UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA

GIANCARLO KLEMM CAMILO

Análise do Sistema de Criptografia Completamente Homomórfico de Gentry

Prof. Luiz Carlos Pessoa Albini, Ph.D.
Orientador

Curitiba, Dezembro de 2015

Análise do Sistema de Criptografia Completamente Homomórfico de Gentry

Giancarlo Klemm Camilo

Projeto Final de Curso submetido ao Departamento de Ciência da Computação da Universidade Federal do Paraná como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Giancarlo Klemm Camilo

Aprovado por:

Prof. Luiz Carlos Pessoa Albini, Ph.D.

Prof. Nome do participante banca 1, D.Sc.

Prof. Nome do participante banca 2, Ph.D.

Prof. Nome do participante banca 3, Ph.D.

CURITIBA, PR - BRASIL

Dezembro de 2015

Agradecimentos

Agradeço à...

RESUMO

Análise do Sistema de Criptografia Completamente Homomórfico de Gentry

Giancarlo Klemm Camilo

Dezembro/2015

Orientador: Luiz Carlos Pessoa Albini, Ph.D.

Resumo...

ABSTRACT

Análise do Sistema de Criptografia Completamente Homomórfico de Gentry

Giancarlo Klemm Camilo

Dezembro/2015

Advisor: Luiz Carlos Pessoa Albini, Ph.D.

Abstract...

Lista de Abreviaturas e Siglas

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

Sumário

Capítulo 1

Criptografia Completamente Homomórfica

1.1 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 de computação em nuvem, uma vez que ela torna possível a confidencialidade neste sistema. A solução apresentada por Gentry para o problema de Criptografia Completamente Homomórfica, atualmente considerado a melhor solução para tal problema, é descrita e seu método de construção é explicado.

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. 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.

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

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.

1.2 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, 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 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. 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 Possíveis Aplicações.

Porém, só depois de 30 anos após a invenção do RSA uma solução plausível foi encontrada. Ela foi apresentada por Gentry em * e sua construção é explicada em detalhes na próxima seção.

1.3 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, o algoritmo *Evaluate* deve computar qualquer algoritmo de computador. Um modelo equivalente é a máquina de Turing, que pode simular qualquer algoritmo usando adições, subtrações e multiplicações[2]. Logo, nosso algoritmo só precisa computar essas três 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 '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 funcções devem ter o mesmo tamanho, a complexidade para descriptografar deve ser independente da complexidade das funções suportadas por *Evaluate*. Além disso a função *Evaluate* deve ser eficiente, ou seja, deve depender polinomialmente somente do tamanho da chave e da complexidade das funções suportadas (A complexidade das funções pode ser medida pelo tempo ela requere 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, de modo que 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 é completamente homomórfico.

[2]Multiplicações modulo 2

1.3.1 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. A chave pública pk é uma lista de inteiros que são criptografias de zero[3]. Os textos da chave pública são criptografados como $c = sk * q + 2r$, onde q é um inteiro aleatório de λ^5 bits e $2r < sk/2$.

- $Encrypt(pk, m)$: Um bit m é criptografado como $m + 2 * r + 2 * sum$, 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[4], logo a descriptografia funciona).

Além destas três funcões nosso sistema tem uma outra função chamada *Avaluate*. Essa função recebe um circuito [5] com t entradas e conjunto de textos criptografados $\{ c_1, c_2, ..., c_t \}$ 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.

[3] A lista tem tamanho polinomial em λ

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

[5] Representado por operações XOR e AND

1.3.2 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.

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}.

1.3.3 Alterações do no sistema parcialmente homomórfico

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.

1.3.4 Segurança

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

1.4 Aplicações

Possível aplicações para um sistema completamente homomórfico... Could Computing Verifiable computing Secure multi-party computation Homomorphic signatures for network coding

1.5 Implementação do algoritmo

Um implementação do do sistema descrito por Gentry em 2008 foi encontrada no GitHub e...

1.6 Testes e Comparações

Utilizando o a implementação explicada na seção anterior os seguintes testes foram feitos...

1.7 Conclusões

Nunc vitae tincidunt urna. Quisque non rhoncus ligula, eget mattis magna. Proin eget dictum mi. Nulla facilisi. Fusce fermentum tincidunt libero sed aliquam. Fusce in lectus erat. Maecenas blandit, ante ac euismod auctor, turpis purus dictum ligula, vel placerat libero massa ut erat. Donec tristique tincidunt purus vel egestas.