DISSERTAÇÃO DE GRADUAÇÃO

Criptografia Homomórfica

Narciso Busatto Junior

Brasília, dezembro de 2013.

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE GRADUAÇÃO

Criptografia Homomórfica

Narciso Busatto Júnior

Dissertação submetida ao Departamento de Engenharia Elétrica
como requisito parcial para obtenção do grau de
Engenheiro de Redes de Comunicação

FICHA CATALOGRÁFICA

BUSATTO, Narciso Júnior. Criptografia Homomórfica. [Distrito Federal] 2013.
i, 49p. (ENE/FT/UnB, Engenheiro de Redes de Comunicação. 2013).
Monografia de Graduação – Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

BUSATTO, NARCISO JUNIOR (2013). Criptografia Homomórfica. Monografia de Graduação, Departamento de Engenharia Elétrica, Universidade de Brasília, DF, 49p.

CESSÃO DE DIREITOS

NOME DOS AUTORES: Narciso Busatto Júnior

TÍTULO: Criptografia Homomórfica

GRAU/ANO: Engenheiro de Redes de Comunicação / 2013

Narciso Busatto Júnior
Colônia Agrícola Bernardo Sayão, Chácara 3 Lote 9
Condomínio Flor do Cerrado – Guará II
CEP: 71.080-025 – Brasília – DF – Brasil

À minha família

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus por tudo. Pelo amparo, pela permanência ao meu lado, pela inspiração.

Gostaria de agradecer de forma especial minha família, principalmente minha esposa, que não mediu forças para me acompanhar, estando ao meu lado em grande parte desse processo, principalmente nos momentos de problemas e dificuldades. Agradeço por sempre ter apoiado minhas decisões, permitindo que eu realizasse este trabalho com empenho e dedicação; e por ter compreendido tantas horas reservadas aos estudos.

Gostaria de agradecer meus pais, grandes responsáveis pela minha vocação pela área, participando de toda minha formação fundamental, oferecendo-me mais do que a simples e boa educação. Sempre fui cativado à busca pelo aperfeiçoamento pessoal. O convívio e as experiências recebidos foram essenciais para o que sou hoje.

Por fim, também gostaria de agradecer a dois professores: O professor Anderson Nascimento, com quem tive contato desde pequeno e sempre foi uma referência pessoal e profissional para mim. Agradeço pela sua ajuda sempre frequente e primordial, sem a qual todo esse trabalho não teria sido realizado. Agradeço também ao professor Diego Aranha, pelos vários momentos de auxílio neste último ano, apresentando-me tantas ideias de pesquisa, culminando no desenvolvimento deste trabalho, além de estar acessível e disponível em diversos momentos.

RESUMO

Criptografia Homomórfica

Neste trabalho, buscamos compreender as mais diversas funcionalidades inerentes a Criptografia Homomórfica. Baseamos o nosso estudo em criptosistemas de chave pública, observando como a diversidade de mecanismos está sendo utilizada. Apesar de ser um tema comum na área, apenas em 2009, com o trabalho de Ph.D. de Craig Gentry, foi demonstrada a existência de criptosistemas completamente homomórficos. Desde então, diversas novas abordagens surgiram e estão sendo analisadas no contexto científico quanto a sua aplicabilidade e sua resiliência. No decorrer do trabalho, além da apresentação de diversos criptosistemas tanto parcialmente como completamente homomórficos, observaremos aplicações, funcionalidades e aspectos responsáveis pela importância deste tema na Criptografia Moderna.

ABSTRACT

Homomorphic Encryption

In this work, we seek to understand the various features inherent in homomorphic encryption. We base our study on public key cryptosystems, watching as the diversity of mechanisms is being used. Though a common theme in the area, only in 2009, with the Ph.D. work of Craig Gentry has shown the existence of fully homomorphic cryptosystems. Since then, several new approaches have emerged and are being analyzed in a scientific context about its applicability and its resilience. Throughout his work, besides the presentation of several cryptosystems both partially as fully homomorphic, identify applications, features and aspects responsible for the importance of this theme in Modern Cryptography.

SUMÁRIO

Capítulo 1
Introdução

Mesmo estando presente em diversos períodos da história da humanidade, a criptografia tornou-se mais imponente com a conceituação da criptografia de chave pública apresentadas por *Whitfield Diffie* e *Martin Hellman* em 1976 com a publicação do artigo *New Directions in Cryptography*, apresentando o clássico algoritmo de troca de chaves *Diffie-Hellman*. Desde então, a necessidade de privacidade e segurança da informação ganhou uma nova importância dentro de um ambiente globalizado envolvido pela imensa troca de informações. Com o grande desenvolvimento da tecnologia e a internet cada vez mais presente no dia-a-dia de cada indivíduo, a busca por ambientes computacionais cada vez mais protegidos obrigou a procura e o estudo de sistemas criptográficos idealmente seguros.

É importante ressaltar que a criptografia por si só não é capaz de solucionar todos os problemas relacionados à segurança existente dentro de um ambiente qualquer. Falhas na implementação do algoritmo, na utilização de um hardware que pode expor texto claro em alguns aspectos, na escolha de chaves criptográficas inadequadas, enfim, diversas são as fragilidades que um sistema pode gerar, comprometendo a segurança proposta em questão. Um exemplo típico é a necessidade de um servidor, externo ao usuário, decifrar uma mensagem para ser capaz de processá-la e enviá-la de volta, cifrada. Caso esse servidor seja acessado por qualquer indivíduo malicioso, a mensagem em texto claro pode ser manipulada, gerando, portanto, insegurança sobre as operações deste servidor.

A Criptografia Homomórfica (***HE – Homomorphic Encryption***) surge como uma das principais ferramentas para sanar este e diversos outros problemas da criptografia moderna. Com ela, é possível realizar um conjunto de procedimentos computacionais diretamente sobre um texto cifrado sem a necessidade de decifrá-lo, impedindo assim que a mensagem original seja de conhecimento do servidor ou de qualquer usuário que acompanhe a realização de tais procedimentos. Desta forma, é possível garantir a privacidade dos dados nos processos de comunicação e de armazenamento externos ao usuário, sendo possível, por exemplo, delegar cálculos ou realizar o armazenamento de dados para ambientes públicos (*cloud computing*).

Este trabalho realiza uma investigação sobre os aspectos e propriedades inerentes a essa ferramenta. Sua composição está subdividida em 6 capítulos. No capítulo 2 serão apresentados os fundamentos e definições necessários para a compreensão de *HE*. É composto por teoremas, definições, proposições matemáticas baseadas em Teoria dos Números e Matemática Abstrata que implicam no conjunto de esquemas essenciais para fundamentar não só o homomorfismo, mas todo e qualquer recurso necessário aos criptosistemas explicados a seguir. Isso não impede que alguns elementos, mesmo inerentes ao estudo de *HE*s, sejam apresentados nos tópicos nos quais serão citados.

No capítulo 3 serão apresentadas a definição e fundamentação de Criptografia Parcialmente Homomórfica e alguns sistemas criptográficos que implementam *HE*. Identificaremos suas implicações e funcionalidades, analisando a eficiência, a resiliência e a praticidade de cada algoritmo de forma geral. É importante salientar que em parte dos algoritmos que implementam *HE*, o homomorfismo foi observado antes de sua definição formal, ou seja, já era conhecida essa característica nos criptosistemas homomórficos. A busca é pela identificação dos potenciais candidatos a algoritmo padrão. Alguns apresentaram vulnerabilidades críticas e

logo foram descartados. No entanto, o seu estudo auxiliou, em muito, o desenvolvimento de outros criptosistemas aceitos pelo meio científico.

No capítulo 4 serão apresentadas as principais aplicações em *HE*. A motivação deve-se ao fato de algumas pesquisas focarem individualmente cada necessidade específica, como computação em nuvem, databases ou comunicação multiparte. Assim que identificamos as propriedades desejadas e obtemos o *feedback* das aplicações de sucesso, temos maior conhecimento e experiência para corrigir ou implementar funcionalidades para outras aplicações.

No capítulo 5 serão apresentadas a definição e fundamentação da Criptografia Completamente Homomórfica (***FHE – Fully Homomorphic Encryption***), apresentando alguns criptosistemas que implementam esse importante mecanismo. Por fim, no capítulo 6 realizaremos uma breve conclusão e detalhes para trabalhos futuros.

Capítulo 2

Definições, Fundamentos Matemáticos e Algoritmos.

2.1 Introdução

Existe uma linha tênue que separa a Matemática e a Criptografia. Apesar de compartilharem de interesses diferentes, interagem conjuntamente para o desenvolvimento mútuo. Mais especificadamente, a Álgebra e a Teoria dos Números tem sido a principal fonte de algoritmos e teoremas para o estabelecimento de criptosistemas de segurança demonstrável. Por outro lado, a computação é responsável por testes sucessivos e cálculos inoperáveis no ponto de vista da limitação humana. Assim, a evolução de uma destas ciências acarreta no desenvolvimento da outra, fazendo com que seus profissionais, em certo ponto, trabalhem conjuntamente.

Neste trabalho, é importante salientar que a simbologia $\mathbb{Z}/n\mathbb{Z}$ será substituída pelo símbolo $\mathbb{Z}_n$. Apesar $\mathbb{Z}_p$ ser o conjunto dos números p-ádicos, muitos são os profissionais que, por costume, utilizam como a forma equivalente à segunda representação na grande maioria dos trabalhos acadêmicos. Manteremos a representação neste trabalho.

2.2 Definições Iniciais

Definição 2.2.1: Dizemos que a é congruente a b módulo n se $n|(a-b)$ (lê-se n divide $a-b$). Representamos por

$$a \equiv b \ (mod\ n)$$

Informalmente dizemos que b é o resto da divisão de a por n.

Teorema 2.2.2: (Função Totiente de Euler) Representamos por $\varphi(n)$ a quantidade de números menores ou iguais a n que são coprimos com ele mesmo. Caso n seja um número primo, então $\varphi(n) = n - 1$. De forma mais geral, para $n = {p_1}^{r_1}.{p_2}^{r_2} \ldots {p_m}^{r_m}$, com $p_i \neq p_j$ para $1 \leq i, j \leq m$, temos que:

$$\varphi(n) = n.\prod_{i=1}^{m}\left(1 - \frac{1}{p_i}\right)$$

Teorema 2.2.3: (Teorema de Euler) Considere $n \in \mathbb{N}$, $a \in \mathbb{Z}$ e $mdc(a,n) = 1$ temos que:

$$a^{\varphi(n)} \equiv 1 \ (mod\ n)$$

Este teorema costuma ser apresentando conjuntamente com o *Pequeno Teorema de Fermat* que afirma que para um p primo positivo qualquer, se $mdc(a,p) = 1$, temos:

$$a^p \equiv a \ (mod\ p)$$

É importante citar que para um inteiro positivo n, define-se a *função de Carmichael* $\lambda(n)$ para obter-se o menor valor inteiro positivo m tal que

$$a^m \equiv 1 \ \ (mod\ n)$$

Para um inteiro a tal que $mdc(a,n) = 1$.

Teorema 2.2.4: (Teorema de Carmichael) Seja n um inteiro positivo, temos (considere p um primo ímpar)

$$\lambda(n) = \begin{cases} \phi(n), \ \ para\ n = 2, 4, p^k, 2p^k, p \\ \frac{1}{2}\phi(n), \ \ para\ n = 2^k, k \geq 3 \end{cases}$$

Uma propriedade muito utilizada é $\lambda\big(mmc(a,b)\big) = mmc(\lambda(a), \lambda(b))$.

Proposição 2.2.5: Considere n um número natural não nulo, $a, x, y \in \mathbb{Z}$ e $mdc(a,n) = 1$. Portanto, $x \equiv y \ \big(mod\ \varphi(n)\big)$ se, e somente se $a^x \equiv a^y \ (mod\ n)$.

Prova: ($\Rightarrow$) Se $x \equiv y \ (mod\ \varphi(n))$ então existe $k \in \mathbb{Z}$ tal que $x = y + k.\varphi(n)$. Se $mdc(a,n) = 1$, temos que

$$a^x \equiv a^{y+k.\varphi(n)} \equiv a^y.a^{k.\varphi(n)} \equiv a^y.\left(a^{\varphi(n)}\right)^k \ (mod\ n)$$

Pelo Teorema 2.2.3, temos que $\left(a^{\varphi(n)}\right)^k \equiv (1)^k \equiv 1 \ (mod\ n)$. Logo:

$$a^x \equiv a^y \ (mod\ n)$$

($\Leftarrow$) Considere a equação $a^x \equiv a^y \ (mod\ n)$. Considere, sem perda de generalidade, que $x \geq y$. Assim, temos que $x = y + T$ com $T \in \mathbb{Z}$. Substituindo na equação, temos:

$$a^x \equiv a^{y+T} \equiv a^y.a^T \equiv a^y \ (mod\ n)$$

Simplificando, temos:

$$a^T \equiv 1 \ (mod\ n)$$

Observando o Teorema 2.2.3, temos que $\varphi(n) | T$. Portanto, podemos escrever $T = k.\varphi(n)$, para algum $k \in \mathbb{Z}$. Podemos reescrever a igualdade como $x = y + k.\varphi(n)$. Calculando módulo $\varphi(n)$, temos:

$$x \equiv y + k.\varphi(n) \equiv y \ (mod\ n)$$

■

Teorema 2.2.6: (Teorema Chinês do Resto) Considere $m_i \in \mathbb{Z}_+^*$, com $i = \{1, 2, 3, \dots, n\}$ de modo que para todo $i \neq j$, $mdc(m_i, m_j) = 1$, ou seja, todos m_i sejam coprimos entre si. Dadas as equações modulares $x \equiv a_i \ (mod\ m_i)$, existe uma única solução X, tal que

$$x \equiv X \left(mod \prod_{i=1}^{n} m_i \right)$$

Definição 2.2.7: Um inteiro a é um resíduo quadrático módulo n se existe um inteiro x tal que:

$$x^2 \equiv a \ (mod\ n)$$

Caso não exista a congruência, dizemos que a é um não resíduo quadrático módulo n.

Definição 2.2.8: Definimos o símbolo de Lagrange para a inteiro e p um primo ímpar como sendo

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{se } a \text{ for um resíduo quadrático módulo } p \text{ e } a \not\equiv 0 \ (mod\ p) \\ -1, & \text{se } a \text{ for um não resíduo quadrático módulo } p \\ 0, & se\ a \equiv 0 \ (mod\ p) \end{cases}$$

De outra forma, pode-se representar o símbolo pela equação modular

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \ (mod\ p)$$

Definição 2.2.9: Dado p_i primos ímpares distintos, temos $n = {p_1}^{r_1}.{p_2}^{r_2} \dots {p_k}^{r_k}$. Dada a definição anterior, definimos o símbolo de Jacobi, para um a inteiro, como sendo

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{r_1} \cdot \left(\frac{a}{p_2}\right)^{r_2} \cdots \left(\frac{a}{p_k}\right)^{r_k}$$

Definição 2.2.10: (Logaritmo discreto) Considere a equação modular $a^x \equiv b\ (mod\ n)$. Dada a complexidade de obter o valor de x para esse caso, este estudo ficou conhecido como *problema do logaritmo discreto* e escreveremos como

$$x \equiv logd(n)_a\ b \ (mod\ n)$$

Teorema 2.2.11: (Teorema Binomial) Considere a função $L(u) = \frac{u-1}{n}$ (quociente da divisão inteira). Se $x \in \mathbb{Z}_n$, então

Prova: Observe que

$$(1+n)^x = \sum_{i=0}^{x} \binom{x}{i} n^i = 1 + nx + \sum_{i=2}^{x} \binom{x}{i} n^i$$

Como $\sum_{i=2}^{x} \binom{x}{i} n^i \equiv 0 \ \ (mod\ n^2)$, segue:

$$(1+n)^x \equiv 1 + nx \ \ (mod\ n^2)$$

Considere y tal que $y = (1+n)^x \ \ mod\ n^2$. Substituindo, obtemos:

$$x \equiv \frac{y-1}{n} \ \ (mod\ n)$$

Ou seja,

$$x \equiv L((1+n)^x \ \ (mod\ \ n^2))\ (mod\ n)$$

■

Definição 2.2.12: Uma função $f: \mathbb{N} \to \mathbb{R}$ é dita *negligenciável* se para cada inteiro positivo k existe um inteiro $N = N(k)$ tal que para todo $x > N$ temos,

$$|f(x)| \leq \frac{1}{x^k}$$

Essa função pode ser representada por $negl(x)$.

2.3 Grupos

Definição 2.3.1: Seja G um conjunto e $*$ uma operação definida $G \times G \to G$. Definimos um grupo como um par $(G,*)$, tal que:

(i) $\forall a, b, c \in G$, temos que $(a * b) * c = a * (b * c)$ (associatividade).

(ii) Existe um único $e \in G$, chamado de *elemento neutro*, tal que para qualquer $a \in G$, então $a * e = e * a = a$ (Existência do elemento neutro).

(iii) Para qualquer $a \in G$, existe um único $a' \in G$ tal que $a * a' = a' * a = e$. Dizemos que a' é o inverso de a (Existência do elemento inverso).

Definição 2.3.2: Seja $(G,*)$ um grupo. Se para todo $a, b \in G$ é válida a igualdade $a * b = b * a$, então $(G,*)$ é chamado *grupo abeliano*.

Definição 2.3.3: Considere que $a^m = \underbrace{a * a * \ldots a}_{m\ vezes}$. A *ordem de um elemento* $a \in G$ é o menor inteiro m tal que $a^m = e$. Representamos m por $ord_a(G)$.

Definição 2.3.4: Dado H um subconjunto de G, dizemos que $(H,*)$ é um *subgrupo* de $(G,*)$ quando seus elementos obedecem à mesma operação definida para o grupo G. Para tanto, pode-se comprovar a validade da Definição 2.3.1 ou, como H é subconjunto de G, basta apenas verificar

(i) H é um subconjunto não vazio.

(ii) Para todo $a, b \in H$, então $a * b' \in H$, em que b' é o inverso de b.

Nesse caso, as propriedades de finitude, comutatividade e ciclicidade são hereditárias.

Definição 2.3.5: Dado um elemento $g \in G$, definimos como *gerador* se todos os elementos de G possam ser escritos como uma potência de g, como apresentada na Definição 2.3.3. Desta forma, $(G,*)$ é chamado de *grupo cíclico*.

2.3.1 Homomorfismo de Grupos

Definição 2.3.6: Considere $(G,*)$ e $(H,\star)$ grupos com suas respectivas operações. Seja uma aplicação $f : G \to H$ que relaciona elementos de grupo G com elementos do grupo H. Definimos um *homomorfismo* f de $(G,*)$ em $(H,\star)$ quando f satisfaz

$$f(a * b) = f(a) \star f(b)$$

Para todo a e b pertencentes a G.

2.4 Anéis

Definição 2.4.1: Seja R um conjunto e $*$ e $\star$ duas operações definidas $R \times R \to R$. Definimos um anel como um trio $(R,*,\star)$, tal que:

(i) $(R,*)$ é um grupo abeliano.

(ii) Se para todo $a, b, c \in R$, $(a \star b) \star c = a \star (b \star c)$ (Associatividade de $\star$)

(iii) Se para todo $a, b, c \in R$, $a \star (b * c) = (a \star b) * (a \star c)$ (Distributividade de $\star$ em relação à $*$).

Definição 2.4.2: Dado um anel $(R,*,\star)$ que para todo $a, b, c \in R$ que seja válida a igualdade $a \star (b \star c) = (a \star b) \star c$, chamamos *anel associativo*.

Definição 2.4.3: Dado um anel $(R,*,\star)$ para o qual existe um único $e_R \in R$, tal que $a \star e_R = e_R \star a = a$, para todo $a \in R$, chamamos R de *anel com elemento neutro*.

Definição 2.4.4: Dado um anel $(R,*,\star)$, tal que para todo $a, b \in R$ é válida a igualdade $a \star b = b \star a$, chamamos R de *anel comutativo*.

Definição 2.4.5: Dado um anel $(R,*,\star)$ comutativo, dizemos que $a \in R$ um *divisor de zero* se existe $b \in R$ e $b \neq 0$ tal que $a \star b = 0$. Caso o anel comutativo não possuir nenhum divisor de zero, dizemos que o anel é um *domínio de integridade*.

2.4.1 Homomorfismo de Anéis

Definição 2.4.6: Considere $(R,*,\star)$ e $(S,\diamond,\Delta)$ anéis com suas respectivas operações. Seja uma aplicação $\phi : R \to S$ que relaciona elementos de R com elementos de S. Definimos um *homomorfismo* ϕ de $(R,*,\star)$ em $(S,\diamond,\Delta)$ quando ϕ satisfaz

$$(i) \quad \phi(a * b) = \phi(a) \diamond \phi(b)$$

$$(ii) \quad \phi(a \star b) = \phi(a) \Delta \phi(b)$$

Para todo a e b pertencentes a R.

Proposição 2.4.7: Sejam $(R,*,\star)$ e $(S,\diamond,\Delta)$ anéis com elemento neutro (Definição 2.4.3), sendo e_R e e_S os elementos neutros das operações $\star$ e Δ, respectivamente . Se existe um homomorfismo ϕ de $(R,*,\star)$ em $(S,\diamond,\Delta)$, então:

$$\phi(e_R) = e_S$$

Prova: Considere a' o inverso de a em relação a $\star$ e $(\phi(a))'$ o inverso de $\phi(a)$ em relação a Δ. Assim, se $a \in R$, temos que $a \star a' = e_R$ e, portanto, $a' \in R$. Temos:

$$\phi(a \star a') = \phi(a) \Delta \phi(a')$$
$$\phi(e_R) = \phi(a) \Delta \phi(a')$$

Sendo que $\phi(a), \phi(a') \in S$. Se acrescentarmos $(\phi(a))' \Delta \left(\phi(a')\right)'$ dos dois lados da igualdade, temos:

$$\phi(e_R) \Delta \left[(\phi(a))' \Delta \left(\phi(a')\right)'\right] = \phi(a) \Delta (\phi(a))' \Delta \phi(a') \Delta (\phi(a'))'$$
$$\phi(e_R) \Delta \left[\left(\phi(a)\right)' \Delta \left(\phi(a')\right)'\right] = e_S$$

Consideremos por absurdo que $\phi(e_R)$ e $\left[(\phi(a))' \Delta \left(\phi(a')\right)'\right]$ são inversos em relação a Δ. Se considerarmos um elemento $b \in R$, podemos, utilizando os mesmos passos acima, obter:

$$\phi(e_R) \Delta \left[\left(\phi(b)\right)' \Delta \left(\phi(b')\right)'\right] = e_S$$

Assim, $\phi(e_R)$ seria o inverso de $(\phi(x))' \Delta \left(\phi(x')\right)'$ para todo $x \in R$. Apesar de ficar claro que é um absurdo, não definimos o elemento inverso para Δ como característica necessária para a formação de um anel. De qualquer forma, tomemos $x = e_R$. Observe que o inverso de e_R é o próprio e_R, do qual segue:

$$\phi(e_R) \Delta [(\phi(e_R))' \Delta (\phi(e_R))'] = e_S$$

Como $(\phi(e_R))'$ é o inverso de $\phi(e_R)$, temos:

$$[\phi(e_R) \Delta (\phi(e_R))'] \Delta (\phi(e_R))' = e_S$$

$$(\phi(e_R))' = e_S$$
$$\phi(e_R) = e_S$$

■

Proposição 2.4.8: Pela proposição anterior (Proposição 2.4.7), temos que $\phi(a') = \big(\phi(a)\big)'$.

Prova: Na Proposição 2.4.7, provamos que $\phi(e_R) = e_S$. Retornando para a equação

$$\phi(e_R) = \phi(a) \vartriangle \phi(a')$$

Temos que:

$$\phi(a) \vartriangle \phi(a') = e_S$$

Que implica em

$$\phi(a') = \big(\phi(a)\big)'$$

■

2.5 Ideais

Definição 2.5.1: Considere um anel $(R,*,\star)$ e um subconjunto I de R. Se I corresponde a um subgrupo de R com relação à operação $*$, e se para quaisquer $x \in I$ e $r \in R$, temos $x \star r \in I$, então dizemos que I é dito um *ideal direto*. Se, do contrário, temos $r \star x \in I$, então dizemos que I é dito um ideal esquerdo.

Caso R for um anel comutativo, então é denominado apenas como *ideal*, uma vez que é tanto ideal direito como esquerdo. Chamamos também de *ideal próprio* aquele que é distinto do anel subjacente.

Definição 2.5.2: Define-se como *ideal primo* o ideal I se para $a \star b \in I$ e $a, b \in R$, então ou $a \in I$ ou $b \in I$.

2.6 Corpos

Definição 2.6.1: Considere um conjunto $\mathbb{F}$ dado que os grupos $(\mathbb{F},*)$ e $(\mathbb{F}^*,\star)$ sejam abelianos (Denição 2.3.2). Dizemos que $(\mathbb{F},*,\star)$ é um *corpo*.

Definição 2.6.2: Seja $\mathbb{Z}_p = \{0,1,2,\dots,p-1\}$. Desta forma, temos que $(\mathbb{Z}_p,+,\times)$ forma um corpo. Esse corpo é denotado por $\mathbb{F}_p$ e é denominado como *corpo de Galois de ordem p*.

Definição 2.6.3: Considere $\mathbb{L}$ um subconjunto do corpo $\mathbb{K}$. Se $\mathbb{L}$ for um corpo, preservando às mesmas operações de $\mathbb{K}$, dizemos que $\mathbb{L}$ é definido como *subcorpo* e $\mathbb{K}$ é denominado *corpo de extensão* em relação a $\mathbb{L}$.

Definição 2.6.4: Um corpo que não possui subcorpo é chamado de *corpo primo*.

2.7 Reticulados

Definição 2.7.1: Um reticulado é um conjunto de pontos no espaço n-dimensional com uma estrutura periódica. Dessa forma, é um subconjunto discreto de $\mathbb{R}^n$ sob a adição de vetores em $\mathbb{R}^n$. Seja $b_1, b_2, b_3, \dots, b_k$ de k vetores linearmente independentes em $\mathbb{R}^n$. Então, define-se o reticulado gerado por esses vetores como

$$L(b_1, b_2, b_3, \dots, b_k) = \left\{ \sum_{i=1}^{k} a_i b_i \,|a_i \in \mathbb{Z} \right\}$$

Por essa definição, $\{b_1, b_2, b_3, \dots, b_k\}$ forma uma base para esse reticulado, o qual tem dimensão k. Em outras palavras, é um espaço vetorial discretizado.

2.8 Curvas Elípticas

Definição 2.8.1: Uma curva elíptica $\mathbb{E}$, definida sobre um corpo $\mathbb{K}$, é um conjunto de pontos $P = (x, y)$ com $x, y \in \overline{\mathbb{K}}$ tais que $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, para $a_i \in \mathbb{K}$. Além disso, as derivadas parciais $2y + a_1x + a_3$ e $3x^2 + 2a_2x + a_4 + a_1y$ não assumem valor nulo simultaneamente, para um dado ponto $P = (x, y)$. Quando as derivadas são simultaneamente nulas, dizemos que a curva é não singular.

Definição 2.8.2: Seja $P \in \mathbb{E}/\mathbb{K}$ um ponto arbitrário de uma curva elíptica $\mathbb{E}$ definida sobre um corpo $\mathbb{K}$. Dado $n \in \mathbb{N}$, definimos *multiplicação escalar*, denotada como $[n]P$, correspondendo a soma de P com ele próprio n vezes.

2.9 Algoritmos

Algoritmo 2.9.1: (*Baby-step giant-step*) é um algoritmo utilizado para calcular o logaritmo discreto (Definição 2.2.12), vasculhando elementos intermediários (*algoritmo meet-in-the-middle*). Dado um grupo cíclico G de ordem n, tome um elemento α tal que α é gerador de G, ou seja, todo elemento β do grupo G pode ser escrito por $\beta \equiv \alpha^x$, com $x = \{1,2,3, \dots, n-1\}$.

O algoritmo é desenvolvido no processo de reescrita de x como $x = i.m + j$, sendo $m = \lceil \sqrt{n} \rceil$ e $0 \leq i, j < m$. Assim, temos que:

$$\beta(\alpha^{-m})^i \equiv \alpha^j$$

Desta forma, são pré-calculados diversos valores para α^j. Ajusta-se o valor de m e testam-se valores para i verificando a congruência em questão. Caso a congruência seja obtida, obtém-se x.

Capítulo 3
Criptografia Homomórfica

O conceito foi inicialmente apresentado por *Ronald Rivest*, *Leonard Adleman* e *M. L. Dertouzos* [$RAD78$] com a expressão homomorfismos privados (*privacy homomorphisms*), provavelmente observando o homomorfismo multiplicativo que envolvia o criptosistema RSA, criado pelos dois primeiros pesquisadores conjuntamente com o criptólogo *Adir Shamir*. Logo o conceito tornou-se fonte de pesquisa e responsável por grande expectativa na área, sendo diversos os trabalhos na busca de sua devida implementação.

3.1 Homomorfismo

Utilizando a Definição 2.3.6, considere dois grupos com suas respectivas operações binárias $(M,*)$ e $(C,\star)$. O conjunto M será o conjunto de todas as mensagens possíveis, em texto claro, que o criptosistema utiliza como entrada para sua cifração. O conjunto C, por sua vez, é o conjunto de todos os textos cifrados disponíveis para a decifração. As operações $*$ e $\star$ são definidas como operações lógicas quaisquer. Desejamos criar uma aplicação de cifração $E : M \to C$, de modo que $E(m) = c$ sendo $m \in M$ e $c \in C$. Ela será um *homomorfismo* se satisfizer a seguinte condição: Dadas $m_1, m_2 \in M$ duas mensagens quaisquer,

$$E(m_1 * m_2) = E(m_1) \star E(m_2)$$

Com $E(m_1), E(m_2), E(m_1 * m_2) \in C$. Note que, por meio desta propriedade, uma aplicação é realizada sobre as mensagens em texto claro sem qualquer necessidade de decifração do texto cifrado. O resultado será ainda cifrado, equivalente a mensagem original cifrada após a realização da operação desejada.

3.2 Criptosistemas Parcialmente Homomórficos

Preservamos a sigla ***HE*** uma vez que entendemos que todo criptosistema homomórfico estabelece inicialmente essa relação de parcialidade. No entanto, no meio acadêmico, pode ser encontrada a sigla ***PHE - Partially Homomorphic Encryption***. Veremos posteriormente que, com certa frequência, para se obter um criptosistema *FHE*, os criptólogos inicialmente adotam um criptosistema *PHE*, realizando as modificações necessárias para alcançar a completude do homomorfismo. Partimos do pressuposto de que todo criptosistema com essas características é parcialmente homomórfico e o representaremos como *HE* (além disso, *PHE* também refere-se a *parallel homomorphic encryption*, o que pode gerar confusão).

3.2.1 Unpadded RSA [RSA77]

Trata-se de um criptosistema que de chave pública no grupo $\mathbb{Z}_n$, tal que n é da forma pq, sendo p e q primos grandes.

Geração de Chaves: Um par de chaves de k bits é gerado da seguinte forma:
1. Geram-se primos grandes p e q com $k/2$ bits e obtemos $n = pq$.
2. Escolhe-se $e \in \mathbb{Z}_n^*$ de modo que exista $d \in \mathbb{Z}_n^*$ que obedeça a equação modular $e.d \equiv 1\ (mod\ \varphi(n))$.

Temos a chave pública (*public key*) $pk = (n, e)$ e a chave privada (*secret key*) $sk = (d)$.

Cifração: Para cifrar uma mensagem $m \in \mathbb{Z}_n$, temos:

$$E_{pk}(m) \equiv c \equiv m^e \ \ (mod\ n)$$

Decifração: Para decifrar uma mensagem $c \in \mathbb{Z}_n$, temos:

$$D_{sk}(c) \equiv m \equiv c^d \ \ (mod\ n)$$

Esquema: O algoritmo é dado por:

$$D_{sk}\left(E_{pk}(m)\right) \equiv D\left(m^e (\ mod\ n)\right) \equiv \left(m^e\ (mod\ n)\right)^d (mod\ n) \equiv m^{ed}\ \ (mod\ n)$$

Como $e.d \equiv 1\ (mod\ \varphi(n))$, podemos reescrever como $e.d = 1 + k.\varphi(n)$, com $k \in \mathbb{Z}$. Dessa forma, temos

$$D_{sk}\left(E_{pk}(m)\right) \equiv m^{ed} \equiv m^{1+k.\varphi(n)} \equiv m.m^{k.\varphi(n)}\ \ (mod\ n)$$

Utilizando o **Teorema 2.2.3**, temos

$$D_{sk}\left(E_{pk}(m)\right) \equiv m.\left(m^{\varphi(n)}\right)^k \equiv m.(1)^k \equiv m\ \ (mod\ n)$$

Homomorfismo: O homomorfismo multiplicativo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv {m_1}^e {m_2}^e \equiv (m_1 m_2)^e\ \ (mod\ n) \equiv E_{pk}(m_1.m_2)$$

Graduação: Podemos dizer, portanto, que o RSA é o início do nosso estudo, pois foi a partir dele que as propriedades sobre homomorfismo foram observadas e pela primeira vez descritas em um trabalho científico. No entanto, trata-se de um algoritmo extremamente frágil. Apesar de ser baseado na dificuldade de dois importantes elementos da teoria dos números: A dificuldade do logaritmo discreto e da fatoração dos naturais, o algoritmo tem caráter determinístico (o algoritmo sem a utilização de um *padding* adequado, por si só, é extremamente frágil).

3.2.2 Rabin [Rab79]

Trata-se de um criptosistema que de chave pública no grupo $\mathbb{Z}_n$, tal que n é da forma pq, sendo p e q primos grandes.

Geração de Chaves: Um par de chaves de k bits é gerado da seguinte forma:

1. Geram-se primos grandes p e q com $k/2$ bits tais que $p \equiv q \equiv 3 \pmod 4$ e obtemos $n = pq$.

Assim, temos $pk = (n)$ e a chave privada é o par $sk = (p, q)$.

Cifração: Para cifrar uma mensagem $m \in \mathbb{Z}_n$, temos:

$$E_{pk}(m) \equiv c \equiv m^2 \pmod n$$

Decifração: Para decifrar uma mensagem $c \in \mathbb{Z}_n$, aplicando o Teorema Chinês do Resto (**Teorema 2.2.8**), temos:

$$D_{sk}(c) \equiv m \equiv \sqrt{c} \pmod n$$

Esquema: O algoritmo é dado por:

$$D_{sk}\left(E_{pk}(m)\right) \equiv D\left(m^2 (\bmod n)\right) \equiv \sqrt{m^2 \pmod n} \pmod n \equiv \sqrt{m^2} \pmod n$$

Logo

$$D_{sk}\left(E_{pk}(m)\right) \equiv m \pmod n$$

Homomorfismo: O homomorfismo multiplicativo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv {m_1}^2{m_2}^2 \equiv (m_1 m_2)^2 \pmod n \equiv E_{pk}(m_1.m_2)$$

Graduação: Assim como o RSA, trata-se igualmente de um algoritmo extremamente frágil, pois também possui caráter determinístico. É baseado na dificuldade da fatoração dos naturais. Sua diferenciação se dá pela sua decifração gerar três resultados falsos, o que pode tornar impossível a sua utilização em um sistema homomórfico. Afinal, para eliminar a ambiguidade da decifração é necessário o acréscimo de algum tipo de *preenchimento* que precisa ser conhecido para sua correta decifração.

3.2.3 ElGamal [ElG84]

Trata-se de um criptosistema que de chave pública no grupo cíclico $\mathbb{Z}_p$, tal que p seja primo ímpar e $p - 1$ tenha um fator primo grande.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. De um grupo cíclico $\mathbb{Z}_p$ de ordem p é escolhido um gerador g.
2. Escolhe-se um valor aleatório $x \in \{1, \dots, q - 1\}$.

3. Calcula-se $h = g^x$.

Assim, temos $pk = (p, g, h)$ e a chave privada é $sk = (x)$.

Cifração: Para cifrar uma mensagem $m \in \mathbb{Z}_p^*$, temos:

1. Escolhe-se um valor aleatório $y \in \mathbb{Z}_{p-1}$.
2. Calcula-se:

$$c_1 \equiv g^y \ (mod\ p)\ e\ \ c_2 \equiv m.h^y \ (mod\ p)$$

Obtendo o texto cifrado $c = (c_1, c_2)$

Decifração: Para decifrar uma mensagem $c = (c_1, c_2)$, calcula-se:

$$D_{sk}(c) \equiv m \equiv c_2({c_1}^x)^{-1} \ (mod\ p)$$

Esquema: O algoritmo é dado por:

$$D_{sk}\left(E_{pk}(m)\right) \equiv m.h^y(g^{xy})^{-1} \equiv m.g^{xy}(g^{xy})^{-1} \equiv m \ (mod\ p)$$

Homomorfismo: O homomorfismo multiplicativo é dado por:

$$\begin{aligned} E_{pk}(m_1).E_{pk}(m_2) &\equiv (g^{x_1}, m_1.h^{x_1})(g^{x_2}, m_2.h^{x_2}) \ (mod\ p) \\ &\equiv (g^{x_1+x_2}, (m_1.m_2).h^{x_1+x_2}) \ (mod\ p) \equiv E_{pk}(m_1.m_2) \end{aligned}$$

Graduação: Diferentemente dos dois anteriores, o criptosistema ElGamal é probabilístico. O cuidado na escolha de um primo p e, principalmente, do gerador g é essencial para a segurança do algoritmo, pois o seu homomorfismo está relacionado à sua maleabilidade.

3.2.4 Goldwasser-Micali [GM82]

Trata-se de um criptosistema que de chave pública no grupo $\mathbb{Z}_n$

Geração de Chaves: Um par de chaves de k bits é gerado da seguinte forma:

1. Geram-se primos grandes p e q com $k/2$ bits e obtemos $n = pq$.
2. Obtêm-se algum x não resíduo quadrático (**Definições 2.2.8 a 2.2.11**) tal que

$$\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1 \Longrightarrow \left(\frac{x}{n}\right) = 1$$

Assim, temos a chave pública (x, n) e a chave privada é o par (p, q).

Cifração: Para cifrar uma mensagem $m = (m_1 m_2 m_3 \ldots m_t)$ com t bits, temos que, para cada m_i, gera-se um valor aleatório y_i pertencente à classe dos resíduos quadráticos módulo n, ou seja, de modo mais direto, $mdc(y_i, n) = 1$. Obtém-se:

$$E_{pk}(m_i) \equiv c_i \equiv x^{m_i}{y_i}^2 \ (mod\ n)$$

Gerando a mensagem cifrada $c = (c_1 c_2 c_3 \dots c_t)$.

Decifração: Para decifrar uma mensagem $c = (c_1 c_2 c_3 \dots c_t)$, para cada i, observa-se se c_i é um resíduo quadrático módulo n. Caso seja, então $m_i = 0$. Caso contrário, $m_i = 1$. Deste modo, obtém-se uma cadeia de bits como mensagem decifrada $m = (m_1 m_2 m_3 \dots m_t)$.

Esquema: O algoritmo é baseado na decisão se um determinado valor m é um resíduo quadrático módulo n. O cálculo só é possível conhecendo-se p e q, da seguinte forma

1. Calcula-se $m_p \equiv m \ (mod\ p)$ e $m_q \equiv m \ (mod\ q)$.
2. Utilizando o Teorema Chinês do Resto (**Teorema 2.2.6**), se $m_p^{\frac{(p-1)}{2}} \equiv 1 \ \ (mod\ p)$ e $m_q^{\frac{(q-1)}{2}} \equiv 1 \ \ (mod\ p)$, então m é resíduo quadrático módulo n.

Homomorfismo: O homomorfismo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv x^{m_1} {y_1}^2 x^{m_2} {y_2}^2 \equiv x^{(m_1+m_2)}(y_1 y_2)^2 \ (mod\ n) \equiv E_{pk}(m_1 \oplus m_2)$$

Graduação: Trata-se de um algoritmo mais robusto que o anterior, sendo igualmente probabilístico. Este algoritmo trabalha diretamente com um conjunto específico de aspectos de Teoria dos Números que lhe gera o nível de segurança esperado. O esquema consiste em obter uma análise dos símbolos de Lagrange, baseado no conhecimento prévio dos números primos p e q, podendo ser utilizados o Teorema Chinês do Resto e a Lei de Reciprocidade Quadrática para acelerar os cálculos envolvidos. É importante salientar que, graças à probabilidade envolvida, cada chave pública possui a mesma probabilidade de análise que as outras. Sua dificuldade está associada à necessidade de cifrar um bit por vez.

3.2.5 Benaloh [Ben94]

Trata-se de um criptosistema que de chave pública no grupo $\mathbb{Z}_n$.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Escolhe-se um tamanho de bloco r.
2. Geram-se primos grandes p e q tais que $r|p-1$, $mmc((p-1)/r\,, r) = 1$ e $\text{mm}c(q-1, r) = 1$.
3. Obtém-se $n = p.q$
4. Escolhe-se $y \in \mathbb{Z}_n^*$ tal que

$$y^{\frac{(p-1)(q-1)}{r}} \not\equiv 1 \ \ (mod\ n)$$

Assim, temos $pk = (y, n)$ e $sk = (p, q)$.

Cifração: Para cifrar uma mensagem m, escolhe-se um elemento $u \in \mathbb{Z}_n^*$. Assim,

$$E_{pk}(m) \equiv c \equiv y^m u^r \ (mod\ n)$$

Gerando a mensagem cifrada c.

Decifração: Para decifrar uma mensagem cifrada c, é necessário calcular $y' \equiv y^{\frac{(p-1).(q-1)}{r}} \ (mod\ n)$ e $c' \equiv c^{\frac{(p-1).(q-1)}{r}} \ (mod\ n)$. A partir disso, é apenas necessário calcular

$$c' \equiv y'^{m} \ (mod\ n)$$

Se r for pequeno, pode-se obter a mensagem m por testes sucessivos. Caso contrário, pode-se pré-calcular os valores possíveis utilizando o algoritmo *Baby-step giant-step* (**Algoritmo 2.9.1**). A decifração terá um gasto de tempo igual a $O(\sqrt{r})$.

Esquema: O algoritmo é baseado no *problema da residuosidade superior*. Esse problema está relacionado com a dificuldade, dado um valor $n = p.q$ com p e q primos desconhecidos e $a, d \in \mathbb{Z}_n^*$, de se obter x tal que:

$$x^d \equiv a \ (mod\ n)$$

Nesse sentido, a é dito *resíduo d-ário módulo n*. O problema é facilmente resolvido se p e q forem conhecidos, uma vez que a é um resíduo d-ário módulo p se, e somente se

$$a^{\frac{p-1}{d}} \equiv 1 \ (mod\ p)$$

Homomorfismo: O homomorfismo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv y^{m_1}{u_1}^r y^{m_2}{u_2}^r \equiv y^{(m_1+m_2)}(u_1u_2)^c \ (mod\ n) \equiv E_{pk}(m_1 + m_2)$$

Graduação: Trata-se de um algoritmo tão robusto quanto o anterior, uma vez que é baseado no criptosistema Goldwasser-Micali, apresentando igualmente segurança semântica. Enquanto o anterior apresentava a necessidade de cifrar um bit por vez, Benaloh permite a cifração de um bloco de tamanho r por vez.

3.2.6 Okamoto-Uchiyama [OU98]

Trata-se de um criptosistema de chave pública no grupo $\mathbb{Z}_n^*$, tal que n é da forma p^2q, sendo p e q primos grandes.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:
1. Geram-se primos grandes p e q e obtemos $n = p^2q$.
2. Escolhe-se $g \in (\mathbb{Z}/n\mathbb{Z})^*$ tal que $g^P \not\equiv 1 \ (mod\ p^2)$.
3. Calcula-se $h \equiv g^n \ (mod\ n)$.

Assim, temos $pk = (n, g, h)$ e $sk = (p, q)$.

Cifração: Para cifrar uma mensagem $m \in \mathbb{Z}_n^*$, temos:

$$E_{pk}(m) \equiv c \equiv g^m h^r \ (mod\ n)$$

Decifração: Definimos $L(k) = \frac{k+1}{p}$. A decifração é dada por

$$D_{sk}(c) \equiv m \equiv \frac{L\left(c^{p-1} \ (mod\ p^2)\right)}{L\left(g^{p-1} \ (mod\ p^2)\right)} \ (mod\ n)$$

Esquema: O grupo $\mathbb{Z}_n^* \approx \mathbb{Z}_{p^2}^* \times \mathbb{Z}_q^*$. Desta forma, o grupo $\mathbb{Z}_{p^2}^*$ tem um único subgrupo de ordem p. Será chamado por H. Dada a singularidade de H, temos que

$$H = \{x : x \equiv 1 \ (mod\ p)\}$$

Para qualquer elemento $x \in \mathbb{Z}_{p^2}^*$, temos que $x^{p-1}\ mod\ p^2 \in H$ desde que $p|(x^{p-1} - 1)$. O mapa L pode ser entendido como um logaritmo sobre o grupo cíclico H para o grupo aditivo $\mathbb{Z}_p$ e é fácil checar que $L(ab) = L(a) + L(b)$ e que L é um isomorfismo entre dois grupos. Como no caso do logaritmo usual, $\frac{L(x)}{L(g)}$ é, neste sentido, o logaritmo de x na base g.

Desta forma, temos

$$(g^m h^r)^{p-1} \equiv (g^m g^{nr})^{p-1} \equiv (g^{p-1})^m g^{p(p-1)rpq} \equiv (g^{p-1}) \ (mod\ p^2)$$

Portanto, para recuperar a mensagem m, é necessário obter o logaritmo com base g^{p-1}, que é obtido por

$$\frac{L((g^{p-1})^m)}{L(g^{p-1})} \equiv m \ (mod\ p)$$

Homomorfismo: O homomorfismo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv y^{m_1}{u_1}^r y^{m_2}{u_2}^r \equiv y^{(m_1+m_2)}(u_1u_2)^c \ (mod\ n) \equiv E_{pk}(m_1 + m_2)$$

Graduação: A segurança pode ser demonstrada como equivalente a fatoração de n. Trata-se de um criptosistema com segurança semântica baseado no p-subgrupo assumido, no qual pressupõe ser difícil determinar qual elemento $x \in \mathbb{Z}_p^*$ é um elemento do subgrupo de ordem p. É similar ao *problema da residuosidade superior* já citado anteriormente.

3.2.7 Naccache-Stern [NS98]

Trata-se de um criptosistema de chave pública no grupo $\mathbb{Z}_n^*$, tal que n é o produto de dois primos grandes.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Escolhe-se uma família de k pequenos e distintos primos $p_1, p_2, \dots, p_k$.
2. Divide-se em dois conjuntos meio a meio

$$u = \prod_{i=1}^{k/2} p_i \quad e \quad v = \prod_{i=k/2+1}^{k} p_i$$

3. Obtém-se

$$\sigma = uv = \prod_{i=1}^{k} p_i$$

4. Escolhe-se a e b primos grandes tais que $p = 2au + 1$ e $q = 2bv + 1$ sejam primos
5. Obtém-se $n = pq$.
6. Escolhe-se um elemento aleatório $g \in \mathbb{Z}_n$ tal que sua ordem seja $\varphi(n)/4$.

Desse modo, temos $pk = (\sigma, n, g)$ e $sk = (p, q)$.

Cifração: Considere a mensagem $m \in \mathbb{Z}_\sigma$. Escolhe-se $x \in \mathbb{Z}_n$ e calcula-se

$$E_{pk}(m) \equiv c \equiv x^\sigma g^m \pmod{n}$$

Decifração: Para decifração, é necessário encontrar $m_i \equiv m \pmod{p_i}$ para cada $1 \leq i \leq k$. Desta forma, é possível aplicar o Teorema Chinês do Resto (Teorema 2.2.6) e obter $m \pmod{\sigma}$. Dado um texto cifrado c, temos que

$$D_{sk}(c_i) \equiv c^{\varphi(n)/p_i} \pmod{n}, \text{ para } i = \{1, 2, \dots, k\}$$

Esquema: O algoritmo é dado por:

$$\begin{aligned} c_i &\equiv c^{\varphi(n)/p_i} \pmod{n}, \text{ para } i = \{1, 2, \dots, k\} \\ c^{\varphi(n)/p_i} &\equiv x^{\sigma\varphi(n)/p_i} x^{m\varphi(n)/p_i} \pmod{n} \\ &\equiv g^{(m_i + y_i p_i)\varphi(n)/p_i} \pmod{n} \\ &\equiv g^{m_i \varphi(n)/p_i} \pmod{n} \end{aligned}$$

sendo $m_i \equiv m \pmod{p_i}$. Escolhendo-se valores pequenos para p_i, m_i pode ser obtido por força bruta, comparando c_i com $g^{j\varphi(n)/p_i}$ para $j = \{1, \dots, p_i - 1\}$. Aplicando o Teorema Chinês do Resto, obtém-se $D_{sk}\big(E_{pk}(m)\big)m \pmod{\sigma}$.

Homomorfismo: O homomorfismo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) \equiv {x_1}^\sigma g^{m_1} {x_2}^\sigma g^{m_2} \equiv (x_1 x_2)^\sigma g^{(m_1 + m_2)} \pmod{n} \equiv E_{pk}(m_1 + m_2)$$

Graduação: Trata-se de um algoritmo baseado no *problema da residuosidade superior*. É uma generalização do criptosistema Benaloh, para o qual $k = 1$. O criptosistema probabilístico apresenta segurança semântica demonstrável.

3.2.8 Paillier [Pai99]

Trata-se de um criptosistema de chave pública, com caráter probabilístico assimétrico. É baseado no problema da residuosidade superior.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Escolhe-se dois primos grandes p e q independentes e aleatórios entre si de modo que $mdc(pq, p-1, q-1) = 1$. Esta propriedade garante que se os dois primos são de tamanhos equivalentes, ou seja, $p, q \in 1\|\{0,1\}^{s-1}$ para o parâmetro de segurança s.
2. Calcule $n = pq$ e $\lambda = mmc(p-1, q-1)$.
3. Escolhe-se um inteiro aleatório $g \in \mathbb{Z}_{n^2}^*$.
4. Garantir que $n|\#\langle g\rangle$ verificando a existência do inverso multiplicativo

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

sendo $L(u) = \frac{u-1}{n}$.

Assim, temos $pk = (n, g)$ e $sk = (\lambda, \mu)$. No caso de p e q com tamanhos equivalentes, uma variante mais simples para as etapas de geração de chaves seria $g = n + 1$, $\lambda = \varphi(n)$ e $\mu \equiv \varphi(n)^{-1} \pmod n$

Cifração: Para cifrar de uma mensagem $m \in \mathbb{Z}_n$, escolhe-se aleatoriamente $r \in \mathbb{Z}_n^*$. Calcula-se:

$$E_{pk}(m) \equiv c \equiv g^m r^n \pmod{n^2}$$

Decifração: Decifrando o texto cifrado $c \in \mathbb{Z}_{n^2}^*$, calcula-se

$$D_{sk}(c) \equiv m \equiv L\left(c^{\lambda} \pmod{n^2}\right).\mu \pmod n$$

Esquema: O algoritmo utiliza diversos teoremas. Um deles é o *teorema de Carmichael*, que afirma que, para todo elemento $\omega \in \mathbb{Z}_{n^2}^*$:

$$\begin{cases} \omega^{\lambda} \equiv 1 \pmod n \\ \omega^{n\lambda} \equiv 1 \pmod{n^2} \end{cases}$$

Aplicando o teorema acima:

$$D_{sk}\left(E_{pk}(m)\right) \equiv c^{\lambda} \equiv (g^m r^n)^{\lambda} \equiv g^{\lambda m} r^{\lambda n} \equiv g^{\lambda m} \pmod{n^2}$$

Utilizando a equação modular $(1+n)^x \equiv 1 + nx \pmod{n^2}$, temos:

$$g^{\lambda m} \equiv ((1+n)^{\alpha}\beta^n)^{\lambda m} \equiv (1+n)^{\alpha\lambda m}\beta^{n\lambda m} \equiv (1+n)^{\alpha\lambda m} \equiv 1 + \alpha\lambda mn \pmod{n^2}$$

Aplicando a função L na equação de decifração, temos:

$$\frac{L(1+\alpha\lambda mn)}{L(1+\alpha\lambda n)} \equiv \frac{\alpha\lambda m}{\alpha\lambda} \equiv m \pmod n$$

Homomorfismo: O homomorfismo aditivo é dado por:

$$E_{pk}(m_1, r_1).E_{pk}(m_2, r_2) \equiv g^{m_1}{r_1}^n g^{m_2}{r_2}^n \equiv g^{(m_1+m_2)}(r_1 r_2)^n \ (mod\ n^2) \equiv E_{pk}(m_1 + m_2)$$

ou

$$E_{pk}(m_1, r_1).g^{m_2} \equiv g^{m_1}{r_1}^n g^{m_2} \equiv g^{(m_1+m_2)}{r_2}^n \ (mod\ n^2) \equiv E_{pk}(m_1 + m_2)$$

O homomorfismo multiplicativo é dado por

$$\left(E_{pk}(m_1, r)\right)^{m_2} \equiv (g^{m_1} r^n)^{m_2} \equiv g^{(m_1 . m_2)} r^{n.m_2} \ (mod\ n^2) \equiv E_{pk}(m_1 . m_2)$$

ou, aplicando uma constante k

$$\left(E_{pk}(m, r)\right)^{k} \equiv (g^{m} r^n)^{k} \equiv g^{(km)} r^{nk} \ (mod\ n^2) \equiv E_{pk}(km)$$

Graduação: Trata-se do principal algoritmo de chave pública no qual se aplica homomorfismo à sua cifração, por apresentar uma quantidade maior de operações aplicáveis. É um algoritmo baseado em diversos teoremas, além de boa parte dos problemas matemáticos verificados até agora. Desta forma, o criptosistema oferece segurança semântica contra ataques de texto claro escolhido (IND-CPA). No entanto, por se tratar de um sistema maleável (essencial aos *HE*s), não oferece segurança contra ataques de texto cifrado escolhido (IND-CCA2). Após a inserção ao algoritmo de uma função *hash*, houve uma melhora significativa, sendo a segurança demonstrada segura na utilização de um modelo de oráculo aleatório. A sua superioridade em quantidade de operações aplicáveis é responsável pela sua referência em toda pesquisa, tornando-se um algoritmo extremamente versátil e, consequentemente, muito utilizado nos diversos protocolos que envolvem homomorfismo.

3.2.9 Damgård-Jurik [DJ01]

Trata-se de um criptosistema de chave pública, uma generalização do criptosistema Paillier.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Escolhe-se dois números primos grandes p e q, aleatórios e distintos.
2. Calcula-se $n = pq$ e $\lambda = mmc(p-1, q-1)$.
3. Escolhe-se um elemento $\mathbb{Z}^*_{n^{s+1}}$ de modo que $g \equiv (1+n)^j x \ (mod\ n^{s+1})$ para um j co-primo com n e $x \in H$.
4. Utilizando o Teorema Chinês do Resto, escolhe-se d de modo que $d \ mod\ n \in \mathbb{Z}^*_n$ e $d \equiv 0 \ (mod\ \lambda)$. Por exemplo, d poderia ser λ como no esquema original da criptografia Paillier.

Assim, $pk = (n, g)$ e $sk = (d)$.

Cifração: Para cifração de uma mensagem $m \in \mathbb{Z}_{n^s}$. Escolhe-se aleatoriamente $r \in \mathbb{Z}^*_{n^{s+1}}$ e calcula-se o texto cifrado:

$$E_{pk}(m) \equiv c \equiv g^m . r^{n^s} \ (mod\ n^{s+1})$$

Decifração: Para decifração de um texto cifrado $c \in \mathbb{Z}^*_{n^{s+1}}$, calcula-se

$$D_{sk}(c) \equiv c^d \ mod\ n^{s+1}$$

Sobre o resultado, aplica-se o mecanismo de decifração do criptosistema Paillier para obter a expressão jmd. Obtém-se o inverso multiplicativo módulo n^s de jd e calcula-se a mensagem original por:

$$D_{sk}(c) \equiv (jmd).(jd)^{-1} \equiv m \ \ (mod\ n^s)$$

Esquema: O algoritmo utiliza os mesmos parâmetros que o criptosistema Paillier. A única diferença refere-se ao fato de que, se c for um texto cifrado válido, da decifração segue:

$$c^d = \left(g^m r^{n^s}\right)^d = \left((1+n)^{jm} x^m r^{n^s}\right)^d = (1+n)^{jmd \ mod\ n^s}\left(x^m r^{n^s}\right)^{d \ mod\ \lambda}$$
$$= (1+n)^{jmd \ \ mod\ n^s}$$

Obtém-se jmd pelo mecanismo de decifração de Paillier. Uma forma de simplificar o algoritmo é definir $g = n+1$ e obter um valor para d tal que $d \equiv 1\ (mod\ n^s)$ e $d \equiv 0\ \ (mod\ \lambda\ )$. Desta forma, a decifração reduz-se a

$$c^d \equiv (1+n)^m \ \ (mod\ n^{s+1})$$

Homomorfismo: O homomorfismo obedece aos mesmos critérios do criptosistema Paillier. O homomorfismo aditivo é dado por:

$$E_{pk}(m_1, r_1).E_{pk}(m_2, r_2) \equiv g^{m_1} {r_1}^{n^s} g^{m_2} {r_2}^{n^s} \equiv g^{(m_1+m_2)}(r_1 r_2)^{n^s}\ (mod\ n^{s+1}) \equiv E_{pk}(m_1+m_2)$$

ou

$$E_{pk}(m_1, r_1).g^{m_2} \equiv g^{m_1} {r_1}^{n^s} g^{m_2} \equiv g^{(m_1+m_2)} {r_2}^{n^s}\ (mod\ n^{s+1}) \equiv E_{pk}(m_1+m_2)$$

O homomorfismo multiplicativo é dado por

$$\left(E_{pk}(m_1, r)\right)^{m_2} \equiv \left(g^{m_1} r^{n^s}\right)^{m_2} \equiv g^{(m_1.m_2)} r^{n^s.m_2}\ (mod\ n^{s+1}) \equiv E_{pk}(m_1.m_2)$$

ou, aplicando uma constante k

$$\left(E_{pk}(m, r)\right)^{k} \equiv \left(g^{m} r^{n^s}\right)^{k} \equiv g^{(km)} r^{n^s k}\ (mod\ n^{s+1}) \equiv E_{pk}(km)$$

Graduação: Trata-se de uma generalização do criptosistema Paillier. Desta forma, mantém as mesmas características de segurança que o anterior. A diferença consiste em uma sofisticação para demais casos, o que amplia a segurança do algoritmo.

3.2.10 McEliece [McE78]

Trata-se de um criptosistema de chave pública, com algoritmo probabilístico e decifração determinística.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Gera-se uma matriz geradora $G_{l\times n}$ do código de Goppa, do qual existe um algoritmo $C(\cdot)$ capaz de corrigir até t erros.
2. Gera-se uma matriz não-singular aleatória $S_{l\times l}$.
3. Gera-se uma matriz de permutação aleatória $T_{n\times n}$.
4. Determina-se $P = S.G.T$.

Assim, $pk = (P, t)$ e $sk = (S, G, P)$.

Cifração: Para cifração de uma mensagem m de tamanho l. Escolhe-se aleatoriamente e um vetor aleatório de tamanho n com peso de Hamming t e calcula-se o texto cifrado:

$$E_{pk}(m) = c = mP + e$$

Decifração: Para decifração de um texto cifrado c de tamanho n, calcula-se a matriz inversa T^{-1} e S^{-1} e calcula-se

$$D_{sk}(c) = C(cT^{-1}).S^{-1} = m$$

Esquema: O algoritmo é dado por:

$$cT^{-1} = (mS)G + eT^{-1}$$

Corrigindo com o algoritmo $C(\cdot)$, temos:

$$C(cT^{-1}) = C\left((mS)G + eT^{-1}\right) = m.S$$

Multiplicando pela matriz inversa S^{-1}

$$D_{sk}\left(E_{pk}(m)\right) = C(cT^{-1}).S^{-1} = m.S.S^{-1} = m$$

Homomorfismo: Considere $W_H(\cdot)$ uma função que define o peso de Hamming. Seja $W_H(e_1), W_H(e_2) \in \left(0, \frac{t}{2}\right]$, então $e_3 = e_1 + e_2$ e $W_H(e_3) \in (0, t]$. Assim o homomorfismo aditivo é dado por:

$$E_{pk}(m_1).E_{pk}(m_2) = (m_1P + e_1) + (m_1P + e_2) = (m_1 + m_2)P + e_3 = E_{pk}(m_1 + m_2)$$

Note que

$$E_{pk}\left(\sum_{i=1}^{k} m_i\right) = \sum_{i=1}^{k} E_{pk}(m_i) \quad \Leftrightarrow \quad \forall i \in \{0, \dots, k\}, W_H(e_i) \in \left(0, \frac{t}{k}\right]$$

O homomorfismo misto é dado por:

$$m_2.E_{pk}(m_1) = m_1m_2P + m_2e_1 = (m_1.m_2)P + e_2 = E_{pk}(m_1.m_2)$$

Se definirmos o produto de matrizes de mesma ordem como sendo $\left(r_{ij}\right)_{l\times n} = \left(a_{ij}\right)_{l\times n}.\left(b_{ij}\right)_{l\times n} = \left(a_{ij}.b_{ij}\right)_{l\times n}$, então podemos afirmar que $W_H(e_2) \in (0, t]$.

O criptosistema não permite o homomorfismo multiplicativo.

Graduação: Trata-se do primeiro esquema a utilizar a randomização no processo de cifração, o que o diferencia completamente de todos os outros criptosistemas até agora vistos. Sua consequência primordial é ser um criptosistema pós-quântico, ou seja, ser computacionalmente seguro contra um ataque quântico eficiente.

3.2.11 Boneh, Goh e Nissim [BGN05]

Trata-se de um criptosistema devido à Boneh, Goh e Nissim, sendo o primeiro a permitir ambas as operações sobre um bloco cifrado de tamanho fixo: adição e multiplicação.

Geração de Chaves: Um par de chaves é gerado da seguinte forma:

1. Escolhem-se dois grandes primos q, r e calcula-se $n = q.r$.
2. Encontra-se uma curva elíptica supersingular $E/\mathbb{F}_p$ com um ponto P de ordem n e seja $\mathbb{G} = \langle P \rangle$.
3. Escolhe-se $Q' \overset{R}{\leftarrow} \mathbb{G}\backslash\{\infty\}$ e defina $Q = [r]Q'$, então Q tem ordem q.
4. Seja $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mu_n \subset \mathbb{F}_{p^2}$ o emparelhamento modificado Weil (construído a partir do emparelhamento de Weil usando um mapa de distorção).

Assim, $pk = (E, \hat{e}, n, P, Q)$ e $sk = (q)$

Cifração: Para cifração, escolhe-se $t \overset{R}{\leftarrow} [1, n]$ e calcula-se o texto cifrado:

$$E_{pk}(m) = c = [m]P + [t]Q$$

Decifração: Para decifração, calcula-se $\tilde{P} = [q]P$ e $\tilde{c} = [q]c$, do qual temos:

$$D_{sk}(c) = \log_{\tilde{P}} \tilde{c} = m'$$

Note que:

$$c = [m]P + [t]Q \Rightarrow \tilde{c} = [mq]P + [qt]Q = [mq]P = [m]\tilde{P}$$

A decifração sobre será eficiente se o espaço de mensagens a serem codificadas for pequeno como na variante ElGamal.

Esquema: Uma das bases do criptosistema é a utilização de grupos de curvas elípticas cuja ordem é um número composto n que possui difícil fatoração, diferentemente de outros sistemas anteriores nos quais utilizamos grupos de ordem prima. Escolhem-se dois números primos secretos p e r e obtém-se $n = p.r$. Em seguida, encontra-se o menor inteiro l tal que $4ln - 1$ seja um primo p e seja E a curva elíptica $y^2 = x^3 + x$ sobre $\mathbb{F}_p$. Como $p \equiv 3\ (mod\ 4)$, a curva E é supersingular com $\#E(\mathbb{F}_p) = p + 1 = 4ln$. Dessa forma, existe um ponto $P \in E(\mathbb{F}_p)$ de ordem n, que pode ser obtido escolhendo-se um ponto aleatório $P' \in E(\mathbb{F}_p)$ e definindo $P = [4l]P'$.

Homomorfismo: Para o homomorfismo aditivo, escolha-se $t \overset{R}{\leftarrow} [1, n]$. Assim temos:

$$E_{pk}(m_1) + E_{pk}(m_2) = c' = c_1 + c_2 + [t']Q \in \mathbb{G}$$

Já o homomorfismo multiplicativo, escolha-se $u \overset{R}{\leftarrow} [1, n]$. Assim, temos:

$$E_{pk}(m_1) \times E_{pk}(m_2) = d = \hat{e}(c_1, c_2) \cdot e(Q, Q)^u \in \mu_n$$

É fácil notar que $c_1 = [m_1]P + [t_1]Q$ é a cifração de m_1 e que $c_2 = [m_2]P + [t_2]Q$ é a cifração de m_2, ou seja, a soma é uma cifração re-randorizada de $m_1 + m_2$. Com a mesma ideia, temos:

$$\begin{aligned} E_{pk}(m_1) \times E_{pk}(m_2) &= \hat{e}([m_1]P + [t_1]Q, [m_2]P + [t_2]Q) \cdot e(Q,Q)^u \\ &= \hat{e}(P,P)^{m_1 m_2} \hat{e}(P,Q)^{m_1 t_2 + m_2 t_1} \hat{e}(Q,Q)^{t_1 t_2 + u} \end{aligned}$$

Pela degeneração do emparelhamento, $g = \hat{e}(P,P)$ é um elemento de $\mathbb{F}_{p^2}$ de ordem n e $h = \hat{e}(Q,Q)$ é um elemento de $\mathbb{F}_{p^2}$ de ordem q. Além disso, $\hat{e}(P,Q)^q = \hat{e}(P,[q]Q) = 1$, então $\hat{e}(P,Q)$ tem ordem q do mesmo modo e é igual a h^a para algum a. Então, temos:

$$d = E_{pk}(m_1) \times E_{pk}(m_2) = g^{m_1 m_2} h^{u'}$$

Para algum u'. Desde que h tenha a ordem de q, elevar a potência q permite decifrar d. Dado $\tilde{g} = g^q$ e $\tilde{d} = d^q$, temos:

$$Dec_{sk}{}'(d) = \log_{\tilde{g}} \tilde{d}$$

Além disso, se tomarmos $u \overset{R}{\leftarrow} [1,n]$, temos:

$$d_1 +' d_2 = d' = d_1 . d_2 . h^u$$

Para obter uma cifração de m em μ_n, poderíamos cifrar m diretamente como $g^m h^u$ ou cifrar m em $\mathbb{G}$ usando Enc e um par de textos cifrados com uma cifração para 1. Note que o algoritmo de multiplicação move o texto cifrado de $E(\mathbb{F}_p)$ para μ_n. Desde que não haja emparelhamento em μ_n (ou de forma mais geral, em $\mathbb{F}_{p^2}^*$), não é possível realizar mais nenhuma multiplicação.

Em resumo, podemos cifrar mensagens no grupo $\mathbb{G}$ e usar algoritmos como $+$, $\times$ e $+'$ para avaliar qualquer polinômio quadrático ℓ-variável no m_i.

Graduação: Como dito anteriormente, trata-se do primeiro criptosistema a permitir ambas operações, no entanto, com a desvantagem de apenas uma única multiplicação pode ser realizada, apesar de a adição nada mais ser do que uma variante de ElGamal. A esta característica, que posteriormente explicaremos melhor, chamamos de criptosistema homomorficamente restrito ou homomorficamente limitado (*somewhat homomorphic*).

Capítulo 4
Aplicações

Desde 2009, diversas áreas identificaram a importância em adaptar seu funcionamento a esta inovadora ferramenta. Mesmo que para algumas seja apenas mais um elemento de segurança a contribuir, para outras, um criptosistema homomórfico pode ser a solução definitiva para os seus principais requisitos de segurança. Diversos são, inclusive, os trabalhos orientados para um único propósito, afinal cada evento exige uma funcionalidade diferenciada, um serviço específico. Entender as principais implicações de *HE* permite ao usuário identificar potencialidades específicas para a destinação desejada.

Nesse sentido, veremos inicialmente algumas propriedades inerentes à *HE* e, como consequência, identificaremos suas principais aplicações, uma vez que têm sido referência para pesquisas na área, além de observarmos o que é esperado da implementação de sistemas homomórficos idealmente seguros para cada caso.

4.1 Algumas Aplicações

Os principais objetivos ao seu utilizar *HE* já foram indiretamente apresentados e serão melhor observados nas aplicações abaixo. Nesse sentido, podemos descrever alguns deles como propriedades inerentes à aplicação dessa ferramenta em um criptosistema para facilitar a identificação de sua utilidade em alguma aplicação.

4.1.1 Prova de Conhecimento Nulo

Esta é uma das primitivas fundamentais de algumas premissas de diversos protocolos criptográficos. A prova do conhecimento nulo (*zero-knowledge proof*) é a capacidade de um sistema manter em segredo as informações privadas de um usuário, mesmo que seja necessário o seu compartilhamento. Assim, é garantido o segredo do conhecimento que se tinha a intenção de comunicar, sem qualquer conhecimento extra.

Podemos exemplificar a utilização de homomorfismo pelo seguinte protocolo. Considere que Alice deseja provar para Bob que existe uma string s tal que $f(s) = 1$, sendo $f(\,)$ uma função arbitrárias eficientemente computável. Dessa forma, seguimos os seguintes passos: Primeiramente, Alice envia para Bob $c = E_{pk}(s)$. Bob, ao receber c, calcula $c' = f(s) = E_{pk}(f(s))$. Por fim, Alice recebe e decifra c' e responde a Bob uma mensagem b que Bob verificará se $b = 1$. Esse protocolo necessita de alguns ajustes, mas demonstra como a utilização do homomorfismo facilita a aplicação dessa primitiva.

4.1.2 Computação em Nuvem

Cloud Computing é o termo que descreve o conjunto de mecanismos disponibilizados, normalmente pela internet, que simulam elementos computacionais presentes em um computador convencional, como armazenamento de memória ou unidades de processamento, só que de

forma distribuída pela rede. Em outras palavras, trata-se da abrangência de diversas funcionalidades que uma rede, como a internet, pode oferecer simulando um computador virtualizado, como um serviço (*as a service*).

Esse mecanismo permite que usuários e empresas possam adotar elementos mais vantajosos e mais baratos, com alta performance, descentralizando parte dos processos realizados por um computador ou um servidor. Mesmo computadores com uma configuração mais simples são capazes de disponibilizar ao seu usuário um desempenho superior.

Um dos maiores questionamentos da área deve-se à segurança dos serviços disponíveis. Algumas empresas de grande porte prontamente disponibilizaram servidores particulares para uma nuvem privada, normalmente construída sobre um *datacenter* próprio, acessados pela própria intranet da empresa, o que lhe garante certo nível de segurança. No entanto, usuários individuais e empresas de pequeno e médio porte, em muitos casos, não têm recursos para investir em uma nuvem privada. A contratação de serviços em uma nuvem pública lida com a desconfiança e falta de segurança que tanto os servidores como o canal pelo qual os dados serão transmitidos oferecem. Já era conhecida a necessidade de criptografia sobre quaisquer aplicações, mas a dúvida sobre criptosistemas idealmente seguros a serem utilizados para a nuvem foi o principal empecilho ao desenvolvimento e evolução desta área. Afinal, ninguém quer que seus dados sejam expostos publicamente, ainda mais por falha de uma empresa que realiza a oferta desse serviço.

Neste aspecto, a *HE* oferece uma funcionalidade realmente prática: Agora um computador cliente ou um servidor pode cifrar os dados que deseja transmitir e enviá-los. Ao receber esses dados, pelo criptosistema homomórfico escolhido, um conjunto de operações pode ser realizado sem a decifração dos dados e o resultado, ainda cifrado, é reenviado para o usuário original. Desta forma, a possibilidade de realizar computações arbitrárias diretamente sobre dados ainda cifrados sem a obrigatoriedade de revelar o seu conteúdo abre uma série de novas oportunidades sobre as funcionalidades dessa ferramenta. Portanto, a composição dos protocolos a serem utilizados é de extrema importância, sendo diferenciados pelo tipo de serviço desejado.

É importante lembrar que, para realizar as operações desejadas sobre os dados cifrados recebidos, o servidor deve possuir uma "máquina" interna para processamento. Poucos são os estudos que buscam um algoritmo capaz de verificar a integridade dos dados obtidos pelo resultado desse processamento, antes do reenvio para o usuário de origem. Caso esse servidor seja visto como uma "caixa preta" que realiza operações lógicas sobre os dados cifrados, o seu mau funcionamento será imperceptível até para o próprio usuário já que a mensagem compartilhada é somente a mensagem cifrada. Para tanto, é necessário ao protocolo o envio frequente de alguns pacotes com dados meramente aleatórios, mas com resultados já pré-calculados, para que após o processamento pelo servidor possam ser comparados com os textos recebidos. A análise detalhada desses pode identificar desde o mau funcionamento do servidor até a tentativa de adulteração de mensagens durante seu trânsito, permitindo igualmente a análise dos procedimentos a serem adotados, desde mero reenvio de uma sequencia referente a um intervalo em falha até a suspeita de que o servidor está comprometido.

4.1.3 Databases

O estudo relacionado às databases é o mesmo relacionado à *cloud computing*, só que os procedimentos se limitam principalmente à pesquisa. A ideia é permitir o usuário consultar informações contidas em seu banco de dados sem o conhecimento dos elementos pesquisados, de modo a promover certa segurança ao acesso a outras informações. Para isso, qualquer protocolo a ser gerado deve buscar o fato de o cliente receber os registros correspondentes à consulta solicitada sem que o servidor armazene ou identifique qualquer informação sobre a consulta. Caso contrário, um usuário malicioso que tem acesso passivo ao sistema identificaria os registros

não processados que logicamente não correspondem à consulta solicitada. Um exemplo típico é o fato de o servidor retornar ao cliente tantos dados quanto o número de registros no banco de dados. Em alguns casos, o servidor poderia identificar algumas informações sobre o número de registros que correspondem à consulta. Para manter a segurança neste quesito, o servidor de tamanho considerável é obrigado a realizar um conjunto de tarefas desnecessárias para não obter informações de registro, o que torna servidores lentos e, em alguns casos, impraticáveis. Além disso, as aplicações online são naturalmente vulneráveis ao vazamento de informações sensíveis, uma vez que, dada a disponibilidade para internet, qualquer adversário pode explorar as falhas de software ou até do próprio criptosistema adotado para obter acesso aos dados privados.

Para isso, o CryptDB é um sistema que fornece confidencialidade prática e demonstrável em detrimento a esses ataques para aplicações lastreadas em banco de dados. Ele funciona através da execução de consultas SQL sobre dados criptografados utilizando uma coleção de esquemas eficientes conscientes. Além disso, pode encadear as chaves de criptografia para as senhas dos usuários, de modo que um item de dados só pode ser decifrado usando a senha de um dos usuários com acesso aos dados. O servidor do banco de dados nunca acessaria os dados. E mesmo que ocorresse o comprometimento dos dados, um adversário seria incapaz de decifrar os dados, a não ser de obter informações durante o manuseio de um usuário habilitado.

A criptografia homomórfica, preferencialmente a *FHE*, é a base para confecção de um sistema ainda mais seguro, que implementa a segurança inclusive sobre o servidor curioso, além de inibir qualquer tipo de ataques ao servidor. Até mesmo porque, por esse método, o registro poderá ser decifrado na máquina do usuário que propôs a consulta.

4.1.4 Agentes Móveis

A segurança que envolve os agentes móveis sempre foi preocupação essencial para toda empresa que oferta o serviço, uma vez que todas essas funcionalidades são oferecidas por ondas de rádio. No Brasil, o padrão de transmissão é o GSM (*Global System for Mobile Communications*), apresentando bandas de 850 MHz, 900 MHz, 1800 MHz e 1900 MHz.

Um usuário que deseja atacar esse sistema provavelmente não terá capacidade de bloquear a chegada do sinal à antena da empresa. No entanto, além de claramente capturar todas as trocas de informações da rede, ele pode identificar fragilidades no processo de verificação de usuário e realizar a “clonagem” de usuários, realizando ligações que serão cobradas diretamente na conta de um usuário desconhecido. Estamos aqui especificando celulares, mas quaisquer sistemas de transmissão por rádio são suscetíveis a esses ataques, como serviços *wi-fi*, TVs por assinatura, dentre outros.

Um esquema de criptografia homomórfica em um grupo não-abeliano especial poderia levar a um sistema criptografia algebricamente homomórfico no campo finito $\mathbb{F}_2$, desta forma, com as operações binárias adequadas, o sistema criptográfico poderia oferecer a possibilidade de criptografar um programa completo ainda executável para rodar no agente móvel. Assim, teríamos a proteção em agentes móveis contra ataques maliciosos.

No ponto de vista das funções criptografadas, as funções permanecem secretas quando utilizados sistemas criptográficos homomórficos, garantindo privacidade. No caso dos dados, esquemas homomórficos oferecem a possibilidade de computar publicamente com dados cifrados, fazendo-os permanecer em segredo. Os dados cifrados seguem o mesmo princípio, sendo possível computar dados secretos de forma pública de modo a permanecer ainda em segredo.

4.1.5 Esquemas de Votação

Antes mesmo da demonstração apresentada por Gentry, diversos esquemas de votação utilizando criptosistemas homomórficos foram introduzidos. A busca por soluções criptográficas gerou a possibilidade de introdução de novos mecanismos de votação, alguns incluindo até mesmo a votação pela internet. Deve-se salientar que na maior parte dos países nos quais esses estudos ocorreram, o voto é facultativo, ou seja, a utilização de mecanismos que dificultem a chegada do eleitor até a máquina de votação acaba por ser responsável por uma parcela considerável de abstenções no dia de eleição.

A grande importância do homomorfismo nesses mecanismos é enorme contribuição sobre os métodos tradicionais, não apenas determinando aspectos de segurança, como também de transparência. Nesse sentido, um criptosistema homomórfico para votação tem como objetivo cinco aspectos importantes:

1. Privacidade do eleitor
   A privacidade mantém as informações de um eleitor privada dos outros usuários caso esse eleitor não deseja ser revelado. Esse aspecto é baseado nos princípios da privacidade e da ausência de recibo de votação (*receipt-freeness*), que é ainda superior a privacidade. Este princípio, descrito por Benaloh e Tuinstra em 1994, indica que qualquer eleitor é incapaz de gerar recibos de votação para provar para outros como votou, ou que ele pode gerar recibos falsos para adversários e os mesmos são incapazes de distinguir sua legitimidade. Dessa forma, o sistema eleitoral não vaza nenhuma informação mesmo se os eleitores forem coagidos. Por fim, um sistema de apresentar resistência à coerção (*coercion-resistance*), definição apresentada por Juels e Jakobsson em 2002, que compõem não somente a ausência de recibos de votação, mas também a defesa contra a randomização, a abstenção forçada e a ataques de simulação.
2. Verificabilidade
   Esse aspecto envolve a integridade, a exatidão, a comprovação individual e a verificabilidade pública. Integridade significa que cada eleitor tem direito a um único voto legítimo e que todos os votos válidos deverão ser computados ao total. Exatidão refere-se ao fato de que se os eleitores agirem corretamente, o resultado da eleição será correto. A comprovação individual garante um mapeamento correto de intenção dos eleitores com os votos recebidos e a verificabilidade pública é a combinação das propriedades citadas, gerando um mapeamento correto de intenções dos eleitores para o resultado final, podendo este procedimento ser verificado. Aqui temos um princípio da verificabilidade universal (*universal verifiability*) apresentado por Sako e Kilian em 1995, de descreve que qualquer um pode verificar que a eleição em questão é justa e o registro público é corretamente computado a partir das cédulas que foram corretamente utilizadas.
3. Confiabilidade
   Nesse aspecto temos os princípios da robustez e da justiça. Robustez refere-se à capacidade/resiliência do sistema eleitoral adotado suportar alguns erros causados por eleitores comuns, bem como o comportamento defeituoso causado por uma minoria das autoridades eleitorais. A justiça garante que nenhum resultado parcial será revelado antes do resultado final ser anunciado, durante o processo de votação.
4. Versatilidade
   O sistema eleitoral adotado deverá ser capaz de lidar com métodos eleitorais diferentes, comportando os mais diversos procedimentos e funcionalidades inerentes às necessidades eleitorais.

5. Usabilidade
   O sistema eleitoral será utilizado por eleitores comuns, pessoas que muitas vezes não possuem qualquer conhecimento especial. Além disso, o sistema deverá ser de fácil funcionamento para as autoridades eleitorais nos quesitos de configuração e controle, sem a necessidade de nenhum equipamento especial ou caro para isso. Sua execução deverá ser eficiente, mesmo que o número de candidatos ou eleitores seja escalonado.

Um fator interessante no cenário de uma eleição é que durante a contagem de votos, deseja-se apenas realizar a soma (contabilização) dos votos de cada eleitor, nem a necessidade de nenhuma outra operação lógica. Nesse sentido, a utilização de criptosistemas parcialmente homomórficos é suficiente para tornar a aplicação idealmente segura. Vejamos um exemplo utilizando o criptosistema Paillier [Pai99].

Considere, para nosso exemplo, que N_E eleitores deverão escolher entre N_C candidatos (logicamente $N_E \geq N_C$). Para computarmos os votos, definimos uma base b (esta poderá ser binária, decimal, hexadecimal, ...), que será usada na cifração dos votos, dada a condição de ser maior que o número de eleitores. Dessa forma, teremos que o primeiro candidato terá por referência b^0; o segundo, b^1; e assim por diante até o último candidato, ${N_C}^{th}$ eleitor, b^{N_C-1}. Consideremos, portanto, que temos $N_E = 8$ eleitores escolherão entre $N_C = 3$ candidatos. Como $b > N_E$, consideraremos o sistema de cifração como decimal. Considerando que cada eleitor pode escolher apenas um candidato, um exemplo de tabela de votação segue abaixo:

Tabela 4.1 – Exemplo de votação

Candidatos / Eleitores	Adams 10^0	Boston 10^1	Chicago 10^2	Voto que será cifrado
Alfa	×			$v = 1$
Bravo		×		$v = 10$
Charlie		×		$v = 10$
Delta	×			$v = 1$
Echo				$v = 0$
Foxtrot	×			$v = 1$
Golf			×	$v = 100$
Hotel	×			$v = 1$

Utilizando Paillier, consideremos $p = 73$ e $q = 97$, logo $n = p.q = 7081$ e $n^2 = 50140561$. Para o algoritmo de geração do par de chaves, utilizemos a função de Carmichael:

$$\lambda = \frac{(p-1)(q-1)}{mdc(p-1, q-1)} = 288$$

Escolhe-se um gerador $g \in \mathbb{Z}^*_{n^2}$ e

$$mdc\left(\frac{g^\lambda \ (mod\ n^2 - 1)}{n}, n\right) = 1 \Rightarrow g = 7082$$

E obtemos

$$\mu \equiv \left(L\left(g^\lambda \ mod\ n^2\right)\right)^{-1} \ (mod\ n) = 4106$$

A cifração será dada por

$$E_{pk}(m_i) \equiv c_i \equiv g^{m_i}.{r_i}^n \ \ (mod\ n^2)$$
$$c_i \equiv 7082^{m_i}.{r_i}^{7081} \ \ (mod\ 50140561)$$

Tabela 4.2 – Dados relacionados a cifração dos votos

Eleitores	Voto que será cifrado m_i	Elemento aleatório r_i	Voto cifrado c_i
Alfa	$v = 1$	493	47025010
Bravo	$v = 10$	2579	19555775
Charlie	$v = 10$	3324	8164480
Delta	$v = 1$	6657	23705018
Echo	$v = 0$	3901	704049
Foxtrot	$v = 1$	5646	31298744
Golf	$v = 100$	1931	4933896
Hotel	$v = 1$	1635	2339754

Para realizar a contabilização dos votos, é necessário calcular:

$$T \equiv \prod_{i=1}^{N_E} c_i \ \ (mod\ n^2)$$

$$T \equiv 27500995\ (mod\ 50140561)$$

Para decifração, temos:

$$m = L\left(c^{\lambda}\ (mod\ n^2)\right).\mu\ \ (mod\ n)$$
$$m = L\left(\frac{(27500995^{288}\ mod\ 50140561) - 1}{7081}\right).4106\ \ (mod\ 7081) = 124$$

Observe que a mensagem final recuperada é a soma de votos de cada candidato na ordem invertida dos mesmos.

Tabela 4.3 – Quantidade de votos por candidato

Candidatos	Adams	Boston	Chicago
Total	4	2	1

Algo que deve ser salientado é que, mesmo tendo eleitores realizando os mesmos votos (votando nos mesmos candidatos), é impossível pela utilização de um elemento aleatório r identificar quaisquer informações sobre a votação realizada por um eleitor. Caso a eleição fosse realizada por seções, é apenas necessário um novo produtório para obtenção imediata do total de votos por candidato.

4.1.6 Moedas Digitais

Também conhecidas por Moedas Virtuais, são criptomoedas cuja criação e transferência entre usuários é baseada em protocolos de código aberto de criptografia, realizados sem a existência de qualquer tipo de autoridade central. Assim, seu funcionamento se dá por banco de dados distribuídos entre diversos nós de uma rede *peer-to-peer* na qual colaboradores trabalham com criptografia para realização das funções elementares de segurança e procedimentos de verificação das transações realizadas entre os usuários, completamente alienada de qualquer instituição financeira.

Podemos analisá-las por meio de duas dessa moeda: Bitcoin e Litecoin

Figura 4.1 - Logotipos do Bitcoin e Litecoin, respectivamente
Fonte: *www.wikipedia.com*

O Bitcoin foi a primeira moeda virtual criada em 2008 por Satoshi Nakamoto e rapidamente tornou-se uma das preferências entre usuários da internet uma vez que, tratando-se de um sistema externo de qualquer agência financeira reguladora, os seus usuários não têm quaisquer obrigações legais referentes à moeda. Além disso, uma de suas funcionalidades, chamada de “mineração”, permite que um usuário ganhe um lote de bitcoins ao contribuir com a rede em razão da necessidade de processamento do sistema. Desta forma, usuários que rodem o programa na função geração de moedas podem receber um lote de no máximo 50 bitcoins.

Para se ter uma ideia da importância da moeda para eventos macroeconômicos globais, em janeiro de 2013, a moeda estava cotada em US$ 13,00/BTC. Já no final de novembro, a cotação chegou a US$ 1.207,00/BTC, principalmente pela crise financeira do Chipre, que deu maior respaldo à moeda.

O Litecoin é uma criptomoeda com as mesmas características que o Bitcoin, com poucas diferenças, sendo as principais: Realizar o processamento de cada bloco em 2,5 minutos ao invés dos 10 minutos necessários para processar o bloco de Bitcoin; e utilizar um algoritmo script com nível de segurança que impede que alguns programas terceiros possam realizar o processamento em um tempo inferior, eliminando quaisquer vantagens que um usuário poderia obter sobre um usuário que realiza o processamento coerente com a necessidade da rede. Com a proposta de alcançar o limite de 84 milhões de litecoins, 4 vezes o limite estipulado para o Bitcoin (21 milhões de bitcoins), o Litecoin tornou-se a segunda maior moeda virtual do mundo.

Um dos interesses sobre a evolução do protocolo utilizado para as moedas virtuais é que, ao invés da mera utilização de um único dispositivo, sejam necessários dois dispositivos em cooperação mútua para realização das operações financeiras, de modo que um não seja o único responsável pela informação. Dessa forma, o risco de perda/roubo de numerário virtual reduziria. Além disso, busca-se a implementação de um terceiro elemento na rede, para que este indivíduo

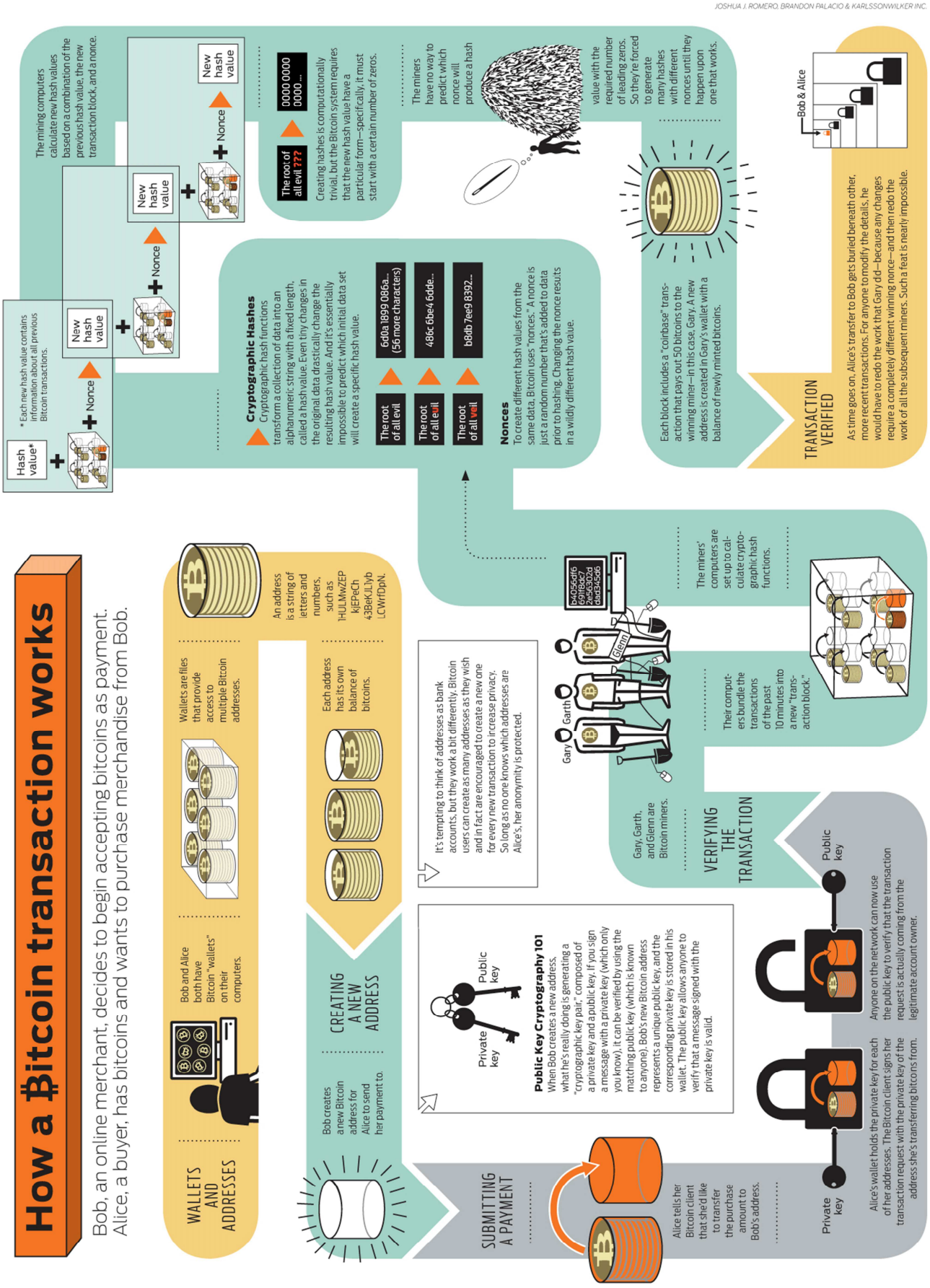

Figura 4.2 – Como funciona uma transação de Bitcoin
Fonte: *http://spectrum.ieee.org*

fosse responsável pela segurança no arquivamento das informações (em alguns trabalhos, é comparado a um cofre). *HE* aparece como forte candidata a sanar essas necessidades, além das diversas outras que possam surgir. O Bitcoin, por exemplo, utiliza um criptosistema de chave pública baseado em curvas elípticas. Com isso, seu protocolo pode ser mantido, sendo apenas adaptado um segundo protocolo de cooperação, que permita que os dados cifrados sejam transitados entre as máquinas, assinando de forma cooperativa as transações intencionadas pelo usuário.

4.1.7 Oblivious Transfer

Oblivious Transfer foi inicialmente introduzido com Rabin em 1981 como um protocolo de computação segura entre duas partes, na qual o emissor possui uma mensagem secreta que ele envia ao receptor, que a recebe com uma probabilidade de $1/2$, sem que o emissor saiba se a mensagem foi recebida ou não. Uma versão ligeiramente diferente desse protocolo tem sido usada atualmente chamada 1-out-of-2 Oblivious Transfer (OT_1^2).

A 1-out-of-2 Oblivious Transfer é uma primitiva criptográfica que permite um receptor obter 1 de 2 elementos mantidos por um emissor, sem obter nenhuma informação sobre o outro elemento e sem o emissor conhecer qual o elemento foi escolhido pelo receptor. O protocolo se baseia sobre o seguinte aspecto: Um emissor E escolhe duas strings de n bits X_0 e X_1 e o receptor R escolhe um bit b e recebe do emissor a strings X_C. O receptor não deve poder obter nenhuma informação sobre X_C, além do emissor não ser capaz de obter qualquer informação sobre o bit b.

Com a utilização de criptosistemas homomórficos, temos observado o surgimento de novos protocolos no âmbito de Oblivious Transfer, inclusive a primitiva criptográfica k-out-of-n ($\mathrm{OT}_{\mathrm{k}}^{\mathrm{n}}$). Desta forma, esses protocolos evoluem para computação multiparte e tornam-se cada vez mais robustos e seguros.

4.1.8 Redes Mistas

Redes Mistas (*mix-nets*) são protocolos que proporcionam o anonimato para os emissores através da coleta de mensagens cifradas de diversos usuários. Este protocolo foi introduzido por Chaum em 1981, descrevendo um protocolo multiparte rodado por um grupo de servidores mix para embaralhar elemento de modo que ninguém pudesse saber a permutação encadeada entre os elementos da entrada e da saída. Quando um emissor E quer enviar uma mensagem anônima a um receptor R de uma forma segura, ele cifra sucessivas vezes a mensagem e envia para um servidor mix que reordena os textos cifrados recebidos e os reenvia, até chegar ao receptor R. Desta forma, os servidores mix coletam todo o texto cifrado, e o reordenam de forma aleatória. Nesse cenário, a privacidade é obtida exigindo que a permutação seja segredo para todos exceto a rede mista.

Um princípio sempre buscado para esse tipo de protocolo é o embaralhamento do segredo verificável (*verifiable secret shuffle*). Um embaralhamento de textos cifrados $e_1, \dots, e_n$ nada mais é que um novo conjunto de textos cifrados $E_1, \dots, E_n$ com o mesmo conjunto de mensagens em texto plano original, só que com sua ordem permutada. A implementação de um criptosistema homomórfico de chave pública nesse caso pode ser compreendida como: Dada a chave pública pk, as mensagens m_1, m_2 e elementos aleatórios r_1, r_2, temos a propriedade:

$$E_{pk}(m_1 m_2; r_1 + r_2) = E_{pk}(m_1; r_1) E_{pk}(m_2; r_2)$$

Como estamos trabalhando com um *HE*, então podemos embaralhar $e_1, \dots, e_n$ selecionando a permutação $\pi \in \sum_n$ e elementos aleatórios $R_1, \dots, R_n$ e calculando

$$E_1 = e_{\pi(1)} E_{pk}(1; R_1)$$
$$E_2 = e_{\pi(2)} E_{pk}(1; R_2)$$
$$\dots$$
$$E_n = e_{\pi(n)} E_{pk}(1; R_n)$$

Se, ainda mais, o criptosistema for semanticamente seguro, expor $E_1, \dots, E_n$ não revela nada sobre a permutação. Ao ponto de que ninguém consegue verificar diretamente se o embaralhamento está correto ou incorreto. Um atacante poderia substituir alguns textos cifrados por outros, o que passaria despercebido. Nesse sentido, a utilização de provas de conhecimento nulo é necessária para correção do embaralhamento.

Esta propriedade, a re-encriptação, é essencial ao processo e responsável pelo dinamismo e segurança, por isso o interesse que seja oferecido diretamente através da utilização de sistemas homomórficos como blocos de construção.

Capítulo 5

Criptografia Completamente Homomórfica

FHE foi um dos diversos problemas em aberto da Criptografia Moderna. Durante 30 anos desde sua enunciação, poucos foram os estudos que trouxeram alguma expectativa de sua demonstração. Para ser mais exato, o melhor resultado obtido antes de 2009 foi o criptosistema Boneh-Goh-Nissim [BGN05] já citado anteriormente. Era um criptosistema que permitia um número ilimitado de adições, mas restringia-se a, no máximo, uma multiplicação. A partir de 2009, com a demonstração sobre a existência de *FHE*s, os estudos direcionaram pela busca de um criptosistema mais eficiente e melhor aplicabilidade em sistema em geral.

5.1 Homomorfismo completo

Utilizando a Definição 2.4.6, considere dois anéis com as mesmas operações binárias $(M,*,\star)$ e $(C,\diamond,\Delta)$. Assim como na demonstração do Capítulo 3, o conjunto M será o conjunto das mensagens em texto claro e o conjunto C, o conjunto de todos os textos cifrados possíveis. As operações $*$, $\star$, $\diamond$ e Δ são operações binárias quaisquer. Da mesma forma, criamos uma aplicação de cifração $E : M \to C$. Ela será um *homomorfismo de anéis* se satisfizer a seguinte condição: Dadas $m_1, m_2 \in M$ duas mensagens quaisquer,

$$E(m_1 * m_2) = E(m_1) \diamond E(m_2)$$
$$E(m_1 \star m_2) = E(m_1) \Delta E(m_2)$$

Com $E(m_1), E(m_2), E(m_1 * m_2), E(m_1 \star m_2) \in C$.

A melhoria sobre o *HE* deve-se a existência de uma quantidade superior de operações lógicas que podem ser realizadas sobre o texto cifrado. Como consequência, cada algoritmo baseia-se em uma quantidade considerável de novas abordagens matemáticas e computacionais que serão vistas abaixo, além de uma maior complexidade na obtenção dos circuitos lógicos.

5.2 Circuito Lógico em um *FHE*

De uma forma mais clara, a essência de um criptosistema completamente homomórfico baseia-se em sua capacidade em realizar cálculos diretamente sobre o texto cifrado que, após a devida decifração, seja identificado o resultado lógico esperado sobre o texto claro. Em outras palavras, o criptosistema mais robusto é aquele que possui uma quantidade maior de operações que possam ser aplicadas sobre suas mensagens codificadas que não interfiram sobre o mecanismo de cifração e decifração, e sim que influenciam somente sobre o texto que foi codificado. Para tanto, entender os mecanismos que envolvem as operações lógicas como um todo é tão importante para a adequação do criptosistema à aplicação requerida.

Podemos compreender um circuito lógico como sendo um grafo acíclico direcionado. Os nós são as portas lógicas e as arestas, os fios. Dependendo da natureza do circuito, a entrada pode ser formada por diversos tipos diferentes de valores, como inteiros ou binários e a

composição das portas correspondentes define a operação lógica a ser realizada sobre os dados. Dessa forma, o circuito $CL_\star$ é definido como um circuito e topologicamente organiza suas portas em níveis que serão executados sequencialmente.

O tamanho do circuito $CL_\star$ é dado pelo número de suas portas desconsiderando as de entrada. A profundidade é dada pelo comprimento do seu trajeto mais longo, a partir de uma porta de entrada para a porta de saída, do seu grafo subjacente direcionado.

Para as operações que envolvem o processamento em bloco, o tamanho deste bloco é diretamente proporcional nível de complexidade exigida pelo sistema. Por esse motivo, a busca por criptosistemas ideais envolve não somente a sua natureza homomórfica, mas a sua capacidade em realizar cálculos com a menor complexidade possível, sendo mais valorizado o processamento bit-a-bit. Apesar de mais lento do que o processamento em bloco, a complexidade exigida é extremamente simples, limitando-se a valores de saída entre valores 0 ou 1.

Em geral, busca-se obter em sistemas criptográficos completamente homomórficos duas operações importantes: a adição e a multiplicação. O motivo é simples: todo criptosistema deve ser capaz de estabelecer parâmetros necessários para implementação de uma porta NAND. A partir da descrição de uma porta NAND, é possível obter qualquer outra porta lógica e, desta forma, criar uma operação lógica desejada. Essa característica, a de ser capaz de recriar qualquer porta lógica, damos o nome de **completude funcional** (*function completeness*).

Caso o *FHE* em questão apresentar homomorfismo aditivo e multiplicativo, note que dada uma mensagem binária (processamento bit-a-bit) qualquer, sua cifração leva a dois possíveis valores possíveis e desconhecidos $E_{pk}(0)$ e $E_{pk}(1)$. Note que:

Tabela 5.1 – Homomorfismo aditivo

c_1	c_2	$c_1 + c_2$	$D_{sk}(c_1 + c_2)$
$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$	0
$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(1)$	1
$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(1)$	1
$E_{pk}(1)$	$E_{pk}(1)$	$E_{pk}(0)$	0

Tabela 5.2 – Homomorfismo multiplicativo

c_1	c_2	$c_1.c_2$	$D_{sk}(c_1.c_2)$
$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$	0
$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$	0
$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(0)$	0
$E_{pk}(1)$	$E_{pk}(1)$	$E_{pk}(1)$	1

As duas tabelas mostram que a operação aditiva implementa uma porta XOR e a operação multiplicativa implementa uma porta AND, respectivamente. Assim, podemos descrever uma porta NAND:

Tabela 5.3 – Implementando uma porta NAND

c_1	c_2	$c_1.c_2$	$c_1.c_2 + E_{pk}(1)$	$D_{sk}(c_1.c_2 + E_{pk}(1))$
$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(1)$	1
$E_{pk}(0)$	$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(1)$	1
$E_{pk}(1)$	$E_{pk}(0)$	$E_{pk}(0)$	$E_{pk}(1)$	1
$E_{pk}(1)$	$E_{pk}(1)$	$E_{pk}(1)$	$E_{pk}(0)$	0

Portanto, caso o *FHE* apresente homomorfismo aditivo e multiplicativo, podemos descrever uma porta lógica NAND com a qual é possível descrever qualquer operação lógica.

5.3 Padrão dos Criptosistemas

Basicamente, todo criptosistema completamente homomórfico de chave pública é formado por um esquema $\mathcal{E}$ composto por 4 algoritmos básicos $\mathcal{E}(KeyGen, Enc, Dec, Evaluate)$

1. $KeyGen$ ( ):
   É o algoritmo de geração de chaves. Esse algoritmo utiliza de aspectos pseudoaleatórios para gerar um par de chaves, sendo uma pública pk e uma privada sk.

2. Enc( ):
   O algoritmo de cifração utilizando a chave pública pk, que gera o texto cifrado c.

3. Dec( ):
   O algoritmo de decifração utilizando a chave privada sk, que recupera a mensagem em texto claro m.

4. $Evaluate$( ) ou $Eval$( )
   O algoritmo em questão recebe como entrada a chave pública pk, um circuito lógico CL do conjunto dos possíveis circuitos homomórficos do criptosistema e um vetor de mensagens cifradas $\langle c_1, \dots, c_t \rangle$ de modo que $c_i = Enc(pk, m_i)$. Desta forma, temos:

$$c = Evaluate(pk, CL_\star, c_1, \dots, c_t)$$

De modo que $c = E_{pk}[CL_\star(m_1, \dots, m_t)]$. Alguns criptosistemas podem apresentar um ou mais algoritmos além dos citados acima, mas os processos de cifração e decifração, além dos aspectos homomórficos envolvidos, seguem este padrão.

5.4 Definições e algoritmos específicos de *FHE*

Com novas abordagens matemáticas e computacionais, algumas definições e algoritmos tornam-se fundamentais para confecção de um *FHE*. Conhecer tanto a nomenclatura como a estratégia esperada de cada processo facilitam a compreensão de como o mecanismo é aproveitado durante a cifração homomórfica.

5.4.1 Corretude

Um esquema $\mathcal{E}(KeyGen, Enc, Dec, Evaluate)$ é dito **correto** se, dado um circuito lógico $CL_\star$ e um par de chaves (pk, sk), é válida a equação

$$D_{sk}\left(Evaluate_{E_{pk}}(pk, CL_\star, c_1, \dots, c_t)\right) = CL_\star(m_1, \dots, m_t)$$

Em outras palavras, um esquema $\mathcal{E}$ é dito correto para uma classe de circuitos lógicos $\mathcal{S}$, se for correto para cada $CL_\star \in \mathcal{S}$. Caso seja correto para todos os circuitos lógicos, então o esquema $\mathcal{E}$ é dito **completamente homomórfico**.

5.4.2 Auto-inicialização (*bootstrapping*)

Dizemos que um esquema $\mathcal{E}$ é d-homomórfico, para $d = d(n)$, se para qualquer profundidade d para um circuito lógico $CL_\star$ (sobre o campo finito $\mathbb{F}_2$) e qualquer conjunto de entradas $m_1, \dots, m_t$, é válido que

$$Pr[Dec_{sk}(Eval(CL_\star, c_1, \dots, c_t) \neq CL_\star(m_1, \dots, m_t)] = negl(n)$$

Conceitua-se também que um esquema homomórfico $\mathcal{E}$ é **compacto** se para todo circuito $CL_\star$, o tamanho do texto cifrado gerado pelo algoritmo $Evaluate(\,)$ é polinomial em relação ao parâmetro de segurança λ e independente do tamanho de $CL_\star$, ou seja, o esquema compacto é completamente homomórfico se este é d-homomórfico para qualquer polinômio de grau d.

Considere um esquema $\mathcal{E}$ que possui a capacidade de avaliar compactamente uma classe de circuitos lógicos $\mathcal{S}$, de modo que, dado um circuito $CL \in \mathcal{S}$, o esquema $\mathcal{E}$ é homomórfico em relação à CL. Dessa forma, existe um circuito CL', cuja estrutura lógica é idêntica à CL, mas aceita como entrada $Enc_{pk}(m)$ ao invés de m. Note que é possível criar um novo esquema $\mathcal{E}'$ a partir de $\mathcal{E}$, que seja capaz de avalizar seu próprio circuito de decifração. Para isso, definimos a variável d como a profundidade de um circuito. Utilizamos a simbologia $\mathcal{E}^{(d)}$ para representar o esquema capaz de avaliar compactamente circuitos com a profundidade máxima d.

Ainda no esquema em questão, considere que a decifração seja implementada por um circuito parametrizado somente pelo parâmetro de segurança λ. O conjunto formado por dois circuitos que recebem como entrada a chave privada sk e dois textos cifrados quaisquer c_1 e c_2 é chamado de **conjunto de circuitos de decifração aumentado**. Em geral, tem-se dois circuitos importantes: $D_{\mathcal{E}}^{(+)}$ e $D_{\mathcal{E}}^{(\times)}$. O primeiro decifra os textos cifrados e soma os resultados, enquanto o segundo também decifra, mas multiplica ao final.

Por fim, dado um esquema $\mathcal{E}$ homomórfico. Se $\mathcal{S}$ é o conjunto de todos os circuitos para os quais $\mathcal{E}$ é correto e $D_{\mathcal{E}} \subseteq \mathcal{S}$, sendo $D_{\mathcal{E}}$ um conjunto de circuitos aumentado, então definimos $\mathcal{E}$ como auto-inicializável.

A **auto-inicialização (*bootstrapping*)** é dada pelo seguinte teorema: Se existe um esquema d-homomórfico cuja profundidade do circuito de decifração é menor do que d, então existe um esquema de criptografia completamente homomórfico nivelado. Além disso, se tal esquema é homomórfico, então possui segurança circular, permanecendo seguro mesmo contra um adversário que tem acesso ao ciclo $E(pk_1, sk_2), E(pk_2, sk_3), \dots, E(pk_{n-1}, sk_n), E(pk_n, sk_1)$, pois não possui vantagem significativa em um jogo de segurança regular.

Esta ferramenta foi apresentada a primeira vez no trabalho de Gentry [$Gen09$]. Parte dos criptosistemas homomórficos tem uma necessidade crítica: que a profundidade dos circuitos seja conhecida e previamente fixada. O grau de criticidade aumenta dado o fato que o criptosistema deve identificar isso para qualquer circuito lógico.

5.4.3 Encriptação Homomórfica Restrita (*Somewhat*)

Também conhecida como Encriptação Homomórfica Limitada (*somewhat homomorphic encryption*) é um criptosistema capaz de realizar operações lógicas, como somas e multiplicações, sobre textos cifrados de maneira homomórfica, mas conforme as operações vão sendo realizadas, o texto cifrado resultante é deformado por um ruído. O algoritmo de decifração é capaz de obter o texto claro desde que o ruído não ultrapasse um certo limiar. Essa é a desvantagem desse tipo de algoritmo: limitar-se a uma quantidade de somas e multiplicações para não comprometer completamente o texto cifrado resultante, caso contrário, é impossível recuperar o texto claro inicialmente codificado.

Uma das práticas da grande maioria dos criptosistemas que utiliza essa técnica é a utilização de um algoritmo conhecido como **reencriptação** (*reencryption*). Trata-se de um algoritmo que recebe um texto cifrado com ruído grande, normalmente acima do limiar, e retorna um texto cifrado com ruído pequeno, abaixo do limiar, o que permite a decifração sem ambiguidade. Para tanto, é necessário um circuito de decifração de baixa profundidade.

5.5 Criptosistemas Completamente Homomórficos

Existem diversas formas de classificar os *FHE*s em geral. A classificação que será adotada nesse trabalho está relacionada com os recursos e algoritmos utilizados em comum. Deve-se ressaltar que, ao surgir um trabalho inovador na área, diversos outros trabalhos surgiram utilizando alguns dos mesmos recursos. Podemos distinguir os criptosistemas em gerações: Criptosistemas gerados (1) por Ideais, (2) por RLWE, (3) por mdc aproximado e (4) por NTRUEncrypt.

Quando falamos gerações, fugimos da noção de tampo e nos baseamos apenas nas ferramentas utilizadas para a confecção de *FHE*s. Apesar de citarmos diversos criptosistemas em cada geração, escolheremos um ou dois para darmos a noção de como funcionam, lembrando que os outros criptosistemas da mesma geração apresentam outras ferramentas, mas em essência estão baseados nos mesmos aspectos, inclusive sendo alguns a mera melhoria de alguns algoritmos de outros.

5.5.1 Criptosistemas gerados por Ideais

Esta geração, a primeira geração, iniciou-se com o trabalho de Craig Gentry e destacou-se por trabalhos ([Gen09], [Gen10], [SV10] e [SS10]) baseados em reticulados ideais, gerando criptosistemas homomorficamente limitados (*somewhat*), apoiados em auto-inicialização (*bootstrapping*) e na análise da profundidade do circuito de decifração.

5.5.1.1 Gentry [Gen09]

O esquema apresentado por Craig Gentry em 2009 surpreendeu a área de pesquisa por demonstrar ser possível um *FHE*. Trata-se de um criptosistema baseado em um ideal I e um anel R, gerando com um ruído e de modo que, dado algum $r \in R$, e é escolhido como elemento de I, tendo a forma $e = r.I$. Uma mensagem m é cifrada acrescentando-se o ruído $m + rI$ e decifrada livrando-se desse ruído por algoritmos que eliminavam o ideal I.

A ideia, portanto, é passar por três principais etapas: Primeiramente, construir um esquema de cifração baseado em reticulados ideais que são homomorficamente restritos (*somewhat*), que significa que está limitada a avaliar polinômios de baixo grau sobre os dados cifrados. Depois disso, realiza-se um esmagamento no circuito de decifração para torna-lo auto-inicializável (*bootstrapping*). Por fim, a auto-inicialização do esquema original ligeiramente aumentada produz o *FHE* desejado.

$KeyGen(\,)$:

O algoritmo de geração de chaves tem como entrada um anel fixo R, bem como uma base B_I de um pequeno ideal $I \trianglelefteq R$, que será usado para incorporar a mensagem dentro de vetor de erro. Além disso, acrescentasse um algoritmo $IdealGen(R, B_I)$ que é utilizado para resultar o par de chaves (pública e privada). A chave pública consiste em uma péssima base B_{pk} de um reticulado ideal J. A chave secreta consiste de uma ótima base de J definida como B_{sk} com vetores curtos e aproximadamente ortogonais. O reticulado ideal J é escolhido tal que $I + J = R$, ou seja, I e J são relativamente primos.

É importante salientar que se consideramos um vetor $\vec{v} \in R$ e B_J é uma base para um ideal $J \trianglelefteq R$, então o valor de $\vec{v}\ mod\ B_J$ é único e pode ser computado eficientemente (*efficiently-computable distinguished representative*). Além disso, $R\ mod\ B_J$ define o conjunto de representantes distintos para $\vec{r} + J$ sobre $\vec{r} \in R$, em relação a determinada base B_J de J.

$Enc(\,)$:

O algoritmo de cifração tem como entrada um vetor de mensagem em texto claro $\vec{m}$ assim como a chave pública B_{pk}. Observe que o espaço de mensagens de texto plano M é um subconjunto de $R\ mod\ B_I$. Acrescentando um algoritmo que experimenta vetores pequenos de classes laterais $\vec{m} + I$, esse resultado é modularmente reduzido à base pública B_{pk}:

$$\vec{c} \equiv \underbrace{\vec{m} + \vec{\imath}}_{\vec{e}}\ mod\ B_{pk}$$

$Dec(\,)$:

O algoritmo de decifração recebe como entrada o texto cifrado e a chave secreta B_{sk} e tem como saída a mensagem original em texto claro:

$$\vec{m} \equiv (\vec{c}\ mod\ B_{sk})\ mod\ B_I$$

$Evaluate(\,)$:

O algoritmo $Eval(\,)$ toma como entrada um circuito CL_* de alguns permitidos do conjunto $\mathcal{S}$ cujas portas lógicas executam operações B_I, a chave pública B_{pk} e um conjunto de textos cifrados $\overrightarrow{c_1}, \overrightarrow{c_2}, \overrightarrow{c_3}, \ldots \overrightarrow{c_t}$. Desta forma, temos o homomorfismo aditivo:

$$Evaluate_{\mathcal{E}}\left(B_{pk}, CL_+, \overrightarrow{c_1}, \overrightarrow{c_2}, \overrightarrow{c_3}, \ldots, \overrightarrow{c_t}\right) = \sum_{i=1}^{t} \overrightarrow{c_i}\ mod\ B_{pk}$$

e o homomorfismo multiplicativo:

$$Evaluate_{\mathcal{E}}\left(B_{pk}, CL_\times, \overrightarrow{c_1}, \overrightarrow{c_2}, \overrightarrow{c_3}, \ldots, \overrightarrow{c_t}\right) = \prod_{i=1}^{t} \overrightarrow{c_i}\ mod\ B_{pk}$$

A razão pela qual este esquema é limitadamente homomórfico deve-se ao fato de que os vetores de erro aumentam a cada operação Note que duas mensagens cifradas $\overrightarrow{c_1} = E(\overrightarrow{m_1}) = \overrightarrow{J_1} + \overrightarrow{e_1}$ e $\overrightarrow{c_2} = E(\overrightarrow{m_2}) = \overrightarrow{J_2} + \overrightarrow{e_2}$. No homomorfismo aditivo temos:

$$CL_+(\overrightarrow{c_1}, \overrightarrow{c_2}) = \overrightarrow{c_1} + \overrightarrow{c_2} = (\overrightarrow{J_1} + \overrightarrow{J_2}) + (\overrightarrow{e_1} + \overrightarrow{e_2})$$

Note que $\overrightarrow{J_1} + \overrightarrow{J_2} \in J$ e $\overrightarrow{e_1} + \overrightarrow{e_2}$ é um erro pequeno. Já o homomorfismo multiplicativo é dado por

$$CL_\times(\overrightarrow{c_1}, \overrightarrow{c_2}) = \overrightarrow{c_1} \times \overrightarrow{c_2} = (\overrightarrow{J_1} \times \overrightarrow{J_2} + \overrightarrow{J_1} \times \overrightarrow{e_2} + \overrightarrow{e_1} \times \overrightarrow{J_2}) + (\overrightarrow{e_1} \times \overrightarrow{e_2})$$

Note que $\overrightarrow{J_1} \times \overrightarrow{J_2} + \overrightarrow{J_1} \times \overrightarrow{e_2} + \overrightarrow{e_1} \times \overrightarrow{J_2} \in J$ e $\overrightarrow{e_1} \times \overrightarrow{e_2}$ também é pequeno. O erro proveniente da multiplicação tende a crescer exponencialmente, sendo capaz de desconfigurar o texto cifrado, impendido uma decifração sem ambiguação. Este problema será resolvido por um algoritmo adicional que atualiza o texto cifrado depois de cada operação. Este procedimento transforma o esquema de criptografia homomórfica restrita em um poderoso esquema complemente homomórfico. Gentry utilizou como base um conjunto de premissas rígidas sobre reticulados ideais, o que dificultou a melhoria do criptosistema por ser uma área de estudo ainda pouco conhecida. Além disso, o criptosistema utiliza o Problema da Soma de Subconjuntos, que é NP-completo, e uma etapa de esmagamento para reduzir a complexidade da decifração.

Esse trabalho, mesmo não sendo aplicável no meio real, foi importante por apresentar a existência de um criptosistema *FHE*, além de seguir uma linha diferenciada em se utilizar um esquema criptográfico *somewhat*, acrescido de um teorema *bootstrapping*. Esta linha passou a ser referência para boa parte dos criptosistemas da primeira geração.

5.5.2 Criptosistemas gerados por LWE e RLWE

Conhecida como a segunda geração, seu primeiro esquema foi apresentado no trabalho de Brakerski e Vaikuntanathan [BV11], que estabeleceu o *FHE* de maneira muito simples, baseada no pressuposto LWE (*learning with errors*). Utilizando reduções conhecidas [Reg05, Pei09], sua segurança é baseada na dificuldade (frequentemente quântica) da aproximação do problema do menor vetor em reticulados de pior caso.

Definimos o problema da seguinte forma: Considere um grupo aditivo $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. Considere também $A_{s,\chi}$ uma distribuição em $\mathbb{Z}_q^n \times \mathbb{T}$ obtido a partir de um vetor escolhido aleatoriamente $a \in \mathbb{Z}_q^n$. Escolhe-se e de acordo com uma distribuição de probabilidades χ em $\mathbb{T}$ e resultando em $(a, \langle a, s\rangle/q + e)$ para algum vetor $s \in \mathbb{Z}_q^n$ onde a divisão é realizada no campo dos reais e a adição em $\mathbb{T}$. O problema $\text{LWE}_{q,\chi}$ é encontrar $s \in \mathbb{Z}_q^n$, dado o acesso de muitas amostras polinomiais obtidas a partir de $A_{s,\chi}$. Esse problema possui uma nova versão, conhecida como problema da decisão LWE (DLWE) na qual o objetivo é distinguir entre produtos internos ruidosos e amostras uniformemente aleatórias de $\mathbb{Z}_q^n \times \mathbb{T}$. O RLWE é o mesmo aplicado a anéis (*rings*). A ideia é que, dado um anel R e dois polinômios $g, t \in R$, determinar se existem os polinômios s e e com pequenos coeficientes tais que $t = g.s + e$, ou se g e t foram escolhidos uniformemente de R.

Uma semelhança nos esquemas baseados em LWE é que os textos cifrados são representados por vetores em $\mathbb{Z}_q$, para algum módulo q. O processo de decifração é essencialmente um produto de computação interna do texto cifrado e o vetor da chave secreta, que produz uma versão ruidosa da mensagem. Como o ruído aumenta a cada operação

homomórfica, a decifração correta é garantida se a amplitude final do ruído estiver abaixo de $q/4$, lembrando que o homomorfismo aditivo aproximadamente duplica o erro, enquanto o homomorfismo multiplicativo aproximadamente eleva-o ao quadrado.

Diversos são os criptosistemas que fazem parte dessa geração. Podemos citar [BV11a, BV11b, BGV12, Bra12, FV12, GHPS12, AP13, GSW13] . Abaixo, apresentaremos um deles [BV11b].

No esquema [BV11b], após L níveis de multiplicação, o ruído cresce de uma amplitude inicial B para B^{2^L}, considerando, por exemplo, uma árvore de multiplicação de profundidade L. Dessa forma, para permitir a decifração correta, é necessário um grande módulo $q \approx B^{2^L}$. Isso afeta tanto a eficiência quanto a segurança do criptosistema (a segurança é inversamente proporcional a razão q/B, logo quanto maior for q para o mesmo B, menor será a segurança do sistema).

Esse esquema foi melhorado em [BGV12], que sugeriu reduzir a dimensão do vetor cifrado após cada multiplicação, o que foi chamado de comutação de módulos (*modulus switching*). Ou seja, a ideia é ir do vetor c sobre $\mathbb{Z}_q$ para o vetor c/w sobre $\mathbb{Z}_{q/w}$ (para algum fator escalonável w). Escalonando, reduzimos o módulo q para o módulo q/w, mas reduzimos o ruído pelo menos fator (de B para B/w). Após L níveis de multiplicação e escalonamento, o ruído ainda possui amplitude B, mas o módulo será q/B^L. Para tanto, é apenas suficiente definir $q \approx B^{L+1}$, que é significativamente menor que o módulo obtido no criptosistema anterior. No entanto, o processo resulta de uma sequência homomórfica complexa responsável pela "descida do módulo". Um fato interessante é que este criptosistema possui uma biblioteca disponível na internet para sua implementação, conhecida como HElib, na qual, além do criptosistema, existem algumas otimizações que aceleram consideravelmente o processo. Ela está implementada em C++ e disponível no endereço *https://github.com/shaih/HElib*.

5.5.2.1 Brakerski e Vaikuntanatham [$BV11$]

Trata-se de um criptosistema que utiliza uma forma diferenciada de gerenciar o ruído por módulo de comutação, baseado no problema LWE, utilizando a auto-inicialização para tornar-se definitivamente um *FHE*.

$KeyGen(\,)$:

Para gerarmos um par de chaves, considere um parâmetro de segurança λ. A partir dele, seja n um polinômio inteiro positivo em λ, k seja um polinômio inteiro positivo em n e um número ímpar q sub-exponencial em n. Considere também χ uma distribuição de ruído que produz pequenos valores. Desta forma, define-se a chave privada $s = (s[1], \dots, s[n]) \in \mathbb{Z}_q^n$ e uma chave pública $(A, v = As + 2e)$ sendo A uma matriz de ordem $k \times n$ escolhida uniformemente de $\mathbb{Z}_q^{k \times n}$ e e escolhido uniformemente de χ^k.

$Enc(\,)$:

Dada uma mensagem m binária, ou seja, $m \in \{0,1\}$, o algoritmo de cifração segue as seguintes etapas: Primeiramente, escolhe-se aleatoriamente $r \in \{0,1\}^k$. Após isso, computa-se $a = A^T r$ e $b = v^T r + m$, gerando como saída o par (a, b). Note que este elemento pertence à $\mathbb{Z}_q^{n+1}$ da mesma forma que a distribuição apresentada na definição de LWE (5.4.4).

$Dec(\,)$:

Para decifrar (a, b) , primeiramente computa-se $b' = b - \langle a, s\rangle = 2e + m \in \mathbb{Z}_q$ considerando e um ruído qualquer. E, por fim, obtém-se a mensagem original calculando $m \equiv b'\ (mod\ 2)$.

Esse esquema é possível uma vez que

$$\langle a, s\rangle = \sum_{i=1}^{n} a_i s_i = A^T r.s$$

Como $As = v - 2e$, temos que $A^T s = v^T - 2e^T$. Logo, temos:

$$\begin{aligned} b' = b - \langle a, s\rangle &= (v^T r + m) - (A^T r.s) \\ &= (v^T r + m) - (v^T r - 2e^T r) \\ &= 2e^T r + m \end{aligned}$$

$Evaluate(\,)$:

Dado o par cifrado (a, b), tal que $a = (a[1], \dots, a[n])$, considere uma função de avaliação linear $f_{a,b}: \mathbb{Z}_q^n \to \mathbb{Z}_q$ tal que para $x = (x[1], \dots, x[n])$

$$f_{a,b} = b - \langle a, x\rangle \equiv b - \sum_{i=1}^{n} a[i].x[i] \ \ (mod\ q)$$

Note que $m = f_{a,b}(s)\ (mod\ 2)$. Observe que essa função permite o homomorfismo aditivo e multiplicativo. Para o aditivo, temos:

$$\begin{aligned} f_{(a,b)}(x) + f_{(a',b')}(x) &= \left(b - \sum_{i=1}^{n} a[i].x[i]\right) + \left(b' - \sum_{i=1}^{n} a'[i].x[i]\right) \\ &= (b + b') - \sum_{i=1}^{n} (a[i] + a'[i]).x[i] \\ &= f_{(a+a',b+b')}(x) \end{aligned}$$

Já para o multiplicativo, temos:

$$\begin{aligned} f_{(a,b)}(x) \cdot f_{(a',b')}(x) &= \left(b - \sum_{i=1}^{n} a[i].x[i]\right) \cdot \left(b' - \sum_{i=1}^{n} a'[i].x[i]\right) \\ &= h_0 + \sum_{i=1}^{n} h_i.x[i] + \sum_{1 \le i \le j \le n} h_{i,j} \cdot x[i] \cdot x[j] \end{aligned}$$

Sendo $h_0 = bb'$, $h_i = -(ba'[i] + b'a[i])$ e $h_{i,j} = a[i]a'[j] + a[j]a'[i]$. Observe que o número de coeficientes é igual a $(n + 1)(n + 2)/2$, logo o texto cifrado é do tamanho equivalente ao quadrado do tamanho de s. Para isso, foi introduzido um método chamado de relinearização (*relinearization*). Considere a substituição da chave secreta $s = (s[1], \dots, s[n])$ pela nova chave secreta $t = (s[1], \dots, s[n], s[1]s[1], s[1]s[2], \dots, s[n]s[n])$. Então, $h_{a,b}(x) = f_{(a,b)}(x) \cdot f_{(a',b')}(x)$ torna-se linear em t. Consequentemente,

$$h_{a,b}(t) \equiv m \cdot m' \ (mod\ 2)$$

Para o esquema ser, de fato, um *FHE*, é necessário o tratamento do ruído. O método de comutação de módulo será melhor explicado no criptosistema seguinte, mas nada mais é do que substituir um texto cifrado $c \in \mathbb{Z}_q^n$ por um texto cifrado $c' \in \mathbb{Z}_p^n$, do qual obtemos pela decifração a mensagem original m.

5.5.3 Criptosistemas gerados por MDC Aproximado

Os criptosistemas dessa geração são baseados na dificuldade do problema do mdc aproximado (*approximate gdc*) é um problema que consiste em, para um conjunto de parâmetros (ρ, η, γ), dado um número polinomial de elementos da distribuição $\mathcal{D}_{\gamma,\rho}(p)$, para um inteiro ímpar p escolhido aleatoriamente, obtenha-se p.

Diversos algoritmos utilizaram essa ferramenta como fundamento e continuam em plena pesquisa [vDGHV10, CMNT11, CNT12, CLT13, KLYC13] . Demonstraremos o primeiro deles para compreender como é possível se obter *FHE* a partir desse problema.

5.5.3.1 van Dijk, Gentry, Halevi e Vaikuntanathan [vDGHV]

Para o esquema, definem-se quatro parâmetros (além do parâmetro de segurança λ): γ é o comprimento de bits dos inteiros na chave pública; η é o comprimento de bits da chave secreta; ρ é o comprimento de bits do ruído (é a distância entre os elementos da chave pública e os múltiplos mais próximos da chave privada); e r é o número de inteiros na chave pública. Esses parâmetros devem obedecer às seguintes restrições: $\rho = \omega(\log \lambda)$, para proteger o criptosistema de ataques de força bruta sobre o ruído; $\eta \geq \rho . \Theta(\lambda \log^2 \lambda)$, para dar suporte ao homomorfismo para a profundidade suficiente dos circuitos capaz de aplicar o "*squashed decryption circuit*"; $\gamma = \omega(\eta^2 \log \lambda)$ para impedir ataques baseados em reticulados sobre o problema do mdc aproximado; e $r \geq \gamma + \omega(\log \lambda)$ para a correta aplicação da redução para mdc aproximado. Nesse sentido, definimos a distribuição como

$$\mathcal{D}_{\gamma,\rho}(p) = \{\text{escolha } q \in \mathbb{Z} \cap [0, 2^\gamma/p), r \in (-2^p, 2^p) \text{: saída } x = pq + r\}$$

Desta forma, definimos o criptosistema:

$KeyGen(\ )$:

A chave secreta sk é um inteiro primo p de η bits tal que $p \in (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta)$. Para a chave pública pk, obtém-se $x_i \in \mathcal{D}_{\gamma,\rho}(p)$ para $i = 0, \dots, r$. Renomeia-se as amostras de modo que x_0 seja o maior. Reinicia-se o processo a menos que x_0 seja ímpar e $r_p(x_0)$ seja par. A chave pública é $pk = \langle x_0, x_1, \dots, x_r \rangle$.

$Enc(\ )$:

Dada uma mensagem $m \in \{0,1\}$, escolhe-se um subconjunto aleatório $S \subseteq \{1,2, \dots, r\}$ e um inteiro aleatório $r \in (-2^p, 2^p)$ e obtém-se:

$$E_{pk}(m) = c = \left[m + 2r + 2 \sum_{i \in S} x_i \right]_{x_0}$$

$Dec(\,)$:
Dado o texto cifrado c, obtém-se:

$$D_{sk}(c) = m' = (c \; mod \; p) \; mod \; 2$$

$Eval(\,)$:
Dado um circuito binário CL com k entradas e k textos cifrados c_i, aplicam-se portas de adição e multiplicação de CL para textos cifrados, realizando todas as operações sobre os inteiros e retornando um número também inteiro.

5.5.4 Criptosistemas gerados por NTRU

O criptosistema de chave pública NTRUEncrypt, ou simplesmente NTRU, é um criptosistema baseado em reticulados e no problema do menor vetor (*shortest vector problem*). Esse algoritmo foi desenvolvido em 1996 por Jeffrey Hoffstein, Jill Pipher e Joseph H. Silverman. Inicialmente, a intenção era ser uma alternativa para o RSA, por exemplo, mas acabou esquecido por ser frágil a ataques quânticos.

É tipicamente descrito como um criptosistema de base polinomial envolvendo reduções em reticulados. Suas operações são baseadas em objetos em um anel de polinômios truncados $R = \mathbb{Z}_q[X]/(X^N - 1)$. Ele depende de três parâmetros de segurança (N, p, q) e de quatro conjuntos $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r$ e $\mathcal{L}_m$ de polinômios de grau $N - 1$ com pequenos coeficientes inteiros. A adição de dois polinômios é definida como a soma emparelhada de coeficientes de mesmo grau e a multiplicação é definida como produto de convolução.

Dois trabalhos se destacaram nesta geração [LTV12, BLLN13] que não ganhou mais força pela dúvida da comunidade científica sobre sua resiliência a diversos ataques diferentes.

Capítulo 6
Conclusão

Vimos neste trabalho que o conhecimento da essência sobre a Criptografia Homomórfica tem papel fundamental para o futuro da Criptografia Moderna, uma vez que apresenta uma inovadora ferramenta que possui um enorme diferencial na escolha de criptosistemas idealmente seguros.

Identificamos diversos criptosistemas, tanto parcialmente como completamente homomórficos, que apresentam níveis diferenciados de segurança e aplicabilidade. Alguns dos principais estão surgindo com um conjunto de importantes mecanismos, tanto matemáticos como computacionais, até então desconhecidos ou simplesmente não reconhecidos. Um exemplo que citamos é o próprio trabalho de Gentry. O seu trabalho introduziu a construção de um esquema de cifração que é limitadamente homomórfico, além de simplificar a função de decifração com técnicas de “esmagamento” e a introdução da auto-inicialização para utilização do homomorfismo esperado. Como dito anteriormente, apesar de ser impraticável, este trabalho ganhou importância não apenas por apresentar a demonstração de um problema em aberto. Um dos principais motivos do grande reconhecimento é a introdução de novos mecanismos que permitiram a adaptação de tantos outros criptosistemas na obtenção de um *FHE*.

É importante salientar igualmente que diversos trabalhos iniciaram gerações de *FHE*s. Com isso, queremos ressaltar a necessidade do indivíduo que estuda e busca criar criptosistemas homomórficos diferenciar sua pesquisa na procura de problemas matemáticos com visam tanto o aumento da dificuldade computacional como a simplificação de processos demasiadamente demorados. Lembramos que, apesar dos trabalhos serem principalmente teóricos, muitos são disponibilizados por bibliotecas pela internet com otimizações e mecanismos que tornem o processo de cifração mais ágil e aplicável às necessidades individuais.

Seja pelo conhecimento dos criptosistemas homomórficos e suas aplicações, seja pela possibilidade da implementação ou melhora de criptosistemas completamente homomórficos, todo estudo nesta direção possui extrema relevância para o contexto da criptografia em geral.

Referências Bibliográficas

[AP13] Jacob Alperin-Sheriff and Chris Peikert: *Practical Bootstrapping in Quasilinear time*. CRYPTO 2013.

[Ben94] Josh Benaloh. *Dense Probabilistic Encryption*. 1994.

[BGN05] Dan Boneh, Eu-Jin Goh, Kobbi Nissim: *Evaluating 2-DNF Formulas on Ciphertexts*. TCC 2005.

[BGV12] Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan: *(Leveled) fully homomorphic encryption without bootstrapping*. ITCS 2012.

[BLLN13] Joppe W. Bos and Kristin Lauter and Jake Loftus and Michael Naehrig: *Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme*, 2013.

[Bra12] Zvika Brakerski: *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP*. CRYPTO 2012.

[BV11a] Zvika Brakerski, Vinod Vaikuntanathan: *Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages*. CRYPTO 2011.

[BV11b] Zvika Brakerski, Vinod Vaikuntanathan: *Efficient Fully Homomorphic Encryption from (Standard) LWE*. FOCS 2011.

[CLT13] Jean-Sébastien Coron and Tancrède Lepoint and Mehdi Tibouchi: *Batch Fully Homomorphic Encryption over the Integers*, 2013.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, Mehdi Tibouchi: *Fully Homomorphic Encryption over the Integers with Shorter Public Keys*, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi: *Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers*. EUROCRYPT 2012.

[DJ01] Ivan Damgård, Mads Jurik*: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System.* 2001.

[ElG84] Taher ElGamal. *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. CRYPTO 1984.

[FV12] Junfeng Fan and Frederik Vercauteren: *Somewhat Practical Fully Homomorphic Encryption*, 2012.

[Gen09] Craig Gentry: *Fully homomorphic encryption using ideal lattices*. STOC 2009.

[Gen10] Craig Gentry: *Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness*. CRYPTO 2010.

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart: *Ring Switching in BGV-Style Homomorphic Encryption*. SCN 2012.

[GM82] S. Goldwasser, S. Micali. *Probabilistic encryption and how to play mental poker keeping secret all partial information.* 14th Symposium on Theory of Computing, 1982.

[GSW13] Craig Gentry and Amit Sahai and Brent Waters: *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based*. CRYPTO 2013.

[KLY13] Jinsu Kim, Moon Sung Lee, Aaram Yun, Jung Hee Cheon: *CRT-based Fully Homomorphic Encryption over the Integers*, 2013.

[LTV12] Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan: *On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption*. STOC 2012.

[McE78] R. J. McEliece. *A Public-Key Cryptosystem Based On Algebraic Coding Theory*. 1978.

[NS98] David Naccache and Jacques Stern. *A New Public Key Cryptosystem Based on Higher Residues*. 1998.

[OU98] Okamoto, Tatsuaki; Uchiyama, Shigenori. *A new public-key cryptosystem as secure as factoring.* EUROCRYPT 1998.

[Pai99] Paillier, Pascal. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.* EUROCRYPT 1999.

[Pei09] C. Peikert. *Public-key cryptosystems from the worst-case shortest vector problem*. STOC, 2009.

[Rab79] Rabin, Michael. *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. MIT Laboratory for Computer Science, 1979.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. *On data banks and privacy homomorphisms.* Foundations of Secure Computation. Academic Press, 1978.

[Reg05] Oded Regev. *On lattices, learning with errors, random linear codes, and cryptography*. STOC, 2005.

[RSA77] Rivest, R.; A. Shamir; L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, 1977.

[SS10] Damien Stehlé and Ron Steinfeld: *Faster Fully Homomorphic Encryption*. ASIACRYPT 2010.

[SV10] N. P. Smart and F. Vercauteren: *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*. PKC 2010.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan: *Fully Homomorphic Encryption over the Integers*. EUROCRYPT 2010