Um estudo sobre esquemas totalmente homomórficos sobre números inteiros

Guilherme Rodrigues Bilar, *Universidade Federal de São Carlos - UFSCar*

Abstract — **This article has the objective of presenting a general state-of-art study about the main fully homomorphic encryption systems, with a focus on system based in integer algebra, specially the DGHV scheme and the improvements proposed by Coron. We made a mapping and a in deep study about the main fully homomorphic cryptosystems, from the original thesis by Gentry, in 2009, to more recent studies, like the schemes based on the learning with errors problem, and the key reduction techniques proposed by Coron.**

Index Terms — **Homomorphism, cryptography, DGHV**

Resumo — **Este artigo tem como objetivo apresentar um estudo generalizado dos principais esquemas totalmente homomórficos existentes, tendo como foco os esquemas totalmente homomórficos baseados na álgebra de números inteiros, mais especificamente o esquema DGHV e as melhorias propostas por Coron. Foi realizado um mapeamento e um estudo profundo principais esquemas criptográficos totalmente homomórficos, partindo da tese original de Gentry em 2009, até estudos mais recentes, como os esquemas baseados no problema de aprendizagem de erros, e as técnicas de redução de chave pública propostas por Coron.**

Palavras-chave — **Homomorfismo, criptografia, DGHV**

I. INTRODUÇÃO

A criptografia moderna tem como base problemas matemáticos relacionados a teoria de números, exemplo disto é o algoritmo RSA que usa a fatoração de números inteiros em números primos para a geração de chaves assimétricas [1], outro exemplo disso é o uso de logaritmos discretos na criptografia de curvas elípticas [2], entretanto, com o surgimento da teoria dos computadores quântico e do algoritmo de Shor, esse tipo de solução torna-se vulnerável a ataques quânticos [3]. Para proteger dados, até mesmo da criptoanálise quântica houve o surgimento de uma nova área dentro da segurança da informação, chamada de Criptografia Pós-Quântica, que estuda a criação de algoritmos criptográficos resistentes a ataques quânticos, tendo como base outros problemas matemático-computacionais, como, por exemplo, algoritmos baseados em hash, baseados em reticulados, baseados em códigos, entre outros. O tempo de ataque para todos esses algoritmos é exponencial, mesmo para um computador quântico [4].

Entre características oferecidas pela a criptografia pós-quântica destaca-se o homomorfismo, onde um esquema totalmente homomórfico oferece a capacidade de manipularmos informação cifrada, de forma a realizar operações com as mesmas, tanto soma quanto multiplicação, sendo possível manter a integridade dos dados e garantir sua confidencialidade. O primeiro esquema totalmente homomórfico foi apresentado por Gentry em 2009, utilizando de reticulados, este que suportava uma quantidade ilimitada de somas e multiplicações. A limitação quebrada foi a de que, a cada operação o ruído presente nas cifras se acumulava até que o processo de decriptação não conseguia mais retornar o dado original. Gentry então demostrou uma técnica para se reduzir o processo de decrypt de dados, sendo possível representá-lo como um polinômio de baixo grau nos bits do texto cifrado e da chave privada, dessa forma, permitindo que o esquema pudesse avaliar o seu próprio circuito de decriptação de forma homomórfica, e, como resultado, diminuindo o ruído apresentado no dado cifrado. Com isso, o esquema de Gentry pode executar computações arbitrarias nos dados cifrados, e, abriu caminho para que suas ideias fossem aplicadas em outros esquemas parcialmente homomórficos com o intuito de produzir esquemas FHE (do inglês: Fully Homomorphic Encrypt).

II. OS ESQUEMAS TOTALMENTE HOMOMÓRFICOS

Por definição um esquema criptográfico é dito totalmente homomórfico quando este permite executar de forma implícita a adição e multiplicação de dados, manipulando apenas os textos cifrados. [5]

Atualmente os esquemas totalmente homomórficos são baseados em três classes de problemas matemáticos: reticulados ideais, como fora proposto inicialmente por Gentry, anéis de números inteiros, proposto por Dijk, Gentry, Halevi e Vaikuntanathan, e, LWE (do inglês: Learning With Erros), proposto pro Brakerski e Vaikuntanathan.

Gentry em seu trabalho original faz uso de reticulados ideias para criar um esquema parcialmente homomórfico, capaz de executar um número limitado de operações no dado cifrado. Posteriormente nesse trabalho, Gentry demonstrou o procedimento de bootstraping, no qual o esquema parcialmente homomórfico pode ser transformado em um esquema totalmente homomórfico através da operação de reencriptação do dado cifrado [6]. Este esquema foi posteriormente implementado por Gentry e Halevi [10], seguido de pequenas alterações e melhorias. O esse trabalho inicial tornou-se referência e um ponto de partida para outras pesquisas e para a criação de outros esquemas FHE.

Após o esquema utilizando reticulados de Gentry, foi proposto o esquema DGHV, em 2010, que, segue a mesma linha de pensamento do trabalho original de Gentry, mas, utilizando um esquema mais simples, que utiliza apenas aritmética de números inteiros para a construção do esquema parcialmente homomórfico que serve como base deste. Sobre

esses esquema temos melhorias propostas por Coron [7][8]; que tornam o custo computacional do esquema menor, e que diminuem o tamanho proibitivo da chave pública do esquema original, que era por volta de λ^10 bits, onde lambda é o fator de segurança da chave.

Brakerski e Vaikuntanathan desenvolveram um esquema baseado no problema de aprendizagem com erros (sigla em inglês: LWE) e aprendizagem com erro em anel (sigla em inglês: RLEW), os mesmos introduziram no meio cientifico uma nova técnica de redução de dimensão e uma nova técnica de troca de módulo. O grande diferencial desse esquema é que o ruído no texto cifrado cresce de forma quasi-linear, e, o bootstraping, essencial nos outros sistemas, passa a não parte necessária para o funcionamento do sistema, mas sim uma otimização.

O uso de reticulados na implementação de um esquema totalmente homomórfico não se mostrou uma das melhores alternativas, de acordo com Coron [7] e de Gentry [9], pois as chaves públicas geradas tinham tamanho estimado na ordem de $\tilde{O}(\lambda^{10})$ Na implementação de Coron [7], na qual foram utilizados números inteiros na geração de chaves, obteve-se uma redução para um tamanho estimado de $\tilde{O}(\lambda^{7})$ da chave pública utilizando um diminuição quadrática dos elementos da chave. Em outro trabalho, mais uma vez por Coron [8] foi possível obter chaves públicas com tamanho estimado de $\tilde{O}(\lambda^{5})$, através de uma técnica onde os elementos da chave são gerados on-the-fly por meio de um RNG (do inglês: Random Number Generator), e, apenas correções desses números são armazenados.

A figura I ilustra os principais tipos de esquemas totalmente homomórficos existentes, o primeiro esquema totalmente homomórfico conhecido foi o proposto por Gentry em sua tese original, apresentada em 2009, o mesmo incluía pela primeira vez a primitiva recrypt, que dava ao esquema, originalmente parcialmente homomórfico, a característica de ser totalmente homomórfico. Este esquema utiliza como base reticulados ideais. Utilizando a tese de Gentry, e seu esquema de bootstraping como base, Halevi em parceria com o próprio Gentry, implementaram em 2010, o esquema totalmente homomórfico Gentry-Halevi, este basicamente o esquema original de Gentry, com inclusão de algumas melhorias. Dentre elas destaca-se uma optimização na geração de chaves do esquema SHE (do inglês: Somewhat Homomorphic Scheme), sendo que esta não mais requer um polinômio de inversão total.

Ainda em 2010, Dijk, Gentry, Halevi e Vaikuntanathan (DGHV), propuseram um esquema totalmente homomórfico que utiliza apenas álgebra modular sobre o anel de números inteiros. Este mesmo esquema foi analisado por Coron em 2011, que propôs duas técnicas de redução de chave pública para o esquema, dando origem a duas variantes do esquema DGHV. A primeira variante de Coron, chamada de DGHV com chave reduzida, conta com a adição de novos parâmetros quadráticos adicionados as primitivas do esquema, armazenando apenas um pequeno conjunto da chave pública para então gerar a chave pública completa em tempo de execução. A segunda variante de Coron, chamada de DGHV com compactação de chave pública com troca de modulo, utiliza geradores de números pseudoaleatórios e armazena apenas as sementes e os fatores de correção para os números pseudoaleatórios, o que permite recuperar o valor dos parâmetros da chave pública em tempo de execução.

Em 2011, Brakerski e Vaikuntanathan, propuseram um novo tipo de esquema totalmente homomórfico, este é baseado no problema de aprendizagem com erros (sigla em inglês: LWE), este em especifico aplicando a problemas de aprendizagem de erros em anel (da sigla em inglês: RLW), este aplica os resultados conhecidos para o RLWE diretamente no esquema. Já em 2012, Brakerski, Gentry e Vaikuntanathan, apresentaram um novo esquema totalmente homomórfico com base no problema LWE, este esquema funciona sem a necessidade de operação de bootstrapping porem o mesmo é mantido no esquema por questões de eficiência.

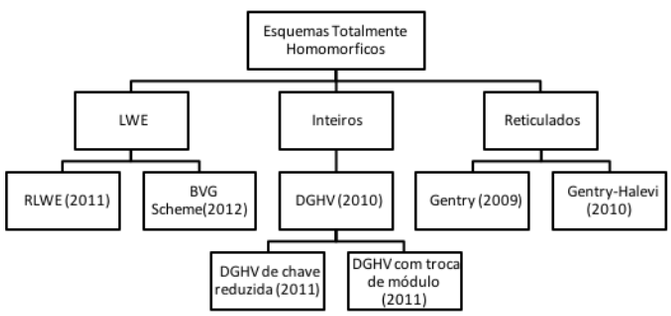

Figura I – hierarquia básica dos sistemas FHE
Fonte: Própria

A. DGHV e Melhorias

Neste tópico é descrito o esquema DGHV original, assim como são descritas as alterações propostas por Coron et. al., as quais atingiram uma notável taxa de compressão no tamanho da chave pública, com relação ao esquema original.

1) DGHV sobre inteiros

Gentry define que DGHV sobre inteiros utiliza como base um conjunto de inteiros públicos, $x_i = p.q_i + r_i, 0 \leq i \leq \tau$ onde o inteiro p é secreto [9], sendo dado um parâmetro de segurança λ, os seguintes parâmetros descrevem o esquema SHE, que gera o FHE sobre inteiros :

- γ é o comprimento em bits de x_i's.
- η é o comprimento em bits da chave secreta p.
- ρ é o comprimento em bits do ruído r_i.
- τ é o número de x_i's na chave pública.
- ρ' é um parâmetro de ruído secundário utilizado para cifrar.

Que devem seguir as seguintes restrições, para garantir a funcionalidade e segurança do sistema:

- $\rho = \omega(log\lambda)$, para proteção contra ataques de força bruta direcionados ao ruído

- $\eta \geq \rho.\Theta(\lambda log^2\lambda)$ para que seja possível realizar operações homomórficos para avaliar o "circuito de decriptação reduzido"
- $\gamma = \omega(\eta^2.log\lambda)$ para frustrar ataques baseados em retículos com aproximação pelo problema de MDC (Mínimo Divisor Comum)
- $\tau \geq \gamma + \omega(log\lambda)$ para reduzir a aproximação por MDC
- $\rho^{'} = \rho + \omega(log\lambda)$ para o parâmetro de ruído secundário

Dados esses parâmetros é possível gerar as primitivas do esquema DGHV, sendo que para um inteiro ímpar p de η-bit, seja utilizada uma distribuição sobre inteiros de γ-bit:

$$\mathcal{D}_{\gamma,\rho}(p) = \{\text{Escolha } q \leftarrow \mathbb{Z} \cap \left[0, \frac{2^{\gamma}}{p}\right), r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho}) : \text{Saída } x = q.p + r\}$$

2) Primitivas Do DGHV

As primitivas deste esquema são quatro, KeyGen, Encrypt, Decrypt e Evaluate. KeyGen é a primitiva responsável pela geração de chaves do esquema, Encrypt responsável por gerar o texto cifrado, Decrypt responsável por decifrar o texto cifrado, até estas três primitivas são comuns e usuais aos esquemas homomórficos [10]. A primitiva Evaluate foi inclusa com o propósito de reduzir o ruído gerado, Dijk et al descreve o algoritmo de Evaluate como:

> "O algoritmo *Evaluate* tem como entrada uma chave pública pk, um circuito *C*, e uma tupla de textos cifrados $\vec{c} = \langle c_1, \dots, c_t \rangle$ (um para cada entrada de bit de C), que tem como saída outra cifra c" [9]

Sendo assim obtemos as primitivas como sendo:

KeyGen(λ)

Sendo a chave privada um inteiro ímpar de η-bits, $p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^{\eta})$

Para a chave pública amostramos $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$ para $i = 0, \dots, \tau$. De forma que $x_0$seja maior, recomeçando a não ser que x_0 seja um número ímpar e ainda $r_{\rho}(x_0)$. Assim a chave pública é pk=$\langle x_0, x_1, \dots, x_{\tau} \rangle$.

Encrypt $(pk, m \in \{0,1\})$.

Escolhe-se um subconjunto $S \subseteq \{1,2, \dots, \tau\}$ aleatório, e um inteiro aleatório r entre $\left(-2^{\rho^{'}}, 2^{\rho}\right)$, e gera o texto cifrado c como:

$$c = \left[m + 2r + 2\sum_{i \in S} x_i\right]_{x0}$$

Evaluate$(pk, C, c_1, \dots, c_t)$.

Dado o circuito C com t bits de entrada e t textos cifrados c_i, aplicar a adição e multiplicação das portas lógicas de C nos textos cifrados, executando todas as operações sobre os inteiros, e retornar o inteiro resultante.

Decrypt(sk, c).

Tem como saída $m \leftarrow (c \bmod p) \bmod 2$. Note que $c \bmod p = c - p.\lfloor c/p \rceil$e p é impar, invés podendo computar: $m \leftarrow [c]_2 \oplus [\lfloor c/p \rceil]_2$.

Assim é descrito o esquema SHE, sendo este semanticamente seguro contra ataques de aproximação de MDC.

3) DGHV Sobre Inteiros Com Chave Pública Reduzida

Coron utilizou uma variante do esquema DGHV, onde foi adicionado um novo parâmetro β, sendo manipulado números inteiros $x'_{i,j}$, na forma de $x'_{i,j} = x_{i,0}.x_{j,1} \bmod x_0$ para $1 \leq i, j \leq \beta$. Dessa forma, apenas 2β inteiros precisam ser armazenados para gerar os $\tau = \beta^2$ inteiros utilizados para a criptografia. Em outras palavras, o sistema executa a criptografia utilizando elementos na forma quadrática da chave pública, ao contrário da forma linear adotada anteriormente. Tal permite reduzir o tamanho da chave pública de τ para $2\sqrt{\tau}$ inteiros de γ bits [7].

Além disso, a fim de tornar esse esquema totalmente homomórfico, são necessários λ^3 elementos y com comprimento de $\kappa = \gamma + 2 + \lceil \log_2(\lambda + 1) \rceil$, o que aumentaria o tamanho da chave pública de $\tilde{\mathcal{O}}(\lambda^7)$ para $\tilde{\mathcal{O}}(\lambda^8)$. Coron então propôs que apela o primeiro dos elementos de y fosse armazenado na chave, e, os outros gerados a partir de uma função geradora de números aleatórios. Dessa forma, o tamanho da chave é mantido na ordem de λ^7, e, os elementos de y são recuperados em tempo de execução [7].

Resumidamente, a principal modificação nesse sistema é o armazenamento da chave privada, e, encriptações da chave privada em forma quadrática. Os elementos são então recuperados on-the-fly quando necessário nas primitivas de encrypt, decrypt e recrypt.

a) Construção

Coron, em seu trabalho [7], demonstra uma melhoria no esquema DGHV, principalmente com relação com comprimento da chave pública do esquema SHE. Originalmente a chave pública possuía um comprimento na ordem de $O(\lambda^{10})$, foi reduzida para $O(\lambda^7)$. A principal técnica para se conseguir essa diminuição foi armazenar apenas um subconjunto da chave pública, e, recuperar a chave completa on-the-fly por meio de multiplicações dos elementos desse subconjunto. Coron também demonstra que seu sistema modificado ainda é semanticamente seguro, porém sobre uma variante do problema do MDC aproximado.

A seguir é apresentada uma descrição completa do esquema proposto por Coron, conforme disponível em (CORON et.al, 2011).

b) KeyGen(1^{λ}).

Gera um número ímpar p com η bits. Escolhe um número inteiro $q_0 \in [0, 2^{\gamma}/p)$, escolhido como um produto aleatório de λ^2-bits primos, e $x_0 = q_0 \cdot p$. Gera β pares inteiros de $x_{i,0}, x_{i,1}$ dentro do intervalo de $1 \leq i \leq \beta$: $x_{i.b} = p.q_{i.b} + r_{i,b}, 1 \leq i \leq \beta, 0 \leq b \leq 1$, onde, $r_{i,b}$ são inteiros entre $(-2^{\rho}, 2^{\rho})$ e $q_{i,b}$ são inteiros aleatórios de ordem $[0, q_C)$. Sendo assim $pk^* = (x_0, x_{1,0}, x_{1,1}, \dots, x_{\beta,0}, x_{\beta,1})$.

Também gera os vetores $s^{(0)}$ e $s^{(1)}$, de comprimento $\lceil\sqrt{\Theta}\rceil$, que seguem a condição de que $s_1^{(0)} = s_1^{(1)} = 1$, para cada um dos $\kappa \in [0, \sqrt{\theta})$ e $b = 0,1$, onde há pelo menos um bit não zero entre os $s_i^{(b)}$'s, $k\lfloor\sqrt{B}\rfloor < i \leq (k+1)\lfloor\sqrt{B}\rfloor$, com $B = \Theta/\theta$, e S sendo $S = \{(i,j): s_i^{(0)}.s_i^{(1)} = 1\}$, contendo exatamente θ elementos.

Inicializa um gerador **f** de números pseudoaleatórios de todo o sistema com uma semente aleatória se, usando assim f(se) para gerar $u_{i,j} \in [0, 2^{\kappa+1})$para $1 \leq i,j \leq \lceil\sqrt{\Theta}\rceil$, $(i,j) \neq (1,1)$. Então, atribuindo $u_{1,1}$ de forma que:

$$\sum_{(i,j)\in S} u_{i,j} = x_p \bmod 2^{\kappa+1}$$

Onde $x_p \leftarrow \lfloor 2^{\kappa}/p \rfloor$. Assim, computando a cifra de $\sigma^{(b)}$dos vetores $s^{(b)}$, escolhendo para cada $i \in [1, \lceil\sqrt{\Theta}\rceil]$ e $b = 0,1$, inteiros aleatórios $r'_{i,b} \in (-2^{\rho}, 2^{\rho})$ e $q'_{i,b} \in [0, q_0)$, é determinado que:

$$\sigma_i^{(b)} = s_i^{(b)} + 2r'_{i,b} + p.q'_{i,b} \bmod x_0$$

Desde modo, temos como saída da primitiva KeyGen(), a chave privada sendo $sk = (s^{(0)}, s^{(1)})$, e a chave pública como $pk = (pk^*, se, u_{1,1}, \sigma^{(0)}, \sigma^{(1)})$.

c) Encrypt (pk, $m \in \{0,1\}$).

Escolha uma matriz de números aleatórios $b = (b_{i,j})_{1\leq i,j\leq\beta} \in [0,2^{\alpha})^{\beta\times\beta}$ e um inteiro aleatório r no intervalo $(-2^{\rho'}, 2^{\rho'})$. Retorne o texto cifrado como:

$$c^* = m + 2r + 2\sum_{1\leq i,j\leq\beta} b_{ij} \cdot x_{i,0} \cdot x_{j,1} \bmod x_0$$

d) Evaluate

Add(pk, c_1^*, c_2^*). Retorna $c_1^* + c_2^* \bmod x_0$

Mult(pk, c_1^*, c_2^*). Retorna $c_1^* \cdot c_2^* \bmod x_0$

e) Expand (pk, c^*).

Esse procedimento, a expansão do texto cifrado, recebe um texto cifrado c^* e computa a matriz associada z. Podemos pensar nesse procedimento como separado tanto do Encrypt quanto de Decrypt, já que pode ser executada de forma pública utilizando apenas o texto cifrado e dados públicos. Para tal, para cada $1 < i,j < \sqrt{\Theta}$, primeiramente compute o número inteiro aleatório $u_{i,j}$ usando o gerador de números pseudoaleatórios f(se), então calcule $y_{i,j} = u_{i,j}/2^{\kappa}$ e então compute $z_{i,j}$:

$$z_{i,j} = [c^* \cdot y_{i,j}]_2$$

Mantendo apenas $n = \lceil\log_2(\theta+1)\rceil$ bits de precisão após o ponto binário. Defina a matriz $z = (z_{i,j})$. Retorne o texto cifrado expandido $c = (c^*, z)$.

f) Decrypt (sk, c^*, z).

Retornar $m \leftarrow \left[c^* - \left\lfloor\sum_{i,j} s_i^{(0)} \cdot s_j^{(1)} \cdot z_{ij}\right\rceil\right]_2$

g) Recrypt (pk, c^*, z).

Para se executar o procedimento de recrypt, aplique o circuito de decriptação ao texto cifrado expandido Z e às chaves secretas encriptadas $\sigma_i^{(b)}$. Retorne o resultado como o texto cifrado renovado c_{new}^*.

Assim define-se de maneira completa o esquema DGHV Totalmente Homomórfico de chave pública reduzida. A primitiva Evaluate foi fragmentada, dando origem a outras a outras duas primitivas de nome Add e Mult. Essas primitivas mapeiam no espaço do texto-puro, respectivamente, as funções lógicas XOR e AND.

h) Parâmetros concretos e experimentais

Com as análises executadas em sua variante do esquema DGHV, levando em conta os ataques conhecidos, Coron chegou aos valores da Tabela 1. Parâmetros para os parâmetros do esquema.

TABELA 1. PARÂMETROS

PARÂMETRO	Λ	P	Ɛ	I	B	Θ	Є
TOY	42	16	1088	$1{,}6\times10^5$	12	144	15
SMALL	52	24	1632	$8{,}6\times10^5$	23	533	15
MEDIUM	62	32	2176	$4{,}2\times10^6$	44	1972	15
LARGE	72	39	2652	$1{,}9\times10^7$	88	7897	15

Fonte: Coron 2011

Vale ressaltar que, embora λ seja nomeado "parâmetro de segurança", ele não deve ser tomado da mesma forma que os valores em bits de segurança comuns em outros esquemas criptográficos. No caso do esquema homomórfico em questão, é mais adequado chamar os parâmetros de "níveis de segurança", sendo eles, por si só, inadequados para comparação com outros esquemas não homomórficos.

A Tabela 2 descreve os tempos de execução e tamanho das chaves públicas geradas pelos parâmetros de segurança inserido no esquema criptográfico, tais parâmetros representam e definem o nível de segurança dos pares de chaves gerados e por consequência de todo o esquema criptográfico, os valores reis destes parâmetros são restritos, sendo Toy equivalente a 42 bits de segurança, Small equivalente a 52 bits de segurança, Medium equivalente a 62 bits de segurança e Large equivalente a 72 bits de segurança. Na primitiva KeyGen, essa responsável pela geração dos pares de chaves públicas e privadas tem-se um taxa de aumento no tempo de execução aproximada de oito vezes conforme

aumentamos o valor dos parâmetros indo de Toy a Large, destaca-se também os baixos tempos de execução da primitiva Decrypt, isso se deve ao fato de que a representação do circuito de baixa profundidade da primitiva ter sido optimizado onde a chave secreta é representada de uma maneira diferente, esse mesmo circuito é utilizado na primitiva de Recrypt.

TABELA 2. TEMPOS DE PROCESSAMENTO E TAMANHOS DE CHAVE

Parâmetro	KeyGen	Encrypt	Decrypt	Recrypt	Pubkey
Toy	4.38s	0.05s	0.01 s	1.92s	0.95 MB
Small	36 s	0.79s	0.01 s	10.5s	9.6 MB
Medium	5min9s	10s	0.02 s	1min20s	89 MB
Large	43min	2min57s	0.05 s	14min33s	802 MB

Fonte: [Coron et al., 2011]

B. DGHV com compressão de chave e mudança de módulo

No trabalho intitulado "Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers", além de demonstrar um ataque a esse sistema com complexidade de $\tilde{\mathcal{O}}(2^{\rho})$, Coron obteve uma implementação feita no software matemático SAGE cuja chave pública possuía 10.3MB de tamanho, ao contrário de 802MB do seu trabalho anterior. Nesse trabalho, o comprimento da chave pública foi reduzido ainda mais de $\tilde{\mathcal{O}}(\lambda^7)$ para $\tilde{\mathcal{O}}(\lambda^5)$.

A principal inovação proposta por Coron nesse esquema, é que, ao invés de se armazenar os elementos da chave, e, criptografia desses elementos é armazenada apenas a correção do valor com relação a um gerador de números aleatórios. Dessa forma, os dados a serem armazenados são menores, e, o dado completo é recuperado on-the-fly pelas primitivas de encrypt, recrypt, decrypt, expand e recrypt. Além disso, é descrita uma técnica de troca de módulo, que permite a esse esquema utilizar o framework sem boostraping proposto por Brakerski, Gentry e Vaikuntanathan.

Anteriormente foi apresentado os tempos de execução e tamanho das chaves públicas da primeira de uma técnica de redução de chave pública [7], entretanto, destaca-se o tamanho das chaves geradas, mesmo que reduzido, ainda é inviável para aplicações, sendo que com o parâmetro Large a chave gerada possui 802 MB, já a chave gerada por essa segunda variação de Coron [8], (ver tabela 2), com o mesmo parâmetro de segurança, gerou uma chave pública de 10.3 MB, isso deve-se ao fato de que nesta implementação é empregada uma técnica de redução de chave diferente, e a própria manipulação dos dados é feita na forma de vetor, ao em vez de matriz como é realizada na implementação anterior, podemos também notar uma redução no tempo da primitiva Recrypt, isso deve a uma adaptação feita do framework, proposto no esquema BVG.

1) KeyGen($\mathbf{1}^{\lambda}$)

Gerar um número ímpar p com η bits. Escolhe um número inteiro $q_0 \in [0, 2^{\gamma}/p)$, sendo $x_0 = q_0 \cdot p$. Inicializa um gerador de números pseudoaleatórios f_1 com uma semente aleatória se_1. Aplicando assim $f_1(se_1)$ para gerar o conjunto de inteiros $\chi_i \in [0, 2^{\gamma}/p)$ para $1 \le i \le \tau$, computando assim:

$$\delta_i = \langle \chi_i \rangle_p + \xi_i \cdot p - r_i$$

onde $r_i \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$ e $\xi_i \leftarrow \mathbb{Z} \cap \left[0, \frac{2^{\lambda+\eta}}{p}\right)$. Sendo $pk^* = (se_1, x_0, \delta_i, \dots, \delta_{\tau})$ sendo os inteiros correspondentes a x_i para $1 \le i \le \tau$, definido por $x_i = X_i - \delta_i$.

Adicionalmente gera um vetor randômico de bits s de comprimento Θ, sujeito as condições que de $s_1 = 1$, para cada $k \in [0, \theta)$, onde se tem no máximo um bits não zero entre s_i para cada $k \cdot B + 1 \le i < (k+1) \cdot B + 1$, onde $B = \lfloor \Theta/\theta \rfloor$, e que o peso de Hamming de s seja exatamente θ.

Inicializa um gerador de números pseudoaleatórios f_2, utilizando uma semente aleatória se_2, gerando inteiros $u_i \in [0, 2^{k+1})$ para $2 \le i \le \Theta$, onde $k := \gamma + n + 2$. Assim, obtemos u_1 da seguinte maneira:

$$\sum_{i=1}^{\Theta} si \cdot u_i = x_p \bmod 2^{k+1}$$

onde $x_p \leftarrow \lfloor 2^{\kappa}/p \rceil$.

Inicializa-se um gerador de números pseudoaleatórios f_3, utilizando uma semente aleatória se_3, computando assim codificações de σ do vetor s da seguinte maneira: utiliza-se $f_3(se_3)$ para gerar inteiros $\chi'_i \in [0, 2^{\gamma}[$ para cada $1 \le i \le \Theta$, gerando assim inteiros aleatórios $r'_i \in (-2^{\rho}, 2^{\rho})$ e $\xi'_i \in [0, 2^{\gamma+\eta}/p)$, assim tendo:

$$\delta'_i = \langle \chi'_i \rangle_p + \xi'_i \cdot p - 2 \cdot r'_i - s_i$$

A codificação correspondente de $\sigma_i des_i$ sendo definida como:

$$\sigma_i = \chi'_i - \delta'_i$$

Obtém-se assim como saída a chave privada $sk = s$ e a chave pública $pk = (pk^*, se_2, u_1, se_3, \delta')$.

2) Encrypt (pk, $m \in \{0,1\}$)

Recuperar os inteiros x_i da pk^*. Escolhendo assim um vetor aleatório inteiro $b = (b_i)_{1 \le i \le \tau} \in [0, 2^{\alpha})^{\tau}$ e um inteiro aleatório r em $(-2^{\rho'}, 2^{\rho'})$. Computando a cifra como:

$$c^* = m + 2r + 2 \sum_{i=1}^{\tau} b_i \cdot x_i \bmod x_0$$

3) Add(pk, c_1^*, c_2^*)

Retorna $c_1^* + c_2^* \bmod x_0$

4) Mult(pk, c_1^*, c_2^*)

Retorna $c_1^* \cdot c_2^* \bmod x_0$

5) Expand(pk, c^*)

O procedimento de expansão do texto cifrado recebe um texto cifrado c^* e computa a matriz associada expandindo o texto cifrado. Para tal, cada $1 < i < \Theta$, primeiramente

computa-se o número inteiro aleatório u_i usando o gerador de números pseudoaleatórios $f_2(se_2)$, então calcule $y_{i,j} = u_{i,j}/2^{\kappa}$ e então compute z_j :

$z_i = \lceil c^* \cdot y_i \rfloor_2$

Mantendo apenas $n = \lceil \log_2(\theta + 1) \rceil$ bits de precisão após o ponto binário. Definindo o vetor $z = (z_i)$. Retornando o texto cifrado expandido $c = (c^*, z)$

6) Decrypt(sk, c^*, z)

Retornam $\leftarrow \left[c^* - \lfloor \sum_{i=1}^{\Theta} s_i \cdot z_i \rceil\right]_2$

7) Recrypt(pk, c^*, z)

Recupera os bits da chave privada criptografada σ_i da pk. Aplicando o circuito de decodificação ao texto cifrado expandido z e aos bits da chave secreta criptografada σ_i. Tendo como saída um texto cifrado com o ruído reduzido como c^*_{novo}.

Assim finalizamos a descrição completa do esquema DGHV Totalmente Homomórfico de chave pública reduzida, a primitiva Evaluate, foi fragmentada, dando origem a outras a outras duas primitivas de nome Add e Mult, que emulam, respectivamente, o comportamento das portas logicas XOR e AND.

III. SEGURANÇA DO SISTEMA CRIPTOGRÁFICO

Gentry demonstra que esse esquema criptográfico, devido as limitações de escopo de parâmetros e a relação entre eles, é seguro de acordo com o problema do MDC aproximado: Dado um conjunto de inteiros $x_0, x_1, \ldots, x_\tau$ todos escolhidos de forma aleatória próximos a múltiplos de um inteiro longo p, encontrar esse "divisor comum próximo" p. [10]

A. Ataques

Ainda no seu trabalho, Gentry demonstra uma simples técnica de ataque de força bruta, onde o MDC para os ruídos r_1', r_2' é calculado. Essa técnica garante que o número p será encontrado, mas, possui tempo de execução de aproximadamente $2^{2\rho}$. A definição para o problema do MDC aproximado especifica para o esquema homomórfico sobre inteiros é: Dado um número polinomial de amostras da distribuição $\mathcal{D}_{\gamma,\rho}(p)$ para um inteiro ímpar aleatório de η bits p, retorne p.

Outros ataques foram propostos, mas, não serão discutidos no escopo desse trabalho.

IV. CONCLUSÃO

Neste artigo foi apresentado o estado da arte dos principais esquemas criptográficos totalmente homomórficos com ênfase nos esquemas sobre inteiros, destacando-se os parâmetros e primitivas dos esquemas derivados do DGHV original, bem como as melhorias propostas por Coron [8]. Como foi mencionado, barreiras relacionadas ao tamanho da chave pública e tempo de execução foram quebradas, mas ainda não estão para aplicação.

Brakerski, Gentry e Vaikuntanathan propuseram [12] um framework para esquemas totalmente homomórficos onde o ruído aumenta somente de maneira linear ao invés de ser exponencial. Esquemas totalmente homomórficos baseados no problema de LWE são relativamente recentes, no entanto, notáveis avanços estão sendo realizados com os mesmo sendo que Brakerski e Vaikuntanathan propuseram uma implementação parcial sem a característica totalmente homomórfico.

Sendo assim, ainda temos um grande campo de pesquisas a serem desenvolvidas com relação a sistemas criptográficos totalmente homomórficos, tanto como no que se refere a esquemas baseados em números inteiros a esquemas totalmente homomórficos em geral.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. M. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2): p.120–126, 1978.

[2] MILLER, V. Uses of elliptic curves in cryptography. In Advances in Cryptology, Crypto 85, Lecture Notes in Computer Science, p. 417–426. Springer, 1985

[3] SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In Shor, P. W., editor, 35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994), p. 124–134. IEEE Comput. Soc. Press, 1994.

[4] BERNSTEIN, D. J., BUCHMANN, J. A., DAHMEN E. Post-Quantum Cryptography, Chicago and Darmstadt, 2008.

[5] AHITUV, N., LAPID, Y., e NEUMANN, S. Processing Encrypted Data, In Comm. of the ACM, vol. 20, p. 777-780, 1987

[6] GENTRY, C. A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University, 2009, Disponivel em:http://crypto.stanford.edu/craig.

[7] CORON, J.S., MANDAL, A., NACCACHE, D. e TIBOUCHI, M., Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In P. Rogaway (Ed.), CRYPTO 2011, LNCS, vol. 6841, Springer, p. 487-504.

[8] CORON, J. S., NACCACHE, D., & TIBOUCHI, M. (2012). Public key compression and modulus switching for fully homomorphic encryption over the integers. In Advances in Cryptology–EUROCRYPT 2012, p. 446-464. Springer Berlin Heidelberg.

[9] DIJK, M. VAN, GENTRY, C., HALEVI, S. e VAIKUNTANATHAN, V., Fully homomorphic encryption over the integers. In H. Gilbert (Ed.), EUROCRYPT 2010, LNCS, vol. 6110, Springer, p. 24-43, 2010.

[10] GENTRY, C., e HALEVI, S., "Implementing Gentry's fully-homomorphic encryption scheme," Advances in Cryptology-EUROCRYPT 2011, p. 129-148, 2011.

[11] GENTRY, C., Fully Homomorphic Encryption Using Ideal Lattices, Symposium on Theory of Computing - STOC, p.169-178, 2009.

[12] BRAKERSKI, Zvika; GENTRY, Craig; VAIKUNTANATHAN, Vinod. (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ACM, 2012. p. 309-325.