Advances in Science, Technology and Engineering Systems Journal
Vol. 4, No. 5, 30-38 (2019)
www.astesj.com
Special Issue on Advancement in Engineering and Computer Science

ASTES Journal
ISSN: 2415-6698

Fully Homomorphic Encryption Scheme Based On Complex Numbers

Khalil Hariss*[1,2], Maroun Chamoun[1], Abed Ellatif Samhat[2]

[1] *Université Saint Joseph, ESIB, CIMTI, Mar Roukoz, Lebanon*

[2] *Lebanese University, Faculty of Engineering, CRSI, Hadath, Lebanon*

ARTICLE INFO

Article history:
Received: 15 June, 2019
Accepted: 20 August, 2019
Online: 06 September, 2019

Keywords:
Complex Numbers
Cloud Systems
Fully Homomorphic
Asymmetric Scheme
Bootstrapping
Approximate GCD

ABSTRACT

In this paper, we present a new Somewhat Homomorphic Encryption (SHE) scheme using computation over complex numbers. We then use Bootstrapping technique to make the scheme Fully Homomorphic (FH) and supports unbounded number of circuit depth. In addition to its homomorphic properties and security level, a main characteristic of the proposed new scheme is its simplicity as it is merely based on addition and multiplication operations over complex numbers. The new scheme is implemented under Python using SAGEMath library and evaluated. Then a crypt-analysis based on Approximate GCD problem is done. A comparison with the BGV, a well known Fully Homomorphic Encryption (FHE) scheme, shows that this new scheme is an efficient homomorphic encryption scheme.

1 Introduction

An encryption scheme is said to be FH, if it allows to perform addition and multiplication operations explicitly over the cipher-texts while performing implicitly the same operations over the plain-texts. This new type of ciphering is very required in cloud systems mainly when users treat the cloud as a third untrusted party. With Homomorphic Encryption (HE) the cloud is able to process over encrypted storage and the privacy of the user is preserved at the cloud side. In Figure. 1, we show the case of a mobile user sending an encrypted query to the cloud using HE, thus the cloud is able to process encrypted query over encrypted data to return an encrypted answer. The user can do the decryption.

The notion of homomorphism was first introduced by Rivest, Adleman and Dertouzo [1] as privacy homomorphism after the invention of RSA crypto-system [2]. The basic RSA crypto-system is a multiplicative homomorphic scheme that allows to perform multiplication operations over the cipher-texts. Give, for example, RSA public key, $p_k = (N, e)$ and cipher $c_i = (m_i)^e mod(N)$ for a plain-text m_i. It is simple to demonstrate that $\prod_i c_i = (\prod_i m_i)^e mod(N)$. Several works followed the RSA scheme. A state of art of HE is given in [3] – [6]. Some of the known homomorphic schemes are: Paillier crypto-system [7, 8], Domingo Ferrer crypto-system [9, 10] which is a HE scheme based on polynomial calculations. The Enhanced MORE [11] is another FHE based on matrix calculation and the NOHE [12] scheme is a lightweight FHE that profits from the simplicity and the homomorphic behavior of logic NOT. The most valued work in this domain was given by Craig Gentry in his Ph.D. thesis in 2009 [13, 14]. The work of Gentry was inspired from lattice based cryptography. Gentry first introduced a SHE scheme that supports a bounded number of operations over the cipher-texts. Then he developed a new refresh mechanism called Bootstrapping [15] – [19] to make the scheme FH and supports unbounded operations. An important FHE scheme is the BGV crypto-system developed by Brakerski, Gentry and Vaikuntanathan in [20, 21]. Since BGV scheme is also Somewhat Homomorphic (SH), Modulus Switching (MS) is a technique used with BGV to extend the circuit's depth during the evaluation procedure.

In this paper, our main motivation is to add a value in homomorphic encryption by designing a new simple and robust FHE based on simple complex numbers operations . We first build a new asymmetric SHE complex-based scheme, then we apply Gentry refresh mechanism (Bootstrapping). In comparison with BGV which is lattice based encryption, our scheme is simply based on complex addition and multiplication operations. In addition, Bootstrapping supports unbounded number of circuit depth in our scheme, while MS extends the BGV evaluation to a limited number.

The rest of the paper is organized as follows: in section 2, we present the basic concept of HE with the properties that can lead to a HE scheme. In section 3, we introduce our new FHE scheme built using complex numbers theory. In section 4 we consider our implementations for the new Complex-based scheme and the

*Corresponding Author: Khalil Hariss, Université Saint Joseph, khalil.hariss@net.usj.edu.lb

https://dx.doi.org/10.25046/aj040504

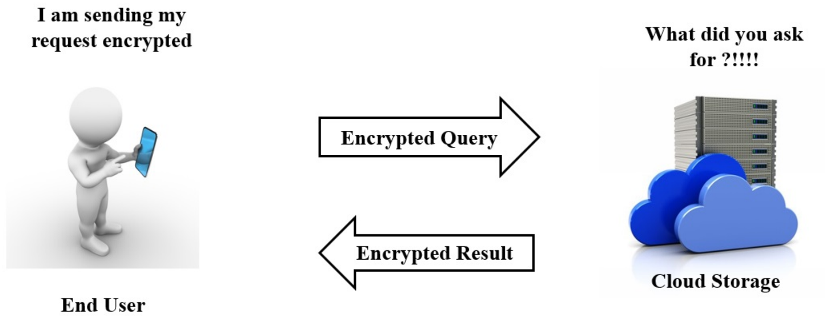

Figure 1: Cloud Scenarion Under Homomorphic Encryption

BGV scheme under Python using SAGEMath Library, followed by a crypt-analysis of our new scheme based on the Approximate GCD problem [22, 23, 24]. And finally the conclusion, comparison between the two schemes and future works are listed in section 5.

2 Homomorphic Encryption

In this work, we focus on asymmetric schemes. Consider an asymmetric scheme α, where p_k is the public key, s_k is the secret key, c_i is the ciphered text of a plain-text m_i. A homomorphic scheme has 3 different basic functions that are: $KeyGen_\alpha$ that generates (p_k, s_k), $Encrypt_\alpha(p_k, m_i)$ that outputs cipher-text c_i, $Decrypt_\alpha(s_k, c_i)$ that outputs the plain-text m_i.

2.1 Evaluation Function

In addition to the three basic functions listed above, a homomorphic scheme has also an evaluation function defined by $Evaluate_\alpha$ that takes as input the public key p_k, a circuit L and a tuple of cipher-texts $C = (c_1, c_2, c_3, ..., c_t)$, cipher of the input plain-texts vector $(m_1, m_2, ..., m_t)$. The evaluation function outputs a cipher-text Ψ given by:

$$\Psi = Evaluate_\alpha(p_k, L, C) \quad (1)$$

The scheme is homomorphic if we have: $\Psi = Encrypt_\alpha(p_k, L(m_1, m_2, ..., m_t))$.

2.2 Homomorphic Properties

When the circuit L performs a certain operation over the encrypted data as written in 1, the cloud is computing a predefined Boolean function f that can be written in a polynomial form. A polynomial form is a set of addition and multiplication gates. Thus to build a HE scheme we should satisfy these two basic properties:

1. Addition

$$Enc_\alpha(p_k, m_1) + Enc_\alpha(p_k, m_2) = Enc_\alpha(p_k, m_1 + m_2) \quad (2)$$

2. Multiplication

$$Enc_\alpha(p_k, m_1) \times Enc_\alpha(p_k, m_2) = Enc_\alpha(p_k, m_1 \times m_2) \quad (3)$$

where m_1 and m_2 are two plain-texts in $\{0, 1\}$ and $Enc_\alpha(p_k, m_i)$ is the encryption function.

3 Homomorphic Complex Scheme

In this section, we build our scheme using complex numbers. We first list the security parameters and then we detail our proposal.

3.1 Parameters

Based on a security parameter λ, different other parameters are generated as listed in [15] to build the new scheme:

1. γ: the bit length of the real and imaginary parts in the public key ($\gamma = \omega(\eta^2 log_2\lambda)$).
2. ρ: the bit-length of the real and imaginary parts in the noise ($\rho = \omega(log_2(\lambda))$.
3. η: the bit length of the secret key ($\eta \geq \rho.\Theta(\lambda log_2^2(\lambda))$.
4. ρ': extra noise parameter.
5. τ: The number of public keys ($\tau \geq \gamma + \omega(log_2\lambda)$).

3.2 Somewhat Homomorphic Complex Scheme Construction

The SH complex scheme construction is based on the following steps:

1. Secret key: The secret key s_k is defined by an η bit prime integer p.
2. Public keys: We build a public key bank (PK) formed of τ complex numbers having the following form: $PK = \{(p_{k1}{}^h + ip_{k2}{}^h) = ((pq_1{}^h + \epsilon_1^h) + i(pq_2{}^h + \epsilon_2{}^h), 1 \leq h \leq \tau\}$. For each complex public key $(p_{k1}{}^h + ip_{k2}{}^h)$, we build different parameters as follow:
   (a) random integer: $q_j^h \longleftarrow Z \cap [0, \frac{2^\eta}{p})$, random noise parameter: $\epsilon_j{}^h \longleftarrow Z \cap (-2^\rho, 2^\rho)$ where $j \in \{1, 2\}$.
   (b) $(p_{k1}{}^h, p_{k2}{}^h)$ should have different parities.
   (c) $(\epsilon_1{}^h, \epsilon_2{}^h)$ should have the same parity.

3. Encryption Function: The encryption of any bit m is given by the following equation:

$$c = Enc(p_k, m) = ({p_{k1}}^h + i{p_{k2}}^h) \times ({R_1}^h + i{R_2}^h) + {r_1}^h + i{r_2}^h + m \quad (4)$$

where $({p_{k1}}^h + i{p_{k2}}^h)$ is chosen randomly from the public key bank (PK), ${R_1}^h + i{R_2}^h$ is a random complex chosen such that $|{R_1}^h| <<< \frac{p}{6|{\epsilon_1}^h - {\epsilon_2}^h|}$, $|{R_2}^h| <<< \frac{p}{6|{\epsilon_1}^h + {\epsilon_2}^h|}$, ${r_1}^h + i{r_2}^h$ is a random complex chosen between $(-2^{\rho'}, 2^{\rho'})$ such that ${r_1}^h$ and ${r_2}^h$ have the same parity and $|{r_1}^h - {r_2}^h| \ll \frac{p}{6}$(Choosing the intervals of R_1^h, R_2^h, r_1^h, r_2^h is explained in the upcoming decryption section).

4. Decryption Function:
Following 4, we can demonstrate that $c = Enc(p_k, m) = ({p_{k1}}^h + i{p_{k2}}^h) \times ({R_1}^h + i{R_2}^h) + {r_1}^h + i{r_2}^h + m = (p{q_1}^h{R_1}^h - p{q_2}^h{R_2}^h + {R_1}^h{\epsilon_1}^h - {R_2}^h{\epsilon_2}^h + {r_1}^h + m) + i(p{q_1}^h{R_2}^h + p{q_2}^h{R_1}^h + {R_2}^h{\epsilon_1}^h + {R_1}^h{\epsilon_2}^h + {r_2}^h)$.
The decryption of the cipher c is done as follows:

$$\begin{aligned} Intermediate = real(c) - imag(c) = \\ p({q_1}^h{R_1}^h - {q_2}^h{R_2}^h - {q_1}^h{R_2}^h - {q_2}^h{R_1}^h) + \\ {R_1}^h({\epsilon_1}^h - {\epsilon_2}^h) - {R_2}^h({\epsilon_1}^h + {\epsilon_2}^h) + ({r_1}^h - {r_2}^h) + m. \\ m = mod(modNear(Intermediate, p), 2). \end{aligned} \quad (5)$$

Where $modNear(x, p) = y$, such that $y \in (-\frac{p}{2}, +\frac{p}{2})$.
Decryption works as long as:
$\frac{-p}{2} \le {R_1}^h({\epsilon_1}^h - {\epsilon_2}^h) - {R_2}^h({\epsilon_1}^h + {\epsilon_2}^h) + ({r_1}^h - {r_2}^h) + m \le \frac{p}{2}$.
In addition $({\epsilon_1}^h - {\epsilon_2}^h)$, $({\epsilon_1}^h + {\epsilon_2}^h)$ and $({r_1}^h - {r_2}^h)$ are even integers.
The decryption condition is satisfied as long as: $|{R_1}^h| <<< \frac{p}{6|{\epsilon_1}^h - {\epsilon_2}^h|}$, $|{R_2}^h| <<< \frac{p}{6|{\epsilon_1}^h + {\epsilon_2}^h|}$, $|{r_1}^h - {r_2}^h| << \frac{p}{6}$ and each pair of $({\epsilon_1}^h, {\epsilon_2}^h)$, $({r_1}^h, {r_2}^h)$ is formed of two integers having the same parity.

3.3 Complex Homomorphic Properties

We show in this section that the proposed scheme satisfies the homomorphic properties. Given two cipher-texts c_1, c_2 respectively for 2 different plain-texts m_1, m_2 encrypted using the complex scheme listed in section 3.2.
$c_1 = ({p_{k1}}^h + i{p_{k2}}^h)({R_1}^h + i{R_2}^h) + {r_1}^h + i{r_2}^h + m_1 = (p{q_1}^h{R_1}^h - p{q_2}^h{R_2}^h + {R_1}^h{\epsilon_1}^h - {R_2}^h{\epsilon_2}^h + {r_1}^h + m_1) + i(p{q_1}^h{R_2}^h + p{q_2}^h{R_1}^h + {R_2}^h{\epsilon_1}^h + {R_1}^h{\epsilon_2}^h + {r_2}^h)$. $c_2 = ({p_{k1}}^t + i{p_{k2}}^t)({R_1}^t + i{R_2}^t) + {r_1}^t + i{r_2}^t + m_2 = (p{q_1}^t{R_1}^t - p{q_2}^t{R_2}^t + {R_1}^t{\epsilon_1}^t - {R_2}^t{\epsilon_2}^t + {r_1}^t + m_2) + i(p{q_1}^t{R_2}^t + p{q_2}^t{R_1}^t + {R_2}^t{\epsilon_1}^t + {R_1}^t{\epsilon_2}^t + {r_2}^t)$.

1. Addition

$$\begin{aligned} c_1 + c_2 = \\ (p{q_1}^h{R_1}^h - p{q_2}^h{R_2}^h + p{q_1}^t{R_1}^t - p{q_2}^t{R_2}^t + {R_1}^h{\epsilon_1}^h \\ - {R_2}^h{\epsilon_2}^h + {R_1}^t{\epsilon_1}^t - {R_2}^t{\epsilon_2}^t + {r_1}^h \\ + {r_1}^t + m_1 + m_2) + i(p{q_1}^h{R_2}^h + p{q_2}^h{R_1}^h + \\ p{q_1}^t{R_2}^t + p{q_2}^t{R_1}^t + {R_2}^h{\epsilon_1}^h + {R_1}^h{\epsilon_2}^h + {R_2}^t{\epsilon_1}^t \\ + {R_1}^t{\epsilon_2}^t + {r_2}^h + {r_2}^t). \end{aligned} \quad (6)$$

Decryption of $c_1 + c_2$ is obtained by calculating $mod(modNear(real(c_1 + c_2) - imag(c_1 + c_2), p), 2)$ having the following form:
$pQ_{add} + R_{add} + m_1 + m_2$ such that $R_{add} = {R_1}^h({\epsilon_1}^h - {\epsilon_2}^h) - {R_2}^h({\epsilon_1}^h + {\epsilon_2}^h) + {R_1}^t({\epsilon_1}^t - {\epsilon_2}^t) - {R_2}^t({\epsilon_1}^t + {\epsilon_2}^t) + ({r_1}^h - {r_2}^h) + ({r_1}^t - {r_2}^t)$.
Decryption works as long as R_{add} is even and $\frac{-p}{2} \le R_{add} \le \frac{p}{2}$.

2. Multiplication
Similar to addition, decryption of $c_1 \times c_2$ is obtained by calculating $mod(modNear(real(c_1 \times c_2) - imag(c_1 \times c_2), p), 2)$. We can demonstrate that $real(c_1 \times c_2) - imag(c_1 \times c_2)$ have the following form:
$pQ_{mult} + R_{mult} + m_1 m_2$ such that $R_{mult} = ({r_1}^t{R_1}^h + m_2{R_1}^h - {r_2}^t{R_2}^h)({\epsilon_1}^h - {\epsilon_2}^h) - ({r_1}^t{R_2}^h + {r_2}^t{R_1}^h + m_2{R_2}^h)({\epsilon_1}^h + {\epsilon_2}^h) + ({r_1}^h{R_1}^t + m_1{R_1}^t - {r_2}^h{R_2}^t)({\epsilon_1}^t - {\epsilon_2}^t) - ({R_1}^t{r_2}^h + {r_1}^h{R_2}^t + m_1{R_2}^t)({\epsilon_1}^t + {\epsilon_2}^t) + ({R_1}^h{R_1}^t{\epsilon_1}^h - {R_2}^h{R_2}^t{\epsilon_1}^h)({\epsilon_1}^t - {\epsilon_2}^t) + ({R_2}^h{R_2}^t{\epsilon_2}^h - {R_1}^h{R_1}^t{\epsilon_2}^h)({\epsilon_1}^t + {\epsilon_2}^t) - ({R_1}^h{R_2}^t{\epsilon_1}^h + {R_2}^h{R_1}^t{\epsilon_1}^h)({\epsilon_1}^t + {\epsilon_2}^t) - ({R_2}^h{R_1}^t{\epsilon_2}^h + {R_1}^h{R_2}^t{\epsilon_2}^h)({\epsilon_1}^t - {\epsilon_2}^t) + {r_1}^h({r_1}^t - {r_2}^t) - {r_2}^h({r_1}^t + {r_2}^t) + m_2({r_1}^h - {r_2}^h) + m_1({r_1}^t - {r_2}^t)$.
Decryption also works as long as R_{mult} is even and $\frac{-p}{2} \le R_{mult} \le \frac{p}{2}$. We can simply demonstrate that R_{add} and R_{mult} are even since each couple of $({r_1}^h, {r_2}^h)$, $({r_1}^t, {r_2}^t)$, $({\epsilon_1}^h, {\epsilon_2}^h)$ and $({\epsilon_1}^t, {\epsilon_2}^t)$ is formed of two integers having the same parity, but having R_{add} and R_{mult} between $\frac{-p}{2}$ and $\frac{p}{2}$ is not always satisfied.
As a conclusion our new complex scheme is SH that supports a bounded number of addition and multiplication operations over the cipher-texts.

3.4 Making the Complex Scheme Fully Homomorphic

To make our complex scheme FH, we apply Bootstrapping [15, 17] in order to reduce the noise level after each operation. Starting from a complex cipher c, the plain-text m can be calculated following this equation $m \leftarrow [c^* - \lfloor \frac{c^*}{p} \rceil]_2$ where $c^* = real(c) - imag(c)$ since this decryption equation is much simpler than Equation. 5. First of all we can use Gentry's transformation to squash the decryption circuit. In this transformation we add to the public key some extra information about the secret key, and use this extra information to post process the cipher-text c^*. The post processed cipher-text c^* can be decrypted more efficiently than the original cipher-text.

3.4.1 Bootstrapping:

Consider the evaluation procedure given in 1, we use instead of the arithmetic circuit L, the decryption circuit D_α of the SH scheme α. The main concept of Bootstrapping is that we have a cipher-text ψ_1 that encrypts m under p_{k1} that we want to refresh. s_{k1} is the secret key related to the public key p_{k1}. s_{k1} is encrypted under another public key p_{k2}. Let $\overline{s_{k1j}}$ be the encrypted secret bits j of s_{k1} under p_{k2}. Bootstrapping is given by this algorithm:

$$\begin{aligned} &Recrypt(p_{k2}, D_\alpha, <\overline{s_{k1j}}>, \psi_{1j}),\\ &Set \quad \overline{\psi_{1j}} \leftarrow Encrypt(p_{k2}, \psi_{1j}) \\ &Output \quad \psi_2 \leftarrow Evaluate_\alpha(p_{k2}, D_\alpha, \langle\langle\overline{s_{k1}}\rangle, \langle\overline{\psi_{1j}}\rangle\rangle) \end{aligned} \tag{7}$$

In 7, the function $Evaluate_\alpha$ takes as input the bits of s_{k1} and ψ_1, each encrypted under p_{k2} to evaluate homomorphically the decryption circuit D_α. The output ψ_2 is an encryption under p_{k2} of $Decrypt_\epsilon(s_{k1}, \psi_1) = m$. By applying 7, we are removing the error vector associated to the first cipher and adding another error vector. The progress is made as long as the second error vector in ψ_2 is shorter than the primitive in ψ_1.

In Figure. 2, we present the evaluation procedure of a circuit with high depth giving birth to a cipher c with a high noise level.

As given in Figure. 3, for a cipher c consider $D_c(sk) = Decrypt_{sk}(c)$, this operation is called squashing the decryption circuit (will be explained in the upcoming section) and $D_c(.)$ is a low depth polynomial in sk. Bootstrapping consists of evaluating $D_c(.)$ using the encryption of the secret key sk.

3.4.2 Squashing the Decryption Circuit:

Bootstrapping is possible as long as the decryption circuit is an arithmetical circuit of low depth. Squashing is the required transformation in making it possible as listed in[15, 16, 17, 18, 19]. Following Gentry's Squashing technique, we add three different extra parameters κ, θ, Θ that are function of the security parameter λ. $\kappa = \frac{\gamma\eta}{\rho'}$, $\theta = \lambda$, $\Theta = \omega(\kappa.log(\lambda))$. For a secret key $s_k^* = p$ and a public key p_k^* from the original complex homomorphic scheme α, we add to the public key a set $\overrightarrow{Y} = \{y_1, y_2, y_3, ..., y_\Theta\}$ of rational numbers in $[0, 2)$ with κ bits of precision, such that there is a sparse subset $S \subset \{1, 2, 3, ..., \Theta\}$ of size θ with $\sum_{i\in S} y_i \approx \frac{1}{p}(mod2)$.

The secret key becomes the indicator of the subset S. The scheme of section 3.2 becomes FH complex by first applying the following modifications:

1. KeyGen:
Generate $s_k^* = p$ and p_k^* as before. set $x_p \leftarrow \lfloor\frac{2^\kappa}{p}\rceil$. Choose randomly a Θ bit vector $\overrightarrow{s}$ with hamming weight θ, $\overrightarrow{s} = \langle s_1, s_2, ..., s_\Theta\rangle$ and let $S = \{i; s_i = 1 \text{ in vector } S\}$. Choose Θ random integers $u_i \in Z \cap [0, 2^{\kappa+1})$, $i = 1, 2, 3, ..., \Theta$ subject to the condition that $\sum_{i\in S} u_i = x_p mod(2^{\kappa+1})$. Set $y_i = \frac{u_i}{2^\kappa}$ and $\overrightarrow{Y} = \{y_1, y_2, ..., y_\Theta\}$. Hence each y_i is a positive number smaller than 2 with κ bits of precision after the binary point such that $[\sum_{i\in S} y_i]_2 = (\frac{1}{p}) - \Delta_p$ for some $\Delta_p < 2^{-\kappa}$.

2. Encrypt and Evaluate Algorithm:
Generate the cipher-text $c^* = real(c) - imag(c)$ from a complex cipher c. Then for each $i \in \{1, 2, 3, ..., \Theta\}$, set $z_i \leftarrow [c^*.y_i]_2$ keeping only $n = \lceil log\theta\rceil + 3$ bits of precision after the binary point for each z_i. Output both c^* and $\overrightarrow{z} = \langle z_1, z_2, ..., z_\Theta\rangle$

3. Decrypt and Output Algorithm:

$$m \leftarrow [c^* - \lfloor\sum_{i\in S} s_i z_i\rceil]_2 \tag{8}$$

Lemma 1. *For every cipher-text $(c^*, \overrightarrow{z})$ that is generated by evaluating a circuit C, it holds that $\sum_i s_i z_i$ is within $\frac{1}{4}$ of an integer.*

Proof. Fix an arithmetic circuit L, public keys and secret keys generated with respect to the security parameter λ, with $\{y_i\}_{i=1}^{\Theta}$ the rational numbers in the public key and $\{s_i\}_{i=1}^{\Theta}$ the secret key bits. If we recall that y_i's were chosen that: $[\sum_i s_i y_i]_2 = (\frac{1}{p}) - \Delta_p$. We fix a circuit L, and t cipher-texts $\{c_i\}_{i=1}^{t}$ that encrypts the input to L and denote: $c = Evaluate_\epsilon(pk, L, c_1, c_2, .., ..., c_t)$, we need to establish this equation:

$$\lfloor\frac{c^*}{p}\rceil = \lfloor\sum_i s_i z_i\rceil(mod2) \tag{9}$$

where $c^* = real(c) - imag(c)$ and the z_i's are computed as $[c^*.y_i]_2$ with only $\lceil log\theta + 3\rceil$ bits of precision after the binary point, hence $[c^*.y_i]_2 = z_i - \Delta_i$ with $|\Delta_i| \le \frac{1}{16\theta}$.

$[(\frac{c^*}{p}) - \sum s_i z_i]_2 = [(\frac{c^*}{p}) - \sum_i s_i[c^* y_i]_2 + \sum s_i\Delta_i]_2 = [(\frac{c^*}{p}) - c^*[\sum s_i y_i]_2 + \sum s_i\Delta_i]_2 = [(\frac{c^*}{p}) - c^*.(\frac{1}{p} - \Delta_p) + \sum s_i\Delta_i]_2 = [c^*.\Delta_p + \sum s_i\Delta_i]_2$. □

We claim that the final quantity inside the brackets has magnitude at most $\frac{1}{8}$. By definition, since c^* is a valid cipher-text output, the value $\frac{c^*}{p}$ is within $\frac{1}{8}$ of an integer. Together all these facts imply the lemma. To establish the claim, we first observe that $|\sum s_i\Delta_i| \le \theta \times \frac{1}{16\theta}$. Regarding $c^*\Delta_p$, recall that the output c^* is obtained by evaluating the circuit L on the cipher-text c_i as listed in [15], for any polynomial P that implements the circuit L, for any $\alpha \ge 1$, if P's input has magnitude at most $2^{\alpha(\rho'+2)}$, its output magnitude at most $2^{\alpha(\eta-4)}$. In particular, when P's are fresh cipher-texts, which have magnitude at most 2^γ, P's output cipher-text c^* has magnitude at most $2^{\gamma(\eta-4)(\rho'+2)} < 2^{\kappa-4}$. Thus $|c^*\Delta_p| < \frac{1}{16}$.

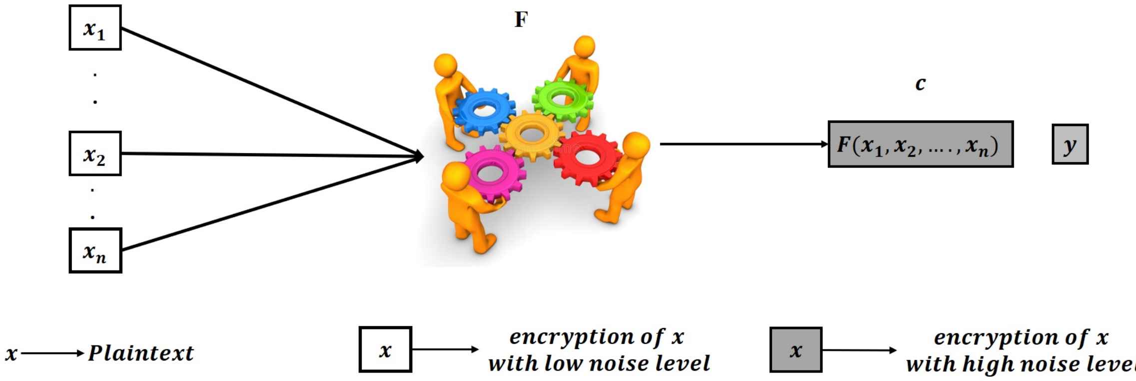

Figure 2: Circuit Evaluation with High Noise

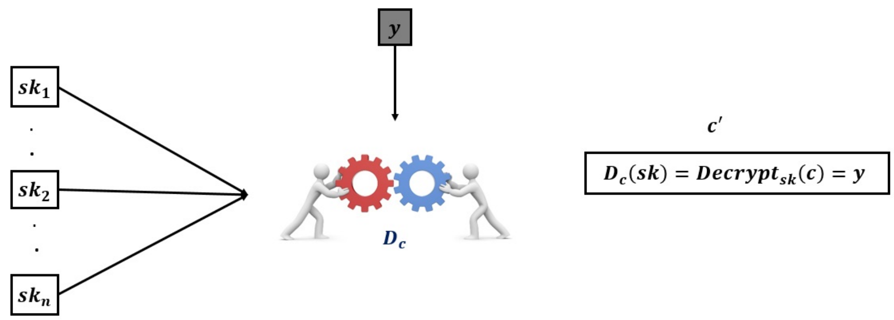

Figure 3: Fresh Cipher Generation

3.4.3 Practical Implementation of Squashing and Bootstrapping:

For the squashed decryption circuit given in 8, the implementation of $\sum_{i \in S} s_i z_i$ (such that $z_i = (z_{i1}, z_{i2}, ..., z_{in})$ is a vector of $n = \lceil log\theta \rceil + 3$ bits, where $1 \leq i \leq \Theta$), can be done with lower computational complexity as implemented in [16, 19]. This new implementation is based on dividing the new secret key $\overrightarrow{s}$ in θ boxes of $B = \frac{\Theta}{\theta}$ bits each, where each box has a single bit having the value 1. This will lead us to obtain a grade school addition algorithm that requires $O(\theta^2)$ multiplications instead of $O(\Theta.\theta)$. The secret key $\overrightarrow{s}$ is divided into $s_{k,i}$, the i^{th} secret key bit in box k, where $1 \leq k \leq \theta$ and $1 \leq i \leq B$. The resultant equation is:

$$m \leftarrow (c^* - \lfloor \sum_{k=1}^{\theta} (\sum_{i=1}^{B} s_{k,i} z_{k,i}) \rceil) mod(2) \quad (10)$$

We denote that the sum $q_k = \sum_{i=1}^{B} s_{k,i} z_{k,i}$ is obtained by adding B numbers, only one being nonzero. The decryption equation is now:

$$m \leftarrow (c^* - \lfloor \sum_{i=1}^{\theta} (q_k) \rceil) mod(2) \quad (11)$$

Where the q_k's are rational in $[0, 2)$ with n bits of precision after the binary point. Another form of the decryption equation is given by:

$$m = [c^*]_2 \oplus [\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil]_2 \quad (12)$$

Based on 12, we can deduce that the parity of the plain-text m is related to the parity of the primitive cipher-text c^* and the parity of $[\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil]_2$.

In order to make this concept much clearer, a simple implementation of bootstrapping, using low values, is explained in the following example. Giving randomly $\Theta = 4$, $\theta = 2$, $B = \frac{\Theta}{\theta} = 2$, and a precision bit $n = 1$. The secret key $\overrightarrow{s} = [b_1, b_2, b_3, b_4]$ such that ($b_1 = 1$ and $b_2 = 0$) or ($b_1 = 0$ and $b_2 = 1$), ($b_3 = 1$ and $b_4 = 0$) or ($b_3 = 0$ and $b_4 = 1$) based on the implementation of 10. Initial values for evaluating the decryption circuit using the secret key $\overrightarrow{s}$ without encryption are $s_1 = [s_{11} \quad s_{12}] = [b_1 \quad b_2]$, $s_2 = [s_{21} \quad s_{22}] = [b_3 \quad b_4]$, the z values are taken just as an example $z_1 = [z_{11} \quad z_{12}] = [[10] \quad [01]]$, $z_2 = [z_{21} \quad z_{22}] = [[11] \quad [01]]$. We start by applying the evaluation procedure over the plaintexts by calculating $Sum = \sum_{k=1}^{\theta} \sum_{i=1}^{B} s_{k,i} z_{k,i} = \sum_{k=1}^{2} \sum_{i=1}^{2} s_{k,i} z_{k,i} = (b_1 z_{11} + b_2 z_{12}) + (b_3 z_{21} + b_4 z_{22}) = [b_1 \quad b_2] + [b_3 \quad b_3 + b_4] =$

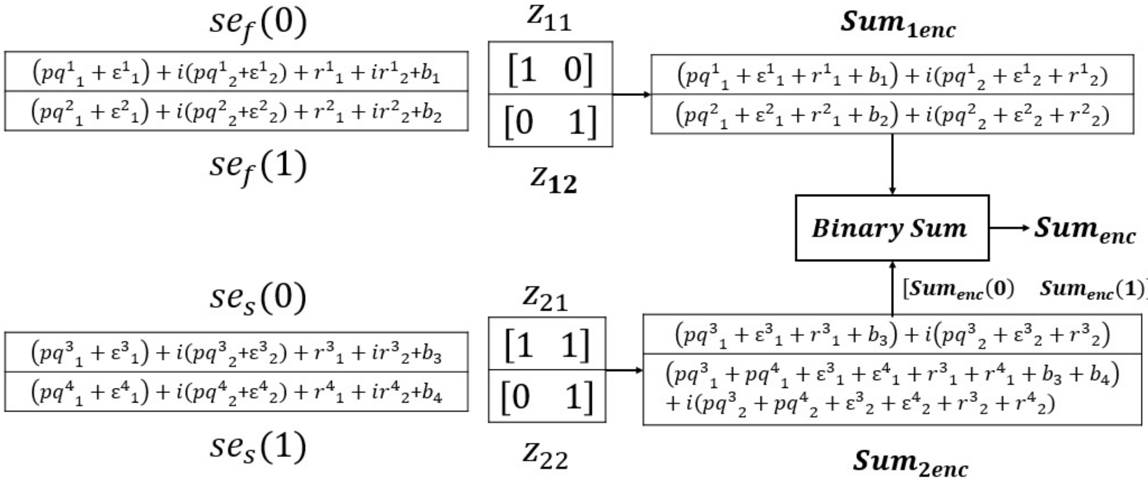

Figure 4: Circuit Evaluation over Cipher-texts.

$(Sum_1) + (Sum_2)$, then we apply the binary summation to calculate $Sum = Sum_1 + Sum_2 = [b_1 + b_3 \quad b_2 + b_3 + b_4 + b_1 b_3] = [Sum(0) \quad Sum(1)]$ ($Sum(0) = Sum_1(0) + Sum_2(0)$ and $Sum(1) = Sum_1(1) + Sum_2(1) + Sum_1(0)Sum_2(0)$). As a result, the parity of $\lfloor Sum \rceil$ is given by $b_1 + b_3 + b_2 + b_3 + b_4 + b_1 b_3$. Next step is to do the circuit evaluation over the complex cipher-texts. Let $se_f = [se_1 \quad se_2]$ be the cipher of the secret key $s_1 = [b_1 \quad b_2]$ and $se_s = [se_3 \quad se_4]$ be the cipher of the secret key $s_2 = [b_3 \quad b_4]$ using the encryption procedure listed in 4 with random values $(R_1^h + iR_2^h) = 1$.
$se_f = [se_f(0) \quad se_f(1)] = Enc(s_1) = [(pq_1^1 + \epsilon_1^1) + i(pq_2^1 + \epsilon_2^1) + r_1^1 + ir_2^1 + b_1 \quad (pq_1^2 + \epsilon_1^2) + i(pq_2^2 + \epsilon_2^2) + r_1^2 + ir_2^2 + b_2]$.
$Sum_{enc} = \sum_{k=1}^{\theta}\sum_{i=1}^{B} se_{k,i}z_{k,i} = \sum_{k=1}^{2}\sum_{i=1}^{2} se_{k,i}z_{k,i} = Sum_{1enc} + Sum_{2enc}$, where $Sum_{1enc} = se_f(0)z_{11} + se_f(1)z_{12}$ and $Sum_{2enc} = se_s(0)z_{21} + se_s(1)z_{22}$. An implementation of circuit evaluation over the cipher-texts is given in Figure.4.
As shown in Figure.4, after applying the binary summation over Sum_{1enc} and Sum_{2enc} we obtain $Sum_{enc} = [Sum_{enc}(0) \quad Sum_{enc}(1)]$ such that $Sum_{enc}(0) = p(Q_{0R} + iQ_{0I}) + (\epsilon_1^1 + \epsilon_1^3) + i(\epsilon_2^1 + \epsilon_2^3) + (r_1^1 + r_1^3) + i(r_2^1 + r_2^3) + \mathbf{b_1} + \mathbf{b_3}$.
$Sum_{enc}(1) = p(Q_{1R} + iQ_{1I}) + (\epsilon_1^1\epsilon_1^3 - \epsilon_2^1\epsilon_2^3) + (r_1^3\epsilon_1^1 + r_1^1\epsilon_1^3 - r_2^3\epsilon_2^1 - r_2^1\epsilon_2^3) + (r_1^1 r_1^3 - r_2^1 r_2^3) + b_1\epsilon_1^3 + b_3\epsilon_1^1 + b_1 r_1^3 + b_3 r_1^1 + (\epsilon_1^2 + \epsilon_1^3 + \epsilon_1^4) + (r_1^2 + r_1^3 + r_1^4) + \mathbf{b_1 b_3} + \mathbf{b_2} + \mathbf{b_3} + \mathbf{b_4} + i((\epsilon_1^1\epsilon_2^3 + \epsilon_2^1\epsilon_1^3) + (r_2^3\epsilon_1^1 + r_1^1\epsilon_2^3 + r_2^1\epsilon_1^3 + r_1^3\epsilon_2^1) + (r_2^1 r_1^3 + r_1^1 r_2^3) + b_1\epsilon_2^3 + b_3\epsilon_2^1 + b_1 r_2^3 + b_3 r_2^1 + (\epsilon_2^2 + \epsilon_2^3 + \epsilon_2^4) + (r_2^2 + r_2^3 + r_2^4))$.
Since $mod(modNear(real(Sum_{enc}(0)) - imag(Sum_{enc}(0)), p), 2) = b_1 + b_3$ and $mod(modNear(real(Sum_{enc}(1)) - imag(Sum_{enc}(1)), p), 2) = b_1 b_3 + b_2 + b_3 + b_4$, as long as the encryption parameters are chosen such that the scheme is SH (i.e Sum_{1enc} and Sum_{2enc} are formed of complex ciphers that can be added and multiplied homomorphically as long as the scheme is SH). Based on all the calculations listed above, a fresh cipher-text of c^* can be written as:

$$fresh_{cipher} = \lfloor Sum_{enc}(0) + Sum_{enc}(1) \rceil + [c^*]_2 + \sum_j (p_{k_1}^{\ j} + ip_{k_2}^{\ j})(R_1^j + iR_2^j) \qquad (13)$$

4 Implementation and Security Analysis

In order to validate our work, first we did an implementation of our new Complex-based scheme, then we compared the evaluation of the logic circuit $C = (b_0 \oplus b_1) \bullet (b_2 \oplus b_3)$ using the two schemes: Complex and BGV.
As a brief overview of the BGV [20, 21], it is a FHE scheme that works over bit level and built using Lattice based cryptography. The security of this scheme depends on the hardness of Learning With Error (LWE) introduced by Oded Regev in [25], that relies on the complexity of solving a noisy linear system. Starting from a security parameter λ, we have a secret key $s \in Z_p^{[1,n]}$ and a cipher-text $c \in Z_p^{[n,1]}$. The hardness of LWE resides in taking the lattice dimension: $n \approx poly(\lambda)$ and the ring dimension: $p \approx poly(n)$ following [21, 25]. Homomorphic addition is achieved by simply adding the two cipher-texts, while homomorphic multiplication is done by calculating the Tensor product of the two different cipher-texts which increases the dimension exponentially. Key Switching (KS) is a new technique introduced to reduce the cipher dimension after each homomorphic multiplication. The basic BGV scheme is SH since the noise level will increase with circuit depth, therefore MS is another technique introduced to reduce the noise level after each arithmetic operation and extend to a higher circuit depth.
Finally a crypt-analysis of the new scheme is validated with the GACD attack. All simulations are done under Python with SAGE-Math Library using a machine having the following specifications (CPU: Intel Xeon, E5 – 2630, 2.40 GHZ, 8 CORES, 128 GB RAM).

4.1 Implementations and Results

1. **First Implementation** : First implementation with the new Complex-based scheme is done with three different security layers (**small**: $\lambda = 42$, $\rho = 26$, $\eta = 988$, $\gamma = 147456$, $\Theta = 150$, $\tau = 158$), (**medium**: $\lambda = 52$, $\rho = 41$, $\eta = 1558$, $\gamma = 843033$, $\Theta = 555$, $\tau = 572$), (**large**: $\lambda = 62$, $\rho = 56$, $\eta = 2128$, $\gamma = 4251886$, $\Theta = 2070$, $\tau = 2110$) with extra noise parameter $\rho' = \eta - \rho$, θ the hamming weight of vector $\overrightarrow{s}$ is equal to 15, n the precision after the binary point of each

Table 1: First Implementation Results

Parameters	*KeyBankGen*	*Encrypt*	*Decrypt*
small	0.136439000002 *s*	0.00102999999945 *s*	0.000683999998728 *s*
medium	2.732191 *s*	0.00130100000024 *s*	0.00339099999837 *s*
large	48.049091 *s*	0.00596199999927 *s*	0.0216390000023 *s*

Table 2: Second Implementation Results

Parameters	Complex mean execution time	BGV mean execution time
small	0.4102108 *s*	0.35041905 *s*
medium	2.8407822 *s*	2.42358273 *s*
large	27.40187732 *s*	38.52468052 *s*

z_i is taken as 4. Table. 1 shows in terms of execution time the generation of public key bank of τ complex public keys, 1 bit encryption and decryption.

2. **Second Implementation** : In the second implementation, we did the evaluation procedure of the logic circuit $C = Plain_{output} = (b_0 \oplus b_1) \bullet (b_2 \oplus b_3)$, (i.e. $Cipher_{Output} = ((c_0 + c_1) \times (c_2 + c_3)$, where $c_i = Enc(b_i),\ for\ 0 \leq i \leq 3)$ using the two schemes (Complex and BGV) with the same level of security λ. For BGV the three layers of implementation are (**small**: $\lambda = 42$, $n = 42$, $p \approx O(n^{20})$, $q \approx O(\frac{n^{20}}{2})$), (**medium**: $\lambda = 52$, $n = 149$, $p \approx O(n^6)$, $q \approx O(\frac{n^6}{2})$), (**large**: $\lambda = 62$, $n = 370$, $p \approx O(n^6)$, $q \approx O(\frac{n^6}{2})$). Table.2 shows a comparison in terms of mean execution time between the two implementations for 100 iterations (Evaluation procedure for the Compplex-based scheme is done with Bootstrapping mechanism while with BGV is achieved with KS and MS). In addition, results have shown that noise is efficiently reduced for the two schemes. One can see that for large case, the proposed Complex based-scheme performs better that BGV.

4.2 Security Analysis:

The crypt-analysis of the homomorphic encryption schemes will consider the techniques used to build such schemes. The security of the BGV-lattice based scheme relies on the hardness of solving LWE problems [25]. As for our new complex scheme, it uses mathematical operations and the typical attack in this case is General Approximate Common Divisor (GACD). Given a public key bank $PK = \{x_1, x_2, ..., x_\tau\}$ where $x_j = pq_j + r_j$ for $1 \leq j \leq \tau$, the basic idea of this attack is to reveal the value of the secret key p starting from PK. The secret key p can be revealed using different types of algorithms like the approximate GCD of two numbers discussed in [22], the approximate GCD of many numbers using the SDA algorithm [23] by applying a lattice based attack with *LLL* algorithm. Recently a new improved attack was introduced and implemented in [24], [19]. In our crypt-analysis, we tried to apply this new Approximate GCD attack. Given the public key bank $PK = \{x_j, 1 \leq j \leq \tau\}$, where $q_j \in [0, \frac{2^\gamma}{p})$ and $r_j \in [0, 2^\rho)$ are chosen uniformly and independently at random. The algorithm is as follows: For $j = 1, 2, ..., \tau$, let:

$$y_j = \prod_{i=0}^{2^\rho - 1} (x_j - i) \tag{14}$$

Equation 14 shows clearly that p divides the GCD $g = gcd(y_1, y_2, ..., y_\tau)$. To build this attack and depending on the choice of the (q_j, r_j), we will try to find a certain bound B not much larger than 2^ρ that with a high probability, all the prime factors of g except p are smaller than this bound B. The probability that all the prime factors of g except p are smaller than B is done based on [19]: "For every prime $p \geq B$ other than p, not all the x_j's are congruent to one of $(0, 1, ..., 2^\rho - 1) \quad mod p$". This happens with probability very close to $1 - (\frac{2^\rho}{p})^s$. Hence, the probability that all the prime factors of g except p are smaller than B is essentially given by the following Euler product:

$$P_{s,\rho}(B) = \prod_{p \geq B, \quad p \neq p} (1 - \frac{2^{s\rho}}{p^s}) \tag{15}$$

Based on [19] and using the usual prime counting function $\pi(x)$ explained in [26], we can demonstrate that 15 converges to some positive number smaller than 1 and satisfies the following lemma:

Lemma 2. *For any* $B > 2^{\rho + \frac{1}{s}}$, *we have:*
$$1 - P_{s,\rho}(B) < \frac{2s}{s-1} \times \frac{2^{s\rho}}{B^{s-1} logB}$$

In our simulation, we picked $B = 2^{\frac{s+1}{s-1}\rho}$ and we got a success probability $P_{s,\rho}(B) > 1 - 2^{-\rho}$. The GACD attack is given by the pseudo code of Algorithm 1.

Algorithm 1 *GACD Attack*

```
procedure p=GACD_(η,γ, ρ)
    p ← random_prime(0, 2^η)
    s ← ρ
    B ← ⌊2^((s+1)/(s−1) ρ)⌋
    Fa ← Factorial(B)
    x ← p × random_integer(0, 2^(γ−η)) + random_integer(0, 2^ρ)
    i ← 0
    g ← 1
    while i < 2^ρ do
        g ← g × (x − i)
        i ← i + 1
    end while
    j ← 1
    while j ≤ s do
        x ← p×random_integer(0, 2^(γ−η))+random_integer(0, 2^ρ)
        i ← 0
        z ← 1
        while i < 2^ρ do
            z ← z × (x − i)
            i ← i + 1
        end while
        g ← Greatest_divisor_of_gcd(g, z)_prime_to_Fa
        if ⌊log2(g)⌋ ≤ η then
            Break
        end if
        j ← j + 1
    end while
return g
end procedure
```

Due to the limited resources of our machine (CPU: Intel Xeon, E5-2630, 2.40 GHZ, 8 CORES, 128 GB RAM), the proposed GACD attack with the security levels related to λ is not feasible since the polynomial y_j given in 14 is of degree 2^ρ with coefficients of size γ and requires a memory of size $2^\rho\gamma$ bits. The required size of each level is: **small** : 1.125 Terra Byte, **medium** : 210759 Terra Byte, **large** : 34831286272 Terra Byte, while our machine is only 128 GB RAM.

5 Conclusion

In this paper, we profited from the simplicity of complex numbers properties by proposing a new SWE scheme based on complex numbers. We applied Gentry refresh mechanism to make our scheme FH. We then implemented our new scheme with the BGV using SAGEMath library. As a comparison with BGV, a well known FHE scheme, our new scheme is an efficient homomorphic scheme and performs better than BGV in terms of execution for large implementation. In addition, our scheme is simply based on simple complex operations rather than lattice based cryptography (homomorphic complex multiplication is done without dimension expansion rather than Tensor product) and Bootstrapping can support unbounded circuit depth, while MS used with BGV is limited to some circuit depth. Finally a crypt-analysis based on GACD attack is presented. Future work will consider the implementation of the GACD Attack given in section 4.2 with a more powerful machine in order to evaluate the approximate attack time.

Acknowledgment This work has been partially funded with support from the Lebanese University.

References

[1] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations on Secure Computation, Academia Press*, 1978, pp. 169–179. [Online]. Available: http://www.oalib.com/references/14708317

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, no. 1, pp. 96–99, Jan. 1983. [Online]. Available: http://doi.acm.org/10.1145/357980.358017

[3] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 83:1–83:33, Dec. 2017. [Online]. Available: http://doi.acm.org/10.1145/3124441

[4] L. Zhang, Y. Zheng, and R. Kantoa, "A review of homomorphic encryption and its applications," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, ser. MobiMedia '16. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 97–106. [Online]. Available: http://dl.acm.org/citation.cfm?id=3021385.3021405

[5] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey, "Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 108–117, March 2013. [Online]. Available: http://doi.acm.org/10.1109/MSP.2012.2230219

[6] S. Fau, R. Sirdey, C. Fontaine, C. Aguilar Melchor, and G. Gogniat, "Towards practical program execution over fully homomorphic encryption schemes," in *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2013)*, Compiègne, France, Oct. 2013, pp. –. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00917061

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-48910-X_16

[8] M. Nassar, A. Erradi, and Q. M. Malluhi, "Paillier's encryption: Implementation and cloud applications," in *2015 International Conference on Applied Research in Computer Science and Engineering (ICAR)*, Oct 2015, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/7338149

[9] J. D. i Ferrer, "A new privacy homomorphism and applications," *Information Processing Letters*, vol. 60, no. 5, pp. 277 – 282, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019096001706

[10] J. Domingo-Ferrer, "A provably secure additive and multiplicative privacy homomorphism*," in *Information Security*, A. H. Chan and V. Gligor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 471–483. [Online]. Available: https://dl.acm.org/citation.cfm?id=744660

[11] K. Hariss, H. Noura, and A. E. Samhat, "Fully enhanced homomorphic encryption algorithm of more approach for real world applications," *Journal of Information Security and Applications*, vol. 34, no. Part 2, pp. 233 – 242, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212616303052

[12] K. Hariss, H. Noura, A. E. Samhat, and M. Chamoun, "Design and realization of a fully homomorphic encryption algorithm for cloud applications," in *Risks and Security of Internet and Systems*, N. Cuppens, F. Cuppens, J.-L. Lanet, A. Legay, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2018, pp. 127–139. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-76687-4_9

[13] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford, CA, USA, 2009, aAI3382729. [Online]. Available: https://dl.acm.org/citation.cfm?id=1834954

[14] G. Craig, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[15] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology – EUROCRYPT 2010*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–43. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-13190-5_2

[16] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Advances in Cryptology – EUROCRYPT 2011*, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 129–148. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-20465-4_9

[17] C. Gentry, S. Halevi, and N. P. Smart, "Better bootstrapping in fully homomorphic encryption," in *Public Key Cryptography – PKC 2012*, M. Fischlin, J. Buchmann, and M. Manulis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–16. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-30057-8_1

[18] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology – CRYPTO 2011*, P. Rogaway, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 487–504. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-22792-9_28

[19] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *Advances in Cryptology – EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 446–464. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-29011-4_27

[20] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Proceedings of the 31st Annual Conference on Advances in Cryptology*, ser. CRYPTO'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 505–524. [Online]. Available: http://dl.acm.org/citation.cfm?id=2033036.2033075

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 309–325. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090262

[22] N. Howgrave-Graham, "Approximate integer common divisors," in *Cryptography and Lattices*, J. H. Silverman, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 51–66. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44670-2_6

[23] J. C. Lagarias, "The computational complexity of simultaneous diophantine approximation problems," *SIAM J. Comput.*, vol. 14, no. 1, pp. 196–209, Feb. 1985. [Online]. Available: http://dx.doi.org/10.1137/0214016

[24] Y. Chen and P. Q. Nguyen, "Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers," in *EUROCRYPT 2012*, ser. Lecture Notes in Computer Science, D. Pointcheval and T. Johansson, Eds., vol. 7237, IACR. Cambridge, United Kingdom: Springer, Apr. 2012, pp. 502–519. [Online]. Available: https://hal.inria.fr/hal-00864374

[25] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: ACM, 2005, pp. 84–93. [Online]. Available: http://doi.acm.org/10.1145/1060590.1060603

[26] E. Bach and J. Shallit, *Algorithmic Number Theory*. Cambridge, MA, USA: MIT Press, 1996. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-04053-9_2