circledast r_2 + h \circledast r_1 \circledast m_2 + h \circledast r_2 \circledast m_1 + m_1 \circledast m_2) \\ &= g^2 \circledast (r_1 \circledast r_2) + f \circledast g \circledast r_1 \circledast m_2 + f \circledast g \circledast r_2 \circledast m_1 + f^2 \circledast m_1 \circledast m_2 \in \mathcal{R}_{N,q} \\ &= m_1 \circledast m_2 \in \mathcal{R}_{N,p} \end{aligned}$$

Suponha uma mensagem cifrada $c_1 = {}_{NTRU\_ENCRYPT(m_1,h)}$ e suponha também que não sabemos se ela é ou não o produto entre duas outras mensagens cifradas. Vimos que o algoritmo é correto quando realizamos a convolução entre f e c_1. Vejamos o que acontece quando consideramos f^2:

$$\begin{aligned} f^2 \circledast (c_1) &= f^2 \circledast (h \circledast r_1 + m_1) \\ &= f \circledast g \circledast r_1 + f^2 \circledast m_1 \in \mathcal{R}_{N,q} \\ &= m_1 \in \mathcal{R}_{N,p} \end{aligned}$$

Portanto não é necessário saber se uma mensagem cifrada é de fato o produto entre duas outras mensagens cifradas.

Então

$$\begin{aligned} &NTRU\_DECRYPT(NTRU\_ENCRYPT(m_1,h) \circledast NTRU\_ENCRYPT(m_2,h), f^2) = \\ &= NTRU\_DECRYPT(NTRU\_ENCRYPT(m_1,h), f) \circledast \\ &\circledast NTRU\_DECRYPT(NTRU\_ENCRYPT(m_2,h), f) \end{aligned}$$

e segue que a definição de multiplicação homomórfica 2.2 é válida.

Assim como no caso da adição homomórfica, vamos generalizar essa propriedade para a multiplicação de M inteiros. A primeira limitação que devemos notar é a possibilidade de um *overflow*: como o grau máximo do polinômio resultante é $N-1$, poderemos multiplicar M polinômios de grau $\lfloor \frac{N-1}{M} \rfloor$. Aqui, definiremos $r(x) = \prod_{j=1}^{M} m_j(x) = \prod_{j=1}^{M} \sum_{i=0}^{\lfloor \frac{N-1}{M} \rfloor} a_{i,j} x^i = \sum_{i=0}^{M \lfloor \frac{N-1}{M} \rfloor} c_i x^i$ como o produto de M inteiros.

$$
\begin{aligned}
f^M \circledast (\prod_{i=1}^{M} c_i(x)) &= f^M \circledast (\prod_{i=1}^{M} (h \circledast r_i + m_i)) \\
&= f^M \circledast (h^M \circledast \prod_{i=1}^{M} r_i + [\text{potências de } h^j, 1 \leq j \leq M-1] + \prod_{i=1}^{M} m_i) \\
&= g^M \circledast \prod_{i=1}^{M} r_i + [\text{potências de } g^j \circledast f^{M-j}, 1 \leq j \leq M-1] \\
&+ f^M \circledast \prod_{i=1}^{M} m_i \in \mathcal{R}_{N,q} \\
&= \prod_{i=1}^{M} m_i \in \mathcal{R}_{N,p}
\end{aligned}
$$

Conforme visto anteriormente, para que não haja nenhuma perda de informação durante a decifração, devemos saber qual o valor máximo que um coeficiente do polinômio $r(x)$ pode ter e então nos certificar de que o primo p seja maior que esse valor. Para isso, consideraremos o caso em que os coeficientes dos fatores $m_j(x)$ possuem valor máximo:

$$r_{max}(x) = \prod_{j=1}^{M} \sum_{i=0}^{\lfloor \frac{N-1}{M} \rfloor} x^i$$

Ao multiplicarmos dois fatores, o polinômio resultante (parcial) será $\sum_{i=0}^{\lfloor \frac{N-1}{M} \rfloor} (x^i + \cdots + x^{\lfloor \frac{N-1}{M} \rfloor + i})$. Esse polinômio será multiplicado por outro fator, teremos novo polinômio resultante $\sum_{j=1}^{\lfloor \frac{N-1}{M} \rfloor} \sum_{i=1}^{\lfloor \frac{N-1}{M} \rfloor} (x^{i+j} + \cdots + x^{\lfloor \frac{N-1}{M} \rfloor + i + j})$ e assim sucessivamente.

Apesar dos cálculos acima tornarem-se muito complexos para obtermos uma fórmula fechada em função de M e N, as fórmulas parciais permitem obter um limite inferior Ωn^{m-1}, além de um dispositivo prático.

Considere $N = 9$ e $M = 4$. Multiplicaremos polinômios de grau $\lfloor \frac{N-1}{M} \rfloor = 2$ e coeficientes com valor máximo, isso é, $1 + x + x^2$. As tabelas a seguir descrevem o dispositivo prático e possuem como entrada os coeficientes dos polinômios.

	x^0	x^1	x^2	
x^0	1			1
x^1	1	1		2
x^2	1	1	1	3
x^3		1	1	2
x^4			1	1

	x^0	x^1	x^2	
x^0	1			1
x^1	2	1		3
x^2	3	2	1	6
x^3	2	3	2	7
x^4	1	2	3	6
x^5		1	2	3
x^6			1	1

	x^0	x^1	x^2	
x^0	1			1
x^1	3	1		4
x^2	6	3	1	10
x^3	7	6	3	16
x^4	6	7	6	19
x^5	3	6	7	16
x^6	9	3	6	10
x^7		9	3	4
x^8			9	1

É importante notar que o termo central possui sempre o maior valor e que os demais termos são simétricos em relação a ele. Generalizando, obtemos um algoritmo para calcular o maior coeficiente do polinômio $r(x)$:

Algorithm 7 Calcula o maior coeficiente de um polinômio binário no anel módulo $x^N - 1$ após M multiplicações

```
function HOMOMORPHIC_MULTIPLICATION_PARAM(N, M)
    n ← ⌊(N−1)/M⌋
    v ← (n + 1) ∗ [1] [4]
    for i ← 2 · · · M do
        w ← (in + 1) ∗ [0]
        for j ← 0 · · · ⌊(in+1)/2⌋ do
            for k ← j · · · 0, −1 do
                if k ≤ n||k ≤ n − j then
                    w[k]+ = v[k]
                else
                    break
                end if
            end for
        end for
        for j ← ⌊(in+1)/2⌋ + 1 · · · in do
            w[j] ← v[in − j]
        end for
        v ← w
    end for
    return v[⌊(nM+1)/2⌋]
end function
```

4.5 Fundamentação teórica: Reticulados

Apresentamos anteriormente a interpretação do problema sob polinômios. Porém, para que possamos compreender a dificuldade na qual o problema está baseado, faz-se necessário introduzir conceitos para que possamos entender o problema sob o ponto de vista de reticulados.

[4] cria uma lista com $n + 1$ posições preenchidas com 1

Definição 4.3. *Sejam $b_1, b_2, ..., b_n \in \mathbb{R}^m$ vetores linearmente independentes tal que $n \leq m$, então um reticulado $\mathcal{L}$ onde esses vetores formam uma base é definido por:*

$$\mathcal{L}(b_1, b_2, ..., b_n) = \left\{ \sum_{i=1}^{n} \alpha_i b_i \mid \alpha_i \in \mathbb{Z} \right\}$$

Informalmente, podemos ainda definir um reticulado como o conjunto de pontos de intersecção em uma grade n-dimensional infinita e regular (mas não necessariamente ortogonal).

Abordaremos neste trabalho apenas casos em que $m = n$.

Definição 4.4 (SVP)**.** *Dado um reticulado $\mathcal{L}$, o problema do vetor mais curto consiste em encontrar um vetor $v \in \mathcal{L} \mid v \neq 0$ que minimize a norma Euclidiana $|v|_2$.*

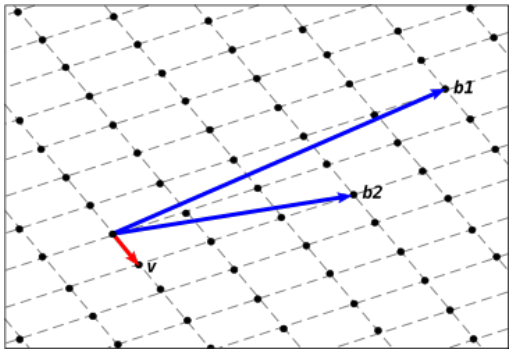

Figura 4.1: O vetor v é o mais curto do reticulado gerado por b_1 e b_2.

Definição 4.5 (CVP)**.** *Dado um reticulado $\mathcal{L}$ e um vetor $w \in R^n$, w não necessariamente em $\mathcal{L}$, o problema do vetor mais próximo consiste em encontrar um vetor $v \in L$ que minimize a norma Euclidiana $||w - v||$.*

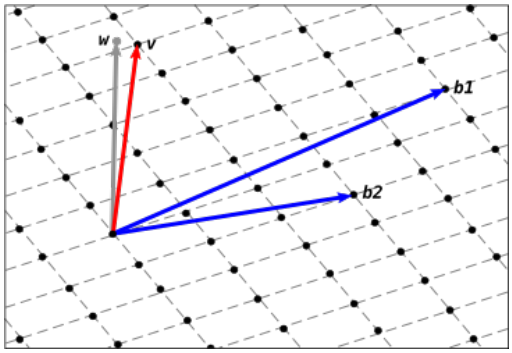

Figura 4.2: O vetor v do reticulado gerado por b_1 e b_2 é o mais próximo de w.

Definição 4.6. *Um vetor* $[v_0, v_1, ..., v_n$ *é dito curto se* $\exists c$ *tal que* $1 \leq c \ll q$ *e* $|v_i| \leq c, \forall i \in \{0, 1, ..., n\}$.

De maneira geral, ambos os problemas querem achar a base mais eficiente do reticulado, formada pelos vetores mais curtos e ortogonais o possível.

Vejamos agora a estrutura utilizada pelo criptossistema e qual sua relação com o problema difícil cuja segurança do NTRU é baseada.

Definição 4.7. *Um reticulado convolucional modular* $\mathcal{L}(B)$ *do tipo* (N, q) *é gerado pelas linhas da matriz* $2N$*-dimensional* B*:*

$$B = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right] = \left[\begin{array}{c|c} I_N & C(h) \\ \hline 0 & q \cdot I_N \end{array}\right]$$

onde $C(h)$ *é a matriz circulante gerada pelo vetor* h *e o elemento* $C_{i,j} = h_{(j-i) \pmod N}$

Note que precisamos apenas de aproximadamente $N \log q$ bits para representar a base B.

Considere o vetor $[f \mid g] \in \mathcal{L}(B)$, onde as N primeiras componentes correspondem aos coeficientes da chave privada f, e as outras N componentes correspondem ao polinômio g.

Como $f = p \cdot f' + 1$, $g = p \cdot g'$ e f' e g' são polinômios ternários, então seus coeficientes em valor absoluto serão menores ou iguais a $p \ll q$ e portanto, da definição 4.6, o vetor $[f \mid g]$ é um vetor curto. Sua norma Euclidiana é $|[f \mid g]|_2 = \sqrt{d_f(-p)^2 + d_f(p)^2 + d_g(-p)^2 + d_g(p)^2} = p\sqrt{2d_f + 2d_g} = O(\sqrt{N})$.

Sabemos que $h = f_q \circledast g = f^{-1} \circledast g \pmod q$, então $\exists v$tal que$f \circledast h = g + q \cdot v$. Portanto o reticulado $\mathcal{L}(B)$ contém o vetor $[f \mid g] = [f \mid -v]B$:

$$[f \mid -v] \cdot \left[\begin{array}{c|c} I_N & C(h) \\ \hline 0 & q \cdot I_N \end{array}\right] = [I_N f + 0 - v \mid C(h)f + q \cdot I_N - v] = [f \mid C(h)f - q \cdot v] = [f \mid g]$$

A chave pública h forma uma base completa conforme a definição 4.7. No entanto h não é uma base curta uma vez que seus coeficientes estão no anel $\mathcal{R}_{N,q}$ e não podemos garantir que $|h_i| \ll q, \forall i \in \{0, 1, ..., N-1\}$. Por esse motivo, consideramos h uma base longa.

Então, dada uma base longa h de um reticulado $\mathcal{L}(B)$, um atacante teria que achar o vetor curto $[f \mid g]$ nessa base para poder decifrar a mensagem, o que não é mais difícil que aproximar o SVP a um fator de $O(\sqrt{N})$.

Atualmente, sabemos que aproximar o SVP a um fator constante é um problema NP-difícil [Micciancio(2001)] e que aproximar o SVP a um fator de $O(\sqrt{N})$ está na classe de complexidade NP ∩ co–NP [Aharonov e Regev(2005)], porém não sabemos se está em P.

Vejamos quais os passos que um atacante com conhecimento de um vetor curto próximo $[f' \mid g']$ poderia tomar:

Seja $[f' \mid g'] = [f' \mid -v']$ um vetor curto em $\mathcal{L}(B)$ para algum vetor v' tal que $f' * h = g' + q \cdot v'$. Suponha que $q \cdot v'$ seja inversível módulo p, com $z = (q \cdot v')^{-1} \pmod p$.

Seja também

$$\begin{aligned} a' &= f' \circledast c = f' \circledast (m + h \circledast r) \\ &= f' \circledast m + f' \circledast h \circledast r = f' \circledast m + (g' + q \cdot v') \circledast r \\ &= f' \circledast m + g' \circledast r + q \cdot v \circledast r \\ &= f' \circledast m + g' \circledast r \in \mathcal{R}_{N,q} \end{aligned}$$

Para parâmetros devidamente escolhidos e $[f' \mid g']$ curto o suficiente, a igualdade acima vale não somente para $\mathcal{R}_{N,q}$, mas também para $\mathcal{R}$. Logo a igualdade

$$a' = f' \circledast (m) + g' \circledast (r) \in \mathcal{R}_{N,q} \tag{4.4}$$

é válida sem redução $\pmod q$.

Como $f' \circledast h' - g' = q \cdot v'$ e $z = (q \cdot v')^{-1} \pmod p$, então $(f' \circledast h - g') \circledast z' = 1$. Mas

$$\begin{aligned} a' &= a' \circledast 1 = a' \circledast ((f' \circledast h - g') \circledast z') \\ &= f' \circledast a' \circledast h \circledast z - g \circledast a' \circledast z \in \mathcal{R}_{N,p} \end{aligned}$$

$$a' = f' \circledast (a' \circledast h \circledast z) - g \circledast (a' \circledast z) \in \mathcal{R}_{N,p} \tag{4.5}$$

Comparando-se as expressões 4.4 e 4.5, temos que $m = a' \circledast h \circledast z$ e $r = a' \circledast z$. Portanto, se um atacante conhecesse um vetor $[f' \mid g']$ suficientemente próximo de $[f \mid g]$ ele poderia pré-calcular $d = q \cdot f' \circledast h \circledast (q \cdot v')^{-1}$ e obter $m = c \circledast d \pmod p$

4.6 Taxa de falha e escolha de parâmetros

Uma falha durante o processo de decifração pode ocorrer ao realizarmos a convolução entre uma mensagem cifrada por c pela chave privada f. Se a escolha de parâmetros não for boa, podemos acabar com um dos coeficientes de $f \circledast c = f \circledast m + g \circledast r$ maior do que q em módulo. Como estamos trabalhando no anel $\mathcal{R}_{N,q}$, esse coeficiente sofrerá uma redução módulo q e teremos perda de informação, similar ao que abordamos em 4.3 e 4.4. Portanto, é necessário garantir que $|f \circledast m + g \circledast r|_\infty < q$.

A literatura relacionada realiza uma análise dos parâmetros sob o NTRU não homomórfico [Hoffstein *et al.*(2015)Hoffstein, Pipher, Schanck, Silverman, Whyte, e Zhang], de onde podemos observar que existe uma correlação entre os parâmetros p e q. Entretanto, não encontramos critérios que consideram nível de segurança, número de produtos por termo e número de termos por expressão aritmética, a fim de automatizar a escolha de parâmetros N, p e q em uma aplicação homomórfica.

Capítulo 5

Implementação

Os algoritmos abordados foram implementados com foco na experimentação das propriedades homomórficas porpostas.

5.1 Implementação do Paillier

Inicialmente, foi realizada uma implementação do algoritmo de Paillier com a linguagem *Python 3*. Essa linguagem possui suporte nativo para inteiros de precisão arbitrária e exponenciações modulares (fazendo-se desnecessário a implementação de um algoritmo como double-and-add).

Além das funções já descritas pela teoria, foi também implementado o algoritmo Euclides estendido para inteiros. Caso exista, o algoritmo obtém o inverso a^{-1} de um número $a \pmod{n}$. É utilizado para calcular $\mu \equiv L(g^{\lambda} \pmod{n^2})^{-1} \pmod{n}$ durante o processo de geração de chaves.

Por questões de performance, o algoritmo foi então implementado na linguagem *C*, com o uso da biblioteca de *GMP* (*GNU Multiple Precision Arithmetic Library*); trata-se de uma biblioteca otimizada para aritmética de inteiros de precisão arbitrária. A biblioteca suporta não somente exponenciações modulares, mas também uma implementação nativa para o cálculo do inverso modular e do teste de primalidade de Miller-Rabin, necessário para obtenção de números primos aleatórios de k bits durante o processo de geração de chaves.

Para gerarmos primos aleatórios de k bits, realizamos os seguintes passos: Obtém-se uma cadeia aleatória de k bits a partir da saída $\backslash dev \backslash urandom$ do linux, ou então a partir da instrução *rdrand* caso suportada pelo processador (possui melhor desempenho por ser uma simples instrução a nível de máquina). Em seguida, certifica-se de que a cadeia representa um número ímpar a partir de uma simples operação *AND* com 1. O passo final consiste em aplicar o teste de primalidade de Miller-Rabin que indicará se o número é primo com uma probabilidade de erro inversamente proporcional ao número de iterações do algoritmo. Caso a probabilidade de ser um primo seja pequena, repete-se o processo.

5.2 Implementação do NTRU

O algoritmo também foi inicialmente implementado na linguagem *Python* e depois na linguagem *C*.

Como visto na parte teórica, o criptossistema é construido sobre anéis polinomiais quociente com o ideal $\langle x^N - 1 \rangle$, ou seja, polinômios de grau máximo $N-1$, e coeficientes $\pmod q$ ou $\pmod p$.

Foi implementada uma biblioteca com as principais operações polinomiais utilizadas pelo algoritmo. Além da soma, as duas outras principais operações implementadas foram a convolução entre dois polinômios e o cálculo do inverso de um polinômio no anel. Esse último também é obtido através do algoritmo de Euclides estendido, porém para polinômios.

Algorithm 8 Cálculo do inverso do polinômio f módulo $x^N - 1$

function EUCLIDES_ESTENDIDO_POLINÔMIO(f)
  $u \leftarrow f,\ r \leftarrow 1$
  $v \leftarrow x^N - 1,\ s \leftarrow 0$
  while $grau(u) > 0$ **do**
    if $grau(u) < grau(v)$ **then**
      $u \leftrightarrow v,\ r \leftrightarrow s$
    end if
    $j \leftarrow grau(u) - grau(v)$
    $c \leftarrow \frac{u_{grau(u)}}{v_{grau(v)}}$
    $u \leftarrow u - cx^j v$
    $v \leftarrow v - cx^j s$
  end while
  if $grau(u) = 0$ **then**
    return r/u_0
  else
    Não possui inverso
  end if
end function

Observe que a cada iteração é necessário obter o inverso do coeficiente de maior grau de v, contudo esse deve ser o inverso módulo q, obtido através do algoritmo de Euclides estendido para inteiros.

As operações mais custosas envolvem o cálculo do inverso polinomial durante a geração de chaves, e reduções módulo q. Como q não precisa ser necessariamente um primo, podemos escolhê-lo como uma potência de dois para otimizar as reduções modulares: basta realizar uma operação *AND* entre o número e uma máscara com os $\log q - 1$ últimos bits iguais a 1 e o restantes iguais a 0.

Note que em *C*, como os polinômios são representados por ponteiros de áreas alocadas, a troca entre os vetores $u \leftrightarrow v$ e $r \leftrightarrow s$ no algoritmo de Euclides estendido pode ser feita em tempo constante em vez de $O(N)$.

Apesar da literatura utilizar valores de q que cabem em inteiros de 32 bits, optamos pela utilização da biblioteca *GMP*, onde cada coeficiente do polinômio é um inteiro de precisão múltipla.

Por esperarmos que os números ficassem grandes, preferimos abordar uma implementação inicial em *C* para inteiros de 64 bits. Como eram necessárias operações de multiplicação seguidas de redução modular entre inteiros de 64 bits, fez-se necessário criar uma biblioteca para inteiros de 128 bits. A biblioteca obtida, além de possuir número limitado de bits também possuía performance inferior à *GMP*, otimizada com uso de vetorização e instruções em *assembly*. Por isso preferimos adotar a *GMP* na implementação do NTRU.

5.3 Testes e experimentações

Os testes envolvendo somas homomórficas consistiram em gerar M números inteiros e encriptá-los. A partir disso, multiplicavam-se (Paillier) ou somavam-se (NTRU) os M criptogramas e somavam-se os M inteiros. Por fim. decriptava-se a soma dos M criptogramas a fim de conferir se o resultado obtido era igual ao da soma dos M inteiros.

Conforme visto na seção 4.6, há uma correlação entre a taxa de falha na decifração do NTRU e os parâmetros p e q. Sabemos que no caso da adição homomórfica podemos somar no máximo $p-1$ termos, porém não sabemos calcular de forma precisa qual valor q é necessário.

A partir da relação $|f \circledast m_s + g \circledast r_s|_\infty < q$, necessária para garantir que não ocorra falhas durante a decifração, conseguimos obter um limite superior para o caso aditivamente homomórfico:

$$\begin{aligned}
|f \circledast m_s + g \circledast r_s|_\infty &= |(p \cdot f' + 1) \circledast m_s + p \cdot g' \circledast r_s|_\infty \\
&\leq |(p \cdot f' + 1) \circledast m_s + p \cdot g' \circledast m_s|_\infty \\
&= |(p \cdot f' + p \cdot g' + 1) \circledast m_s|_\infty \\
&< p \cdot |(f' + g' + 1) \circledast m_s|_\infty \\
&= p \cdot |3 \cdot \sum_{i=0}^{N-1} x^i \circledast (p-1) \cdot \sum_{i=0}^{N-1} x^i|_\infty \\
&< 3 \cdot p \cdot (p-1) |\sum_{i=0}^{N-1} x^i \circledast \sum_{i=0}^{N-1} x^i|_\infty \\
&= 3 \cdot p \cdot (p-1) \cdot N
\end{aligned}$$

O limite superior obtido não considera a aleatoriedade dos polinômios ternários e mensagens (polinômios binários, no nosso caso) utilizadas na prática. Com o intuito de refinar o resultado anterior, foi realizado um experimento que consistiu em gerar um par de chaves públicas para valores iniciais de p e q, e somar $p-1$ mensagens cifradas. Caso a decifração fosse bem sucedida após dado número de repetições, p tornava-se o próximo primo maior que p. Caso contrário, q recebia o próximo primo maior que q.

Os dados foram gerados para valores de N primos, próximos dos 32 e 64 bits utilizados para representar inteiros.

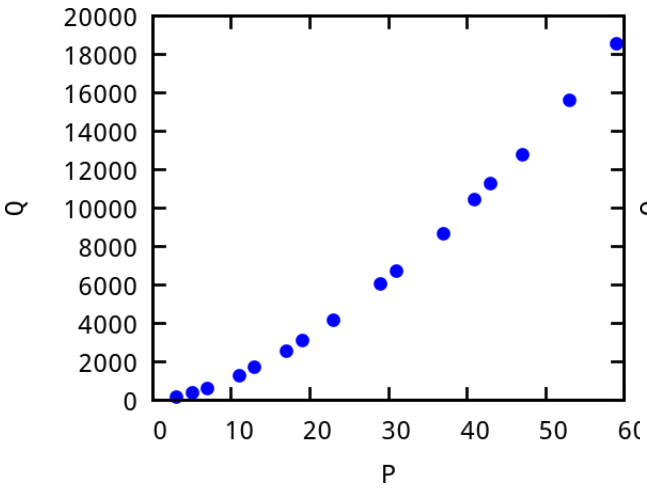

Figura 5.1: Valores de q obtidos após 1000 repetições bem sucedidas de $p-1$ somas homomórficas, onde $N = 31$.

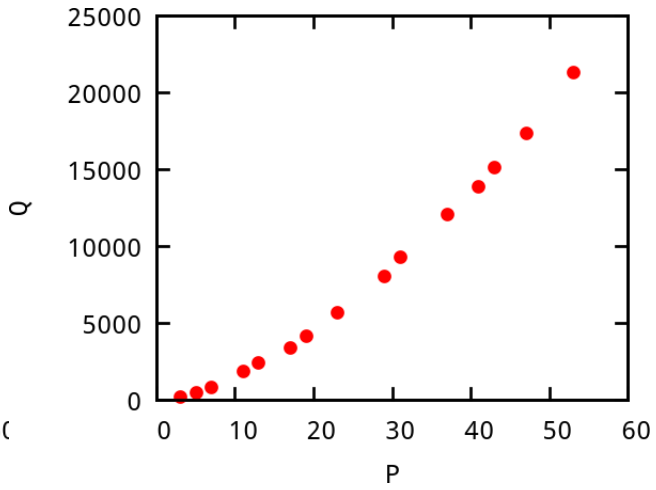

Figura 5.2: Valores de q obtidos após 1000 repetições bem sucedidas de $p-1$ somas homomórficas, onde $N = 61$.

Um exemplo com parâmetros e precisão menores pôde avaliar uma maior quantidade de valores q em função de p.

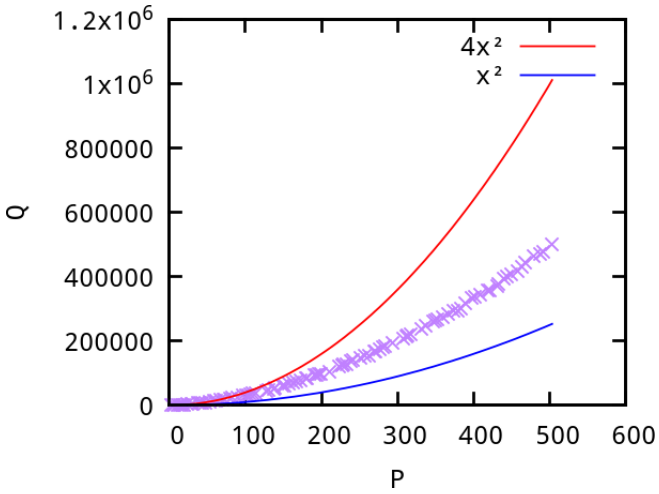

Figura 5.3: Valores de q obtidos após 100 repetições bem sucedidas de $p-1$ somas homomórficas, onde $N = 19$. Os resultados obtidos sugerem que $q \in \Theta(p^2)$

Considerando o caso multiplicativamente homomórfico, a partir do algoritmo 7 foi possível observar qual valor o parâmetro p deve assumir para realizarmos M multiplicações em um reticulado de dimensão N.

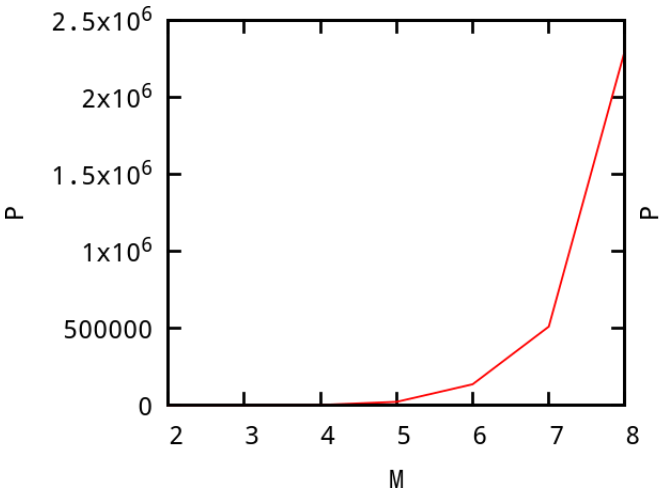

Figura 5.4: Valores de p obtidos para M multiplicações em um reticulado de dimensão $N = 67$

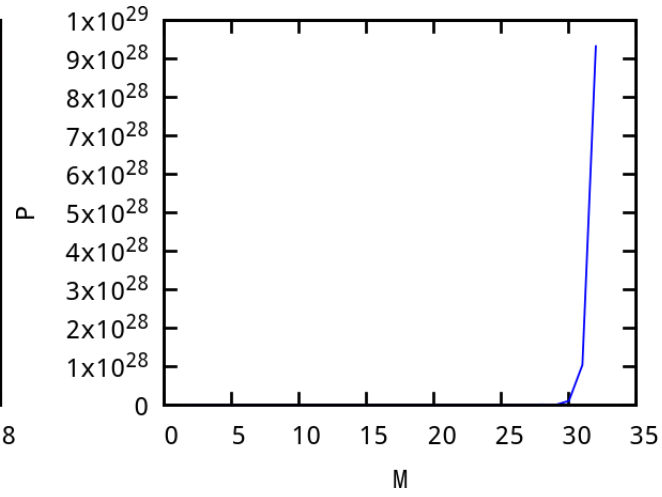

Figura 5.5: Valores de p obtidos para M multiplicações em um reticulado de dimensão $N = 257$

Como podemos observar, os valores de p crescem muito rapidamente em função do número de multiplicações suportadas. Ainda, assim como no caso aditivamente homomórfico, temos que considerar que o parâmetro q deve aumentar em função de p a fim de evitar a taxa de falhas. O teste a seguir nos mostra o quão grande esses parâmetros podem se tornar para um número relativamente pequeno de multiplicações.

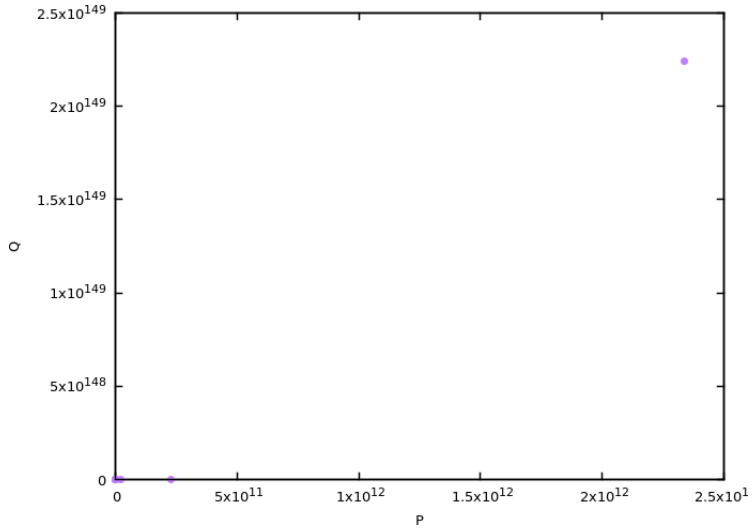

Figura 5.6: Valores de q em função de p, obtidos após 1000 repetições bem sucedidas de $M = [2, ..., 10]$ multiplicações homomórficas em um reticulado de dimensão $N = 257$.

Nesse teste, para $N = 257$ e $M = 10$, temos $p = 2338583373809$ e $q = 22397252385161808061020588281426816305913920097309203176893427298557050406343111370790515067246111334052184046852743973898552638610412341933165072332 3$. Como o crescimento desses parâmetros é grande, a escala utilizada aglomera os pares (p, q), para $M < 9$, próximos à origem do grafico.

Vale ressaltar que, para evitar um overflow, restringimos os graus dos polinômios mul-

tiplicados de tal forma que não atinjam grau igual a N. Realizar uma abordagem menos restritiva dos parâmetros é área de pesquisa futura.

Capítulo 6

Conclusão

Neste trabalho, foi possível avaliar criptossistemas homomórficos sob o ponto de vista teórico e de implementação.

A teoria estudada cobriu dois dos problemas computacionalmente difíceis mais utilizados em criptografia de chave pública atual. O Paillier tem sua segurança baseada no problema dos resíduos quadráticos. Sua teoria é fundamentada na função demonstravelmente bijetora de encriptação e no teorema de Carmichael. O NTRU, apesar de possuir uma interpretação polinomial, é construído sob a teoria dos reticulados (mais especificamente, reticulados convolucionais modulares). Sua segurança advém da dificuldade de se obter vetores curtos, com norma $O(\sqrt{N})$ em vez de $O(q\sqrt{N})$. O segundo algoritmo, além de possuir menor complexidade de tempo e menor tamanho de chaves em relação ao primeiro, é ainda conjecturado pós-quântico.

Por fim, concluímos que os criptossistemas estudados jamais conseguirão suprir todas as necessidades da computação na nuvem, onde volumes cada vez maiores de dados devem ser processados rapidamente por esses algoritmos. Entretanto, isso não impossibilita seu uso em aplicações específicas e de menor porte.

O Paillier nos permite realizar o cálculo de médias simples e até médias ponderadas, onde os pesos de ponderação são públicos (como trabalha com inteiros, a divisão final deverá ser feita pelo próprio usuário). Apesar de ser capaz de realizar grande número de somas de criptogramas na prática, trata-se de um esquema limitado em aplicações reais que exigem cálculos mais complexos.

O criptossistema NTRU mostrou-se viável para uma aplicação que exige um número bem limitado de operações homomórficas. Empiricamente, vimos que o valor q deve aumentar para evitar falhas durante o processo de decifração conforme comportamos um número maior de operações homomórficas. Isso implica em operações mais lentas e chaves maiores. Ainda, determinar parâmetros seguros, bem como a viabilidade de se utilizar a técnica de *bootstrapping*, constituem áreas de pesquisa futura na área.

Referências Bibliográficas

[Aharonov e Regev(2005)] **Aharonov e Regev(2005)** Dorit Aharonov e Oded Regev. Lattice problems in np ∩ conp. *J. ACM*, 52(5):749–765. ISSN 0004-5411. doi: 10.1145/1089023.1089025. URL http://doi.acm.org/10.1145/1089023.1089025. Citado na pág. 27

[Alkim *et al.*(2016)Alkim, Ducas, Pöppelmann, e Schwabe] **Alkim *et al.*(2016)** Erdem Alkim, Léo Ducas, Thomas Pöppelmann e Peter Schwabe. Post-quantum key exchange - A new hope. Em *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, páginas 327–343. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim. Citado na pág. 18

[Barreto(2015)] **Barreto(2015)** Paulo Sérgio Licciardi Messeder Barreto. Lecture notes: Post-quantum cryptography - lattices, 2015. Citado na pág. 17

[Bollauf(2014)] **Bollauf(2014)** Maiara Francine Bollauf. Criptografia baseada em reticulados, 2014. URL http://www.ime.unicamp.br/~campello/geometria/maiara.pdf. Citado na pág. 17

[Boneh *et al.*(2005)Boneh, Goh, e Nissim] **Boneh *et al.*(2005)** Dan Boneh, Eu-Jin Goh e Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. Em *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, páginas 325–341, Berlin, Heidelberg. Springer-Verlag. ISBN 3-540-24573-1, 978-3-540-24573-5. doi: 10.1007/978-3-540-30576-7_18. URL http://dx.doi.org/10.1007/978-3-540-30576-7_18. Citado na pág. 2

[Gentry(2009)] **Gentry(2009)** Craig Gentry. Fully homomorphic encryption using ideal lattices. Em *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, páginas 169–178, New York, NY, USA. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536440. URL http://doi.acm.org/10.1145/1536414.1536440. Citado na pág. 2, 6

[Hoffstein *et al.*(2015)Hoffstein, Pipher, Schanck, Silverman, Whyte, e Zhang] **Hoffstein *et al.*(2015)** Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte e Zhenfei Zhang. Choosing parameters for ntruencrypt. Cryptology ePrint Archive, Report 2015/708, 2015. http://eprint.iacr.org/2015/708. Citado na pág. 17, 28

[Hoffstein *et al.*(1998)Hoffstein, Pipher, e Silverman] **Hoffstein *et al.*(1998)** Jeffrey Hoffstein, Jill Pipher e Joseph H. Silverman. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter NTRU: A ring-based public key cryptosystem, páginas 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-69113-6. doi: 10.1007/BFb0054868. URL http://dx.doi.org/10.1007/BFb0054868. Citado na pág. 17

[Micciancio(2001)] **Micciancio(2001)** Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035. ISSN 0097-5397. doi: 10.1137/S0097539700373039. URL http://dx.doi.org/10.1137/S0097539700373039. Citado na pág. 27

[O'Keeffe(2008)] **O'Keeffe(2008)** Michael O'Keeffe. The paillier cryptosystem: A look into the cryptosystem and its potential application, 2008. URL http://www.tcnj.edu/~hagedorn/papers/CapstonePapers/OKeeffe/CapstoneOKeeffeCryptography.pdf. Citado na pág. 7

[Paillier(1999)] **Paillier(1999)** Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. Em *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, páginas 223–238. Springer. doi: 10.1007/3-540-48910-X_16. Citado na pág. 2, 7

[Pipher(2002)] **Pipher(2002)** Jill Pipher. Lectures on the ntru encryption algorithm and digital signature scheme: Grenoble, june 2002, 2002. URL http://www.math.brown.edu/~jpipher/grenoble.pdf. Citado na pág. 17

[Rivest *et al.*(1978)Rivest, Shamir, e Adleman] **Rivest** *et al.***(1978)** R. L. Rivest, A. Shamir e L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126. ISSN 0001-0782. doi: 10.1145/359340.359342. URL http://doi.acm.org/10.1145/359340.359342. Citado na pág. 4, 7

[Shor(1997)] **Shor(1997)** Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509. ISSN 0097-5397. doi: 10.1137/S0097539795293172. URL http://dx.doi.org/10.1137/S0097539795293172. Citado na pág. 7

[Volkhausen(2006)] **Volkhausen(2006)** Tobias Volkhausen. Paillier cryptosystem: A mathematical introduction, 2006. URL http://www2.cs.uni-paderborn.de/cs/ag-bloemer/lehre/proseminar_WS2005/material/Volkhausen_Ausarbeitung.pdf?PHPSESSID=edd79a27cb022cfa749b4a2baa7ea6f0. Citado na pág. 7