A Comprehensive Study of Fully Homomorphic Encryption Schemes

Majedah Alkharji, Hang Liu, Mayyada Al Hammoshi

1Electrical Engineering and Computer Science, CUA, Washington, DC, {32alkharji, liuh} @cua.edu
2School of Information Computer System VIU, Fairfax, VA, mhammoshi@viu.edu

Abstract

Organizations and individuals have been shifting to the Internet-based cloud computing technology to obtain more efficient and faster computing services in recent years. However, moving to the cloud services has created new security challenges. Confidential information becomes more vulnerable to be leaked due to computation outsourcing to the third-parties. The issue of data breaches could eliminate all the benefits that an organization may get by utilizing the cloud-based computing services. The primary goal of securing information is to provide confidentiality, authenticity, integrity and data privacy. Data encryption has been widely used to protect data during the communication and storage process. However, as users need to process data in the cloud, traditional cryptographic schemes cannot work well because it requires passing the secret keys to the cloud servers so that the servers can obtain the original data and perform the needed computations on the plaintext. Once encrypted data are disclosed for calculations, it can't guarantee the data confidentiality in the cloud; which poses a significant challenge for applications of cloud computing. Homomorphic Encryption (HE) has been proposed to overcome this security issue, which enables mathematical computations directly on encrypted data without the requirement of decryption in the cloud and can provide the same result as obtained by using calculations on the plaintext. Many Fully Homomorphic Encryption (FHE) schemes are designed, which can perform an arbitrary number of additions and multiplications on the ciphertext. In this paper, a comprehensive survey of homomorphic encryption technology using public key algorithms is introduced, then, FHE schemes are reviewed and analyzed. As researchers believe the advancement in the FHE technology will continue, this paper guides the state-of-the-art definitions, properties, algorithms, and schemes of FHE.

Keywords: Homomorphic Encryption (HE), Fully Homomorphic Encryption (FHE), FHE from Learning with Error (LWE), FHE over the Integer (DGHV), Cloud Security, Cryptography.

1. Introduction

Cloud computing means providing on-demand network access to IT sharing computing resources (e.g., servers, storage applications, and networks) using IT components (e.g., hardware, and software) via the internet or private channels [13][47]. The third-party Cloud Service Providers (CSP), such as Microsoft, provide applications that can be used by IT professionals as a platform, on which they can offer services to users flexibly and conveniently. By transmission into the data-centric cloud environment, data will be more readily accessible than before. Moreover, through CSP, a user can warehouse data within a package of cloud remote servers rather than in-house which in turn enhances interaction [1][34].

Given the heightened use of the internet, computer, and cloud computing technology, security considers as a prime requirement to ensure confidentiality, integrity, and accessibility of the information systems resources. The improvement in computational techniques for machine learning has linked with the emergence of the cloud-based computing platforms. Despite the efficient computing solution and economic advantages associated with cloud computing, users are very concerned about security and confidentiality of data stored and processed in the cloud. These security concerns are caused by some security risks such as insider threats, security breach, and potential hackers [13][19][47].

Among the solutions provided for safeguarding the data stored in the cloud is the encryption to prevent unauthorized people from accessing the information. Hence, encryption algorithms must be applied
within the scope of “big data” and “cloud computing” to attain the goal of data protection including confidentiality and integrity. The usage of either symmetric or asymmetric (public key) encryption algorithms “see Figure 1.” are not entirely sufficient with cloud-based scenario [31].

This drawback can be overcome by using Homomorphic Encryption (HE). It is an efficient algorithm that protects information stored in the cloud while guaranteeing its accessibility. For secure data processing in the cloud platform, Fully Homomorphic Encryption (FHE) is a crucial step in enhancing cloud-computing security and ensuring data privacy. It supports evaluating arbitrary operations on ciphertext without exposing the original data. The evaluation process outcomes match the result of calculations when performed on the plaintext [1][13].

The goal of this paper is to provide researchers with detailed documentation of HE and FHE including algorithms, performance, and security assumptions. These concepts should be enough to realize how they work. The following guideline provides a firm basis for the researchers who would like to intensify their knowledge on these subjects.

Figure 1. Asymmetric Encryption Functions Applied to the Cloud

The remaining of the article is organized as follows: the next section introduces basic definitions, functions, and properties of Homomorphic Encryption (HE). Part 3 classifies the classes of HE (partially HE, somewhat HE, or fully HE), and presents the related HE schemes for each type along with their fundamental definitions, algorithms, semantic security, and possible applications. Part 4 in this guideline gives information about the definitions and principles of FHE related assumptions. The following one (part 5) features a comprehensive survey about the improvement in the field of FHE. Section 6 discusses the semantic security of FHE, and the last part presents future work and conclusion.

2. Homomorphic Encryption (HE)

2.1. Definition of homomorphic encryption

The primary objective of encryption is to assure data privacy and confidentiality in both storage and treatment processes. Many conventional cryptographic algorithms have been proposed and implemented to ensure security. When all warehoused information is encrypted, that will solve all the challenges identified with information security such as data security, third-party control, and availability. Accordingly, an encoded version of the data will be given to CSP to perform operations as a means of protecting data in the cloud from loss or breach. However, data cannot be processed in the encrypted format due to inaccessibility. Therefore, the CSP must decrypt the data, then perform the calculations on the data before sending the outcome to the user. Hence, both users and companies should trust the CSP to carry out operations [15][19][31].

The practice shows security weaknesses in the encryption methods since it allows for loss of privacy and confidentiality. Using encryption means that the user will have to provide the CSP with the private key to perform operations. The practice will then lead to the users giving up their confidential information,
which is not the aim of the cloud data storage technology. Consider the possibility of having a technique that allows completing any calculation without getting the original data [31]. Homomorphic encryption refers to “the encryption technology that implies that the procedures can be done on the encrypted data and the matching outcome can be achieved as on original data.” With HE cryptosystems, a firm can encrypt its database and submit it to the cloud, however, the encrypted data can be processed without the private key held by the client [12][28][34]. However, HE has its limitations which include its inability to deal with specific threats such as attacks with selected ciphertext (IND-CCA) and attacks with selected plaintext (IND-CPA). These setbacks emphasize demand for a capability to carry out computations on encrypted data, such ability that offers several critical applications including the capacity to privately outsource calculations [35].

2.2. Functions of homomorphic encryption

An encryption scheme is considered homomorphic if: given a plaintext \((m) = (m_1, m_2)\), one can compute \(E[f(m_1, m_2)]\) from \(E(m_1)\) and \(E(m_2)\), without using \(pk\), where \(f\) might be +, \(\times\), \(\oplus\). Homomorphic encryption permits the conversion of ciphertext \(c(m)\) of text \(m\) to ciphertext \(c(f(m))\) of a function of text \(m\) without revealing the message [12][19][28][34].

**2.2.1. Key generation (G).** The client generates two pairs of keys, public key \((pk)\) and secret/private key \((sk)\), to perform the encryption of plaintext \((m)\).

**2.2.2. Encryption (E).** The client encrypts the plaintext \((m)\) using \(pk\) and produces \(E_{pk}(m)\).

**2.2.3. Sending.** The ciphertext \((c)\) and \(pk\) are delivered to the server.

**2.2.4. Storage.** \(pk\) and \(c\) are stored in the cloud database.

**2.2.5. Request.** The client must send a request to the server to analyze the encrypted information.

**2.2.6. Evaluation (EV).** Server processes the request and evaluates function \(f\) on ciphertext \((c)\) using \(pk\).

**2.2.7. Response.** The cloud provider responds by returning the result to the client.

**2.2.8. Decryption (D).** \(EV(f(c))\) is deciphered by the client applying its secret key to obtain the original data \((m)\).
2.3. Properties of homomorphic encryption

HE enables servers to carry out sophisticated mathematical computations on encrypted records without acknowledging the original message. In more details, given plaintexts, \( m_1 \) & \( m_2 \), and the corresponding ciphertexts \( c_1 \) & \( c_2 \), HE schemes allow the processing of \( c_1 \Theta c_2 \) without applying \( pk_1 \Theta pk_2 \). In that connection, the cryptosystem is additive or multiplicative homomorphic depending on the \( \Theta \) operation, which can be addition or multiplication.

The HE systems can be classified in line with the operation that allows performing on the original data as following [1][19][34][28][12]:

2.3.1. Additive Homomorphic Encryption (AHE). It allows the HE schemes to evaluate raw data. Examples of this scheme are Pailler, GM, Benaloh, and Okamoto-Uchiyama cryptosystems. Scholars emphasize that HE is additive if: \( E(m_1 \oplus m_2) = E(m_1) \oplus E(m_2) \), without knowing \( (m_1) \), and \( (m_2) \).

2.3.2. Multiplicative Homomorphic Encryption (MHE). It refers to systems in which ciphertexts from the ultimate product of plaintexts. RSA and El-Gamal cryptosystems constitute multiplicative homomorphic schemes. Homomorphic encryption is multiplicative if: \( E(m_1 \otimes m_2) = E(m_1) \otimes E(m_2) \), without knowing \( (m_1) \), and \( (m_2) \).

3. History of homomorphic encryption.

The concept of “privacy Homomorphism” introduced by Rivest, Adlema, and Dertouzos in 1978. Although the idea has proposed, the progress made was little within 30 years. In 1982, Goldwasser and Micali suggested a verifiable encryption system known as Goldwasser-Micali (GM), which developed to an outstanding level of safety. This system is additive Homomorphic encryption, but it could only perform just one operation and encrypt a single bit. The GM encryption scheme is the addition of encrypted bits mod 2 (which is, the exclusive- OR function). The Benaloh Cryptosystem is an extension of the Goldwasser-Micali (GM). It was developed in 1994 by Benaloh. Four years later, The Naccache–Stern cryptosystem (NS) was proposed by Naccache and Stern in 1998. The Okamoto–Uchiyama (OU) cryptosystem,“ in the same year by Okamoto and Uchiyama. On the same note, Pascal Paillier was declared another secure provable additive homomorphic encryption scheme in 1999. In late 2000, The Damgård–Jurik cryptosystem (DJ) proposed by Damgård and Jurik. It was a generalization of the Paillier cryptosystem. All these schemes intensively studied and supported either homomorphic addition or multiplication of plaintexts, but not both!

Boneh, Goh, and Nissim developed in 2005 a better semantically secure technology which known as Boneh-Goh-Nissim (BGN) cryptosystem. It allows to execute an arbitrary number of additions but only allowed a single multiplication.

In 2009, Craig Gentry invented the groundbreaking work of fully homomorphic encryption; since then, the first blueprint has interested many researchers. In the next section, details of these cryptosystems will be under each HE categories they are most related to [28][34][35][3][18][36][19].

3.1. Partially Homomorphic Encryption schemes (PHE)

In partially homomorphic encryption, one operation either addition (ex: Paillier, GM cryptosystem), or multiplication (ex: RSA, El-Gamal cryptosystem) can be performed on the ciphertext, but both operations cannot be handled [12]. The following algorithms are different examples of PHE cryptosystems. For more details, Kukucka in his thesis [20] investigated these algorithms theoretically.

3.1.1. Goldwasser-Micali cryptosystem (GM). It is additive HE cryptosystem proposed by Goldwasser and Micali in 1982. It is considered as a probabilistic public key algorithm, but it can encrypt ciphertext bit-by-bit [12]. This scheme is recognized as an important stone for the later studies. Some systems proposed after were treated as generalizations of this one [15]. GM has the XOR homomorphic characteristic, or we can call it addition modulo 2. The security of GM cryptosystem relies on the quadratic residuosity problem [20].
3.1.2. Benaloh cryptosystem. Benaloh Cryptosystem was proposed to improve the poor expansion factor provided by GM Cryptosystem. Instead of bit-by-bit encryption, the Benaloh scheme encrypts the ciphertext block-by-block at once with \( r \) bits length using a technique called “dense probabilistic encryption.” Assume we have \( k \)-bit plaintext, \( n \) is security parameter; this method computes the encryption of \( k \)-bit plaintext to get ciphertext of \( n + k \) bit. A small prime restricts the Benaloh cryptosystem messages. This scheme rests on the difficulty of the higher residuosity problem [20].

3.1.3. Naccache-Stern cryptosystem (NS). Naccache–Stern cryptosystem was classified first as a deterministic public key homomorphic scheme, but it has been demonstrated that after revision, it can be made probabilistic. NS has been counted as a generalization of the Benaloh cryptosystem by reducing the expansion factor of the ciphertext since the messages are limited by the multiplication of many small primes. Regarding time complexity, recovering a plaintext from its matching ciphertext is a little less efficient because the procedure includes decoding the ciphertext modulo each of the small prime factors and then resetting the ciphertext using Chinese remaindering. The security of NS cryptosystem relies on the higher residuosity problem which considered to be intractable more than integer factorization [20][25].

3.1.4. Okamoto-Uchiyama cryptosystem (OU). Like RSA public key cryptography scheme, Okamoto-Uchiyama homomorphic (OU) cryptosystem relies on the challenge of factoring large integer. The primary difference of this system is that it works in the multiplicative group of integers modulo \( n \), where \( n \) in the form \( N = p^r q \) instead of \( N = p q \), where \( p \) and \( q \) are large primes. This cryptosystem is considered homomorphic under addition, subtraction, and multiplication of ciphertext. The semantic security of this probabilistic scheme derives from the p-subgroup assumption, which is very identical to the quadratic residuosity problem and higher residuosity problem [20].

3.1.5. Paillier cryptosystem. Pascal Paillier proposed the new probabilistic asymmetric cryptographic algorithm, which contains an additive homomorphic characteristic. Paillier cryptosystem is considered as an expansion of Okamoto-Uchiyama. The innovation has proved under Decisional Composite Residuosity Assumption (DCRA) [31]. As such, it has numerous applications such as threshold schemes and e-voting systems. “Algorithm 1.” demonstrates the additive property of paillier cryptosystem [15][28][34][1][19].

Assume there are two ciphertexts \( c_1 \) & \( c_2 \) the following illustration demonstrates the additive homomorphic characteristic of the Paillier cryptosystem:

\[
\begin{align*}
  c_1 &= g^{m_1} r_1^e \mod n^2 \\
  c_2 &= g^{m_2} r_2^e \mod n^2 \\
  c_1 \cdot c_2 &= g^{m_1 m_2} r_1^e r_2^e \mod n^2, g^{m_2} r_2^e \mod n^2 \\
  \text{Additive property is: } g^{m_1 m_2 (r_1 r_2)^e} \mod n^2
\end{align*}
\]

3.1.6. Damgard-Jurik cryptosystem (DJ). Damgard-Jurik is a probabilistic asymmetric homomorphic cryptosystem serving addition and subtraction. Similar to Paillier, Damgard-Jurik also based on (DCRA), but the only variation here, is that DJ computes modulo \( n^{s+1} \) instead of \( n^2 \) in Paillier. DJ is a generalization of Pailler’s scheme to groups of \( Z_{n^s+1}^* \), where \( s > 0 \). when \( s \) increases, we will get a decreased expansion. DJ semantic security relies on the assumption of the Decisional Composite Residuosity Problem [15][20].

3.1.7. RSA Algorithm. In 1978, Rivest, Shamir, and Adleman suggested their most widely used public-key cryptosystem. The RSA scheme has a multiplicative homomorphic property, which means, the homomorphic encryption scheme given by RSA is the product of two messages modulo \( n \). RSA semantic security relies on the hardness of the integer factorization problem. “Algorithm 2.” demonstrates the multiplicative property of RSA cryptosystem [34][28][19][26][1][15].

Assume there are two ciphertexts, \( c_1 \) & \( c_2 \), the following illustration demonstrates the multiplicative homomorphic characteristic of the RSA cryptosystem:

\[
\begin{align*}
  c_1 &= m_1^e \mod n \\
  c_2 &= m_2^e \mod n \\
  c_1 \cdot c_2 &= m_1^e m_2^e \mod n
\end{align*}
\]
Multiplicative property is: \((m_1 \cdot m_2)^e \mod n\)

**Algorithm 1. Paillier Algorithm**

<table>
<thead>
<tr>
<th>Key Generation: (G(p, q): pk, sk)</th>
<th>Decryption: (D(c, sk):m)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> ((p, q))</td>
<td><strong>Input:</strong> ((c), \text{and } sk) where (c &lt; n^2), ciphertext ((c) \in Z_{n^2})</td>
</tr>
<tr>
<td>Choose (p, q \in P), where (p, q) are two large prime numbers</td>
<td><strong>Computation:</strong> Compute (m = L(c^e \mod n^2) \cdot L(g^\lambda \mod n^2)^{-1} \mod n)</td>
</tr>
<tr>
<td><strong>Computation:</strong> Compute (\lambda = \text{lcm}(p-1, q-1)) ((\text{Carmichael's function})) Compute (\mu = \text{lcm}(n \cdot (p-1), n \cdot (q-1)))</td>
<td></td>
</tr>
<tr>
<td>Secret key: (sk = (p, q)) or ((\text{equivalently } \lambda))</td>
<td><strong>Output:</strong> ((m))</td>
</tr>
<tr>
<td><strong>Output:</strong> ((c))</td>
<td>Plaintext ((m) \in Z_n)</td>
</tr>
<tr>
<td>Ciphertext ((c) \in Z_{n^2})</td>
<td></td>
</tr>
</tbody>
</table>

**Algorithm 2. RSA Algorithm**

<table>
<thead>
<tr>
<th>Key Generation: (G(p, q): pk, sk)</th>
<th>Decryption: (D(c, sk):m)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> ((p, q))</td>
<td><strong>Input:</strong> ((c), \text{and } sk) where (c &lt; n^2), ciphertext ((c) \in Z_n)</td>
</tr>
<tr>
<td>Choose (p, q \in P), where (p, q) are two large prime numbers</td>
<td><strong>Computation:</strong> Compute (m = L(c^e \mod n^2) \cdot L(g^\lambda \mod n^2)^{-1} \mod n)</td>
</tr>
<tr>
<td><strong>Computation:</strong> Compute (\phi(n) = (p-1) \cdot (q-1)), where (\gcd(n, \phi(n)) = 1)</td>
<td></td>
</tr>
<tr>
<td>Choose (e \in {2, \ldots, \phi(n) - 1}), where (e) is a random integer, such that (\gcd(e, \phi(n)) = 1)</td>
<td></td>
</tr>
<tr>
<td>(\text{public key: } pk = (n, e))</td>
<td><strong>Output:</strong> ((m))</td>
</tr>
<tr>
<td><strong>Output:</strong> ((c))</td>
<td>Plaintext ((m) \in Z_n)</td>
</tr>
<tr>
<td>Ciphertext ((c) \in Z_n)</td>
<td></td>
</tr>
</tbody>
</table>

3.1.8. El-Gamal encryption algorithm. Like RSA, the public key encryption scheme given by El-Gamal is a multiplicative homomorphic encryption cryptosystem. It proposed by Taher El-Gamal in 1984, and its security relied on the hardness of the Diffie-Hellman problem. The next algorithm “Algorithm 3.” demonstrates the multiplicative property of El-Gamal cryptosystem [12][26][28][15].

Assume there are two ciphertexts, \(c_1 = (x_1, y_1)\) \& \(c_2 = (x_2, y_2)\)

The following illustration demonstrates the multiplicative homomorphic characteristic of the El-Gamal cryptosystem:

\[c_1 \cdot c_2 = (x_1, y_1) \cdot (x_2, y_2) = (x_1 \cdot x_2, y_1 \cdot y_2) = g^{k_1} \cdot g^{k_2} \cdot (m_1, \beta^{k_1}) \cdot (m_2, \beta^{k_2}) \mod p\]
Multiplicative property is: \( g^{k_1 \cdot k_2} \pmod{p} \)

### Algorithm 3. El-Gamal Algorithm

**Key Generation:** \( G(p, q): pk, sk \)

**Input:** \( (p, g) \)
- Choose \( p \in P \), where \( p \) is a large prime number
- Choose \( g \in Z_p^* \), where \( g \) is a generator of the cyclic group \( Z_p^* \)
- Choose \( a \in \{2, \ldots, p-2\} \), where \( a \) is a random integer

**Computation:**
- Compute \( \beta = g^a \pmod{p} \)

**Output:** \( (pk, sk) \)
- Public key: \( pk = (p, g, \beta) \)
- Secret key: \( sk = (a) \)

**Encryption:** \( E(m, pk): c \)

- **Input:** \( (m), and pk = (p, g, \beta) \)
- Plaintext \( m \in Z_p \), where \( Z_p = \{0, 1, \ldots, p-1\} \)
- Choose \( k \in \{2, \ldots, p-2\} \), where \( k \) is a random integer

**Computation:**
- Compute \( x = g^k \pmod{p} \)
- Compute \( y = m \cdot \beta^k \pmod{p} \)

**Output:** \( c = (x, y) \)

**Decryption:** \( D(c, sk): m \)

- **Input:** \( c = (x, y), and sk = (a) \)
- Ciphertext \( c \in Z_p \)

**Computation:**
- Compute \( m = x^a \cdot y \pmod{p} \)

**Output:** \( m \in Z_p \)

Regarding PHE schemes’ efficiency, NS permits the least message expansion \( (N/Q) \) as compared to the Benaloh cryptosystem. To ensure that the system remains protected and secure, the lower bound of this expansion rate should be four. Improved schemes have developed with the expansion factor being lowered to increase efficiency. Nonetheless, NS has not considered as suitable as Okamoto-Uchiyama cryptosystem, which is easier to apply and has a constant expansion rate of three. Scholars aimed at reducing the rate but without decreasing the level of security. For instance, Paillier cryptosystem allowed efficient decryption by enabling encryption of many bits during single calculation with a better expansion rate of two. The safety of DJ cryptosystem compares to the Paillier’s first innovation, but this generalization of Paillier permits reduction of the expansion rate to about one. A comparison of Paillier, RSA, DJ, and El-Gamal can be attained assuming the same security factor \( k \) [25].

### 3.2. Somewhat Homomorphic Encryption schemes (SWHE)

Somewhat homomorphic encryption approaches can only evaluate a multiple but a limited number of addition and multiplication activities [12]. SWHE schemes refer to encryption systems that present unique homomorphic characteristics but lacks full homomorphic capacity. Schemes support many addition operations and only single multiplication, but every time the computations are done, they result in “noise” in the ciphertexts that eventually make the decryption impossible [32][31]. Furthermore, in SWHE systems, the ciphertexts could expand in size, hence violating the message requirement. Boneh-Goh-Nissim (BGN) described below is considered as most famous SWHS. For more information about the algorithm and its security, see Kukucka thesis [20].

#### 3.2.1 Boneh-Goh-Nissim (BGN).

In 2005, Boneh, Goh, and Nissim developed a better semantically secure technology which known as Boneh-Goh-Nissim (BGN) cryptosystem. It allows the merging of addition and multiplication with a fixed-size of ciphertexts. With the BGN public key cryptosystem, it became possible to handle an arbitrary number of additions but only allowed a single multiplication. BGN cryptosystem uses bilinear pairings-based to allow the computation of a single homomorphic multiplication of two ciphertexts. Also, it evaluates quadratic formulas on encrypted data (e.g., 2-DNFs) [36][3][18]. BGN is secure under the assumption of the subgroup decision problem. The message expansion degree of BGN cryptosystem represented by \( N/R \), where \( N \) refers to the bit-length of \( n \) while \( R \) denotes the bit-length of \( r \).
3.3 Fully Homomorphic Encryption (FHE)

Fully homomorphic encryption supports an arbitrary number of multiplications and additions, hence, computes any form of function on encrypted information. For all types of computations on the data warehoused in the cloud, FHE must be adopted because it allows execution of operations on encrypted records without decryption. As such, the usage of FHE is a crucial step in enhancing cloud-computing security. The concept of FHE is about as old as the idea of public key encryption. Gentry’s innovative blueprint in 2009 indicated interestingly a reasonable construction of fully homomorphic encryption. The fundamental building stone in Gentry’s project, what’s called “Somewhat” Homomorphic Encryption (SWHE), which depended on the hardness of lattices. FHE is considered as the “holy grail” of cryptography and a primitive for building other cryptographic schemes; However, Gentry’s work does not represent a conclusion to the mission for the Holy Grail [4][19][36]. “Table 1.” presents a comparing between all different HE schemes according to properties, categories, & security assumption.

Table 1. Properties, Categories, and Security Assumption of HE Schemes [1][12][20]

<table>
<thead>
<tr>
<th>HE Scheme</th>
<th>Year</th>
<th>HE Categories</th>
<th>Homomorphic Features</th>
<th>Security Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy Homomorphism</td>
<td>1978</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Goldwasser-Micali (GM)</td>
<td>1982</td>
<td>PHE</td>
<td>XOR</td>
<td>Quadratic residuosity problem</td>
</tr>
<tr>
<td>The Benaloh</td>
<td>1994</td>
<td>PHE</td>
<td>Additive</td>
<td>Higher residuosity problem</td>
</tr>
<tr>
<td>Naccache–Stern (NS)</td>
<td>1998</td>
<td>PHE</td>
<td>Additive</td>
<td>Higher residuosity problem</td>
</tr>
<tr>
<td>Okamoto-Uchiyama (OU)</td>
<td>1998</td>
<td>PHE</td>
<td>Additive</td>
<td>P-subgroup assumption</td>
</tr>
<tr>
<td>Paillier</td>
<td>1999</td>
<td>PHE</td>
<td>Additive</td>
<td>Decisional Composite Residuosity Assumption (DCRA)</td>
</tr>
<tr>
<td>Damgard-Jurik (DJ)</td>
<td>2000</td>
<td>PHE</td>
<td>Additive</td>
<td>Decisional Composite Residuosity Assumption (DCRA)</td>
</tr>
<tr>
<td>RSA</td>
<td>1977</td>
<td>PHE</td>
<td>Multiplicative</td>
<td>Integer factorization problem.</td>
</tr>
<tr>
<td>El-Gamal</td>
<td>1984</td>
<td>PHE</td>
<td>Multiplicative</td>
<td>Diffie-Hellman problem</td>
</tr>
<tr>
<td>Boneh-Goh-Nissim (BGN)</td>
<td>2005</td>
<td>SWHE</td>
<td>Unlimited additions, one multiplication</td>
<td>Subgroup decision problem.</td>
</tr>
<tr>
<td>Gentry’s FHE</td>
<td>2009</td>
<td>FHE</td>
<td>Unlimited additions, and multiplication</td>
<td>Sparse Subset Sum (SSSP) assumption</td>
</tr>
</tbody>
</table>

3.3.1. FHE application. Cloud computation technology widely used in the contemporary world. FHE schemes are applicable in cloud computing to provide security assurance to the users; thereby their information remains confidential and inaccessible by unauthorized personnel [8]. With FHE, one can outsource the mathematical computations on confidential encrypted data to cloud server without requiring the user’s private key. FHE can be applied in databases’ arithmetic operations to maintain the confidentiality of the user’s data. Moreover, Gentry states in his blueprint that FHE permits private requests to a search engine. In this case, the user offers an encrypted query, and the search engine processes an encoded response without ever focusing on the question. Besides, it also allows searching within encrypted information where a user maintains encoded records on a remote server and later retrieve only data that satisfy some Boolean limitations, even though the server can hardly decrypt the files independently. On a broader scale, fully homomorphic encryption enhances the efficiency of protected multiparty computations [3][18].
4. Principles of FHE related assumptions

4.1. Lattice theory

Over the last decade, lattice theory is a remarkable field that started to show up as a foundation in modern cryptography, especially, in the infrastructure of fully homomorphic encryption (FHE). The attraction of lattice-based primitives comes from the fact that their security can often be based on worst-case scenario assumptions [24]. Gentry’s blueprint depended on ideals in different rings, and on the hardness of approximation lattice problems in the polynomial range. Lattice problems have considered as a standard tool in cryptography. Ideal lattices develop FHE Where they inherit natural mathematical Add and Mul operations from the ring since they correspond to ideals in the polynomial ring [3][18][4][20].

Definition 4.1.1. Lattice L. Is a set of vectors in n-dimensional Euclidean vector space with a strong periodic structure. When Euclidean space is at least 2-dimensional, each lattice has infinite entities in infinite bases, while in cryptography, all elements such as the ciphertext, public key, and secret key, (bit strings has fixed length), should be taken from a finite space. Consequently, the lattices utilized in the field of cryptography should be over a finite field.

Figure 3. A 2-dimensional lattice in the Euclidean plane

Definition 4.1.2. The basis of lattice L. A set of n vectors (v₁, ..., vₙ), can be viewed as a basis for a vector space. Lattices have many bases. Some bases are considered as “good,” while others considered as “bad.”

\[ L = \{ a_1v_1 + a_2v_2 + \cdots + a_nv_n : a_1, a_2, \ldots, a_n \in \mathbb{Z} \} \]

Definition 4.1.3. Lattices points. Any point of a lattice is the result of “linear combination” of those basis vectors with “integer coefficients.” Mathematical operations such as addition, subtraction, multiplication are performed on those points located in the vector space. SVP and CVP are the two-major hard lattice computational problems.

Definition 4.1.4. Shortest Vector Problem (SVP). It means finding the shortest vector \( v \) in lattice \( L \) with a nonzero value.

Definition 4.1.5. Approximate Closest Vector Problem (CVP). A problem of finding the vector \( v \) in the lattice \( L \) which is closest to a given target \( t \).

Solution: given a vector \( v \) not in \( L \), draw a domain around the target point \( t \). We have two cases:
- If the basis is “good” such that the basis consists of short vectors that are reasonably orthogonal to one another, then find a vertex \( v \in L \) that is closest to \( t \), a candidate for an approximate closest lattice vector.
- Using a “bad” basis, find the closest lattice vector that solves CVP such that much closer to the target \( t \) than the nearest vertex [29].

### 4.2. Learning with Error (LWE) Problem

**Definition 4.2.1. LWE Problem.** Consider a linear combination of a lattice basis vectors including a small error, the issue of searching and recognizing the difference between noisy random linear functions (with error) and uniformly random vectors is known as the “learning with error” problem. In other words, the problem of finding the closest vector to the vector linked with noise in a given lattice, specifically, by solving Closest Vector Problem (CVP) and/or the linear combination. Hence, the difficulty of resolving LWE restricted to finding a “good” (short or close) basis for a relevant lattice [24][29]. The Learning with Errors (LWE) problem, proposed by Regev [9][23]. As of late, many cryptographers employ LWE in constructing plenty of cryptographic schemes to obtain a high level of security and efficiency [24].

The LWE problem aims to retrieve a secret \( s \in \mathbb{Z}_q^n \), given a series of approximate random linear equations on \( s \), e.g., the input might be as follows:

\[
egin{align*}
14s_1 + 15s_2 + 5s_3 + 2s_4 & \approx 8 \pmod{17} \\
13s_1 + 14s_2 + 14s_3 + 6s_4 & \approx 16 \pmod{17} \\
6s_1 + 10s_2 + 13s_3 + 1s_4 & \approx 3 \pmod{17} \\
10s_1 + 4s_2 + 12s_3 + 16s_4 & \approx 12 \pmod{17} \\
9s_1 + 5s_2 + 9s_3 + 6s_4 & \approx 9 \pmod{17} \\
3s_1 + 6s_2 + 4s_3 + 5s_4 & \approx 16 \pmod{17}
\end{align*}
\]

Each equation corrects up to some small additive error (say, \( \pm l \)), and his goal is to recover \( s \). Answer is \( s = (0, 13, 9, 11) \) [23]. Retrieving \( s \) would be straightforward in case the error not introduced. After about \( n \) equations, \( s \) can be retrieved in polynomial time using “Gaussian elimination.” If there is an error, the problem might be more difficult.

The hardness of LWE assumption can be addressed by the following: Firstly, the best-known algorithms for LWE complexity order is exponential. Secondly, LWE is a natural generalization to the large module of the Learning Parity with Noise (LPN) problem. Finding an efficient algorithm for LPN and understanding its complexity is necessary to get the advantages from the small modulus. Thirdly, Numerous lattice-based cryptographic systems relied straight upon two average-case scenario problems, i.e., Learning with Errors (LWE) problem, and Short Integer Solution (SIS) problem. These two average-case obstacles have been appeared to accede strong lattices hardness guarantees. To be more specific, LWE seemed to be classified as the same level of difficulty with many worst-case scenario issues such as the Shortest Independent Vectors Problem (SIVP), the decision version of Shortest Vector Problem (GapSVP), and the Learning Parity with Noise (LPN) problem. On the same note, SIS seemed to be as hard as comparable worst-case complexity under a polynomial factor in the lattice dimension.

To get back to the point, for cryptographic applications, given a sequence of vectors \( v_1, \ldots, v_n \in \mathbb{Z}_q^n \), cryptographic schemes, which relied on SIS, and LWE problems, usually require rather large key sizes of order \( n' \). From a practical perspective, minimizing the key size to roughly linear size might lead to efficient enhancements [23][24][30].

**Definition 4.2.2. Small Integer Solution (SIS) problem.** Given a sequence of vectors \( v_1, \ldots, v_n \in \mathbb{Z}_q^n \), find a subset of them “a combination with small coefficients” that sums to zero (modulo \( q \)). SIS is defined as the problem of finding short vectors in a random lattice or code. According to Regev [9][23], many algorithms for solving the LWE problem has been proposed, e.g., maximum likelihood algorithm (naïve algorithm), the combinatorial algorithm invented by Blum, Kalai, and Wasserman (BKW) (best-known algorithm) [30]. The other most widely used algorithms to tackle LWE are Lattice basis reduction (LLL) algorithm, and algebraic Algorithms.

**Definition 4.2.3. Maximum Likelihood Algorithm.** The proof of this algorithm started with assuming that \( q \) is polynomial and the error distribution is normal. Then, demonstrating that after about \( O(n) \)
equations, the correct assignment will be the secret $s$. Since it is the only assignment that approximately fulfills the equations, finding $s$ can be achieved by trying all possible $q^e$ assignments. Then, performing an algorithm with running time $q^e=2^{O(\log n)}$ has been obtained, using only $O(n)$ equations [23][30].

**Definition 4.2.4. Blum-Kalai-Wasserman (BKW) combinatorial algorithm.** BKW presented as follows: Firstly, preparing refined running-time estimates for the data and functions requirements, thus understanding and solving the complexity of the LWE problem. Secondly, applying the estimates analysis to different parameters for LWE cryptographic cryptosystems. Finally, comparing with alternative schemes based on lattice reduction. As a result, a “new recovered upper bounds for the concrete hardness of these LWE-based schemes” is provided. It has shown that BKW algorithm exceeds previous estimates for lattice reduction algorithms [30].

**Definition 4.2.5. Lattice basis reduction (LLL) algorithm.** At the cost of an approximate exponential in the number of dimensions, LLL is used to reduce lattice basis in polynomial time. If the approximation is extremely important to the lattice space (modulo $q$), resolving Closest Vector Problem (CVP) outputs an error. For a given $q$, there are different dimensions $n$ (i.e., LWE is believed to be hard) [29].

### 4.3. Ring Learning with Error problem (R-LWE)

Comparing to LWE, R-LWE problem perform calculations on polynomials which have “better complexity” than vectors. A primary open question whether it is possible for cryptographic schemes, which applied LWE, to be more efficient by taking advantage of this additional arithmetic functions. Lyubashevsky, Peikert, and Regev [24] resolved this question by proposing a variant of LWE over rings known as “ring-LWE,” demonstrating that it also enjoys worst-case lattices complexity qualities. R-LWE is a simple expansion of LWE [9][23] to get more security and reduce ciphertext size.

The main idea behind R-LWE is that the vectors can be visible as polynomials modulo the $n^{th}$ cyclotomic polynomial “the unique irreducible polynomial with integer coefficients”, where $n$ is a power of 2. They restricted their algorithm to cyclotomic fields rather than other number fields. According to the authors, the ring-LWE distribution is pseudorandom, assuming that the worst-case lattices problems of the ring-LWE problem are hard for “polynomial-time quantum algorithms.” As a final point, many improvements and security proofs on LWE have quite often counterparts on the first truly practical R-LWE. However, the reasons behind working with R-LWE rather than LWE are that many of LWE-based schemes could be much more effective and practical when utilizing R-LWE instead. [24][29].

### 5. Evolution of FHE

#### 5.1. [G’09] Gentry – First FHE

Craig Gentry, an employee of IBM, invented the first encryption scheme that is fully homomorphic [3][18] based on ideal lattices in various rings. Specifically, the plaintext is a ring element, and the ciphertext is the encrypted plaintext linked with some noise, which related to an ideal. In Gentry’s original discovery, he started with SWHE plan and later “bootstrapped” to generate a fully homomorphic encryption system [31][32]. Gentry suggested a homomorphic scheme, which is roughly speaking similar to a Goldreich–Goldwasser–Halevi (GGH) lattice-based cryptosystem. He utilized ideal lattices to develop a bootstrappable encryption protocol. The reason behind using ideal lattices is because every ciphertext has a noise parameter which grows in the resulting ciphertext after any homomorphic operation applied to the original ciphertexts [10][31]. He later demonstrated that with a proper key generation technique, the security of that plan could be decreased to the worst case scenario of some lattice problems in ideal lattices. But this scheme is not yet bootstrappable, so Gentry portrayed in a change to squash the decryption scheme, by minimizing the degree of the decryption polynomial [16].

According to Gentry [3][18], the abstract of FHE is straightforward; He began his work with some assumptions as described in the following:

1. Given ciphertexts that encrypt $m_1, ..., m_i$, FHE should allow anybody to output a ciphertext that encrypts $f(m_1, ..., m_i)$ for any function $f$, as long as that function can be proficiently performed. The inputs, outputs, and middle value are constantly encoded, no information about $m_i, ..., m_i$ or $f(m_1, ..., m_i)$, or any plaintext value must leak.
2. A FHE scheme $e$ must have an effective function $\text{Evaluate } e$ that, given a valid $e$ key pair $(sk, pk)$, any circuit $y$, and any ciphertexts $c_i \leftarrow \text{Encrypt } e(pk, \pi_i)$, outputs:

$$c \leftarrow \text{Evaluate } e(pk, y, c_1, \ldots, c_i), \text{ such that Decrypt } e(sk, c) = y(\pi_1, \ldots, \pi_i).$$

3. Assume you have some encryption procedures with a “noise parameter” joined to each ciphertext, in which encryption produces a ciphertext with small noise, i.e., $< n$, whereas decryption performs as long as the noise is smaller than some threshold $N >> n$.

4. Consume that you have algorithm re-encrypt that takes a ciphertext $E(m_1)$ or $E(m_2)$ with noise $N' < N$. It provides a “new” ciphertext with noise parameters sufficiently smaller than $\sqrt{N}$. This re-encrypt calculation is sufficient to build FHE scheme out of the SWHE scheme.

5. Besides, suppose you have calculations $Add$ and $Multiply$ that can take ciphertexts $E(m_1)$ and $E(m_2)$ and provide $E(m_1 + m_2)$ for addition and $E(m_1 \cdot m_2)$ for multiplication. However, at the cost of adding or multiplying the noise parameters, this promptly provides a “SWHE” scheme that can deal with circuits of multiplicative depth almost $\log \log N - \log \log n$.

His strategies were like those utilized as a part of server-aided cryptography, where a client with a moderate device that needs to assign the more significant part of the decryption work to a server without permitting the server to decrypt entirely. Gentry required a second computational hardness presumption, like ones that have been concentrated on with regards to server-aided cryptography.

Gentry’s innovation can be summarized into three stages:

Stage 1: Construct Some-what Homomorphic Encryption (SWHE) scheme, in particular, an encryption plan which is eligible for evaluating “low-degree” polynomials on decrypted data homomorphically. In other words, which supports assessing a limited number of operations (many addition operations and one multiplication like the Boneh-Goh-Nissim cryptosystem) [20][35][6].

Stage 2: “Squashing” the SWHE decryption circuit until it is straightforward enough to be handled within the homomorphic capacity of the SWHE scheme. The goal is to get reasonably reduced decryption circuit complexity, thus changing the plan into a bootstrappable protocol, which has the same homomorphic ability. Squashing helps to figure out whether we can apply the bootstrapping hypothesis to the SWHE schemes, in other words, determine whether they equipped for assessing their own decryption circuits. The approach of squashing procedure accomplished by including a “clue” about the secret key to the evaluation key. To be more specific, instead of using the original secret key, an extra “hint” about the secret key added inside the public key, known as Sparse Subset-Sum Problem (SSSP). The public key enlarged with a large set of vectors, to such an extent that there exists an extremely sparse subset of them that indicates the secret key. Furthermore, this “extra indication” was insufficient to decrypt a ciphertext output by the first plan, but it could be utilized to “enlarge” the ciphertext, hence build another fresh ciphertext. Comparing to schemes like RSA or El-Gamal, which rely on exponentiation, Gentry’s essential FHE project depended on various complexity assumptions. The most complicated one is the difficulty of a decisional version of the Sparse Subset-Sum Problem (SSSP) that employed in squashing the decryption circuit. The processed ciphertext of the hidden plan can be decrypted with a low-degree polynomial in the bits of the ciphertext and the new secret key “equivalently a circuit of small depth” and acquires a bootstrappable cryptosystem [16][10].

Stage 3: Bootstrapping technique to get FHE scheme. After squashing stage, SWHE scheme is ready to evaluate “low-degree” polynomials and support a limited number of operations. To acquire FHE cryptosystem from SWHE scheme, Gentry gave a fabulous bootstrapping hypothesis. He demonstrated that critical SWHE scheme, a ciphertext could be “refreshed” by running the decryption circuit on it homomorphically using an encrypted private key, which brings about a minimized noise. It is clear that the noise vector roughly doubles in size for each addition evaluation, and squares for each multiplication evaluation. As a result, the decryption process could output mistaken raw data. At the point when we get a large or noisy ciphertext, the cryptographer can use the SWHE scheme to assess the decryption circuit using the encrypted secret key. Given two refreshed ciphertexts, one can perform an unlimited number of homomorphic computations (either addition or multiplication), which could not perform on the original ciphertexts because of the linked noise.

The fundamental reason for bootstrapping is to encrypt plaintext utilizing one key and perform operations until the error brought into the ciphertext reaches a specific margin. Then, perform the re-encryption function on the already encrypted (ciphertext) using the encrypted secret key, after that, decrypt using the first public key. The secret key is encrypted under the same public key $pk$, a necessity that referred to as “circular security,” i.e., it should be capable of encrypting its own particular secret key, and evaluate the function which is sufficient to permit HE concerning addition and multiplication.
The hard point of this technique is to attain a scheme that supports evaluating “high-enough degree” polynomials, and at the same time has decryption circuit that considered as “low-enough degree” polynomials. Whenever the degree of evaluated polynomials exceeds the decryption polynomials (multiply by 2), the scheme is known as “bootstrappable”, “and then it can be transformed to FHE scheme $3\,[18][32][16][35][36][20][6][10]$.

5.2. Gentry’s implementations

5.2.1. [SV’10] Smart and Vercauteren – first improvement. The initial effort to improve Gentry's fully homomorphic public key encryption scheme [2] made in 2010 by Smart and Vercauteren. Their construction followed Gentry's technique in producing FHE scheme from SWHE scheme, but they used “principle ideal lattices” of a prime determinant. They demonstrated that the public and private keys represented by two large integers (paying little attention to their dimension), and the secret key in decryption strategy represented by one large integer. They could realize the fundamental of SWHE scheme, yet they were not ready to support sufficiently large parameters to make Gentry's squashing procedure experience. Accordingly, they were not able to acquire a bootstrappable functionality or FHE scheme. Comparing to Gentry's original system, their scheme has smaller ciphertext expansion and key size. One issue in their execution was the complexity of key generation procedure for the SWHE scheme because they should generate many nominees to find one whose determinant is prime. Besides, they evaluated that the squashed decryption technique will have a degree of few hundreds. To support this methodology with their parameters, they must utilize a lattice dimension of at least $n = 227(= 1.3 \times 108)$, which is well past the capacities of the key generation process [2][16][31][10].

5.2.2. [SS’10] Stehle and Steinfeld – faster FHE. Scholars depicted an optimization considering ideal lattices and its examination [5] to obtain a faster FHE scheme. In their project, they analyzed the complexity of Gentry’s scheme related to the Sparse Subset Sum (SSSP) assumption in a more aggressive way. Also, they presented a probabilistic decryption process that actualized with a mathematical circuit of “low multiplicative degree.” After these changes together applied, fully homomorphic encryption scheme became faster, with a $O(\lambda^{3.5})$ bit complexity per elementary binary Add/Mul gate [5].

5.2.3. [GH’11a] Gentry and Halevi - first working implementation. This work [16] is an optimized version of the Smart–Vercauteren “principal-ideal lattices” cryptosystem [2] to handle the slow key generation procedure. They proposed a few major and minor optimizations along with bootstrapping method and squashing the decryption circuit. For key generation procedure, instead of requiring prime determinant, their scheme required that the Hermite Normal Form (HNF) of the lattice has a particular form. For decryption circuit, they did not require “full polynomial inversion” since they decrypted using a “simpler decryption circuit.” They used a single coefficient of the secret inverse polynomial, but the variation here is that they used “modular arithmetic” instead of “rational division.” As for the bootstrappable scheme, the public key includes examples of the Sparse Subset-Sum Problem (SSSP) which have a “very space-efficient representation.” The public key has encryption of all the secret key bits in the FHE scheme. Also, to improve the storing space for all encrypted data, they utilized a “space-time tradeoff.” To speed-up encryption, they utilized an effective algorithm for “batch evaluation” of many polynomials. The private key in their implementation is a binary vector of length “$S \approx 1000$”, the only $s = 15$ bits set to one, while the other bits set to zero. By representing the secret key in $s$ groups of $S$ bits, they got an important speedup. According to four different security levels (“toy,” “small,” “medium,” and “large”), their implementation with lattices has tested of several dimensions. From a “toy” setting in dimension $512$, to “small,” “medium,” and “large” settings in dimensions 2048, 8192 and 32768 respectively. Regarding the public-key size ranges, the size from $70\,Mb$ for the “small” setting, to $2.3\,Gb$ for the “large” setting [31][10][16].

5.2.4. [SV’11] Smart and Vercauteren - FHD supports SIMD. Gentry’s scheme encrypts and decrypts a plaintext of only 1-bit length. For this reason, scholars thought about improving particular operations, which could be processed on many bits in parallel to minimize runtime. When Smart-Vercauteren presented their variation of Gentry's blueprint [2], they specified that their cryptosystem could support SIMD style operations (Single Instruction, Multiple Data). Smart-Vercauteren recalled their SWHE
variation [2] and proved that it could support SIMD operations [33] in the finite field of characteristic two by modifying key generation. They demonstrated the possibility of choosing parameters for Gentry and Halevi implementation [16] to enable such SIMD operations, re-encrypting all data elements separately in parallel, thus obtaining FHE from SWHE scheme. They proved how such SIMD operations could be used to execute different higher level missions by exploring two situations which are implementing AES encryption homomorphically and seeking an encrypted database on a remote server [33][31].

5.2.5. [GH’11b] Gentry and Halevi - FHE without squashing. Scholars developed a new FHE approach without squashing [17] as the hybrid of SWHE and a “compatible Multiplicatively Homomorphic Encryption” (MHE) scheme depends on ideal lattices. It demonstrated how to bootstrap excluding the method of “squashing” the decryption circuit. Accordingly, this leveled FHE scheme constructed by excluding the necessity to assume the difficulty of the sparse subset sum problem (SSSP), thus, replaced with the decisional Diffie–Hellman (DDH) assumption. The primary strategy is to express the decryption procedure of SWHE schemes as a depth-3 (ΣΠΣ) algebraic circuit of a specific structure. Because of the particular form of the decryption circuit, the transformation to the MHE scheme should be possible without evaluating anything homomorphically. Consequently, at the stage of assessing this circuit through the bootstrapping technique, the authors developed an optimization of their level FHE scheme, where the whole leveled FHE ciphertext tentatively “compressed” into a one MHE plan (e.g., El-Gamal) ciphertext. In other words, the SWHE scheme should be able to evaluate the MHE scheme’s decryption circuit, rather than its own decryption circuit, thus getting rid of the “circularity” that made squashing step required. In the end, they showed the possibility to substitute the MHE scheme by additively homomorphic encryption (AHE) scheme, which is capable of encrypting discrete logarithms. This substitution allowed them to develop a leveled FHE scheme whose semantic security relied on the worst-case scenario of the Shortest Independent Vector Problem (SIVP) over ideal lattices (Ideal-SIVP) where the ciphertext length is reduced [17][31].

5.3. FHE Based on Learning with Error (LWE) and (Ring-LWE)

5.3.1. [BV’11a] Brakerski and Vaikuntanathan – FHE from R-LWE (PLWE). Experts presented FHE from Ring-LWE and security for key-dependent messages [7]. The experts proposed SWHE technique that is extremely simple to understand and apply. Then, they transformed it into FHE scheme using the same methods proposed by Gentry [3][18], i.e., “squashing” and “bootstrapping” techniques. The R-LWE assumption permits to eliminate the worst-case hardness on ideal lattices, thus providing a simple and secure scheme when encrypting “polynomial functions” of the private key. Their public key encryption scheme relies on the “Polynomial Learning with Errors” (PLWE) assumption, which is a simplified form of R-LWE, i.e., proposed by Lyubashevsky, Peikert, and Regev [24]. It has proved that the scheme is somewhat homomorphic, i.e., “circular secure.” Also, FHE can be achieved by bootstrapping, using “Gentry-style” squashing [7].

5.3.2. [BV’11b] Brakerski and Vaikuntanathan – FHE from LWE. Right after that, same experts constructed a novel FHE project based on standard learning with error (LWE) from Regev [9][23], known as (BV) scheme [4]. This design is unique as it does not follow the Gentry blueprint [3][18] and DGHV scheme [11] over the integers. Unlike Gentry’s blueprint which included new and comparatively untested cryptographic presumptions, BV cryptosystem works under a standard, well-realized, and hard solving cryptographic assumptions (LWE). However, it is easy to understand and execute and has very short ciphertexts making it more productive than prior ones [20][23][32]. The first step in BV scheme is a new method called “re-linearization” somewhat homomorphic encryption without ideals. Instead of employing SWHE scheme relying on the necessity of solving complexity assumptions on ideals in different rings. Re-linearization’s security depends only on the hardness of solving standard “short vector” problems on arbitrary (not necessarily ideal) lattices. The second step is “dimension-modulus switching” fully homomorphic encryption without squashing. It permits to exclude the necessity of the “squashing step” and bypass the difficulty of the sparse subset-sum problem (SSSP), hence reducing the ciphertext size and the decryption complexity of the scheme [31].
5.3.3. [LNV’11] Lauter, Naehrig, and Vaikuntanathan – FHE from R-LWE. Scholars proposed FHE based on Ring-LWE [21]. It is an implementation of the “Somewhat” public key encryption scheme from BV scheme [4] proposed by Brakerski and Vaikuntanathan while employing the computer algebra system Magma. In a nutshell, they thought that it is enough to implement “SWHE” scheme since it can be much faster, and more practical than FHE schemes. So, most of these applications support many addition operations, yet only a limited number of multiplications. Moreover, the re-linearization technique proposed in BV, which minimizes the size of the ciphertext to two ring components, has been employed in this implementation. They executed experiments using Magma’s polynomial algebraic for all calculations (addition and multiplication) in the ring of polynomials modulo a prime number, thus providing a similar efficiency with the same level of homomorphism and security. As a result, they proved that “encryption for the sum of 100 128-bit numbers can be calculated from the individual ciphertexts in 20 milliseconds on a laptop running Magma” [21].

5.3.4. [BGV’12] Brakerski, Gentry, and Vaikuntanatha – leveled FHE without bootstrapping. Within the same period, Brakerski, Gentry, and Vaikuntanatha constructed leveled BGV technology without bootstrapping [6] on techniques of BV scheme [4] while using R-LWE problem from [24]. Nowadays, because BGV encryption scheme significantly enhances efficiency and level of security on the “weaker assumptions,” it is considered as the first existing scheme proved practically in real-life applications. There are two versions of the BGV cryptosystems: one handles the integer vectors which based on learning with errors (LWE) problem [9][23], while the other version handles the integer polynomials which based on Ring-learning with errors (R-LWE) problem [24]. The main contribution of BGV cryptosystem was a new strategy of constructing a leveled FHE schemes that able to evaluate “arbitrary polynomial-size circuits” while eliminating the costly bootstrapping procedure. It is considered as a Public key (asymmetric) encryption scheme that encrypts bits where the noise vector grows only linearly with “multiplicative depth” rather than exponentially.

They started somewhat homomorphic encryption (SWHE) scheme based on “Ring LWE” assumptions [24] that have $2^L$ security against known attacks since it is much more efficient. In previous schemes which worked over ideal lattices, sub-exponential factors used, also a parameter $d$ (i.e., indicating the degree of the polynomials to be evaluated). But, in BGV scheme, security is based on lattice problems with “quasi-polynomial approximation factors” giving an exponential improvement. Moreover, the experts used a parameter $L$ (i.e., indicating the number of levels of the arithmetic circuit to be evaluated). Since BGV has per-gate computation only “quasi-linear” in the security parameter, the researchers provided several optimizations techniques to their FHE scheme which explained in the following:

1. Re-linearization procedure was used to reduce the dimension of the ciphertext and key sizes.
2. Dimension reduction was performed in the BV scheme [4] to accomplish a FHE instead of using squashing methods, while in this project, the “modulus switching” procedure was bundled into a “dimension reduction” technique, and then, named separately and examine carefully.
3. Modulus switching was refined to manage noise brought into ciphertexts during homomorphic multiplication operations without knowing the secret key, and without bootstrapping.
4. A combination of previous two procedures that minimize the multiplicative depth of the decryption circuit was applied. According to the authors, BV scheme re-linearization/ modulus switching methods can be used to convert a ciphertext $c_1$ (decrypted using one private key vector $s_1$) to a different ciphertext $c_2$ that encrypts the same plaintext. But in this scheme, a ciphertext $c_1$ (decrypted using a second secret key vector $s_2$) was transformed to a different ciphertext $c_2$.
5. Batching technique was the first optimization in the scheme. It permits to minimize the per-gate calculation from quasi-linear in the security parameter $\lambda$ to “polynomial.” This method accomplished by packing multiple plaintexts into each ciphertext homomorphically rather than one. However, its security gives approximately the same level of efficiency.
6. Bootstrapping was reemployed as an optimization rather than a requirement. It allows us to achieve per-gate computation quasi-quadratic in the security parameter, independent of the depth of the circuit evaluated.
7. Combining batching with the bootstrapping method is a great mix. With batching the bootstrapping optimization, circuits whose levels mostly have width at least $\lambda$ can be homomorphically evaluated with only $O(\lambda)$ per-gate computation, independent of the number of levels. In other words, batching
homomorphic evaluation of the decryption function permits to reduce the per-gate calculation by another factor of $\lambda$, from $O(\lambda^2)$ to $O(\lambda)$ (independent of $L$).

The resulting FHE scheme remains secure under a harder assumption. The others obtained a result that was like LWE scheme, however, in case of poor performance, they provided some extra optimizations. For relying on R-LWE, they had two different results:

- While eliminating bootstrapping method, and security depends upon the hardness of R-LWE for an approximation factor exponential in $L$, the result was a leveled FHE scheme that can evaluate $L$-level arithmetic circuits, where the per-gate calculation is $O(\lambda \cdot L^2)$.

- While using bootstrapping technique as an optimization rather than a requirement. However, security is based on the hardness of R-LWE for quasi-polynomial factors; the result was a leveled FHE scheme with $O(\lambda^2)$ per-gate calculation, independent of $L$ [6][20][28][14][29].

### 5.3.5. [GHS’12a] Gentry, Halevi, and Smart – FHE with polylog overhead.

At crypto’12, experts proposed FHE scheme with polylog overhead [46]. Under the R-LWE assumption, they illustrated that polylogarithmic overhead could be used to accomplish re-encryption procedure and homomorphic evaluation of arbitrary (wide enough) arithmetic circuits. Specifically, for security parameter $\lambda$, they constructed FHE schemes that can evaluate any width-$\Omega(\lambda)$ circuit with $t$ gates in time $t \cdot \text{polylog}(\lambda)$. To get low overhead, they recalled the batch homomorphic evaluation techniques of both FHE SIMD [33] and BGV scheme [6], which showed that homomorphic evaluations could be applied to “packed” ciphertexts that encrypt vectors of plaintext elements. Also, they proposed permuting/routing methods to move plaintext elements across these vectors efficiently, thus implementing general arithmetic circuit in a batched version without unpacking the plaintext vectors. Moreover, they introduced some optimizations that can enhance homomorphic evaluation in specific scenarios [46].

### 5.3.6. [GHS’12b] [GHS’15] Gentry, Halevi, and Smart - evaluation for the AES circuit.

Experts implemented a variant [38] of leveled FHE BGV scheme [6] that based on R-LWE, using the batch method from [46] while applying SIMD techniques from [33]. In this work, scholars illustrated AES-specific optimizations that can evaluate the AES-128 circuit in three different techniques, along with several tools for FHE evaluation which include a variant of the key-switching method from [4], and the modulus-switching technique from BGV [6]. Their implementation was built on NTL C++ library running over GMP as a software platform, and running on a large-memory machine. The first implementation took about 36 hours to evaluate an entire ten rounds AES encryption operation. Therefore, Using SIMD techniques, they could have processed 54 AES blocks in each evaluation in just under 40 minutes per AES block. The second implementation took just over 60 hours to evaluate the AES operation. However, they processed 720 AES blocks in each evaluation on the rate of just over five minutes per AES block. The third implementation, it showed an increase in time complexity; however, it turned out to be less competitive and attractive when encrypting a single block [38].

In 2015, same experts recalled their early implementation of leveled HE [38] and provided an updated version [39]. Their execution was built on top of the HElib library, running on a small laptop using 3 to 3.7 GB of RAM. When implementing without bootstrapping, it took about 4 minutes to evaluate an entire AES-128 encryption operation. Therefore, Using SIMD techniques, they could have processed 120 AES blocks in each evaluation in just under 2 seconds per AES block. When implementing with bootstrapping, it handled 180 blocks in only over 18 minutes, meaning 6 seconds/block. The scholars observed that “byte-slicing” and “bit-slicing” implementations could consume fewer levels per round function [39].

### 5.3.7. [GHS’12c] Gentry, Halevi, and Smart - better bootstrapping in FHE.

Gentry in collaboration with other scholars thought of an improvement of Gentry’s bootstrapping procedure [27] and joined their strategy with the SIMD homomorphic calculation. The major obstacle in the bootstrapping technique of Gentry’s breakthrough is the requirement to evaluate the modular arithmetic reduction operation homomorphically. It was done by simulating a “binary modular reduction circuit,” using bit operations on integer numbers that represented on the binary. The authors’ approach bypasses the reduction of one integer modulo another homomorphically to some degree, by using an arithmetic modulus near a power of two. It has proved that it is faster and simpler to depict and actualize than the standard binary circuit approach. Their strategy permits saving the encryption of the private key as a single ciphertext, hence minimizing the size of the public key. Their scheme can be joined with the SIMD homomorphic calculation procedures [33] as well, to run a bootstrapping technique that could be made in time.
A Comprehensive Study of Fully Homomorphic Encryption Schemes
Majedah Alkharji, Hang Liu, Mayyada Al Hammoshi

“quasilinear” in the security parameter. This last part requires expanding the methods from previous work to process arithmetic over some rings besides over fields. To be more specific, their scheme works with modulo very close to a power of two, instead of over characteristic two fields [27][31].

5.3.8. [Bra’12] Brakerski - FHE from classical GapSVP. Brakerski proposed FHE cryptosystem without modulus switching technique method from classical GapSVP [40]. He presented a new tensoring method for LWE-based FHE scheme. Comparing to all previous candidates, in which the ciphertext noise grows quadratically with every multiplication “before refreshing”, the noise in his work only grows linearly. This technique was used to construct a scale-invariant FHE scheme, whose properties only relied on the ratio between the modulus \( q \) and the initial noise level, and not on their absolute values. The scheme has some advantages, e.g., throughout the evaluation process, the same modulus was used instead of a ladder of moduli required while using the “modulus switching” method. Besides, complexity can be reduced from the worst-case scenario of the GapSVP assumption (with quasi-polynomial approximation factor), while previous works’ security could only exhibit a quantum reduction from GapSVP [40].

5.3.9. [GSW’13] Gentry, Sahai, and Waters - attribute based FHE. Experts presented LWE-based GSW homomorphic encryption construction [41]. They compared their project with prior LWE-based cryptosystems. In GSW plan, the approximate eigenvector method is used instead of expensive and complicated re-linearization procedure. To make their scheme faster and more straightforward, they accomplished homomorphic evaluation operations by just completing square matrix addition or multiplication. Unlike previous constructions which require getting the user-specific encrypted secret key, the encryption and homomorphic evaluations can be done by any user has only public parameters while eliminating the evaluation key. In this work, a “compiler” is used to transform LWE-based IBE scheme into identity-based FHE scheme. Also, the recent identity-based FHE scheme is transformed into Attribute-Based Encryption (ABE) scheme that allows data encrypted by the same index to be performed homomorphically without evaluation key [41].

5.3.10. [BV’14] Brakerski and Vaikuntanathan - lattice-based FHE. Brakerski and Vaikuntanathan recalled GSW scheme [41] and constructed a lattice-based FHE approach [42] under LWE assumption for polynomial approximation factors (DLWE). They proved that their leveled FHE matches the optimal hardness for traditional “non-homomorphic” lattice-based public key encryption scheme up to the small factor \( c > 0 \). The cryptosystem consists of three main observations: First, circuit evaluation in FHE usually means the homomorphic arithmetic operations, however, in this work, it is “Noise-bounded sequential evaluation” procedure that manages the size of noise efficiently. Second, using Barrington’s Theorem, larger circuit classes is sequentialized to transform into polynomial length. Third, to get the best-known approximation factor up to \( c \), a variant application of the dimension-modulus reduction of BV scheme [4] is employed to minimize the ciphertext size [42].

5.3.11. [SP’14] Sheriff and Peikert - faster bootstrapping with polynomial error. Researchers implemented a straightforward variant of GSW LWE-based encryption scheme [43] with a tighter analysis of error growth under homomorphic calculations. Comparing to prior projects, they proposed a new and faster bootstrapping algorithm with smaller runtime and approximation factor. Avoiding Barrington’s Theorem, an elementary arithmetic procedure is used instead. To bootstrap with \( 2\lambda \) security under conventional assumptions, their method requires only a quasi-linear \( O(\lambda) \) number of homomorphic operations on GSW ciphertext, which is quasi-optimal for bitwise schemes that encrypt just one bit per ciphertext like GSW system [43].

5.4. FHE scheme over the integers (DGHV) and its improvements

5.4.1. [DGHV’10] Dijk, Gentry, Halevi, and Vaikuntanathan - FHE scheme over the integer. Gentry collaborated with van Dijk, Halevi, and Vaikuntanathan to construct FHE technique called (DGHV) [11]. This work is more straightforward than Gentry’s initial one since all mathematical operations are done over the integers using only “elementary modular arithmetic computation” rather than ideal lattices over a “polynomial ring.” They adopted the same “squash decryption circuit” method to get a bootstrappable scheme and then applied refreshing ciphertext procedure to get FHE scheme. Nonetheless, keeping in
mind the end goal to understand the full homomorphism, the DGHV also performed a re-encryption technique before mathematical operations to reduce the noise components, which extraordinarily raised the calculation complexity. This technique made a high commitment to the advancement of FHE. The primary accomplishment was the plaintext comprised of integers as opposed to one bit. Also, they minimized the security of their SWHE scheme to find an approximate gcd integer, i.e., a list of integers, that is “near-multiples” of an invisible integer, gives an output of a hidden integer. Consequently, the development of the DGHV Construction depends on the complexity of the common divisors issue, defined by the prior work of Howgrave-Graham. This effortlessness comes to the detriment of public key size in $\hat{O}(\lambda^{10})$, which is considered too large for any functional framework [11][10][20][31].

5.4.2. [CMNT’11] Coron, Mandal, Naccache, and Tibouchi – shorter PK for DGHV. They proposed an improvement of DGHV FHE scheme [10], i.e., working over integers with smaller public keys. Their work minimizes the public key size of the SWHE scheme from $\hat{O}(\lambda^{10})$ to $\hat{O}(\lambda^{2})$. According to the authors, “the idea consists in storing only a smaller subset of the public key and then generating the full public key on the fly by combining the elements in the small subset multiplicatively.” Rather than performing the encryption with a linear form, a quadratic form in the public key components has been used to get a shorter public key. They demonstrated that the cryptosystem remains secure, in light of a more powerful variation of the approximate GCD assumption. The second contribution was to depict the first implementation of the DGHV scheme over the integers under their variation while borrowing some of the optimizations from the Gentry-Halevi implementation [16] of Gentry’s breakthrough [3][18]. From Stehle and Steinfeld [5] they utilized the repeated analysis of the sparse subset sum assumption; however, because of the elevation in the error likelihood for their set of parameters, they did not use the probabilistic decryption procedure. Their main limitation was to define a secure collection of concrete parameters. Their method was to implement the known attacks, measure their running time and extrapolate for large parameters; Then, they can fix the concrete parameters according to the desired level of security. They attained almost the same level of performance as the Gentry-Halevi implementation [16]. To be more accurate, they use the same four security levels, even though they might not be similar due to the different concepts of “security bits.” They defined the security parameters as “toy,” “small,” “medium” and “large,” corresponding to 42, 52, 62 and 72 bits of security. With a public key size of 800 MB, Encryption and re-encryption take 3 minutes and 14 minutes for “large” parameters. This result proved that FHE could be performed utilizing basic mathematical operations [10].

5.4.3. [CNT’12] Coron, Naccache, and Tibouchi – shorter pk compression and modulus switching for DGHV. Same scholars constructed another improvement in which a compression approach [22] for minimizing the public key size from $\hat{O}(\lambda^{2})$ down to $\hat{O}(\lambda^{2})$. They acquired an implementation of the FHE scheme with a 10.1 MB public key rather than 802 MB utilizing comparable security parameters. The experts proposed “public key compression” method to increase the efficiency and decrease the public key size of DGHV schemes. Then they proposed what it’s called “extension to higher degrees” to expand the quadratic encryption procedure of DGHV’s first improvement [10] to higher degrees to get a shorter public key. They demonstrated that a specific family of quadratic hash functions is sufficiently close to being “pairwise independent,” thus proving that the scheme remains semantically secure under the approximate-GCD assumption. Also, “modulus switching and leveled DGHV Scheme” proposed to show how to apply (BGV) FHE scheme without bootstrapping [6] with the DGHV scheme over the integers [11][22].

5.4.4. [CLT’13] Coron et al. – batch DGHV. Same experts and others expanded the FHE DGHV scheme to obtain a batch DGHV scheme [44] with similar features of batch R-LWE-based FHE schemes [6][38][46]. Batching method allows obtaining scheme that encrypts and handle plaintexts vector as a single ciphertext homomorphically. In this work, the authors proposed two different techniques depending on two variant assumptions. The first method depends on a new decisional approximate-GCD problem, while the second one relied on the more classical computational error-free approximate-GCD assumption. Also, given the ciphertext and the public key, they showed how to accomplish permutations arbitrarily on the vector of plaintext. Comparing with the running time of Gentry et al. at Crypto 2012 [38], These contributions provided a better performance of the homomorphic evaluation and encrypted up to 531 AES ciphertexts with a cost of about 12 minutes per AES ciphertext even with the bootstrapping procedure [44].
5.4.5. [CLT’14] Coron, Lepoint, and Tibouchi - scale invariant FHE over the integers. At Crypto 2012, Brakerski constructed LWE-based scale-invariant FHE scheme without modulus switching from classical GapSVP [40]. Coron et al. described a variant of DGHV scheme with the same scale-invariant technique, but it is based on Approximate-GCD assumption [45]. Their scheme does not use modulus switching, and the noise grows linearly in the number of levels instead of exponentially. It has a single secret modulus $p$ to be homomorphically evaluated in the multiplicative depth of the circuit. They constructed a leveled FHE scheme; therefore, using bootstrapping method, the scheme can be transformed into pure FHE scheme. They also homomorphically evaluated the full AES encryption circuit and provided concrete parameters and timings. Compared to prior implementations, this scheme obtained a competitive execution. According to the authors, they had “about 23 seconds (resp. 3 minutes) per AES block at the 72-bit (resp. 80-bit) security level on a mid-range workstation.” Finally, they showed the equality between the classical computational Approximate-GCD problem and the (error-free) decisional Approximate-GCD problem introduced by [44]. This equivalence permits to avoid the extra noise appended during encryption in all the integer-based FHE schemes described so far [45]. “Table 2.” lists FHE schemes in order along with, how they work, and the complexity of these cryptosystems.

Table 2. FHE Scheme, Brief Description, and Security Assumption of HE Schemes

<table>
<thead>
<tr>
<th>FHE Scheme</th>
<th>Year</th>
<th>Scheme Outline</th>
<th>Security Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gentry’s FHE</td>
<td>2009</td>
<td>First FHE scheme, based on ideal lattices</td>
<td>The hardness assumption of SSSP</td>
</tr>
<tr>
<td>Smart and Vercauteren</td>
<td>2010</td>
<td>1st Improvement of Gentry’s scheme with a small key and ciphertext size, using “principal-ideal lattices.”</td>
<td>The complexity of key generation procedure (finding small principal ideal lattice)</td>
</tr>
<tr>
<td>Stehle and Steinfeld</td>
<td>2010</td>
<td>Two main improvements of Gentry’s scheme to obtain a faster FHE scheme</td>
<td>The hardness assumption of SSSP</td>
</tr>
<tr>
<td>Gentry and Halevi</td>
<td>2011</td>
<td>1st working Implementation of Gentry’s scheme by some optimizations</td>
<td>The hardness assumption of finding small principal ideal lattice</td>
</tr>
<tr>
<td>Smart and Vercauteren</td>
<td>2011</td>
<td>FHE scheme enables SIMD operations</td>
<td>The decision variant of BDDP, or SSSP</td>
</tr>
<tr>
<td>Gentry and Halevi</td>
<td>2011</td>
<td>FHE without squashing cryptosystem using depth-3 arithmetic circuits</td>
<td>The decisional (DDH) assumption, or SIVP problem over ideal lattices (Ideal-SIVP)</td>
</tr>
<tr>
<td>Brakerski and Vaikuntanathan</td>
<td>2011</td>
<td>FHE from R-LWE and Security for Key Dependent Messages</td>
<td>The hardness of PLWE Problem (a simplified form of R-LWE)</td>
</tr>
<tr>
<td>Brakerski and Vaikuntanathan</td>
<td>2011</td>
<td>FHE scheme based on LWE (BV scheme)</td>
<td>The hardness of LWE Problem</td>
</tr>
<tr>
<td>Lauter, Naehrig, and Vaikuntanathan</td>
<td>2011</td>
<td>Implementation of FHE scheme based on R-LWE</td>
<td>The hardness of R-LWE Problem</td>
</tr>
<tr>
<td>Brakerski, Gentry, and Vaikuntanathan</td>
<td>2012</td>
<td>Leveled FHE scheme without bootstrapping (BGV scheme)</td>
<td>R-LWE for an approximation factor exponential, or R-LWE for quasi-polynomial approximation factors</td>
</tr>
<tr>
<td>Gentry, Halevi, and Smart</td>
<td>2012</td>
<td>FHE with polylog overhead</td>
<td>The hardness of R-LWE Problem</td>
</tr>
<tr>
<td>Gentry, Halevi, and Smart</td>
<td>2012</td>
<td>Homomorphic evaluation for the AES circuit</td>
<td>The hardness of R-LWE Problem</td>
</tr>
<tr>
<td>Gentry Halevi, and Smart</td>
<td>2012</td>
<td>Improvement of Gentry’s bootstrapping, then join it with SIMD operations</td>
<td>The quasi-polynomial approximation factors</td>
</tr>
<tr>
<td>Brakerski</td>
<td>2012</td>
<td>FHE from classical GapSVP (scale-invariant FHE scheme)</td>
<td>The hardness of LWE Problem</td>
</tr>
</tbody>
</table>
Applying homomorphic encryption on the cloud computing is a fresh idea of security. The exploration of HE schemes highlights essential concepts regarding the generation of cryptographic needs and leads to improve cloud computing benefits thus enhancing the client satisfaction. The role of adopting HE algorithms by the CSP to ensure the security of clients’ confidential data cannot be underestimated. It is used to provide more efficient security services and support easy retrieval of the data. Cloud computing draws researchers’ attention to develop practical FHE schemes. Therefore, applications of fully homomorphic encryption have increased in the recent times with the spread of cloud computing. In fact, the current level of the fully homomorphic encryption applications points towards the importance of the further research to improve these schemes and address its weaknesses. Precisely, the most efficient FHE method is still very costly and suffer from poor performance. Also, performing computations utilizing FHE takes quite long. However, these limitations regarding speed and ability to manage the massive load of data can be overcome as schemes improve.

Comprehensively, this paper has simplified many definitions related to HE. The role of HE in the existing applications have been investigated, and the current state of the art has been reviewed and presented systematically.

Future work will be applying two different FHE schemes and compare their effectiveness. One is based on machine learning with Error (LWE), and the other is FHE over the integers using elementary modular arithmetic computation (DGHV). The third task is to apply the FHE schemes to cloud database applications.

8. References


Acknowledgment

Majedah Alkharji would like to thank Ministry of Education - Saudi Arabia Culture Mission (SACM) for funding her Ph.D. scholarship.

\^Majedah Alkharji M.SC.

Majedah Alkharji is a current computer science Ph.D. candidate in the Department of Electrical Engineering and Computer Science at the Catholic University of America (CUA), Washington DC. In 2011, she received her master degree in Computer Science and got a certificate in engineering management from the Catholic University of America (CUA). She got the bachelor degree in computer science from the King Saud University at the kingdom of Saudi Arabia in 1999 and have six years teaching experience in the field of computer science, Saudi Arabia. She has been published four research papers.

\^Dr. Hang Liu Ph.D.

Hang Liu received his Ph.D. degree in Electrical Engineering from the University of Pennsylvania. He joined the Catholic University of America as an Associate Professor in the Department of Electrical Engineering and Computer Science in 2013. Before joining CUA, he had more than ten years of research experience in the networking industry and worked in senior research and management positions at several companies. He also led several industry-university collaborative research projects. He was an adjunct professor of WINLAB, the ECE Dept., Rutgers University from 2004 to 2012. Dr. Liu has published more than 100 peer-reviewed papers in leading journals and conferences and received two best paper awards and one best student paper award. He is the inventor/co-inventor of over 80 granted and pending international patents.

\^Dr. Mayyada Al Hammoshi Ph.D.

Mayyada Al Hammoshi holds a Ph.D. in Computer Science, Computer Networking, and Communications from the College of Computer Science and Math at Mosul University (Iraq). She served as the principal investigator for the Omani Research Council Grant's research toward the development of a multiagent grid computing facility based on graphics processing units. In her more than 20 years of experience, Dr. Al Hammoshi has served as a faculty member, research department head, and journal editorial board member. She has been published in more than 15 referred journals.