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

[resumo]

Palavras-chave:

ABSTRACT

[abstract]

Keywords:

CAPÍTULO 1

INTRODUÇÃO

Esta monografia tem como objetivo apresentar o problema da Criptografia Completamente Homomórfica com suas 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 [2], atualmente considerada a melhor solução para tal problema, é descrita com relação a criptografia sobre inteiros [4] e seu método de construção é explicado seguindo os seus passos de contrução originais e padrões de implementação utilizados em [1].

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 verificar a segurança deste sistema de criptografia, verificando o que garante que nenhuma informação seja liberada quando operações são aplicadas ao texto criptografado uma vez que uma dica sobre a chave privada é inserida na chave pública.

Além disso, testes foram realizados utilizando uma implementação deste sistema de criptografia [1] e seus resultados foram comparados com outros sistemas de criptografia conhecidos, como RSA.

Concluímos com algumas observações 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.

[1] A implementação escolhida é baseada na criptografia com relação a inteiros, que é explicada em [**?**]

CAPÍTULO 2

CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA

2.1 Origens

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

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.

2.2 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 descritos na seção seguinte.

2.3 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 primeira solução é mais teórica e generalizada então a construção explicada e testada nesta monografia é baseada no sistema completamente homomórfico sobre inteiros de Gentry, Halevi, Dijk e Vaikuntanathan que se baseia nos mesmo princípios.

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 nossos 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 podemos 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 arqumentos 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 transaformado em um texto criptografado por b, sem que a chave privada de a ou o textos original sejam comprometidos. Ou seja, 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.

3.6 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.

CAPÍTULO 4

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

Antes de começar a construção precisamos 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*;
- As saídas dessas funções devem ter o mesmo tamanho;
- A complexidade para descriptografar deve ser independente da complexidade das funções suportadas por *Evaluate*;
- E a função *Evaluate* deve ser eficiente, ou seja, deve depender polinomialmente

[1] Multiplicações modulo 2

somente do tamanho da chave e da complexidade das funções suportadas (A complexidade das funções pode ser medida pelo tempo requerido em uma máquina de turing ou, análogamente, pelo tamanho de um circuito boleano necessário para computar a função).

Para chegar a um sistema completamente homomórfico, Gentry mostra a construção de um sistema **parcialmente homomórfico**, onde 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**.

4.1 Sistema parcialmente homomórfico

Para um parâmetro de segurança λ, o sistema parcialmente homomórfico funciona da seguinte maneira:

- $KeyGen(\lambda)$: A chave privada sk é um inteiro de λ^2 bits. Seguindo o modelo de [**?**], a chave pública pk é uma lista de inteiros que são criptografias de zero[2]. Os textos da chave pública são criptografados como $c = sk \cdot q + 2r$, onde q é um inteiro aleatório de λ^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 simplismente $(c \mod sk) \mod 2$ (Se os textos criptografados da chave pública tiverem baixo ruído, o texto criptografado c vai ter baixo ruído[3], logo a descriptografia funciona).

Além destas três funcões nosso sistema tem uma outra função chamada *Avaluate*. Essa função recebe um circuito [4] com t entradas e conjunto de textos criptografados $\{c_1, c_2, ..., c_t\}$

[2] A lista tem tamanho polinomial em λ.
[3] Ruído é gerado quando um texto é criptografado. Ele se da por $(c \mod sk)$
[4] Representado por operações XOR e AND

e retorna o resultado do circuito (Operações de soma, subtração 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 podemos somar os textos criptografados como inteiros (Através da função Avaluate) e depois descriptografa-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 temos que ter uma função que diminua o ruído até que outra operação possa ser executada.

4.2 De parcialmente para completamente homomórfico

Antes de continuarmos, vamos definir alguns conceitos sobre nosso 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 disso as respostas não são mais confiáveis.
- Inversamente à função *Encrypt*, *Decrypt* retira o ruído.

De acordo com [3] Para que nosso sistema possa ser completamente homomórfico, a própria função *Decrypt* é usada para diminuir o ruído. Então o sistema deve que ser capaz de 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, temos que descriptografar o texto encriptografado por pk_1 enquanto ele estiver criptografado por pk_2. Para isso temos 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 representado 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.

Ou seja, 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 (A por 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 limite 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}.

4.3 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 caracteristica, 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 integer 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[5] 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 pre-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

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

relativamente pequeno de números (Resultado da função de criptografia usando a dica de p).

As seguintes alterações foram feitas as funções:

- 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 α com somatório igual a $1/p$. A chave secreta gerada sk^* é um vetor esparço $s \in \{0, 1\}$ com peso Hamming α. 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:

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.4 Como as operações funcionam

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

CAPÍTULO 5

SEGURANÇA

- probabilidade de 'adivinhar' o plain text e a chave privada - mostrar calculo de probabilidade

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.1 Ataques conhecidos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation

ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

CAPÍTULO 6

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 no paper Bootstrapping Homomorphic Encryption in less than a second.

6.1 O que está implementado

A biblioteca ofereçe funções para:

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

6.2 Diferenças entre a implementação e o sistema de Gentry

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation

[1] Pode ser encontrada em https://github.com/lducas/FHEW

ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

CAPÍTULO 7

TESTES E RESULTADOS

7.1 Arquitetura usada

A arquitetura do computador usada 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)

7.2 Algoritmo usado para comparação

O algoritmo RSA foi usado como comparação com o sistema completamente homomórfico pois ele é usado em larga escala atualmente, além de ser a origem da CCH. A implementação utilizada foi a [1] por ser próxima a um RSA puro (sem otimizações) e por ser na mesma linguagem de programação que a implementação do CCH. Desse modo a compilação pode ser feita com o mesmo compilador e nível automático de otimização (-O1).

7.3 Metodología

Podemos 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. Ainda na seção de velocidade olhamos para o tempo geração e o tamanho das chaves geradas pelos sistemas.

[1] Pode ser encontrada em ?

Para comparar a segurança podemos olhar para estratégias comuns de ataques e calcular 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.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

7.4 Comparações e Resultados

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 *.

Sistema	Criptografia	Descriptografia	Operação NAND
RSA	21 Kb/s	21 Kb/s	-
CCH	214 b/s	188 b/s	?

Tabela 7.1: Média dos tempos dos testes

Outra diferença é o tamanho das chaves geradas onde o RSA gera chaves com tamanhos X e Y para chave pública e secreta respectivamente e o CCH gera chaves com tamanhos Z e W para chave pública e secreta respectivamente. O tempo para criptografar e descriptografar é o que faz com que a CCH não possa ser usada em larga escala, mas é possivel ver avanços que estão sendo feitos para tornar essa criptografia mais utilizável. Um exemplo é * que mostra um modo de gerar uma chave comprimida (De 2GB para 20Mb?), enquanto * mostra várias otimizações que tendem a diminuir o tempo para criptografar e descriptografar.

Com relação a segurança destes dois sistemas é possível dizer que nos dois casos ela se baseia no problema NP-Complexo de máximo divisor comum, Lorem ipsum dolor sit

amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

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[2] 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.

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

CAPÍTULO 8

CONCLUSÃO

O sistema de criptografia copletamente homomórfico descrito por Gentry resolve o problema de delegar processamento com privacidade, sem que a segurança seja comprometida. Apesar do método de construção ser 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. Mas podemos há avanços sendo feitos e baseado neste modelo vários outros sistema de criptografias parecidos já foram criados e publicados, cada vez mais próximo de um modelo que possa realmente ser usado.

BIBLIOGRAFIA

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

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

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

[4] 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.

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