On DGHV and BGV Fully Homomorphic Encryption Schemes

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

[1]Lebanese University-Faculty of Engineering- CRSI, Hadath, Lebanon
[2]Saint Joseph University-Faculty of Engineering-CMTI, Mar Roukoz, Lebanon
khalil.hariss@net.usj.edu.lb; Maroun.chamoun@usj.edu.lb; samhat@ul.edu.lb

***Abstract*—Homomorphic Encryption (HE) is a new cryptographic topic that allows untrusted parties to compute over encrypted data. This new encryption scheme is very famous in a cloud scenario, because it leverages cryptographic techniques in the cloud by allowing it to store encrypted data, and to process encrypted query over it. In this paper, we consider two important HE schemes: Dijk-Gentry-Halevi-Vaikutanathan (DGHV) and, Brakerski-Gentry-Vaikunathan (BV-BGV). The first one is based on computing over real integers while the other one is based on *LWE* (Lattice based Encryption). We present a detailed explanation of both schemes showing their efficient techniques including Bootstrapping and Modulus switching and we finally draw several conclusions.**

Keywords—HE, FHE, DGHV, BGV, BV, LWE.

I. INTRODUCTION

Homomorphic Encryption (HE) is a modern cryptographic technique that will be used for cloud systems. HE provides privacy and protection in the cloud setting. Users storing their encrypted data at the cloud side, can send encrypted queries to the cloud and the latter is able to perform encrypted queries over encrypted data. In addition to cloud querying, HE can be used in a lot of real world modern applications such as e-Voting, Spam filters, Attribute Based Encryption (ABE)[1], Data aggregation [2], etc. Fig. 1 shows an implementation of HE in real world applications. In 1978, Rivest et al.[3] introduced the notion of Privacy Homomorphism that allows partial computations over encrypted data. Since this date, designing and implementing a Fully Homomorphic Encryption (FHE) scheme that allows full computation over encrypted data, became a big challenge for researchers and cryptographers. Several works had considered the design and implementation of such scheme. Pallier Encryption Scheme [4] is an additive homomorphic encryption scheme that allows only add operations over encrypted data, Domingo Ferrer Crypto-system [5], [6] is a FHE scheme based on polynomial calculations. MORE, Enhanced MORE [7], [8] and PORE [9], are FHE schemes based on linear equations. The most valued work was introduced by Craig Gentry in 2009 [10], [11]. Several works followed Gentry's contribution and the most two important schemes are: The DGHV which is an asymmetric encryption scheme based on the computation over real integers ([12], [13]). The basic DGHV scheme provides the homomorphic behavior for a limited circuit depth. Bootstrapping ([12], [13]) is a technique introduced in the literature to make the scheme fully homomorphic and supports unlimited circuit depth. Another important Encryption scheme is the BGV scheme ([16][17]). It is based on the hardness of Learning With Error (LWE). The basic scheme of BGV is homomorphic for a limited number of circuit depth. Modulus switching is a new technique used to make the scheme fully homomorphic and supports an unlimited circuit depth. But DGHV and BGV are not practical for real world applications due to storage overhead.
The rest of this paper is decomposed as follow: In section II, we introduce the properties of FHE as well as the definition of the asymmetric homomorphic encryption scheme. In section III, we present in detail DGHV scheme in both symmetric and asymmetric scenarios. BV-BGV schemes are discussed in section IV. Finally, we conclude this paper by a comparison between DGHV and BV-BGV in section V.

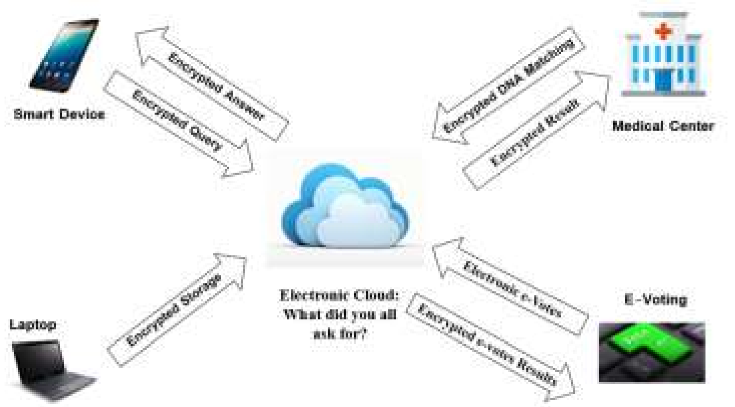


Fig. 1. Homomorphic implementation

II. HOMOMORPHIC ENCRYPTION

We give in this section an explanation about homomorphic encryption mainly for an asymmetric scenario.

A. Homomorphic Scheme

An asymmetric homomorphic encryption scheme ε is defined by 4 functions:

- $KeyGen_\varepsilon()$: generates pair of keys, public key pk and secret key sk.
- $Encrypt_\varepsilon(pk, \pi_i)$: outputs a cipher-text ψ_i, where π_i is a plain-text.
- $Decrypt_\varepsilon(sk, \psi_i)$: outputs the plain text π_i, where ψ_i is a cipher-text.
- $Evaluate_\varepsilon$: This function takes as input the public key pk, a circuit C and a tuple of cipher-texts Ψ
  $\Psi = (\psi_1, \psi_2, \psi_3, \ldots\ldots, \psi_t)$.

The evaluation function outputs a cipher-text ψ.

$$\psi = Evaluate_{\varepsilon}(pk, C, \Psi) \quad (1)$$

The scheme is homomorphic if
$\psi = Encrypt_{\varepsilon}(pk, C(\pi_1, \pi_2, \pi_3, \dots, \pi_t))$.

B. Homomorphic Properties

As written in equation (1), we are computing any function f, that the circuit C performs, over the encrypted data. Any function f can be written as Boolean function. Any Boolean is written as a polynomial form and since any polynomial form is a set of addition and multiplication operations, to build a FHE scheme we should satisfy the two following homomorphic properties:

1) Addition:

$$Enc_{\varepsilon}(pk, x_1) + Enc_{\varepsilon}(pk, x_2) = Enc_{\varepsilon}(pk, x_1 + x_2) \quad (2)$$

2) Multiplication:

$$Enc_{\varepsilon}(pk, x_1) * Enc_{\varepsilon}(pk, x_2) = Enc_{\varepsilon}(pk, x_1 * x_2) \quad (3)$$

Where x_1, x_2 are two plain-texts in $\{0,1\}$ and *Encε(pk,x)* is the Encryption function.

III. DGHV SCHEME

This scheme is built upon integers, the first step to build DGHV is to consider the following symmetric scheme:

A. Symmetric scheme:

1) The basic symmetric scheme

The secret key is an odd integer, chosen from some interval $p \in [2^{\eta-1}, 2^{\eta})$ where η is the bit length of the secret key.

The cipher-text of a plain-text $m \in \{0,1\}$ is given by:

$$c = Encrypt(p, m) = pq + 2r + m \quad (4)$$

Where the integers q, r are chosen randomly in some other intervals, such that $2r$ is smaller than $p/2$ in absolute value. In this scheme, the cipher-text is an integer whose residue $mod\ p$ has the same parity of the plaintext m.

$$Decrypt(p, m) = (c\ modp)mod2 \quad (5)$$

The decryption works as long as the noise r is much smaller than p.

Computing the decryption equation is also possible by the following equation:

$$m' \leftarrow [c - \lfloor c/p \rceil]_2 \quad (6)$$

2) Homomorphic behavior:

Suppose that we have two plain-texts m_1, m_2, with two cipher-texts respectively $c_1 = pq_1 + 2r_1 + m_1$, $c_2 = pq_2 + 2r_2 + m_2$.

a) Addition:

$$c_1 + c_2 = p(q_1 + q_2) + 2(r_1 + r_2) + m_1 + m_2 \quad (7)$$

Decryption works as long as $2(r_1 + r_2) \in [-p/2, p/2]$.

b) Multiplication:

$$\begin{aligned} c_1 * c_2 = p(&pq_1q_2 + 2r_2q_1 + q_1m_2 + 2q_2r_1 \\ &+ m_1q_2) \\ &+ 2(2r_1r_2 + m_2r_1 + m_1r_2) \\ &+ m_1m_2. \end{aligned} \quad (8)$$

Decryption works as long as $2(2r_1r_2 + m_2r_1 + m_1r_2) \in [-p/2, p/2]$.

This scheme seems to be "Somewhat homomorphic", it can be used to evaluate circuits with limited depth. To make the scheme fully homomorphic by reducing the noise level, bootstrapping is a new technique ([12], [13]) introduced in the literature.

B. Asymmetric scheme:

1) Secret parameters:

The DGHV scheme is built based on the symmetric scheme given in section A, using six different parameters as follow and Table I shows the different values:

λ: Security parameter used by KeyGen.
γ: is the bit length of the integer in the public key.
η: is the bit length of the secret key.
ρ: is the bit length of the noise.
τ: is the number of integer in the public key.
ρ': Secondary noise parameter.

TABLE I. SECURITY PARAMETERS

parameter	**value**
ρ	$\omega(\log\lambda)$
η	$\rho.\Theta(\lambda log^2\lambda)$
γ	$\omega(\eta^2 \log\lambda)$
τ	$\gamma + \omega\ (log\lambda)$
ρ'	$\rho + \omega(\log\lambda)$

2) Scheme Construction:

For a specific ($\eta\ bits$) odd positive integer p, we use the following distribution over γ-bit integers:

$$D_{\gamma,\rho}(p) = \{choose\ q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p), \\ r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho}): output\ x = pq + r\}$$

$KeyGen(\lambda)$: The secret key is an odd η-bit integer:

$$p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^{\eta}).$$

For the public key, sample $x_i \leftarrow D_{\gamma,p}$ for $i = 0, \dots\dots, \tau$. Relabel so that x_0 is the largest. Restart unless x_0 is odd and $r_p(x_0)$ is even. The public key is $pk = \langle x_0, x_1, \dots\dots, x_\tau \rangle$.

$Encrypt(pk, m\epsilon\{0,1\})$. Choose a random subset $S \subseteq \{1,2,\dots.,\tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and output

$$c \leftarrow [m + 2r + 2\sum\nolimits_{i\epsilon S} x_i]_{x_0} \quad (9)$$

$Evaluate(pk, C, c_1, c_2, \dots\dots, c_t)$: Given the (binary) circuit C with t inputs, and t ciphertexts c_i, apply the (integer) addition and multiplication gates of C to the cipher-texts, performing all the operations over the integers, and return the resulting integer.

$$Decrypt(sk, c)\text{: Output } m' \leftarrow (c \bmod p) \bmod 2. \quad (10)$$

Remark: Decryption can also be written as:
$m' \leftarrow [c - \lfloor {}^{c}/_{p} \rceil]_2$.

3) Homomorphic properties:
The homomorphic properties of the asymmetric scheme is interpreted as in section (A.2) for the symmetric one.

C. Making the scheme fully homomorphic:

1) Bootstrapping:
A bootstrappable encryption scheme is an encryption scheme that can evaluate its own decryption circuit D [12][13]. We apply equation (1) on the encryption scheme, but instead of using the circuit C, we use the decryption circuit D.
Suppose that ε is bootstrappable with plaintext space P which is {0,1}, and the circuit C is Boolean. And we have a cipher-text ψ_1 that encrypts π under pk_1 which we want to refresh. Suppose that we can decrypt it homomorphically, and we also have sk_1 the secret key for pk_1. sk_1 is encrypted under a second public key pk_2: let $\overline{sk_{1J}}$ be the encrypted secret key bits. Consider the following algorithm.

$$Recrypt_{\varepsilon}(pk_2, D_{\varepsilon}, \langle \overline{sk_{1J}} \rangle, \psi_{1j}).$$

$$Set\ \overline{\psi_{1J}} \overset{R}{\leftarrow} Encrypt\ (pk_2, \psi_{1j}) \quad (11)$$

$$Output\ \psi_2\ \leftarrow\ Evaluate_{\varepsilon}(pk_2, D_{\varepsilon}, \langle \langle \overline{sk_{1J}} \rangle, \langle \overline{\psi_{1J}} \rangle \rangle)$$

Above, $Evaluate$ takes as input the bits of sk_1 and ψ_1, each encrypted under pk_2. Then ε is used to evaluate the decryption circuit homomorphically. The output ψ_2 is thus an encryption under pk_2 of $Decrypt_{\varepsilon}(sk_1, \psi_1) = \pi$. Applying the decryption circuit D_{ε} removes the error vector associated to the first cipher-text, but $Evaluate_{\varepsilon}$ simultaneously introduces a new vector error. Intuitively, we have made progress as long as the second error vector is shorter. Bootstrapping is possible if decryption can be expressed as an arithmetic circuit of low depth.

2) Squashing the decryption circuit
Squashing the decryption circuit is a transformation applied over the decryption circuit to make it an arithmetic circuit of low depth [11][12],[13]. In this transformation, we add to the public key some extra information about the secret key, and we use extra information to "post process" the cipher-text. The post-processed cipher-text can be decrypted more efficiently than the original cipher-text, thus making the scheme bootstrappable. But this requires a larger cipher-text, and also introducing another hardness assumption.
Let κ, θ, Θ be three more parameters, which are function of λ. We use κ=${}^{\gamma\eta}/_{\rho'}$, θ=λ and Θ=ω($\kappa.\log\lambda$). For a secret key $sk^* = p$ and a public key pk^* from the original somewhat homomorphic scheme ε. We add to the public key a set $\vec{y} = \{y_1, y_2, \dots \dots, y_{\Theta}\}$ of rational numbers in [0,2) with κ bits of precision, such that there is a sparse subset $S \subset \{1,2,3, \dots, \Theta\}$ of size θ with $\sum_{i \epsilon S} y_i \approx {}^{1}/_{p}\ (mod\ 2)$. We also replace the secret key by the indicator vector of the subset S. In more details, we modify the encryption scheme from section B as follows:

a) $KeyGen.$: Generates $sk^* = p$ and pk^* as before. Set $x_p \leftarrow \lfloor {}^{2^{\kappa}}/_{p} \rceil$, choose at random a Θ bit vector with hamming weight θ, $\vec{s} = \langle s_1, s_2, \dots., s_{\Theta} \rangle$, and let $S = \{i{:}\ s_i = 1\}$.
Choose at random integers $u_i \epsilon Z \cap [0, 2^{\kappa+1}), i = 1, \dots\dots, \Theta$, subject to the condition that $\sum_{i\ \epsilon S} u_i = x_p\ (mod\ 2^{\kappa+1})$. Set $y_i = {}^{u_i}/_{2^{\kappa}}$ and $\vec{y} = \{y_1, y_2, \dots\dots, y_{\Theta}\}$. Hence each y_i is a positive number smaller than two, with κ bits of precison after the binary point. Also $[\sum_{i \epsilon S} y_i]_2 = ({}^{1}/_{p}) - \Delta_p$ for some $|\Delta_p| < 2^{-\kappa}$.
Output the secret key $sk = \vec{s}$ and public key $pk = (pk^*, \vec{y})$.

b) $Encrypt$ and $Evaluate$: Generate a cipher-text c^* as before. Then for $i\ \epsilon\ \{1,2,3, \dots., \Theta\}$, 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 $\vec{z} = \langle z_1, z_2, \dots\dots, z_{\Theta} \rangle$.

c) $Decrypt$: outputs

$$m' \leftarrow [c^* - \lfloor \sum_{i \epsilon S} s_i z_i \rceil]_2. \quad (12)$$

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

e) Proof: Fix a public and secret keys, generated with respect to 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. Recall that the $y_i's$ were chosen so that

$$[\textstyle\sum_i s_i y_i]_2 = ({}^{1}/_{p}) - \Delta_p \text{ with } |\Delta_p| \leqslant 2^{-\kappa}.$$

Fix an electric circuit C, and t cipher-texts $\{c_i\}_{i=1}^{t}$ that encrypt the inputs to C, and denote
$c^* = Evaluate(pk, C, c_1, c_2, \dots, c_t)$,
we need to establish that

$$\lfloor {}^{c^*}/_{p} \rceil = \lfloor \sum_i s_i z_i \rceil\ (mod\ 2) \quad (13)$$

Where the z_i's are computed as $[c^*.y_i]_2$ with only $\lceil \log\theta \rceil$+3 bits of precision after the binary point, so $[c^*.y_i]_2 = z_i - \Delta_i$ with $|\Delta_i| \leqslant {}^{1}/_{16\theta}$.

We have:

$$\left[({}^{c^*}/_{p}) - \sum s_i z_i\right]_2 = \left[({}^{c^*}/_{p}) - \sum s_i [c^*.y_i]_2 + \sum s_i \Delta_i\right]_2$$

$$= \left[({}^{c^*}/_{p}) - c^*.\left[\sum s_i y_i\right]_2 + \sum s_i \Delta_i\right]_2$$

$$= \left[({}^{c^*}/_{p}) - c^*.({}^{1}/_{p} - \Delta_p) + \sum s_i \Delta_i\right]_2$$

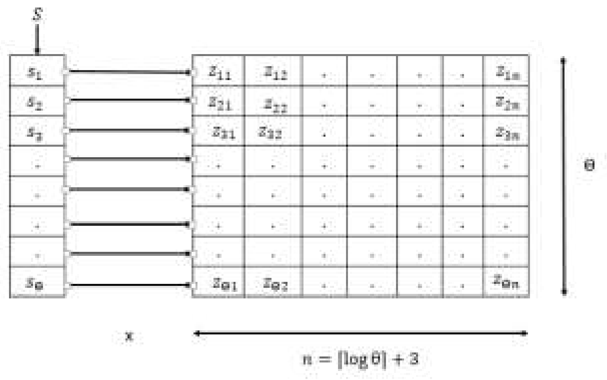
$$= \left[c^*.\Delta_p + \sum s_i\Delta_i\right]_2$$

We claim that the final quantity inside the brackets has magnitude at most $1/_8$. By definition, since c^* is a valid ciphertext output, the value $c^*/_p$ is within $1/_8$ of an integer. Together, these facts imply the lemma.

To establish the claim, observe that $|\sum s_i\Delta_i| \leqslant \theta.\frac{1}{16\theta} = 1/_{16}$. Regarding $c^*\Delta_p$, recall that the output cipher-text c^* is obtained by evaluating the circuit C on the cipher-text c_i As listed in [12], for any polynomial P that implements the cicuit C, for any α≥1, if P's input have magnitude at most $2^{\alpha(\rho'+2)}$, its output has magnitude at most $2^{\alpha(\eta-4)}$. In particular, when P's input are fresh ciphertext, 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, $|c^*\Delta_p| < 1/_{16}$ and the claim follows.

3) Practical Example of Bootstrapping and Squashing:

After Squashing the decryption circuit, the latter have the form given in equation (12). $\sum_{i\epsilon S} s_i z_i$ is implemented as given in Fig. 2.



S
s_1
s_2
s_3
s_Θ
z_{11} z_{12} z_{1n}
z_{21} z_{22} z_{2n}
z_{31} z_{32} z_{3n}
$z_{\Theta1}$ $z_{\Theta2}$ $z_{\Theta n}$
Θ
x
n = ⌈log θ⌉ + 3

Fig. 2. Multiplication after Sqaushing

In [14][15] a multiplication with lower computational complexity than the one given in Fig. 2 is explained. In this new implementation the secret key $\vec{s}$ is divided in θ boxes of $B = \Theta/_\theta$ bits each, such that each box has a single 1-bit in it. This enables to obtain a grade-school addition algorithm that requires $O(\theta^2)$ multiplications of bits instead of $O(\Theta.\theta)$.

The secret key $\vec{s}$ is divided into $s_{k,i}$ the i^{th} secret key bit in box k, where $1 \leqslant k \leqslant \theta$ and $1 \leqslant i \leqslant B$. In this case we have this equation:

$$m \leftarrow c^* - \lfloor \sum_{k=1}^{\theta} (\sum_{i=1}^{B} s_{k,i}z_{k,i}) \rceil \, mod(2) \qquad (14)$$

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

$$m \leftarrow c^* - \lfloor \sum_{i=1}^{\theta} (q_k) \rceil \, mod\,(2) \qquad (15)$$

Where the q_k's are rational in [0,2) with n bits of precision after the binary point.
Based on decryption equations above, decryption can be written as:

$$m = [c^* - \lfloor c^*/_p \rceil]_2 = [c^* - [\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil]_2 \qquad (16)$$

The same equation can be written as:

$$m = [c^*]_2 \oplus [\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil]_2 \qquad (17)$$

The parity of m is related to the parity of c^* and the parity of $\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil$, during bootstrapping we should conserve the parity of $\lfloor \sum_{i=1}^{\Theta} s_i z_i \rceil$.
We show a toy implementation of bootstrapping using low values just to clarify the ideas.
Suppose that we have $\Theta = 4$ and $\theta = 2$. In this case we have $B = \Theta/_\theta = 2$, and precision bit: $n = 1$.
$\vec{s} = [b_1\ b_2\ b_3\ b_4]$,
$((b_1 = 1\ and\ b_2 = 0)$ or $(b_1 = 0\ and\ b_2 = 1))$.
$((b_3 = 1\ and\ b_4 = 0)$ or $(b_4 = 0\ and\ b_3 = 1))$.
$z = [[1\ 0], [0\ 1], [1\ 1], [0\ 1]]$.
Let us try to evaluate the decryption circuit using the secret key $\vec{s}$ without encryption.
$s_1 = [s_{11}\ s_{12}] = [b_1,\ b_2]$ $s_2 = [s_{21}\ s_{22}] = [b_3\ , b_4]$
$z_1 = [z_{11}\ z_{12}] = [[1\ 0], [0\ 1]]$
$z_2 = [z_{21}\ z_{22}] = [[1\ 1], [0\ 1]]$.
$Sum = \sum_{k=1}^{\theta}\sum_{i=1}^{B} s_{k,i}z_{k,i} = \sum_{k=1}^{2}\sum_{i=1}^{2} s_{k,i}z_{k,i} =$
$b_1z_{11} + b_2z_{12} + b_3z_{21} + b_4z_{22}$.
$sum1 = b_1z_{11} + b_2z_{12} = [b_1,\ b_2]$ (b_1 is the real part, b_2 is the decimal part).
$sum2 = b_3z_{21} + b_4z_{22} = [b_3,\ b_3 + b_4]$ (b_3 is the real part, $b_3 + b_4$ is the decimal part).
To add $sum1$ and $sum2$, we apply the binary addition listed in lemma 7 of [14]. The result is
$sum = sum1 + sum2 = [b_1 + b_3,\ b_2 + b_3 + b_4 + b_1b_3]$
($b_1 + b_2$ is the real part and $b_2 + b_3 + b_4 + b_1b_3$ is the decimal part).
The parity of $\lfloor sum \rceil = \lfloor \sum_i s_i z_i \rceil$ is given by

$$[b_1 + b_3 +\ b_2 + b_3 + b_4 + b_1b_3]_2$$

Let us repeat the same procedure, but this time we evaluate the decryption circuit using the encryption of the secret key $\vec{s}$.
For simplicity suppose that the largest public key x_0 is pq_0.

$se_1 = Enc(s_1) = [pq_1 + 2r_1 + b_1\ , pq_2 + 2r_2 + b_2]$
$se_2 = Enc(s_2) = [pq_3 + 2r_3 + b_3\ , pq_4 + 2r_4 + b_4]$
$r_i\ \epsilon\ [-2^\rho, 2^\rho]\ i\ \epsilon\{1,2,3,4\}$

$$Sum_{enc} = \sum_{k=1}^{\theta}\sum_{i=1}^{B} se_{k,i}z_{k,i} = \sum_{k=1}^{2}\sum_{i=1}^{2} se_{k,i}z_{k,i} =$$
$$(pq_1 + 2r_1 + b_1)z_{11} + (pq_2 + 2r_2 + b_2)z_{12} + (pq_3 + 2r_3 + b_3)z_{21} + (pq_4 + 2r_4 + b_4)z_{22}.$$

$$sum1_{enc} = (pq_1 + 2r_1 + b_1)z_{11} + (pq_2 + 2r_2 + b_2)z_{12} = [pq_1 + 2r_1 + b_1, pq_2 + 2r_2 + b_2]$$
$$sum2_{enc} = (pq_3 + 2r_3 + b_3)z_{21} + (pq_4 + 2r_4 + b_4)z_{22} = [pq_3 + 2r_3 + b_3, p(q_3 + q_4) + 2(r_3 + r_4) + b_3 + b_4].$$

Also to add $sum1_{enc}, sum2_{enc}$, we apply the same binary addition listed in lemma 7 of [14] and the result is

$$\lfloor sum_{enc} \rceil = \lfloor \sum_i se_i z_i \rceil = [sum1_{enc} + sum2_{enc}]_{x_0} =$$
$[p(q_1 + q_3 - k_1 q_0) + 2(r_1 + r_3) + b_1 + b_3 \quad p(q_2 + q_3 + q_4 - k_2 q_0 + pq_1 q_3 + 2r_3 q_1 + b_2 q_1 + 2rq_3 + b_1 q_3) + 2(r_3 + r_4 + r_1 r_3 + r_1 b_3 + b_1 r_3 + r_2 + b_2 + b_3 + b_4 + b_1 b_3] = [sum^1_{enc} \quad sum^2_{enc}]$, Such that $r_i \epsilon [-2^{\rho}, 2^{\rho}]$, $i \epsilon \{1,2,3,4\}$.

$$ciphertext_{fresh} = \lfloor sum_{enc} \rceil + [c^*]_2 \tag{18}$$

The parity of $ciphertext_{fresh}$ in equation (18) is preserved based on equation (17), the fresh cipher has lower noise than c^* because during bootstrapping the secret key bits are encrypted by sampling the noise from $[-2^{\rho}, 2^{\rho}]$, while the initial cipher-text is obtained by sampling the noise from $\left[-2^{\rho'}, 2^{\rho'}\right]$, given that $\rho < \rho'$ as given in TABLE I.

IV. BV, BGV SCHEMES

The *BV* [16] (Brakerski-Vaikunathan) and *BGV* [17] (Brakerski-Gentry-Vaikunathan) consist of applying the *DGHV* over Lattices. They are two encryption schemes based on the hardness of *LWE (*Learning With Error*)* [18][19]. *LWE* is a lattice based encryption scheme. We first explain the Lattice based encryption scheme [20], then *LWE* and finally we will introduce the *BV, BGV* schemes.

A. Lattices Definition [20]:

1) Given n linearly vectors in R^m $\{b_1, b_2, b_3, \ldots, b_n\}$, a lattice of basis $\{b_1, b_2, b_3, \ldots\ldots, b_n\}$ is defined as linear combination of the n vectors [20].

$$a_1 b_1 + a_2 b_2 + a_3 + \cdots \ldots \ldots + a_n b_n$$

$$a_1 \begin{bmatrix} b_{11} \\ b_{12} \\ \cdot \\ b_{1m} \end{bmatrix} + a_2 \begin{bmatrix} b_{21} \\ b_{22} \\ \cdot \\ b_{2m} \end{bmatrix} + a_3 \begin{bmatrix} b_{31} \\ b_{32} \\ \cdot \\ b_{3m} \end{bmatrix} + \ldots \ldots + a_n \begin{bmatrix} b_{n1} \\ b_{n2} \\ \cdot \\ b_{nm} \end{bmatrix} \tag{19}$$

The matrix representation is given by $L(B) = A.B$ where

$$A = [a_1\ a_2 \ldots . a_n] \epsilon\ Z^{(1,n)},$$

$$B = \begin{bmatrix} b_{11} & b_{12} & \ldots. & \ldots. & b_{1m} \\ b_{21} & b_{22} & \ldots. & \ldots. & b_{2m} \\ . & . & \ldots. & . & \ldots.. \\ . & . & \ldots\ldots. & . & \ldots\ldots. \\ b_{n1} & b_{n2} & \ldots.. & \ldots.. & b_{nm} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{bmatrix} \epsilon R^{(n,m)}$$

$L(B)$ is an additive sub-group of R^m.
The rank of the lattice is n and its dimension is m, we say that the lattice is full rank if $n = m$.

2) Two bases are equivalent when they generate the same lattice.
Two bases B_1 and B_2 are equivalent if and only if $B_2 = B_1 U$, where U is a unimodular matrix, $Det(U) = \pm 1$.
Two bases B_1 and B_2 are equivalent, $Det(B_1) = \pm Det(B_2)$, $|Det(B_1)| = |Det(B_2)|$.

3) Determinant of a lattice

$$L(B) = Det(L) = |Det(B)|.$$

4) Fundamental parallelepiped:
We define the fundamental parallelepiped by:

$$P(B) = \{\alpha_1 b_1 + \alpha_2 b_2 + \alpha_3 b_3 + \cdots .. + \alpha_n b_n, \alpha_i \epsilon [0,1)\} \tag{20}$$
$$Det(L) = Det(B) = vol(P(B)).$$

5) Bad base vs Good base:

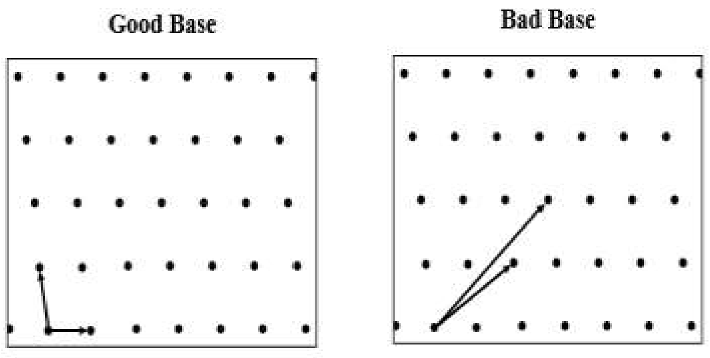


Fig. 3. Good base Vs Bad base

If we carefully compare the two bases given in Fig. 3, we can see that two bases generate the same lattice, but the first one is considered as good base because it is formed of small vectors while the other one is a bad basis because it is formed of large vectors.

6) Succesive Minima and Minskowski's Theorem
We define $\lambda_1(L)$ as the length (Euclidean Distance) of the shortest non zero vector in L.
More generally, $\lambda_k(L)$ denotes the smallest radius of a ball containing k linearly independent vectors $(1 \leqslant k \leqslant n)$.
An explanation of successive minima is given briefly in Fig. 4, where $\lambda_1(L)$ and $\lambda_2(L)$ are drawn.

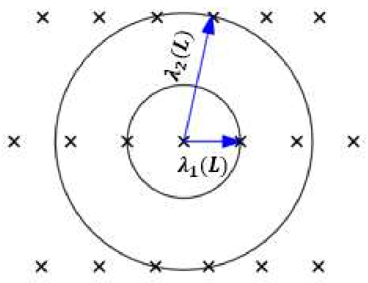


Fig. 4. Succesive Mininma

Minkowski's first theorem: for any full-rank lattice L of rank n,

$$\lambda_1(L) \leqslant \sqrt{n}(Det(L))^{1/n}. \tag{21}$$

Minkowski's second theorem: for any full-rank lattice L of rank n,

$$(\prod_{i=1}^{n} \lambda_i(L))^{1/n} \leqslant \sqrt{n}(Det(L))^{1/n}. \tag{22}$$

7) Some lattice problems

- SVP_h: Shortest Vector Problem: For $h \geq 1$, given a basis B of $L = L(B)$, find $z \in L$ such that $0 \neq ||z|| \leqslant h.\lambda_1(L)$.
- CVP_h: Closest Vector Problem: For $h \geq 1$, given a basis B of $L = L(B)$, find $z \in L$ such that $||t - z|| \leqslant h.dist(t, L)$.
- Using good basis, Babai's algorithm [20] resolves the CVP_h with $h = 2^{n/2}$.
  (i.e. finding $z \in L$ such that $||t - z|| \leqslant 2^{n/2}.dist(t, L)$).

8) Lattice based encryption

A simple asymmetric encryption scheme using Lattice is as follows:

- Secret Key: Secret Key is chosen as a good basis of the lattice $L(B)$.
- Public Key: Public Key is chosen as a bad basis of the lattice $L(B)$.
- It is easy to generate the bad basis from the good one, but the inverse is hard.
- Encryption consists of creating a problem (such as CVP_h) where it is hard to resolve it using the bad basis, but resolving it with good basis is simple (using Babai's algorithm with good basis).
- Example: to encrypt, we write the message as a vector e, we randomly choose $z \in L$, we need $x = z + e$, decryption is finding $z \in L$ close to x.

B. Learning With Error (LWE)

1) LWE is a machine learning problem[18] [19], where we need to solve a hard problem, we have to learn the values of s from a sample of the form $(a_i, \langle s, a_i\rangle + e_i)$, a_i are uniformly chosen from Z_q^n and e_i a noise distribution over Z_q.

Given the following security parameters,

n: Dimension, $q \geq 2$: modulus, $\alpha \ll 1$: error rate,

we define two different problems:

Search Problem: find $s \in Z_q^n$ given a noisy random inner products

$$\begin{aligned} a_1 &\leftarrow Z_q^n, & b_1 &= \langle s, a_1\rangle + e_1 \\ a_2 &\leftarrow Z_q^n, & b_2 &= \langle s, a_2\rangle + e_2 \\ & & &\dots \\ a_m &\leftarrow Z_q^n, & b_m &= \langle s, a_m\rangle + e_m \end{aligned} \tag{23}$$

Errors $e_i \leftarrow \chi$=Gaussian over Zq, param αq.

Decision problem: distinguish (a_i, b_i) calculated in equation (23) from a uniform (a_i, b_i).

2) LWE matrix form

Suppose that we have

$$A = [a_1\, a_2 \dots a_n] = \begin{bmatrix} a_{11} & a_{21} & \cdot & a_{m1} \\ a_{12} & a_{22} & \cdot & a_{m2} \\ \cdot & \cdot & \cdot & \cdot \\ a_{1n} & a_{2n} & \cdot & a_{mn} \end{bmatrix} \in Z_q^{(n,m)}$$

$s^t = [s_1\, s_2 \dots\dots s_n] \in Z_q^{(1,n)}$, $e^t = [e_1\, e_2 \dots\dots e_m] \in Z_q^{(1,m)}$

The *LWE* matrix form of the Equation (23) can be written as

$$b^t = s^t A + e^t \in Z_q^{(1,m)} \tag{24}$$

Equation (24) is lattice equation, where $s^t A$ forms the lattice points. In lattice base encryption scenario, e^t can be considered as the plaintext, and decryption is a CVP_h problem that consists of finding the closest lattice point to b^t.

C. BV-BGV schemes

1) Basic Scheme

The *BV* and *BGV* are two homomorphic encryption schemes that are similar, but differ a little during the modulus switching phase. The *BV* was published by the authors of [16] in 2011, while the *BGV* was published by the authors of [17] in 2012. The two schemes consist of applying the *DGHV* over Lattices. The hardness of these two schemes is based on the hardness of *LWE.*

We will suppose that the scheme for now is a symmetric scheme, that works over the bit level. The secret key is $s \in Z_q^{(n,1)}$ and the cipher-text is $c \in Z_q^{(n,1)}$.

Decryption is given by this equation

$$\begin{aligned} b &= \underbrace{\big((\langle s, c\rangle mod(q))mod\,(2)\big)}_{vector\ form} \\ &= \underbrace{\big(((s.c)mod(q))mod(2)\big)}_{matrix\ form} \end{aligned} \tag{25}$$

Decryption works only if the noise is small enough: $((s.c)mod\ q) << \alpha q$.

The cipher-text c is close to an orthogonal space to s, and the plain-text is encoded in the parity of distance.

2) Building the homomorphic scheme

The scheme is an instance of Error Correcting Code (ECC), addition is valid as long as the noise is small.
The question is, how we can build the multiplicative homomorphic given that cipher-texts are vectors?
To build the homomorphic multiplication, we use tensor product given in the next equation:

$$M = u \otimes v = \{M_{ij} = (u_i v_j)\} \quad (26)$$

It is easy to demonstrate that:

$$s(u \otimes v)s^t = \langle s, u\rangle\langle s, v\rangle = (s.u)(s.v) \quad (27)$$

If the noise level is low after the multiplication operation, decryption can be written as:

$$\begin{aligned}&(s(u \otimes v)s^t mod(q))mod(2)\\ &\qquad = ((\langle s,u\rangle\langle s,v\rangle)mod(q))mod(2)\end{aligned} \quad (28)$$

To clarify this scheme, we consider a tensor product with dimension 2, the secret key is $s = [s_1\ s_2]$ and two vectors of cipher-texts $u = [u_1\ u_2]$ and $v = [v_1\ v_2]$.

$\langle s, u\rangle = (s_1u_1 + s_2u_2)$ and $\langle s, v\rangle = (s_1v_1 + s_2v_2)$

$M = u \otimes v = \{M_{ij} = (u_i, v_j)\} = \begin{bmatrix} u_1v_1 & u_1v_2 \\ u_2v_1 & u_2v_2 \end{bmatrix}$

$sMs^t = [s_1\ s_2]\begin{bmatrix} u_1v_1 & u_1v_2 \\ u_2v_1 & u_2v_2 \end{bmatrix}\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} =$

$s_1u_1v_1s_1 + s_2u_2v_1s_1 + s_1u_1v_2s_2 + s_2u_2v_2s_2.$

We can simply verify that $sMs^t = \langle s, u\rangle\langle s, v\rangle$

The key point is that we prefer to deal with vectors rather than matrices.

$u \otimes v = \begin{bmatrix} u_1v_1 & u_1v_2 \\ u_2v_1 & u_2v_2 \end{bmatrix}$, we take $c^* = vect(u \otimes v) =$ $[u_1v_1\ \ u_1v_2\ \ u_2v_1\ \ u_2v_2]$.

$s \otimes s = \begin{bmatrix} s_1s_1 & s_1s_2 \\ s_2s_1 & s_2s_2 \end{bmatrix}$, we take $s^* = vect(s \otimes s) =$ $[s_1s_1\ \ s_1s_2\ \ s_2s_1\ \ s_2s_2]$.

We can simply verify that $\langle c^*, s^*\rangle = \langle s, u\rangle\langle s, v\rangle = sMs^t$.

$$Dec(c^*) = ((\langle c^*, s^*\rangle mod(q))mod(2) \quad (29)$$

Equation (29) holds as long as the noise $\langle c^*, s^*\rangle\ mod(q)$ is quite small.
The problem is that the dimension grows exponentially each time we multiply two cipher-texts homomorphically using tensor product, as shown in TABLE II.

TABLE II. HOMOMORPHIC MULTIPLICATION GROWTH

Nb of multiplications	Cipher dimension
0	n
1	$n^2 = n^{2^1}$
2	$n^4 = n^{2^2}$
…..	….
k	n^{2^k}

To resolve the problem listed above, the *key switching* technique is introduced [16][17], we have a cipher-text c^* with respect to an extended secret key $s^* = vect(s \otimes s)$, the main purpose of key switching is to build a low dimension cipher-text c' with respect to a low dimension key s' such that

$$Dec_{s'}(c') = Dec_{s^*}(c^*) \quad (30)$$

To achieve key switching, we have to publish an encryption matrix $M \epsilon Z_q^{[m,n]}$ defined by:

$$\begin{gathered}M(s^* \to s)\ ,\ c' = Mc^*\ ([m, 1] = [m, n][n, 1]).\\ m < n\end{gathered} \quad (31)$$

The different secret keys and cipher-texts are given by
$(s^* = [1|t]\ \epsilon Z_q^n,\ \ c^* \epsilon Z_q^n)$ and $(s' = [1|t'] \epsilon Z_q^m\ , c' \epsilon\ Z_q^m)$

The matrix M is published as shown in Fig. 5,

$$s' \epsilon\ Z_q^{[1,m]}, \qquad M\ \epsilon\ Z_q^{[m,n]}$$

$s'M\ = -t'A + 2e + s^* + t'A = (2e + s)\ \epsilon\ Z_q^{[1,n]}$

$Dec_{s'(c')} = ((s'c')mod(q))mod(2)$

$s'c' = s'Mc = (2e + s^*)c^* = 2(ec^*) + (s^*c^*)$

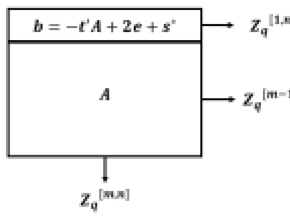


Fig. 5. Matrix M: key switching

The refresh mechanism would work if:

$$((s'c')mod(q))mod(2) = ((s^*c^*)mod(q))mod(2).$$

and (ec^*) should be small, but it is not small because $c^* \epsilon Z_q^n$.
To overcome this , we represent c^* with higher dimension but with lower values. Thus c^* is represented in its binary form

$$\begin{gathered}c^* = [c_i^*],\ c_i^* \text{ is the } i^{th} \text{ entry of } c^* \text{ and can be written}\\ \text{as } c_i^* = \textstyle\sum_{j=0}^{l-1} 2^j c_{ij}^*,\ c_{ij}^*\ \epsilon\ \{0,1\}\end{gathered} \quad (32)$$

Let $b_j = [c^*{}_{1j}\ c^*{}_{2j}\ \ c^*{}_{3j}\ \ \ldots\ldots.c^*{}_{nj}]\ \ 1 \leqslant j \leqslant l - 1$

$$c^* = \sum_{j=0}^{l-1} 2^j * b_j$$

$$\langle s^* c^* \rangle = \sum_{j=0}^{l-1} 2^j \langle b_j, s^* \rangle = 2^0 \langle b_0, s^* \rangle + 2^1 \langle b_1, s^* \rangle + \cdots 2^{l-1} \langle b_{l-1}, s^* \rangle. \tag{33}$$

If we suppose that $c^{**} = [b_0\ b_1 \ldots\ldots b_{l-1}]$ and $s^{**} = [2^0 s^*\ 2^1 s^* \ldots\ldots 2^{l-1} s^*]$, we can simply verify that $\langle s^{**}, c^{**} \rangle = \langle s^*, c^* \rangle$

We publish in this case the matrix $M: M(s^{**} \to s')$ such that

$$c' = Mc^{**}$$

$(the\ dimension\ is\ [m,1] = [m, n*(l-1)][n*(l-1),1])$

$$s^{'}M = -t^{'}A + 2e + s^{**} + t^{'}A = 2e + s^{**}$$
$$s'c' = s'Mc^{**} = (2e + s^{**})c^{**} = 2(ec^{**}) + (s^{**}c^{**}) \tag{34}$$

In this case we can say that:

$$\big((s^{**}c^{**})mod(q)\big)mod(2) = ((s'c')mod(q))mod(2) \tag{35}$$

Because ec^{**} is small enough.

3) Leveled Somewhat Homomorphic Scheme

Level Somewhat Homomorphic Encryption Scheme *(SWHE)* means that we are working level by level. For a circuit of depth d, we generate d random secret keys $s_i = [1|t_i]$ in Z_q^m, $0 \leqslant i \leqslant d-1$.

For each level i of the circuit, we have to publish a public key M_i in $Z_q^{[m,n]}$, for level 0 we publish $M_0 = M(0 \to s_0)$ and for any level i we publish $M_i = M(s_{i-1}^{**} \to s_i)$.

$$s_0 M_0 = 2e_0$$
$$s_i M_i = 2e_i + s_{i-1}^{**} \tag{36}$$

Encryption scheme is given by the following equation:

$$Enc(b) = M_0 r + \begin{bmatrix} b \\ 0 \\ \cdot \\ 0 \end{bmatrix}, \quad r \text{ is a random vector in} \{0,1\}^n \tag{37}$$

Decryption at level i is given by the following equation:

$$Dec(c,i) = \big(\langle s_i c_i \rangle mod(q)\big)mod(2) \tag{38}$$

By applying this *SWHE*, we can directly add cipher-texts at the same level, while during multiplication, we multiply (c_1, i) and (c_2, i), we compute the extended cipher-texts $c^* = vect(c_1, c_2)$, then we calculate the doubly extended c^{**}, and we calculate the fresh cipher-text $c' = Mc^{**}$.

4) Making the scheme fully homomorphic:

The noise at level i is given by $(sc_i)mod(q)$. Noise level gets doubled and squared by multiplication. But in this way we can homomorphically evaluate a limited depth circuit as long as the noise is lower than q. To evaluate deeper circuit, we introduce the Modulus switching technique, that is based on switching into another modulus (different than q) but we are still able to decrypt. The key challenge is that we have a cipher-text c with respect to a secret key s, we will build a new cipher-text c' for some $p < q$ such that

$$\big(\langle c,s \rangle mod(p)\big)mod(2) = (\langle c',s \rangle mod(q))mod(2) \tag{39}$$

Let $p < q$ be odd integers, $c, s \in Z_n^q$, such that:

$$|\langle c,s \rangle mod(q)| < {q}/{2} - {q}/{p}\, ||s||_l \tag{40}$$

Let $c' = rnd_c({p}/{q}c)$, where $rnd_c(.)$ rounds each entry up or down such that $c' = c(mod2)$, then we have

$$|\langle c',s \rangle mod(p)| \leqslant \tfrac{p}{q} |\langle c,s \rangle mod(q)| + ||s||_l \tag{41}$$

$$(\langle c',s \rangle mod(p))mod(2) = \big(\langle c,s \rangle mod(q)\big)mod(2) \tag{42}$$

For $\langle c,s \rangle mod(q)$, we can always find a k such that

$$\langle c,s \rangle - kq \epsilon \left[\frac{-q}{2}, \frac{q}{2}\right]$$

$$|\langle c,s \rangle modq| < \tfrac{q}{2} - \tfrac{q}{p} ||s||_l \Longrightarrow$$

$$\frac{-q}{2} + \frac{q}{p} ||s||_l < \langle c,s \rangle - kq$$

If we multiply the inequality by $\frac{p}{q}$ we obtain the following:

$$\tfrac{-p}{2} + ||s||_l < \ \tfrac{p}{q}\ \langle c,s \rangle - kp < \tfrac{p}{2} - ||s||_l$$

$$\langle c',s \rangle - kp \in [\tfrac{-p}{2} + ||s||_l, \tfrac{p}{2} - ||s||_l] \Longrightarrow \langle c',s \rangle - kp \in [\tfrac{-p}{2}, \tfrac{p}{2}]$$

This proves equation (41).

Given that $\langle c,s \rangle mod\ q = \langle c,s \rangle - kq$ and

$\langle c',s \rangle mod\ p = \langle c',s \rangle - kp$ for the same k.

$(kq)mod\ 2 = (kp)mod\ 2, \qquad p,\ q$ are both odd.

$c' = c(mod2)$.

$$\langle c',s \rangle mod(2) = \langle c,s \rangle mod(2) \Longrightarrow (\langle c',s \rangle - kp)mod(2) = (\langle c,s \rangle - kq)mod(2)$$

$$\big(\langle c,s \rangle mod(q)\big)mod(2) = \big(\langle c',s \rangle mod(p)\big)mod(2).$$

And this proves equation (42).

As we said before BV and BGV are similar, but they differ in the modulus switching phase, the difference is that with BV $p \ll q$, while in $BGV\ p < q$.

During the implementation we start with a large modulus q_0, small noise $\mu << q_0$. After 1st multiplication, noise grows to μ^2, we switch to $q_1 \cong {q_0}/{\mu}$, noise is reduced to ${\mu^2}/{\mu} \cong \mu$. After next multiplication layer, noise again grows to μ^2, we switch to $q_2 \cong {q_1}/{\mu}$ to reduce it back to μ.

A practical example is given in TABLE III. where we started with initial modulus $q_0 = \mu^5$, we can deduce how the modulus switching allowed us to evaluate deeper circuit depth. Without modulus switching we stopped at level 3, while with modulus switching we stopped at level 4.

TABLE III. MODULUS SWITCHING VS NO MODULUD SWITCHING

	With modulus switching		Without modulus switching	
	Noise	Modulus	Noise	Modulus
Fresh Cipher-texts	μ	μ^5	μ	μ^5
Level 1	μ	μ^4	μ^2	μ^5
Level 2	μ	μ^3	μ^4	μ^5
Level 3	μ	μ^2	$\underline{\boldsymbol{\mu^8}}$	$\underline{\boldsymbol{\mu^5}}$
Level 4	$\underline{\boldsymbol{\mu}}$	$\underline{\boldsymbol{\mu}}$		

V. CONCLUSION

In this paper, we investigated two important FHE asymmetric schemes: *DGHV* and *BV-BGV*. The first one is based on computing over real integers, while the other one is based on *LWE* (Lattice based Encryption). The two encryption schemes are not practical for real world applications due to storage overhead because for example as listed in [14], the public key size of *DGHV* is 802 MB even after applying public key compression. One main advantage of the *BV-BGV* over the *DGHV* is the modulus switching technique: a cipher-text refresh mechanism in *DGHV* (squashing and bootstrapping) takes 14 minutes, while modulus switching is very simple to be achieved [14]. On the other hand bootstrapping can extend the circuit evaluation to unlimited circuit depth, while the modulus switching extends it to a limited number but larger than the original scheme. Future work will focus on the designing and implementing of a secure FHE scheme practical for real world applications and especially for Cloud applications.

REFERENCES

[1] Dan Boneh and Craig Gentry and Sergey Gorbunov and Shai Halevi and Valeria Nikolaenko and Gil Segev and Vinod Vaikuntanathan and Dhinakaran Vinayagamurthy, Fully Key-Homomorphic Encryption, Arithmetics Circuiy ABE, and Compact Garbled Circuits, Cryptology ePrint Archive, Report 2014/356, 2014,http://eprint.iacr.org/2014/356.

[2] T. D. Ramotsoela and G. P. Hancke, "Data aggregation using homomorphic encryption in wireless sensor networks," *2015 Information Security for South Africa (ISSA)*, Johannesburg, 2015, pp. 1-8.doi: 10.1109/ISSA.2015.7335058

[3] R. Rivest, A. Shamir, and L. Adelman, A method for obtaining digital signatures and public-key cryptosystems, Communication of the ACM 21 (2): 120126, 1978.

[4] M. Nassar, A. Erradi, and Q. Malluhi, Pallier's Encryotion:Implementation and clouds applications Applied Research in Compuer Science and Engineering (ICAR), 2015 International Conference on, Beirut, 2015, pp. 1-5 .doi:10.1109/ARCSE.2015.7338149.

[5] Joseph Domingo Ferrer, A new privacy homomorphism and applications Information Processing Letters, Volume 60, Isssue 5, 9 December 1996, pages 277-282.

[6] Joseph Domingo Ferrer, A Provably Secure Additive and Multiplicative Privacy Homomorphism, Universitat Rovira I Virgli, Dept. of Computer Engineering and Maths, ISC '02 Proceedings of the 5th International Conference on information Security, Pages 471-483, Springer-Verlag London UK, 2002.

[7] Xiao, Liangliang and Bastani, Osbert and Yen, ILing An Efficient Homomorphic Encryption Protocol for Multi-User Systems Citeseer, IACR Cryptology ePrint Archive, volume 2012, year 2012, pp. 193, year 2012.

[8] Khalil Hariss, Hassan Noura, Abed Ellatif Samhat, Fully Enhanced Homomorphic Encryption algorithm of MORE approach for real world applications, Journal of Information Security and Applications, Volume 34, 2017, Pages 233-242, ISSN 2214-2126, http://dx.doi.org/10.1016/j.jisa.2017.02.001.

[9] Kipnis, Aviad and Hibshoosh, Eliphaz, Efficient Methods for Practical Fully Homomorphic Symmetric-key Encrypton, Randomization and Verification IACR Cryptology ePrint Archive, volume 2012, pp. 637, year 2012.

[10] Craig Gentry PhD Thesis "A FULLY HOMOMORPHIC ENCRYPTION SCHEME" Stanford University, Department of Computer Science, September 2009.

[11] Craig Gentry.2009. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of Computing (STOC'09). ACM, New York, NY, USA, 169178:https://doi.org/10.1145/1536414.1536440

[12] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully Homomorphic Encryption over the integers, EURO-CRYPT2010 (LNCS) vol.6110, pp. 24-43.

[13] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. 2011. Fully homomorphic encryption over the integers with shorter public keys. In *Proceedings of the 31st annual conference on Advances in cryptology* (CRYPTO'11), Phillip Rogaway (Ed.). Springer-Verlag, Berlin, Heidelberg, 487-504.

[14] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. 2012. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques* (EUROCRYPT'12), David Pointcheval and Thomas Johansson (Eds.). Springer-Verlag, Berlin, Heidelberg, 446-464. DOI=http://dx.doi.org/10.1007/978-3-642-29011-4_27.

[15] Craig Gentry, Shai Halevi, Implementing Gentry's Fully-Homomorphic Encryption scheme, Cryptology ePrint Archive, Report 2010/520, 2010, http://eprint.iacr.org/2010/520.

[16] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of the 31st annual conference on Advances in cryptology* (CRYPTO'11), Phillip Rogaway (Ed.). Springer-Verlag, Berlin, Heidelberg, 505-524.

[17] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (ITCS '12). ACM, New York, NY, USA, 309-325. DOI=http://dx.doi.org/10.1145/2090236.2090262.

[18] oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing* (STOC '05). ACM, New York, NY, USA, 84-93. DOI: http://dx.doi.org/10.1145/1060590.1060603

[19] Chris Peikert, Lattice-Based Cryptography: Short Integer Solution (SIS) and Learning With Errors (LWE), Georgia Institute of Technology

[20] David A. Madore, Réseaux euclidiens et cryptographie, Journées Télécom-UPS, Le numérique pour tous, Télécom ParisTech, 29 mai 2015.