An Improved Fully Homomorphic Encryption Scheme for Cloud Computing

Mohd Rizuan Baharon[1], Qi Shi[2], Mohd Faizal Abdollah[1], S.M.Warusia Mohamed S.M.M Yassin[1] and Ariff Idris[1]

[1]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, 76100 Malaysia
[2]Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom

***Abstract*:** Business in cloud computing is very popular among Small and Medium Enterprises (SMEs). By leveraging services from the cloud, such companies can migrate all of their in-house operations to cloud at low costs with minimum IT facility requirements such as desktop machines and the Internet. Even though the cloud promises tremendous advantages in terms of computing resources and storage spaces, some of the companies are still reluctant to adopt such a technology because of security concerns. To overcome such problems, a fully homomorphic encryption (FHE) scheme with improved efficiency can be implemented as the scheme allows computation on encrypted data without decryption. In this paper, we propose an improved FHE scheme that uses a symmetric key for encryption together with a protocol to implement the scheme. Furthermore, we also provide an analysis regarding to the noise growth in the processed ciphertext and squashing technique that is required to reduce the noise. This analysis is essential to improve the efficiency of the scheme as the squashing technique is time-consuming.

***Keywords*:** Fully Homomorphic Encryption Scheme, Cloud Computing, Integers, Symmetric Encryption Scheme, Asymmetric Encryption Scheme**.**

1. Introduction

Computation of arbitrary functions on encrypted data is a desired application of many online service providers like the cloud as it offers tremendous benefits to their business and to the users as well. By enabling computation on encrypted data, many applications can be outsourced into the cloud to process the encrypted data without decryption. Such a process can guarantee the security and privacy of the data processed by those applications. With the advent technology provided by the cloud, many organizations have started to think and move their in-house business operations to the cloud. This is due to the cloud providing huge advantages including ample computing resources and storage spaces on an as-pay-as-you-use basis [1]. In addition, those fantastic services can be leveraged in a very convenient way with minimum IT facility requirements such as desktop machines and the Internet. In order to fully leverage services from the cloud, a user needs to outsource its data into the cloud storage. Nevertheless, outsourcing such information that may contain private data raises security and privacy concerns as the safety of the data is not guaranteed [2]–[4]. Thus, to preserve the security and privacy of the data, computation of arbitrary functions on encrypted data is certainly required, which enables the user to perform operations on its data without revealing the data contents to the cloud. To achieve such a requirement, a lot of research has been conducted from various perspectives such as theoretical research and practical applications to enable arbitrary functions to be computed on encrypted data [5]–[11].

Research of computation on encrypted data started 30 years ago when Rivest, Adleman, and Dertouzos invented the RSA cryptosystem [12]. Even the RSA scheme has a special feature which is homomorphic under multiplication, but it is still unable to compute arbitrary functions on encrypted data, as to achieve that, the encryption scheme must support homomorphism in both additions and multiplications [13]. Since then, a lot of methods had been proposed but they were still partially homomorphic or fully homomorphic under certain conditions [14], [15]. This process continued until a smart guy Gentry has found a homomorphic scheme that supports both additions and multiplications so as to enable computation of arbitrary functions on encrypted data. The proposed scheme is called a fully homomorphic encryption (FHE) over Lattices [16]. Since then, all the proposed FHE schemes have followed the blueprint of Gentry's scheme [17], [18]. His achievement has proven that computation on arbitrary encrypted data is achievable and provided a new direction for other researchers to conduct research to allow computation on arbitrary encrypted data.

Even computation on an encrypted data is achievable and many optimizations has been made to improve the existing schemes; their implementation in any practical system is far from practical as efficiency still be the big obstacle. According to [10], [19], [20] this efficiency problem is came from various aspects such as key generation algorithm, the noise growth, and a technique that has been used to reduce the noise. As such in [18], the public key generated with high complexity is unpractical to be implemented in any practical system [21]. Furthermore, as all of the existing schemes based on the blueprint of Gentry's scheme, each ciphertext generated has noise attached for security reason. When the ciphertext is processed, the noise will be growing and once it exceeded the size of the secret key i.e. the threshold; the decryption will be failed. To remedy such a problem, squashing can be used to reduce the noise. However, this technique is time-consuming and need to be minimized or omitted as proposed in [8], [22].

Thus, in this paper we propose a solution that focuses on the key generation, and provide an analysis related to the scheme efficiency. We believed that by concentrating on the complexity of key generation, we will provide a scheme that has better efficiency. Furthermore, the analysis related to the scheme efficiency will give a way to improve the weaknesses and optimize the scheme.

Our contribution in this paper is threefold. First, we proposed an improved FHE scheme by combining symmetric and asymmetric encryption schemes to achieve better efficiency. Our encryption process using a symmetric key will make the encryption faster compared to others that mainly adopt slow

asymmetric encryption schemes [23]. Furthermore, our proposed scheme will reduce the key generation time because there is only one symmetric key required to encrypt the data rather than many public keys proposed in other schemes. The next contribution of this paper is that we have proposed an appropriate protocol to implement our proposed scheme. This protocol has been designed accordingly for the implementation of computation of arbitrary functions on encrypted data in cloud environments. Furthermore, we have provided an analysis of the implication of noise and the number of squashing required for computation on encrypted data.

Our paper is organized as follows. Section 2 describes some background of the existing FHE schemes over integers. Then, in Section 3 we explain our proposed scheme together with a proposed protocol to implement the scheme. In section 4, we provide an analysis on the performance of the proposed scheme. Finally, in Section 5 we conclude our paper and provide some open problems to be further investigated in the future.

2. Related Work

In 2009, Gentry found a technique that allows computation of arbitrary functions on encrypted data [18]. His brilliant idea can be illustrated in Figure 1.

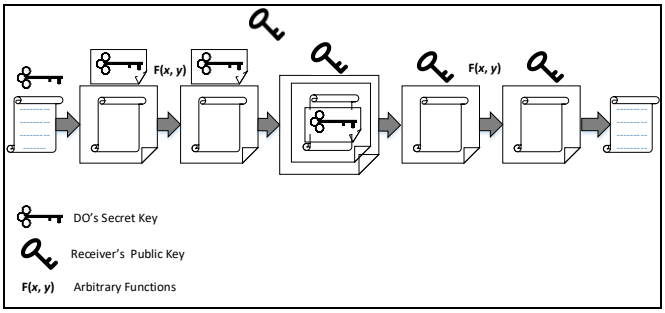

Figure 1. Gentry's FHE scheme data.

In the figure, there are two types of keys; public keys and a secret key where the former is an encryption key while the latter is a decryption key. The public key is used to encrypt an original data item. The encrypted data will be outsourced to a processor for remote processing using low degree polynomial. During the process, noise that attached to each ciphertext will be growing. Thus, to reduce the size of the noise growth in the ciphertext, squashing technique will be implemented. Squashing technique involved two steps; re-encrypt and decryption. The re-encrypt is a technique to prevent the data from being disclosed to the processor once it is decrypted using an encrypted secret key. At the meantime, the decryption is required to remove the growing noise in the processed ciphertext, leaving the ciphertext encrypted by another public key, and has a small amount of noise attached to the data.

The squashing technique is very essential as the purpose is to remove the growing noise in the ciphertext without revealing any information about the plaintext or the secret key to the processor. Without this process, further computation on the encrypted data would be unable to get a correct result, as the noise growth could exceed a set threshold and thus impede the derivation of the correct result. The process of removing the noise will be repeated during the computation on the encrypted data in order to maintain a small amount of noise attached to the ciphertext. Finally, once the computation is completed, the result will be decryptable using the receiver's private key [12].

Since the invention of Gentry's scheme, a lot of improvements on FHE schemes have been proposed. Some of them are based on the complexity of Lattices [11], [16], [20] and others on the simplicity of the Integers [7]–[9], [18], [24]. The aim of all the schemes is solely to allow the computation of arbitrary functions on encrypted data while the security of the processed data is guaranteed in a very efficient way. Those schemes are believed to be able to solve the FHE problem identified 30 years ago. Nonetheless, to implement such schemes in any practical systems, there is still a lot of work to be done to improve their applicability, particularly efficiency.

A FHE scheme over integers that was first proposed by van Dijk et al. [24] seems to be a good potential scheme because it is proposed based on conceptual simplicity. Furthermore, the plaintext of this scheme is considered as integers instead of single bits used in other schemes. Such a scheme provides a better improvement in terms of a bigger plaintext space rather than the plaintext space of $\{0,1\}$. Nevertheless, this scheme suffers from an efficiency issue as the scheme's public key with the complexity of $O(\lambda^{10})$ is too large for any practical system [24].

According to Coron et al. in their works, this drawback can be improved by reducing the public-key size to $O(\lambda^7)$ [7] Such an improvement can be achieved by encrypting with a quadratic form in the public-key element [7]. The details of the way they improved the previous scheme are explained as in the previous section.

2.1 FHE Scheme over Integers

Van Dijk et al. [24] have proposed a FHE scheme using only elementary modular arithmetic with the aim of simplicity. Their FHE scheme is constructed from a somewhat homomorphic scheme, a scheme that can evaluates low degree polynomial on ciphertext. To achieve fully homomorphic properties, they have implemented squashing technique introduced by Gentry. Their scheme only uses addition and multiplication over the Integers rather than using ideal Lattices over polynomial ring to achieve simplicity. Furthermore, in their scheme, they only consider encryption schemes that are homomorphic with respect to Boolean circuits, C consisting of gates for addition and multiplication mod 2. The details of their scheme and parameters used are explained as below.

λ is a security parameter.

ρ is the bit-length of a noise r_i.

η is the bit-length of a secret key p .

γ is the bit-length of the x_i 's such that $x_i = pq_i + 2r_i$.

τ is the number of x_i 's in the public key, pk such that

$$pk = \langle x_1, x_2, ..., x_\tau \rangle .$$

ρ' is a secondary noise parameter used for encryption.

For a specific η -bit odd integer p, the following distribution over γ -bit integers is used:

$$D_{\gamma,\rho}(p) = \{\text{Randomly choose } q_i \leftarrow \mathbb{Z} \cap [0, 2^\gamma / p),$$

$$r_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \text{Output } x_i = p \cdot q_i + r_i\}$$

Where p is the secret key and is a set of all integers.

A homomorphic public key encryption scheme consists of four algorithms which are KeyGen, Encrypt, Decrypt, and Evaluate. The algorithm Evaluate takes as input a public key pk, a circuit C, a tuple of ciphertexts $c=\langle c_1,c_2,...,c_t\rangle$, and output another ciphertext c.

KeyGen: Generate a random odd integer p of η bits. For $0\le i\le\tau$, sample $x_i\leftarrow D_{\gamma,\rho}(p)$. Relabel so that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let $pk=\langle x_1,x_2,...,x_\tau\rangle$ and $sk=p$.

Encrypt $(pk,m\in\{0,1\})$**.** Given a data item m which is either 0 or 1 to be encrypted, choose a random subset $S\subseteq\{1,2,...,\tau\}$ and a random integer $r\in(-2^{\rho'},2^{\rho'})$, and output the ciphertext:

$$c=\left[m+2r+2\sum_{i\in S}x_i\right]_{x_0} \quad (1)$$

Evaluate $(pk,C,c_1,c_2,...,c_t)$ Given a circuit C with t input bits, and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, perform all the additions and multiplications over integers, and return the resulting in integers.

Decrypt (sk,c): Output $m\leftarrow(c \bmod p)\bmod 2$.

The scheme is homomorphic under addition and multiplication. Suppose $c_1(m_1)$ and $c_2(m_2)$ be ciphertexts of plaintexts m_1 and m_2.

2.1.1 The Homomorphic Addition

$$c_1(m_1)+c_2(m_2)=(m_1+2r_1+pq_1)+(m_2+2r_2+pq_2)$$

and the decryption

$$\begin{aligned}&c^{-1}\left(c_1(m_1)+c_2(m_2)\right)\\&=\left(\left((m_1+m_2)+2r'+2pq'\right)\bmod p\right)\bmod 2\\&=m_1+m_2\end{aligned}$$

2.1.2 The Homomorphic Multiplication

$$c_1(m_1)\cdot c_2(m_2)=(m_1+2r_1+pq_1)\cdot(m_2+2r_2+pq_2)$$

and the decryption

$$\begin{aligned}&c^{-1}\left(c_1(m_1)\cdot c_2(m_2)\right)\\&=\left(\left(m_1.m_2+2f(m_1,m_2,r_1,r_2)+2pf(m_1,m_2,r_1,r_2,q_1,q_2)\right)\bmod p\right)\\&\bmod 2\\&=m_1.m_2\end{aligned}$$

The scheme is semantically secure as shown in [24] under the approximate-Greatest Common Divisors (GCD) assumption. The Definition of approximate GCD is given as below.

Definition 1: The (ρ,η,γ)-approximate-GCD problem is: For a random η-bit odd integer p, given polynomially many samples from $D_{\gamma,\rho}(p)$, output p.

2.2 FHE Scheme over Integers with Shorter Public Keys

The technique proposed by Coron et al. [7] works with integers x'_{ij} of the form $x'_{ij}=x_{i,0}\cdot x_{j,1} \bmod x_0$ for $1\le i,j\le\beta$ where β is a new parameter. Only 2β integers $x_{i,b}$ need to be stored in the public key in order to generate $\tau=\beta^2$ integers x'_{ij} used for encryption. In other words, a quadratic form is used for encryption in the public key elements instead of a linear form. This process will reduce the public key size from the size of τ to the size of $2\sqrt{\tau}$ integers of γ bits.

Their technique requires using an error-free x_0, that is, $x_0=q_0\cdot p$ where q_0 is a random square free integer (an integer that is not divisible by perfect square) in $\left[0,2^\gamma/p\right)$. Otherwise the error would grow too large. Furthermore for encryption, they consider a linear combination of x'_{ij}, with coefficients in $\left[0,2^\alpha\right)$ instead of bits; this further reduces the size of the public key. This technique provides the following operations:

KeyGen: Generate a random prime $p\in\left[2^{\eta-1},2^\eta\right)$. Let $x_0=q_0\cdot p$. Generate integers $x_{i,b}$ for $1\le i,j\le\beta$ and $b\in\{0,1\}$:

$$x_{i,b}=pq_{i,b}+r_{i,b}\quad 1\le i\le\beta,\ 0\le b\le 1 \quad (2)$$

where $q_{i,b}$ are random integers in $[0,q_0)$ and $r_{i,b}$ are integers in $\left(-2^{\rho'},2^{\rho'}\right)$. Let $sk=p$ and $pk=\left(x_0,x_{1,0},x_{1,1},...,x_{\beta,0},x_{\beta,1}\right)$.

Encrypt $(pk,m\in\{0,1\})$: Generate a random vector $b=b_{i,j}$ of size $\tau=\beta^2$ and with each element $b_{i,j}$ in $\left[0,2^\alpha\right)$. Generate a random integer r in $\left(-2^{\rho'},2^{\rho'}\right)$. Output the ciphertext:

$$c=m+\left(2r+2\sum_{1\le i,j\le\beta}b_{i,j}\cdot x_{i,j}\cdot x_{j,1}\right)\bmod x_0 \quad (3)$$

Evaluate and Decrypt: These are the same as the ones in the [24], except that the ciphertext is reduced by modulo x_0 after addition and multiplication.

2.3 Comparison of Performance of Both schemes

The scheme proposed by van Dijk et al. has achieved simplicity but less efficiency. To see the improvement in term of public key size, the scheme parameters in [7], [24] and notations like ω and θ as have been implemented to describe the complexity of each parameter setting.

$\rho=\omega(\log\lambda)$ to prevent brute force attacks on the noise.

$\eta=\rho\cdot\theta\left(\lambda\log^2\lambda\right)$ to support the homomorphic operation for evaluating the "squashed decryption circuit".

$\gamma=\omega\left(\eta^2\cdot\log\lambda\right)$ to avoid lattice-based attacks.

$\tau\ge\gamma+\omega(\log\lambda)$ for the reduction to approximate, (GCD).

$\rho'=\rho+\omega(\log\lambda)$ for the secondary noise parameter.

In order to satisfy such constraints, a convenient parameter setting suggested in [24] is as follows:

$$\rho=\lambda,\rho'=2\lambda,\eta=O\left(\lambda^2\right),\gamma=O\left(\lambda^5\right) \text{ and } \tau=\gamma+\lambda.$$

This setting results in a scheme with the public key size of $O(\lambda 10)$. In practice, the bit-length of the public key should be no less than $\gamma = 2^{23}$ bits to avoid lattice attacks. However, the size of the public key generated by this scheme is at least 2^{46} bits, which is too large to be implemented on any real application [7].

This performance has been improved by Coron et al. works. In order to achieve the reduced public key, they have defined some changes in the parameter setting as follows:

$\rho = \omega(\log \lambda)$ to prevent brute force attacks on the noise.

$\eta \geq (2\rho + \alpha) \cdot \theta\left(\lambda \log^2 \lambda\right)$ to support the homomorphic operation for evaluating the "squashed decryption circuit".

$\gamma = \omega\left(\eta^2 \cdot \log \lambda\right)$ to avoid lattice-based attacks.

$\alpha \cdot \beta^2 \geq \gamma + \omega(\log \lambda)$ for the reduction to approximate GCD.

$\rho' = \rho + \omega(\log \lambda)$ for the secondary noise parameter.

To satisfied these conditions, they set $\rho = \lambda, \rho' = 2\lambda, \eta = O\left(\lambda^2\right), \gamma = O\left(\lambda^5\right)$ as in the original scheme, but add up other new parameters such that $\alpha = \lambda, \beta = O\left(\lambda^2\right), \rho' = 4\lambda$. The main difference between the two schemes is that instead of having $\tau = O\left(\lambda^5\right)$ integers x'_i, they only have $2\beta = O\left(\lambda^2\right)$ integers x_i. Thus, the size of the public key is reduced to $O\left(\lambda^7\right)$ instead of $O\left(\lambda^{10}\right)$ [7].

3. The Propose Scheme

In this section, we describe the proposed scheme, the security analysis of the proposed scheme and the proposed protocol to implement the scheme.

3.1 The Proposed Scheme

Our proposed scheme utilizes a combination of symmetric and asymmetric encryption schemes to achieve fully homomorphic properties. This technique is proposed to improve the efficiency of the two schemes introduced in the previous section, as we do not need to generate many public keys for encryption as proposed by other schemes [8], [19], [25], [26]. In our scheme, there are three parties involved which are Data Owner (DO), processor, and receiver. In the scheme as illustrated in Figure 2, two types of keys will be used; public keys and two distinct secret keys where the former is an encryption key whiles the latter are for encryption and decryption. To encrypt the original data item, a secret key will be used instead of public key proposed by other schemes. The encrypted data will be outsourced to a processor together with an encrypted secret key that has been used to encrypt the data. Prior the data to be processed, the data need to be re-encrypted using a receiver's public key and decrypted using the encrypted secret key sent by the DO. This process is essential to allow further computation on the ciphertext as during the process, the size of noise attached to the data will be reduced and without disclosed any information about the data to the processor. The rest of the processes in our scheme are similar to Gentry's approach as before.

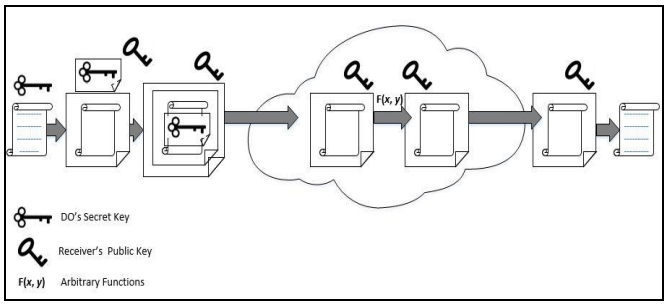

Figure 2. The proposed scheme

Our scheme parameter setting is based on Coron et al.'s scheme introduced as in previous section. Therefore, the size of our public key can be maintained as $O\left(\lambda^7\right)$. Our proposed scheme consists of four algorithms described below.

KeyGen (p, p', x_i): Generates secret keys p and p', which are odd integers such that p and $p' \in \left[2^{\eta-1}, 2^{\eta}\right)$ with η being the bit length of the keys. Generates parameters q_i and r_i randomly such that $q_i \approx 2^{\eta^3}$ and $r_i = 2^{\sqrt{\eta}}$. Generates public keys, $x_i = p'q_i + 2r_i$ for $i = 1, 2, ..., n$ where n is the number of the squashing processes required in the computation to get the desired result.

Encrypt $(p, m \in \{0,1\}, x_i)$: Suppose m be a plaintext such that $m \in \in \{0,1\}$. m is encrypted using a symmetric encryption scheme as in [24].

$$c(m) = pq_i + 2r_i + m \text{ such that } c(m) \in \quad (4)$$

The secret key p is encrypted by DO using a receiver's public key, x_1.

$$\overline{c(p)} = x_1 + p, \qquad \text{s.t. } \overline{c(p)} \in \quad (5)$$

Re-encrypt and decrypt $c(m), x_1, \overline{c(p)}$: The processor will re-encrypt the encrypted data using a receiver's public key. Then, decrypt it using the encrypted secret key sent by the DO. This process is called squashing and it will be repeated along the computation on encrypted data to maintain a small amount of noise attached to the data.

$\overline{c(m)} = \overline{c(c(m))} *_d \overline{c(p)}$, see Definition 2.

$$= (c(m) + x_1) *_d (p + x_1)$$
$$= (c(m) *_d p) + x_1$$
$$= m + x_1$$

Definition 2: Let a and b be integers such that a is a secret key to encrypt a plaintext x and $b = c_a(x)$ is a ciphertext, we then have $a *_d b = c_a^{-1}(b) = x$

Evaluate $\left(C, \overline{c_1(m)}, \overline{c_2(m)}, ..., \overline{c_n(m)}\right)$: Suppose C is a Boolean circuit and $\left\langle \overline{c_1(m)}, \overline{c_2(m)}, ..., \overline{c_n(m)} \right\rangle$ be a tuple of ciphertext. Then, ciphertexts can be evaluated by C as long as the noise growth in the ciphertexts does not exceed the threshold. Otherwise the decryption will be failed [18]. This noise growth can be maintained small by implementing re-encrypt and decrypt on the ciphertext along the computation on encrypted data.

Decrypt $\left(\overline{c(C(m))}, p'\right)$: $\overline{c(C(m))}$ is decrypted using p' (a secret key that poses by the receiver) as below:

$$C(m) = \left(\overline{c(C(m))} \bmod p'\right) \bmod 2 \qquad (6)$$

3.2 Scheme Security

Our scheme's security is based on the hardness of approximation GCD. The scheme is semantically secure under the stronger error-free approximate GCD assumption. The definition of error-free approximate GCD is given by the Definition 3. The proof follows the same strategy as in [24]. For the two specific integers p and q, we define the distribution as in [7]:

$$D'_{\rho}(p, q_0) = \{\text{Randomly choose } q \leftarrow \square \cap [0, q_0),$$

$$r \leftarrow \square \cap (-2^{\rho}, 2^{\rho}) : \text{Output } x = p \cdot q + r\}$$

Definition 3: (*Error-free approximate GCD*)
For a random η-bit prime p, $y_0 = q_0 \ldots p$ where q_0 is a random integer in $[0, 2^{\gamma} / p)$ and polynomially many samples from $D_{\rho}(p, q_0)$, then output p.

Based on [27], the encryption of zero and the encryption of a random message by using the public key encryption instead of the oracle is hard to be differentiated. Thus, the scheme is said to be semantically secure under the stronger error-free approximate GCD problem.

3.3 The Proposed Protocol

This protocol has been designed according to the implementation of the proposed FHE scheme in cloud environment. Three parties involved in the protocol which are a group of Data Owner (DO), Cloud Service Provider (CSP), and a receiver. DO is responsible as data contributor with limited computing resources and storage spaces, CSP is the processor, and the receiver is as a data user. In this protocol, a lot of computing resources and storage spaces are required to process the data contributed by DO. Thus, DO need to outsource the data to the CSP as the CSP provide such facilities as a service. Prior outsourcing the data, the data will be encrypted using the proposed scheme to prevent any information of the data to be disclosed to the CSP and at the same time, the data can be processed in the encrypted form without decryption. Once, the process is completed, result of the process will be sent to the receiver which satisfies a certain policy made by the DO. In the protocol, only the receiver will be able to decrypt the result as the receiver has the decryption key.
The specific details of the proposed protocol will be described as below while the protocol is illustrated in Figure 3.

Stage 1: DO that consists of A_1 and A_2, together generate a secret key Sk_A. They create raw data $V = v_i$ and $W = w_i$ for $i = 1, 2, \ldots, n$, and encrypt them using the secret key, i.e. $c_{Sk_A}(V)$ and $c_{Sk_A}(W)$. They outsource $c_{Sk_A}(V)$ and $c_{Sk_A}(W)$ to the designated cloud storage.

Stage 2: A receiver, B who satisfies a certain policy to use the result of processed data $c_{Sk_A}(V)$ and $c_{Sk_A}(W)$ will outsource their public key to the CSP and keep the corresponding private key by them.

Stage 3: A_1 and A_2 retrieve B's public key from the CSP, use it to encrypt their secret key $c_{Sk_B}(Sk_A)$ and sent it to the CSP.

Stage 4: The CSP encrypts $c_{Sk_A}(V)$ and $c_{Sk_A}(W)$ using Pk_B to produce $c_{Pk_B}\left(c_{Sk_A}(V)\right)$ and $c_{Pk_B}\left(c_{Sk_A}(W)\right)$. The CSP decrypts $c_{Pk_B}\left(c_{Sk_A}(V)\right)$ and $c_{Pk_B}\left(c_{Sk_A}(W)\right)$ homomorphically as below.

$$c_{Pk_B}(Sk_A) *_d c_{Pk_B}\left(c_{Sk_A}(V)\right) = c_{Pk_B}\left(Sk_A *_d c_{Sk_A}(V)\right) = c_{Pk_B}(V)$$

$$c_{Pk_B}(Sk_A) *_d c_{Pk_B}\left(c_{Sk_A}(W)\right) = c_{Pk_B}\left(Sk_A *_d c_{Sk_A}(W)\right) = c_{Pk_B}(W)$$

Then, the CSP computes those data homomorphically.

$$c_{Pk_B}(V) * c_{Pk_B}(W) = c_{Pk_B}(V * W)$$

Stage 5: B retrieves the result from the CSP and decrypts it using B's private key:

$$c_{Sk_B}{}^{-1}\left(c_{Pk_B}(V * W)\right) = V * W$$

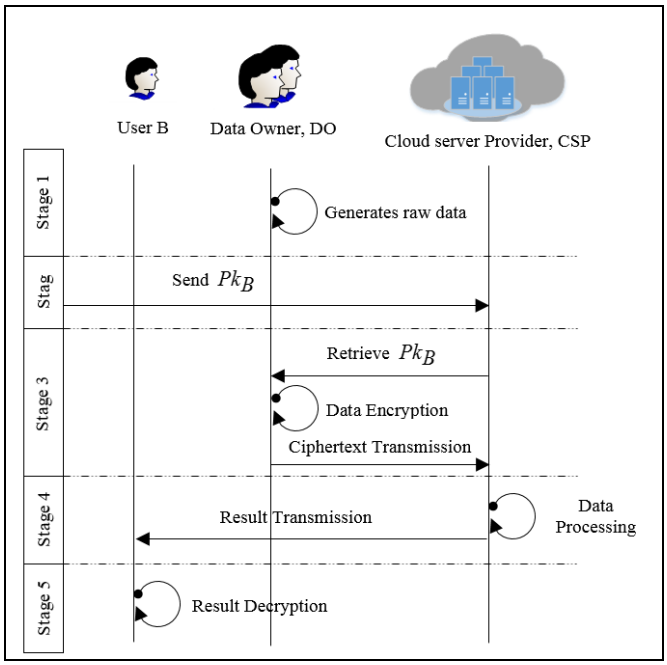

Figure 3. The Proposed Protocol

4. Analysis, Result and Discussion

In this section, we provide the definition of full adder in terms of gates operations. This definition is essential to determine the size of noise during computation on encrypted data. Furthermore, we provide some analysis of the scheme performance according to the number of squashing needed to perform the operations. The results of analysis are illustrated as graphs in Figures 5 and 6.

4.1 Full Adder Definition

In this sub-section, the definition of full adder is provided as the following definition.

Definition 4:
An OR gate can be defined as a combination of XORs and an AND Gates as below:

$$D + E = (D \cdot E) \oplus (D \oplus E) \qquad (7)$$

Definition 5:
Carried output, C_{out} in circuit evaluation (during addition operation) can be illustrated in Figure 4 and defined as below.

$$C_{out} = (A \cdot B) + (C_{IN} \cdot (A \oplus B))$$

$$= (A \cdot B) + (C_{IN} \cdot (A \oplus B)) \oplus (A \cdot B) + (C_{IN} \cdot (A \oplus B))$$

where C_{IN} is a carried input value and S is the output value as shown in Figure 4.

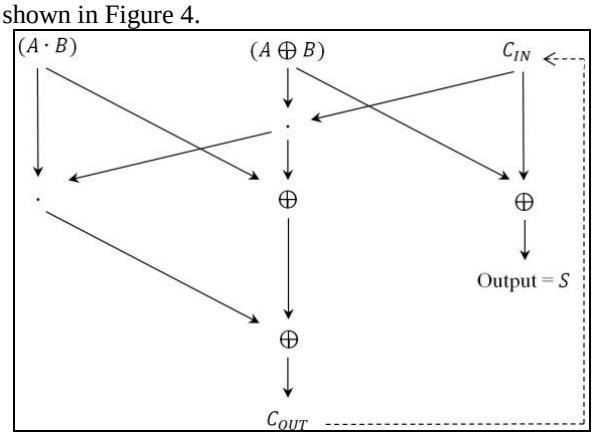

Figure 4. Map Representation

Definition 6:
According to Definition 4 and 5, a single bit addition is defined as below:

$$A + B = A \oplus B \oplus C_{IN}$$
$$= A \oplus B \oplus (A \cdot B) \cdot (C_{IN} \cdot (A \oplus B)) \oplus (A \cdot B) \oplus (C_{IN} \cdot (A \oplus B))$$

5. Result and Discussion

The proposed scheme is believed to provide better efficiency in terms of key generation process as this scheme uses symmetric encryption rather than asymmetric one. The other performance of the scheme's process remains the same as the one in [7]. Furthermore, to provide further analysis on the process involved in this scheme, we have analysed the number of squashing processes required with respect to the noise growth. Too fast noise growth implies frequent squashing operations required and thus degrades the scheme efficiency.

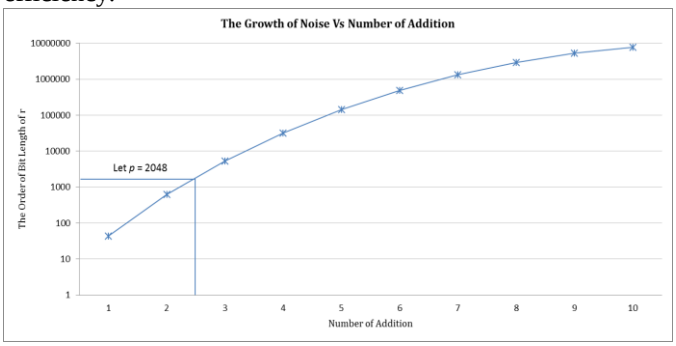

Figure 5. Adding up to 10 16-bits ciphertexts.

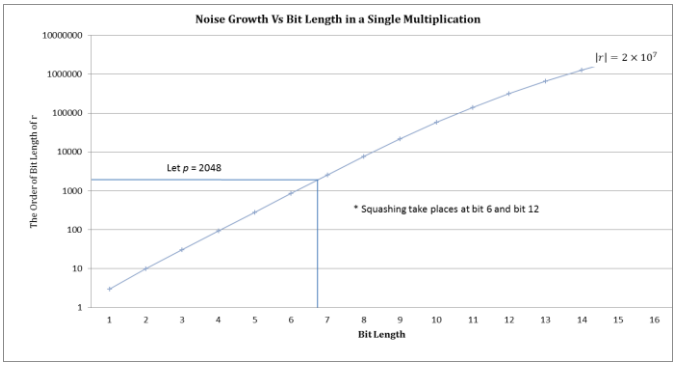

Figure 6. A single multiplication of ciphertexts with the length of up to 16-bits.

To reduce the number of squashing required, we have analysed the implication of noise towards the bit length of ciphertext and the result is shown in Figures 5 and 6.

Figure 5 shows the growth of noise based on the sum of up to 10 16-bits ciphertexts. In this example, we have chosen a secret key of 2048 bits. After the sum of two ciphertexts, the noise has grown to nearly the size of the secret key. Once it exceeds the threshold, then the decryption will fail. Thus, the squashing technique is required to reduce the size of the noise to the size of the original ciphertext. To continue this process, the result of every sum needs to be squashed in order to have a correct result at the end of 10 additions on the distinct ciphertexts.

Furthermore, Figure 6 shows the growth of noise during the product of two 16-bits ciphertexts with the same size of the secret key. After summing the first 6 bits of the ciphertexts, the noise is growing to nearly the size of the secret key. Once it exceeds the size of the key, the same problem happens as for the addition. Thus, squashing is needed to reduce the size of noise to the size of the first bit summation to enable the completion of the multiplication.

Based on the analysis, we can conclude that the number of squashing required in computing on encrypted data is depend on the size of the noise growth with respect to the size of the secret key. In addition, both noise and squashing are very essential to be implemented to provide a secure FHE scheme. However, squashing technique is time-consuming and it should be carefully managed. Thus, in our proposed scheme, we have designed the scheme so that this technique will be done efficiently by the CSP as the CSP has a lot of computing resource. By doing so, we can achieve a FHE scheme with better efficiency while the security of the data is preserved.

6. Conclusions

We have proposed an improved FHE scheme which combines symmetric and asymmetric encryptions. Our scheme uses a symmetric key for encryption, so it offers better efficiency by successfully avoiding the time consuming process for generating many public keys in the most existing schemes. To implement the proposed scheme, we have proposed a protocol which has been designed to operate in cloud environments. Furthermore, we have provided a new analysis of the effect of noise during operations on encrypted data. This result will determine the number of squashing processes required to complete the computation on the encrypted data. In order to provide an efficient FHE scheme, the number of squashing processes should be considered because such a process is time-consuming and thus affects the efficiency of the scheme.

7. Acknowledgement

This work has been supported under Universiti Teknikal Malysia Melaka research grant Gluar/CSM/2016/FTMK-CACT/100013. The authors would like to thank to Universiti Teknikal Malaysia Melaka, CyberSecurity Malaysia and all members of INSFORNET research group for their incredible supports in this project.

References

[1] S. Marston, "Cloud Computing – The Business Perspective," *Sciences-New York*, pp. 1–11, 2011.

[2] G. V. Mini and K. S. A. Viji, "A Comprehensive Cloud Security Model with Enhanced Key Management , Access Control and Data Anonymization Features," *International Journal of Communication Networks and Information Security*, vol. 9, no. 2, pp. 263–273, 2017.

[3] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[4] M. R. Baharon, Q. Shi, D. Llewellyn-Jones, and M. Merabti, "Efficient and secure remote data storing and processing," *European Conference on Information Warfare and Security, ECCWS*, pp. 396–401, 2013.

[5] C. Gentry and S. Halevi, "Implementing Gentry ' s Fully-Homomorphic Encryption Scheme," *30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1–29, 2011.

[6] D. Stehlé and R. Steinfeld, "Faster fully homomorphic encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2010, pp. 377–394.

[7] A. Mandal, D. Naccache, and M. Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter Public Keys," *Proceedings of the 31st annual conference on Advances in Cryptology*, pp. 487–504, 2011.

[8] M. Tibouchi, "Batch Fully Homomorphic Encryption over the Integers," *Lecture Notes in Computer Science*, vol. 7881, pp. 315–355, 2013.

[9] J. Kim, M. S. Lee, A. Yun, and J. H. Cheon, "CRT-based Fully Homomorphic Encryption over the Integers," *IACR Cryptology ePrint Archive 2013*, pp. 1–18, 2013.

[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12*, pp. 309–325, 2012.

[11] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *Proceedings of the 15th international conference on Practice and Theory in Public Key Cryptography*, pp. 1–16, 2012.

[12] G. Davis, "Processing Encrypted Data," *Communications of the ACM.*, vol. 30, no. 9, pp. 777–780, 1987.

[13] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[14] D. Boneh, "Evaluating 2-DNF Formulas on Ciphertexts," *Second Theory of Cryptography Conference, TCC 2005, Cambridge Proceedings*, pp. 325–341, 2005.

[15] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," *Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11*, pp. 113–124, 2011.

[16] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. Dissertation, Stanford University, 2009.

[17] N. P. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," *Proceedings of the 13th international conference on Practice and Theory in Public Key Cryptography*, pp. 420–443, 2010.

[18] M. Tibouchi, "Scale-Invariant Fully Homomorphic Encryption over the Integers," *Lecture Notes in Computer Science*, vol. 8383, pp. 311–328, 2014.

[19] Z. Brakerski, "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP," *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology*, vol. 7417, pp. 868–886, 2012.

[20] C. Gentry and S. Halevi, "Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits," *IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 107–109, Oct. 2011.

[21] Y. Doröz, E. Öztürk, and B. Sunar, "A million-bit multiplier architecture for fully homomorphic encryption," *Microprocessors and Microsystems*, vol. 38, no. 8, pp. 766–775, 2014.

[22] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things : A systematic review of the literature and recommendations for future research," *Journal of Network and Computer Applications*, vol. 97, no. April, pp. 23–34, 2017.

[23] P. Karu, "Practical Comparison of Fast Public-key Cryptosystems," *Telecommunications Software and Multimedia Lab. at Helsinki Univ. of Technology, Seminar on Network Security.*, pp. 1–18, 2001.

[24] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43, 2010.

[25] M. R. Baharon, Q. Shi, and D. Llewellyn-jones, "A New Lightweight Homomorphic Encryption Scheme for Mobile Cloud Computing," *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 618–625, 2015.

[26] J. H. Cheon, H. Hong, M. S. Lee, and H. Ryu, "The Polynomial Approximate Common Divisor Problem and its Application to the Fully Homomorphic Encryption," *Information Sciences*, vol. 326, pp. 41–58, 2016.

[27] Z. Chen, "Fully homomorphic encryption," *University of Wollongong*, pp. 1–77, 2017.