GIANCARLO KLEMM CAMILO

# ANÁLISE DO SISTEMA DE CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICO DE GENTRY

CURITIBA

2015

GIANCARLO KLEMM CAMILO

# ANÁLISE DO SISTEMA DE CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICO DE GENTRY

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Paraná.
Orientador: Prof. Dr. Luiz Carlos P. Albini

CURITIBA

2015

# SUMÁRIO

**RESUMO** **iii**

**ABSTRACT** **iv**

**ABREVIATURAS** **v**

**1 INTRODUÇÃO** **1**

**2 CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA** **3**

2.1 Definição . . . . . 3

2.2 O sistema estudado . . . . . 4

2.3 Construção do sistema . . . . . 4

2.3.1 Sistema parcialmente homomórfico . . . . . 5

2.3.2 De parcialmente para completamente homomórfico . . . . . 6

2.3.3 Simplificando a descriptografia . . . . . 8

2.4 Limitações atuais . . . . . 9

2.5 Segurança . . . . . 9

2.5.1 Padrões de segurança . . . . . 10

2.5.2 Prova de segurança semântica . . . . . 10

2.5.3 Segurança com relação à dica dentro chaves . . . . . 11

**3 APLICAÇÕES** **12**

3.1 Computação em nuvem . . . . . 12

3.2 Buscas seguras . . . . . 12

3.3 Segurança de software . . . . . 13

3.4 Re-criptografia de Proxys . . . . . 13

3.5 Filtros de emails sigilosos . . . . . 13

**4 TESTES E RESULTADOS** **14**

4.1 Implementação do algoritmo . . . . . 14

4.2 Arquitetura usada . . . . . 14

4.3 Algoritmo usado para comparação . . . . . 14

4.4 Metodología . . . . . 15

4.5 Velocidade . . . . . 15

4.6 Segurança . . . . . 16

4.7 Funcionalidade . . . . . 16

**5 CONCLUSÃO** **17**

**BIBLIOGRAFIA** **18**

# RESUMO

A criptografia completamente homomórfica (**CCH**) permite que qualquer função matemática possa ser aplicada aos dados criptografados. Esse conceito surgiu logo após a criação do RSA porém somente anos depois Craig Gentry foi o primeiro a ter uma solução para este problema. A segurança do CCH se baseia em partes do algoritmo que podem ser reduzidas a problemas difíceis. Enquanto esses problemas continuarem a não ter um solução eficiente, o sistema é considerado seguro. Há várias aplicações possíveis para um sistema completamente homomórfico: computação em nuvem com privacidade, buscas seguras, segurança de software, re-criptografia de proxys e filtros de emails sigilosos. Após testes serem feitos comparando a CCH com o RSA é possível ver porque a criptografia completamente homomórfica ainda não é usada em larga escala. No entanto, há vários avanços sendo feitos em otimizações do algoritmo.

**Palavras-chave**: Criptografia Completamente Homomórfica, Segurança, Privacidade

# ABSTRACT

The fully homomorphic encryption (**FHE**) allows mathematical functions to be applied to encrypted data. This concept appeared after the creation of the RSA but only after several years Craig Gentry was the first to come up with a solution to this problem. The security of the FHE scheme is based on parts of the algorithm that can be reduced to hard problems. While this problems remain with no efficient solution, the scheme is considered secure. There are several application that are possible with a fully homomorphic encryption scheme: cloud computing with privacy, secure searches, software security, proxy re-encryption and secure email filters. After tests comparing the FHE with the RSA, it's possible to see why the fully homomorphic encryption is still not ready to be fully used. But there are a lot of advances on optimizing the algorithm.

**Keywords**: Fully Homomorphic Encryption, Security, Privacy

# ABREVIATURAS

**RSA** RSA é um sistema de criptografia de chave pública criado por Ron Rivest, Adi Shamir e Leonard Adleman

**CCH** Criptografia Completamente Homomórfica

**BMS** Bit menos significativo

**DES** Data Encryption Standard

**AES** Advanced Encryption Standard

# CAPÍTULO 1

# INTRODUÇÃO

A criptografia está presente em quase todas as aplicações de hoje e com ela é possível garantir que informações sejam trocadas de modo seguro. Os algoritmos de criptografia podem ser classificados em dois grandes grupos, os simétricos e os assimétricos.

A criptografia simétrica é o método onde o remetente e o destinatário usam a mesma chave, exemplos dessa criptografia são o DES e o AES. A criptografia assimétrica por sua vez tem chave pública para criptografar mensagens e uma chave privada para descriptografar. É comum que comunicações que usam criptografia simétrica façam a troca inicial das chaves utilizando algum algoritmo assimétrico. Dos algoritmos assimétricos, o RSA é o mais utilizado e é o algoritmo que deu origem a criptografia homomórfica.

Esta monografia tem como objetivo apresentar o problema da Criptografia Completamente Homomórfica, soluções atuais, possíveis aplicações e limitações. Essa criptografia, apesar de recente, tem a capacidade de inovar o campo da computação em nuvem, pois ela não necessita que os dados sejam descriptografados para que operações sejam aplicadas nos dados originais.

A solução apresentada por Gentry para o problema de Criptografia Completamente Homomórfica [4], atualmente considerada a melhor solução para tal problema, é descrita nesta monografia com relação a criptografia sobre inteiros [7]. Seu método de construção é explicado seguindo os seus passos de contrução originais e padrões de implementação utilizados em [3].

Possíveis aplicações para esse sistema são discutidas, mostrando qual o papel da criptografia completamente homomórfica nelas e qual a viabilidade de implementação delas. Uma seção é dedicada a descrever a segurança deste sistema de criptografia, verificando o que garante que nenhuma informação seja liberada quando operações são aplicadas ao texto criptografado.

Além disso, testes foram realizados utilizando uma implementação deste sistema de criptografia[1] e seus resultados foram comparados com outro sistema de criptografia assimétrico, o RSA.

Para concluir, algumas observações são feitas sobre as diferenças entre essa criptografia e outras criptografias assimétricas, limitações atuais, possibilidades futuras e melhorias necessárias para fazer com que a criptografia completamente homomórfica possa ser usada em larga escala.

O primeiro capítulo introduz o objeto central do estudo desta monografia. Em seguida, o capítulo dois apresenta formalmente o problema da criptografia completamente homomórfica, mostra os passos utilizados por Gentry para construção do sistema e finaliza com uma análise de segurança da criptografia. O capítulo três mostra possíveis aplicações para um sistema completamente homomórfico. O capítulo quatro mostra os testes realizados e seus resultados com relação a velocidade, segurança e funcionalidade do sistema implementado. Finalmente, o capítulo cinco traz uma conclusão sobre a criptografia estudada.

[1] A implementação escolhida é baseada na criptografia com relação a inteiros.

# CAPÍTULO 2

# CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA

Um sistema de criptografia é chamado de homomórfico quando é possível fazer operações matemáticas nos dados criptografados de modo que quando eles forem descriptografados, estas operações foram aplicadas aos dados originais sem que informações sobre o texto original tenham vazado. Um número $x$ é criptografado e uma operação de multiplicação por 2 é feita nos dados criptografados, quando os dados forem descriptografados o resultado será $2 \cdot x$.

Essa noção de criptografia foi formalmente apresentada logo após a invenção do RSA quando foi observado que ele é homomórfico em relação a multiplicação, porém não para adição. Isso levou a inevitável questão de se é possivel criar um sistema que seja homomórfico com relação a adição e multiplicação (Completamente homomórfico) e o que podemos fazer com tal sistema.

## 2.1 Definição

Um sistema de criptografia é completamente homomórfico quando é possível executar adições e multiplicações sobre os dados criptografados, sem limite no número de operações a serem feitas em sequência. Como qualquer função computacional pode ser representada por adições e multiplicações, isso significa que poderiamos executar qualquer função sem saber os dados originais. Várias aplicações seriam possíveis para um sistema como este, tais como computação em nuvem compatível com privacidade, bancos de dados privados, tercerização de processamento de modo seguro, entre outros.

## 2.2 O sistema estudado

Só depois de 30 anos após a invenção do RSA, uma solução plausível foi encontrada para o problema da criptografia completamente homomórfica. Ela foi apresentada por Gentry em sua tese de doutorado [4]. Sua primeira solução é mais teórica e generalizada. A construção descrita e testada nesta monografia é baseada no sistema completamente homomórfico sobre inteiros [7] de Gentry, Halevi, Dijk e Vaikuntanathan que se baseia nos mesmo princípios.

## 2.3 Construção do sistema

O sistema apresentado por Gentry é um sistema de criptografia assimétrico com quatro algoritmos básicos: *KeyGen*, *Encrypt*, *Decrypt* e *Evaluate*. $KeyGen$ gera uma chave pública (disponível para qualquer um) e uma chave privada, o algoritmo *Encrypt* usa a chave pública para criptografar os dados e o algoritmo *Decrypt* usa a chave privada para descriptografar os dados. A diferença desse modelo para um sistema de chave pública não completamente homomórfico está no algoritmo *Evaluate*. Este algoritmo suporta certas funções que podem ser aplicadas nos dados, de modo que para cada função $f$ o algoritmo *Evaluate* recebe um texto que criptografa os dados $(m_1, m_2, ..., m_i)$ e retorna um texto que criptografa $f(m_1, m_2, ..., m_i)$.

Para que o sistema seja completamente homomórfico, a função *Evaluate* deve computar qualquer algoritmo. Um modelo equivalente é a máquina de Turing, que pode simular qualquer algoritmo usando adições e multiplicações[1]. Logo, o algoritmo só precisa computar essas duas funções.

Antes de começar a construção é necessário formalizar algumas restrições para o sistema de criptografia, uma vez que o objetivo desse tipo de criptografia é que o 'processamento' possa ser delegado de modo seguro.

- A complexidade para descriptografar a saída de *Evaluate* deve ser igual a de descriptografar a saída de *Decrypt*;

[1] Multiplicações modulo 2

- As saídas dessas funções devem ter o mesmo tamanho;
- A complexidade para descriptografar deve ser independente da complexidade das funções suportadas por *Evaluate*;
- A função *Evaluate* deve ser eficiente, ou seja, deve depender polinomialmente somente do tamanho da chave e da complexidade das funções suportadas.

Para chegar a um sistema completamente homomórfico, Gentry mostra a construção de um sistema **parcialmente homomórfico**. Neste sistema o algoritmo *Evaluate* pode computar um número limitado de funções, porém não funções muito complicadas. Em seguida esse sistema é extendido para que possa executar qualquer função, ou seja, ele se torna **completamente homomórfico**.

### 2.3.1 Sistema parcialmente homomórfico

Para um parâmetro de segurança $\lambda$, o sistema parcialmente homomórfico funciona da seguinte maneira:

- $KeyGen(\lambda)$: A chave privada $sk$ é um inteiro de $\lambda^2$ bits. Seguindo o modelo de [8], a chave pública $pk$ é uma lista onde cada posição é um inteiro calculado como $c = sk \cdot q + 2r$, onde $q$ é um inteiro aleatório de $\lambda^5$ bits e $2r < sk/2$.
- $Encrypt(pk, m)$: Um bit $m$ é criptografado como $m + 2 \cdot r + 2 \cdot sum$, onde $sum$ é uma soma de um conjunto de textos criptografados da chave pública, e $r$ é um número aleatório tal que $-(2^p) < r < (2^p)$ para $p = \lambda^2$.
- $Decrypt(sk, c)$ é $(c \mod sk) \mod 2$ (Se os textos criptografados da chave pública tiverem baixo ruído, o texto criptografado $c$ vai ter baixo ruído[2], logo a descriptografia funciona).

Além destas três funcões o sistema tem uma outra função chamada *Avaluate*. Essa função recebe um circuito [3] com t entradas e conjunto de textos criptografados $\{c_1, c_2, ..., c_t\}$ e

[2]Ruído é gerado quando um texto é criptografado. Ele se da por $(c \mod sk)$

[3]Representado por operações XOR e AND

retorna o resultado do circuito (Operações de soma e multiplicação). O resultado do circuito descriptografado deve ser igual a saida do circuito com os textos planos como entrada.

Este sistema de criptografia é homomórfico pois é possível somar os textos criptografados como inteiros (Através da função *Avaluate*) e depois descriptografá-los. Isso funciona pois o ruído tem a mesma paridade que o texto original, porém o ruído aumenta a cada operação até que não é retornado o valor certo ao descriptografar. Para que este sistema seja completamente homomórfico deve haver uma função que diminua o ruído o suficiente para que outra operação possa ser executada.

### 2.3.2 De parcialmente para completamente homomórfico

Conceitos sobre o sistema de criptografia atual:

- O ruído é inserido durante a criptografia de um bit (Função *Encrypt*);
- A função *Avaluate* pode fazer adições, subtrações e multiplicações, mas o ruído é aumentado respectivamente.
- O sistema consegue descriptografar até uma certa quantidade de ruído, após isso as respostas não são mais confiáveis.
- Inversamente à função *Encrypt*, *Decrypt* retira o ruído.

De acordo com [5], para que o sistema possa ser completamente homomórfico, a própria função *Decrypt* pode ser usada para diminuir o ruído. Então o sistema tem que computar sua própria função de descriptografar na função *Evaluate*, ou seja, além de somar, subtrair e multiplicar, é necessário suportar a função de descriptografar. A função *Decrypt* removeria o ruído de uma criptografia, mas o texto plano estaria exposto. Logo, é necessário descriptografar o texto encriptografado por $pk_1$ enquanto ele estiver criptografado por $pk_2$. Para isso segue a seguinte função:

$Recrypt(pk_i, D, sk'_{i-1}, c_{i-1})$:

- $c'_{i-1} = Encrypt(pk_i, c_{i-1,j})$ para $j = 0$ até $j = N$ onde $N =$ numero de bits de $c_{i-1}$
- Return $c = Evaluate(pk_i, D, sk'_{i-1}, c'_{i-1})$

Onde: $pk_i$ é uma das chaves públicas, $sk'_{i-1}$ é um vetor em que cada posição é um bit de $sk_{i-1}$ criptografado usando $pk_i$, $D$ é o circuito que computa a função $Decrypt$, e $c_{i-1}$ é a criptografia de um bit m usando $pk_{i-1}$.

A função $Recrypt$ primeiramente criptografa cada um dos bits de $c_{i-1}$, gerando assim um vetor com os bits de $c_{i-1}$ criptografados. Esse vetor, em conjunto com o vetor $sk'_{i-1}$ são usados como entrada para o circuito $D$, que é executado dentro da função $Evaluate$. Como a função $Decrypt$ está representada por um circuito binário ($D$), e como o sistema consegue executar o circuito $D$ sem que o ruído passe do limite, o resultado $c$ de $Recrypt$ é a criptografia de $Decrypt(sk_i - 1, c_{i-1})$ usando $pk_i$.

O bit $m$ criptografado primeiramente por $pk_{i-1}$, agora está criptografado por $pk_i$. Isso é possível pois após $Encrypt$ o texto está duplamente criptografado, e a função $Evaluate$ remove a criptografia mais interna (da chave $pk_{i-1}$). É possivel notar que a função $Evaluate$ remove o ruído da criptografia por $pk_{i-1}$ por causa do circuito sendo usado, mas ao mesmo tempo introduz um novo ruído quando avalia a criptografia por $pk_i$. Desde que o novo ruído inserido seja menor que o ruído removido, podemos continuar aplicando esse processo sem que o ruído passe do limite.

Somente isso não torna o sistema completamente homomórfico, a função $Recrypt$ somente muda da criptografia de uma chave $pk_k$ para $pk_{k+1}$. Como o objetivo é poder fazer operações nos dados sem que o ruído passe do limite, podemos criar um novo circuito que é igual a $D$, mas com uma operação a mais. Logo, o circuito usado faz a descriptografia e uma operação (Adição ou multiplicação), fazendo com que o sistema seja completamente homomórfico.

A chave pública do sistema consiste de um conjunto de chaves públicas $(pk_1, pk_2, ..., pk_{n+1})$ e a chave privada consiste de um conjunto de chaves privadas criptografas $(sk_1, sk_2, ..., sk_n)$, onde $sk_i$ é criptografada por $pk_{i+1}$.

### 2.3.3 Simplificando a descriptografia

Como foi mostrado na seção anterior, se o sistema conseguir executar sua própria função de descriptografar como um circuito binário, então o sistema pode ser transformado em completamente homomórfico. Mas o sistema parcialmente descrito ainda não tem essa característica, então ele foi alterado para um sistema que tenha a descriptografia simples o suficiente para poder ser executada.

A função de descriptografia do sistema parcialmente homomórfico é:

$$m = (c \mod p) \mod 2$$

Que é equivalente à:

$$m = BMS(c)\ XOR\ BMS(APROX(c/p))$$

Onde BMS retorna o bit menos significativo e APROX arredonda para o inteiro mais próximo.

As funções $BMS$ e $XOR$ podem ser executadas com apenas uma operação lógica, a complexidade está em $APROX(c/p)$. Como $c$ e $p$ podem ser números longos, o ruído gerado pode ser maior do que o sistema pode aguentar. O sistema pode executar polinômios de grau menor que P, mas como $c$ e $p$ tem $P$ bits cada um, e como uma multiplicação[4] geraria um polinômio de aproximadamente grau $P$, o ruído pode passar do limite.

Com o objetivo de tornar a função de descriptografar mais simples, a sua entrada deve ser pré-processada. Parte da complexidade é jogada para a função de criptografar que agora faz um pós processamento, deixando assim menos trabalho para a função de descriptografia. Para isso a função de gerar chave agora incluí uma dica do inteiro $p$. Essa dica é usada durante a criptografia para que a descriptografia fique mais simples. Além das funções de gerar chave e de criptografia serem alteradas, a função de descriptografar que multiplica dois números grandes é substituída agora por uma que soma um conjunto relativamente pequeno de números (Resultado da função de criptografia usando a dica de p).

As seguintes alterações foram feitas as funções:

[4] A divisão no caso é representada como uma multiplicação $c \cdot 1/p$.

- Key-Gen: A dica do inteiro $p$ é inserida na chave pública na forma de um vetor $y = \langle y_1, y_2, ..., y_\beta \rangle$ de modo que haja um subconjunto $S \subset \{1, ..., \beta\}$ de tamanho $\alpha$ com somatório igual a $1/p$. A chave secreta gerada $sk^*$ é um vetor esparço $s \in \{0, 1\}$ com peso Hamming $\alpha$. Essa dica é usada pelas funções *Encrypt* e *Decrypt*.

- *Encrypt*: Após gerar o texto criptografado c, *Encrypt* gera um novo vetor $z = \langle z_1, z_2, ..., z_\beta \rangle$ onde $z_i \leftarrow c \cdot y_i$. Esse vetor é o pre-processamento para deixar menos trabalho ao algoritmo *Decrypt*.

- *Decrypt*: retorna BMS(c) XOR BMS(APROX($\sum_i s_i z_i$)) onde $s_i$ é um elemento de $s$ (Dica na chave privada). Descriptografia funciona pois:

$$\sum_i s_i z_i = \sum_i c \cdot s_i y_i = c/p \bmod 2$$

Com essas alterações a complexidade da função de descriptografar fica menor o suficiente para ser computada pelo sistema dentro da função *Evaluate*. O ruído gerado pela criptografia é removido pela função *Evaluate* ao mesmo tempo que um novo ruído é inserido. Se o ruído final for pequeno o suficiente é possível computar a função de descriptografia e uma operação lógica, desse modo o sistema se torna completamente homomórfico.

## 2.4 Limitações atuais

Grande parte da limitação atual está na velocidade do sistema e no tamanho dos textos criptografados. A criptografia funciona gerando um inteiro para cada bit do texto original, fazendo com que os textos criptografados sejam muito grandes, além das própias funções de criptografar, descriptografar e executar funções também serem lentas para os padrões de hoje.

## 2.5 Segurança

De acordo com Gentry, para que uma criptografia seja considerada segura ela deve atender dois critérios: *one-wayness* e *segurança semântica* [9].

### 2.5.1 Padrões de segurança

*One-wayness* significa que, dado uma chave pública $pk$ e uma mensagem criptografada $c$ por $pk$, deve ser difícil para um adversário gerar o texto plano correspondente. Um algoritmo que rode em tempo polinomial $\lambda$ deve ter uma probabilidade de sucesso menor que $1/\lambda^k$ para uma qualquer constante $k$.

Para que um sistema seja *semanticamente seguro*, dado um texto criptografado $c$ que criptografa o texto plano $m_0$ ou $m_1$, o adversário deve ter 50% de chance de acerto. Se ele tiver mais, significa que há algum processo além de pura probabilidade. Por isso um sistema determinístico[5] não é considerado semanticamente seguro, pois o adversário pode gerar as criptografias dos textos $m_0$ e $m_1$ e comparar com $c$. Isso nos diz que um sistema seguro deve ser probabilístico.

### 2.5.2 Prova de segurança semântica

Apesar do sistema de criptografia descrito ser "maleável"devido à função *Evaluate* que muda a criptografia de uma chave para outra, o sistema continua sendo *semanticamente seguro*. Gentry mostra isso comparando a função de criptografia com o problema de encontrar o *maior divisor comum* (MDC). Euclides mostra que achar o MDC é fácil, porém na função de criptografia o que temos é MDC aproximado.

$$x_1 = s_1 + p \cdot q_1$$

Com $s_i$ muito menor que p e com um parâmetro de segurança $\lambda$, esse problema não pode ser resolvido polinomialmente com relação ao comprimento de bits de $x_1$.

Além disso, Gentry mostra que esse problema de MDC aproximado pode ser reduzido ao problema de segurança semântica do sistema. Desse modo a segurança semântica só pode ser quebrada se o problema de MDC aproximado for possível em tempo polinomial.

[5]Onde há apenas um texto criptografado para cada texto plano.

### 2.5.3 Segurança com relação à dica dentro chaves

Como visto previamente, para que o sistema seja completamente homomórfico uma dica da chave secreta é colocada na chave pública. Essa maleabilidade pode dar a impressão que o sistema deixaria de ser seguro, no entanto Gentry prova que o sistema continua seguro desde que:

- O sistema parcialmente homomórfico que da origem ao sistema completamente homomórfico seja semanticamente seguro.
- Dado um conjunto de números $y$ e outro número $s$, seja difícil achar um subconjunto esparço de $y$ com somatório igual a $s$.

O último representa o problema de descobrir $p$ a partir da dica. Esse problema é chamado de Problema da Soma de Subconjunto Esparço (PSSE) e já foi estudado com relação a sistema de criptografias com servidores. Se o tamanho do vetor e seu peso Hamming forem escolhidos corretamente, ele não pode ser resolvido em tempo polinomial.

# CAPÍTULO 3

# APLICAÇÕES

A maior parte das aplicações para um sistema de criptografia completamente homomórfico está na alteração de aplicações já existentes, de modo que elas possam ser utlizadas de modo completamente seguro. Alguns exemplos são:

## 3.1 Computação em nuvem

Atualmente o uso da 'nuvem' está ficando cada vez mais comum. Porém os dados, por estarem em algum servidor público, não estão completamente seguros e eventualmente a segurança desses servidores pode ser comprometida por meio de ataques ao servidor ou outra atividade ilícita. Hoje o que se pode fazer para ter mais segurança é manter os nossos dados criptografados na nuvem. Mas se quisermos utilizar esses dados, eles devem ser baixados e manipulados localmente, indo de encontro com o propósito da computação em nuvem.

Com a criptografia completamente homomórfica é possível fazer buscar e alterações sobre dados criptografados, ou seja, é possível buscar arquivos por palavra-chave mesmo que o servidor não tenha conhecimento dos dados do arquivos. Deste pode realmente é possível delegar processamento de modo seguro.

## 3.2 Buscas seguras

Também é possível fazer buscar criptografadas em "search-engines". Os argumentos da busca podem ser criptografados pela CCH e usados no servidor em um circuito que representa a função de busca do servidor. As funções desse circuito podem fazer as comparações com os dados criptografados sem nunca saber seus valores reais. O resultado do circuito é um texto criptografado, que pode ser descriptografado pela pessoa que criptografou os

parametros da busca. Desse modo a busca é completamente sigilosa.

## 3.3 Segurança de software

Uma possível aplicação é a segurança de software, onde os valores de um programa permancem criptografados na memória. Cada operação do programa sobre os dados pode ser representada por um circuito de NANDs e aplicada aos dados e os dados só precisam ser descriptografados na saída do programa.

## 3.4 Re-criptografia de Proxys

Utilizando uma propriedade da criptografia completamente homomórfica é possivel fazer com que um texto criptografado por uma chave pública $a$, possa ser transformado em um texto criptografado por $b$, sem que a chave privada de $a$ ou o textos original sejam comprometidos. O proxy dono da chave pública $a$ pode enviar uma mensagem para um outro proxy com chave pública $b$ de modo seguro.

Sistemas que permitem esse comportamento já existem porém eles podem não ser bilaterais, ou seja, conseguem mudar a criptografia de $a$ para $b$ mas não de $b$ para $a$. Ou tem limitações sobre o número de vezes que uma mensagem pode ser recriptografada. A CCH não tem tais limitações.

## 3.5 Filtros de emails sigilosos

Em um sistema onde os emails são criptografados para que apenas o usuario tenha acesso os dados, usando a CCH é possível executar filtros sobre os emails com relação a palavras dentro do email, destinatário, etc, sem que o servidor tenha acesso direto aos dados.

# CAPÍTULO 4

# TESTES E RESULTADOS

## 4.1 Implementação do algoritmo

A implementação usada para testes com criptografia completamente homomórfica foi a biblioteca FHEW[1] criada por Leo Ducas e Daniele Micciancio baseada em [6].

A biblioteca ofereçe funções para:

- Criar chave secreta e chave pública
- Criptografar bit
- Descriptografar bit criptografado
- Executar um NAND homomórfico em dados criptografados

## 4.2 Arquitetura usada

A arquitetura do computador usado para os testes é a seguinte:

- **Processador**: Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz
- **Memória RAM**: 8,00 GB (7,88 GB utilizáveis)
- **Sistema Operacional**: Debian 7 (64 bits)

## 4.3 Algoritmo usado para comparação

O algoritmo RSA foi usado como comparação pois ele é usado em larga escala atualmente, além de ser a origem da CCH. A implementação utilizada foi a biblioteca Crypto++[2] por ser próxima a um RSA puro (sem muitas otimizações) e por ser na mesma linguagem de

[1]Pode ser encontrada em https://github.com/lducas/FHEW
[2]Pode ser encontrada em www.http://cryptopp.com/

programação que a implementação do CCH. Desse modo a compilação pode ser feita com o mesmo compilador e nível automático de otimização (-O1).

## 4.4 Metodología

É possivel dividir a comparação em três etapas: velocidade, segurança e funcionalidades.

Para comparar a velocidade dos dois tipos de criptografia os testes foram realizados em um conjunto de 100 strings de 80 bits cada (10 caracteres). Cada string é criptografada usando RSA e CCH, e em seguida ela é descriptografada. O tempo médio das operações do RSA para as 100 strings é calculado e comparado com as do CCH.

Para comparar a segurança podemos olhar para estratégias comuns de ataques e verificar a probabilidade de tal ataque ser bem sucedido para cada criptografia.

Para comparar a funcionalidade olhamos para o que a criptografia pode oferecer, e baseado nas comparações anteriores quais são os pontos fortes e pontos fracos de cada uma.

## 4.5 Velocidade

É possivel ver que o sistema atual de criptografia completamente homomórfica é muito mais lento do que o RSA. Os tempos obtidos nos testes são podem ser vistos na tabela 4.1.

| Sistema | Criptografia | Descriptografia | Operação NAND |
|---|---|---|---|
| RSA | 21 Kb/s | 21 Kb/s | Não disponível |
| CCH | 214 b/s | 188 b/s | 0,46 b/s |

Tabela 4.1: Média dos tempos dos testes

Os tempos para criptografar e descriptografar são muito mais lentos no CCH, porém o que faz com que a CCH não possa ser usada em larga escala é o tempo para fazer operações homomórficas. É possivel ver avanços que estão sendo feitos para tornar essa criptografia mais utilizável. Várias possíveis otimizações são descritas em [10] e [2] que mostram como diminuir o tempo para criptografar e da função *Recrypt* (Para deixar

as operações mais rápidas). Também é possível notar que o tamanho da chave pública nesta implementação tem aproximadamente 12GB, mas uma implementação do algoritmo encontrada no Github sugere a possibilidade de usar uma chave comprimida com 2MB [1].

## 4.6 Segurança

Tanto a segurança do RSA quanto da CCH é baseada em problemas matemáticos e no fato de eles serem difíceis (sem solução eficiente). A CCH tem sua segurança baseada no problema de MDC aproximado e no PSSE, que já foram mencionados previamente. O RSA também tem sua segurança baseada em dois problemas, o de fatoração de números grandes[3] e o problema RSA[4].

Comparar a complexidade destes problemas não está no escopo desta monografia, mas é possível dizer que os dois sistemas de criptografia são seguros.

## 4.7 Funcionalidade

O RSA é usado na maioria das aplicações que usam criptografia assimétrica, além de possibilitar assinaturas digitais e é considerado um dos algorimos assimétricos mais seguros já construídos. Mas na questão de funcionalidade, apesar do RSA ser muito presente hoje em dia, a CCH é muito mais versátil e torna muitas aplicações possíveis pois ele mantém a segurança do RSA mas o extende para ter total sigilo[5] sobre os dados criptografados. Obviamente cada criptigrafia tem um uso diferente mas é possível ver que a CCH tem a capacidade de revolucionar o campo da criptografia, assim como o RSA fez quando foi criado.

[3]Fatoração de números grandes é a decomposição de um inteiro composto grande em um produto de números menores

[4]O problema RSA pode ser definido como encontrar a $e$-raiz de um número aleatório

[5]Por total sigilo se quer dizer que nem quem faz operações nos dados consegue vê-los

# CAPÍTULO 5

# CONCLUSÃO

O sistema de criptografia completamente homomórfico descrito por Gentry resolve o problema de delegar processamento com privacidade, sem que a segurança seja comprometida. Mesmo que o método de construção seja um tanto extenso, o algoritmo final tem a implementação relativamente simples, apesar de que tende a ficar mais complexo a medida que otimizações são adicionadas.

Esse novo paradigma de criptografia faz com que várias aplicações sejam possíveis e possibilita que aplicações já existentes sejam extendidas para que haja confidencialidade e privacidade (como a computação em nuvem).

A solução atual ainda tem muito a ser melhorada para que possa ser usada em larga escala, devido as limitações de seu tempo de execução e tamanho necessário das chaves para garantir um modelo seguro. Apesar disso, há avanços sendo feitos, vários outros sistemas de criptografias semelhantes já foram criados baseados no modelo de Gentry. Esses novos sistemas trazem novas otimizações e estão cada vez mais próximo de um modelo viável de criptografia completamente homomórfica.

# BIBLIOGRAFIA

[1] J.S. Coron. An implementation of the dghv fully homomorphic scheme. https://github.com/coron/fhe.

[2] Shai Halevi Craig Gentry. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits, 2011. Disponível em: https://eprint.iacr.org/2011/279.

[3] Shai Halevi Craig Gentry. Implementing gentry's fully-homomorphic encryption scheme, 2011. Disponível em: https://eprint.iacr.org/2010/520.pdf.

[4] Craig Gentry. A fully homomorphic encryption scheme. Disponível em: https://crypto.stanford.edu/craig/craig-thesis.pdf.

[5] Craig Gentry. Computing arbitrary functions of encrypted data, 2010. Disponível em: https://crypto.stanford.edu/craig/easy-fhe.pdf.

[6] Daniele Micciancio Léo Ducas. Bootstrapping homomorphic encryption in less than a second, 2014. Disponível em: http://eprint.iacr.org/2014/816.

[7] Shai Halevi Vinod Vaikuntanathan Marten van Dijk, Craig Gentry. Fully homomorphic encryption over the integers, 2010. Disponível em: https://eprint.iacr.org/2009/616.pdf.

[8] Ron Rothblum. Homomorphic encryption: from private-key to public-key, 2010. Disponível em: http://eccc.hpi-web.de/report/2010/146/.

[9] S. Micali S. Goldwasser. Probabilistic encryption, 1984. J. Comput. Syst. Sci., 28(2):270–299.

[10] Craig Gentry Zvika Brakerski. Fully homomorphic encryption without bootstrapping, 2011. Disponível em: https://eprint.iacr.org/2011/277.