Homomorphic Encryption the "Holy Grail" of Cryptography

Dalia Tourky, Mohamed ElKawkagy, Arabi Keshk
Computer Science Department
Higher Institute of Computer Science and Information Systems
Ministry of Higher Education
Egypt
e-mail: Dalia_Tourky@hotmail.com; M_nabil_shams@yahoo.com; arabikeshk@yahoo.com

***Abstract*—Homomorphic encryption is the cryptosystem which allows computations on encrypted data achieving the goal of protecting the privacy of data during communication and storage process. For decades' Homomorphic encryption is the Holy Grail of cryptography, nonetheless constructing algorithms and implementing methods of homomorphic encryption schemes are sophisticated. This paper clarifies the concept, categories and construction methods of Homomorphic encryption by sailing in its complicated algorithms from the starting spark of algebraically homomorphic encryption to the approach of constructing fully homomorphic encryption from SomeWhat homomorphic encryption.**

Keywords-component; privacy of data – computation on encrypted data – homomorphic encryption – fully homomorphic encryption

I. INTRODUCTION

Privacy and confidentially are prime requirements for communication, whereas the traditional encryption algorithms are one of security mechanism that is responsible for providing the privately to digital data during transmission and storage process. A reasonable demand for a new encryption scheme that is capable of preserving the privacy of data over the public cloud or untrusted computer announced.

Homomorphic encryption is the replay for this demand in which it allowed a particular type of computations on encrypted data and generated an encrypted result which when decrypted gives the same result of computations performed on decrypted data.

Privacy homomorphism, the starting spark of homomorphic encryption schemes that proposed in 1978 by Ronald L. Rivest et al. [1]. They introduce for the first time the idea of computing on encrypted data without a prior decryption, and the results was a new challenge and announcement of research questions, is it possible to have a secure scheme that can be able to compute an arbitrary function on encrypted data? Can this secured scheme be implemented and practical?

In the same year 1978, Ronald L. Rivest et al., [2] proposed the first implementation method (RSA) a public key scheme with the homomorphic property. The methodology based on to achieve the semantic security the message is padding with random bits before encryption but. Unfortunately, these padding results lose the homomorphic property.

For three decades the research community received various encryptions schemes with homomorphic properties support simultaneous either adding or multiplying encrypted data, for example, Goldwasser- Maicali [3], El-Gamal [4], and paillier [5].

Similar to the construction of Paillier Dan Boneh et al., [6] homomorphic public –key scheme based on finite groups of composite order that support bilinear maps. Which the first time both arbitrary additions and single multiplication on encrypted data.

Enormous progress obtained in 2009 when Gentry [7] announces his success to convert the somewhat homomorphic scheme into a fully homomorphic scheme that is capable of evaluating an arbitrary number of additions and multiplications on the encrypted data

II. HOMOMORPHIC ENCRYPTION

The lineage of the word homomorphic is to the Greek language which translates as the same form or the same shape, and the core concept of it is the transformation that has the same effect on two different sets of objects.

The principle of homomorphic encryption is a base on abstract algebra term homomorphism, which refers to mapping φ between two groups $(G, \emptyset)$ and $(H, *)$ such that $\varphi(x \emptyset y) = \varphi(x) * \varphi(y)$ for $x, y \in G$ and $\varphi(x), \varphi(y) \in H$.

- Example:

Let G be a group of real number under addition G = under + and H be a group of positive real number under multiplication the H = $\mathbb{R}^+$ with a function f which maps $x \mapsto e^x$, to make sure this a homomorphism, verify that $f(x + y) = f(x) * f(y)$ by the definition of f then.

Homomorphic encryption applies the same concept, consider a cryptosystem c that has encryption function ε, plaintext x_n ,ciphertext c_n such that $\varepsilon(x_n) = c_n$ and some operation :

1) c is consider additively homomorphic iff. $\exists \Delta$: $\varepsilon(x_1) \Delta \varepsilon(x_2) = \varepsilon(x_1 + x_2)$.
2) c is consider multiplicatively homomorphic iff. $\exists \Delta$: $\varepsilon(x_1) \Delta \varepsilon(x_2) = \varepsilon(x_1 x_2)$

III. HOMOMORPHIC ENCRYPTION CATEGORIES

The Three general classes of Homomorphic Encryption illustrated in Figure 1.

A. *Partially Homomorphic Encryption (PHE)*

The encryption scheme that has homomorphic property supporting only one type of operations either addition or multiplication. Example Goldwasser-Maicali encryption scheme.

B. *Somewhat Homomorphic Encryption (SHE)*

The encryption scheme that has thehomomorphic property that supports a limited number of additions and multiplication operations.

C. *Fully Homomorphic Encryption (SHE)*

A homomorphic encryption scheme that can perform an arbitrary number of both additions and multiplications operations.

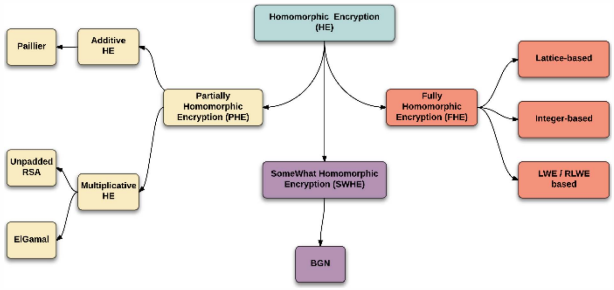

Figure 1. Homomorphic encryption schemes.

IV. SPARTIALLY HOMOMORPHIC ENCRYPTION (PHE)

A. *Paillier Encryption Scheme*

Paillier encryption scheme [5] is a probabilistic public-key algorithm. Its underlying mathematical based on composite residuosity assumption, i.e. n is equal to multiplying two large prime number p and q chosen randomly.

The algorithm:

1) The parameter:
- Two large primes p and q chose randomly.
- Select random integer $g \in \mathbb{Z}_{n^2}^*$.(i.e. g in invertible of n^2)

2) Key generation
- Compute the private key.
- The public key =(g, n).

3) Encryption
- For a message m to be encrypted it has to be less than such that $m \in \mathbb{Z}_n$, the ciphertext $c = g^m r^n \ mode \ n^2$ for random $r \in \mathbb{Z}_n^*$.

4) Decryption
- $m = \frac{L\,(c^\lambda \ mod \ n^2)}{L(g^\lambda \, mod \, n^2)(\ mod \ n)} \ where \ l(u) = (u-1)\,/\,n$

The scheme is an additively homomorphic cryptosystem; this means that given only the public key and the encryption of m_1 and m_2, one can compute the encryption of $m_1 + m_2$

- *Homomorphic properties*

Given two ciphertexts $E(m_1, PK) = g^{m_1} r_1^n (mod\ n^2)$ and $E(m_2, PK) = g^{m_2} r_2^n\ (mod\ n^2)$, where r_1 and r_2 are randomly chosen from$\mathbb{Z}_n^*$.

Consequently, the homomorphic property is the product of two ciphertexts will decrypt to the sum of their plaintexts, i.e.,

$$D\big(E(m_1, PK).E(m_2, PK)(mod\ n^2)\big) = m_1 + m_2 (mod\ n)$$

Because $E(m_1, PK).E(m_2, PK) = (g^{m_1} r_1^n)(g^{m_2} r_2^n)(mod\ n^2)$

$$= g^{m_1+m_2}(r_1 r_2)^n (mod\ n^2)$$

$$= E(m_1 + m_2, PK)$$

- Example

Encrypt the two message$m_1 = 34$ and$m_2 = 16$

With public key $(n, g) = (2501, 92)$ and$r_1 = 5, r_2 = 7$

1) $C_1 = 92^{34}.5^{2501}\,(mod\ n^2) = 1129735$
2) $C_2 = 92^{16}.7^{2501}\,(mod\ n^2) = 5140305$
3) $C_1.C_2 =$ **2010769**
4) $m_1 + m_2$
5) $C_1 + C_2 = 92^5.35^{2501}\,(mod\ n^2) =$ **2010769**

B. *RSA Scheme*

The firstsuccessful public-key scheme with a homomorphic property with a block size of 1024 bit or 309 decimal digits.

- The algorithm:

1) Key generation
- Select two random p and q
- Compute $n = p \times q$
- Compute $\emptyset(n) = (p-1)(q-1)$
- Select $e$such that e is relatively prime to $\phi(n)$
- Determine d such that $d \equiv 1(\ mod\ \emptyset(n))$
- Public key = {n, e}
- private key = {n, d}

2) Encryption: ciphertext $C = M^e \ mod \ n$ where $M < n$

3) Decryption: plaintext$M = C^d \ mod \ n$

The scheme is a multiplicative homomorphic cryptosystem; this means that given only the public key and the encryption of m_1 and m_2 , one can compute the encryption of $m_1 \times m_2$

- *Homomorphic properties (unpadded RSA)*

Given two ciphertexts

$$E(m_1, PK) = (m_1^e \ mod \ n) \text{ and } E(m_2, Pk) = (m_2^e \ mod \ n)$$

$$C_1 = m_1^e mod\ n$$

$$C_2 = m_2^e \ mod \ n$$

$$C_1 \times C_2 = m_1^e \times m_2^e \ mod \ n = (m_1 \times m_2)^e \ mod \ n$$

This proves that unpadded RSA has the homomorphic property but unfortunately is insecure.

- Example

Assume the following RSA keys: $PK = (187,7), SK = (187,23)$ and message $m_1 = 88$ and $m_2 = 99$

Encryption: $C_1 = 88^7 mod\ 187 = 11$

$$C_2 = 99^7 mod\ 187 = 176$$

$$C_1 \times C_2 = 11 \times 176\ mod\ 187 = 66$$

This is the same as$m_1 \times m_2 = 88 \times 99\ mod\ 187 = 110$

$$Enc(m_1 \times m_2) = 110^7 mod\ 187 =$$ **66**

V. THE ESSENTIAL TERMINOLOGY OF HOMOMORPHIC CRYPTOSYSTEM

Any conventional cryptosystem as illustrated in Figure 2 has three basic algorithms $KeyGen_{\varepsilon}$, $Encrypt_{\varepsilon}$ and $Decrypt_{\varepsilon}$, but homomorphic cryptosystem has an additional algorithm $Evaluate_{\varepsilon}$, furthermore homomorphic cryptosystem can be either symmetric or Asymmetric, but most of currently Homomorphic encryption schemes are based on Asymmetric cryptosystem.

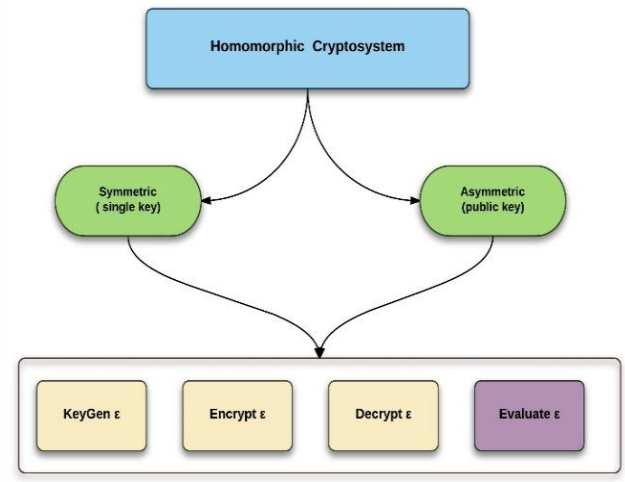

Figure 2. Homomorphic cryptosystem ε = (KeyGen, Enc, Dec, Eval).

Gentry's encryption scheme is homomorphic on Boolean circuits consisting of gates for addition and multiplication mod 2; This means that the plaintext space of encryption is limited to {0,1}. [8]

The general process and components of Asymmetric Homomorphic encryption scheme Asymmetric Homomorphic encryption scheme is illustrated in Figure 3 and described as the following.

- $\boldsymbol{KeyGen}_{\varepsilon}$: running in a time of polynomial which is a security parameter λ that specifies the bit-length of the keys and outputs keys (PK, Sk), where PK is the public key and Sk is the private key.
- $\boldsymbol{Encrypt}_{\varepsilon}$: converts plaintext $x_1, x_2, \ldots\ldots, x_n$ to ciphertext as $c_i \leftarrow Encrypt_{\varepsilon}(PK, m_i)$where m is a plaintext bit or integer.
- $\boldsymbol{Decrypt}_{\varepsilon}$: converts ciphertext to plaintext as $m_i \leftarrow Decrypt_{\varepsilon}(SK, c_i)$.
- $\boldsymbol{Evaluate}_{\varepsilon}$: The additional algorithm for homomorphic encryption scheme which allows aset of function f to be computed on any ciphertext $c_1, \ldots\ldots\ldots, c_t$ as $c' \leftarrow Evaluate_{\varepsilon}(PK, f, c_1, \ldots\ldots\ldots, c_t)$. The function f can be circuit-base represented by an arithmetic circuit or Boolean circuit or non-circuit based defined as a mathematical function.
- ***C-homomorphism:*** Let C be a class of functions. A scheme HE is C-homomorphism if $c' \leftarrow Evaluate_{\varepsilon}(PK, f, c_1, \ldots\ldots\ldots, c_t)$ and $f(x_1, x_2, \ldots\ldots, x_i) = Decrypt_{\varepsilon}(SK, c_i)$.
- ***Circuit-privacy*** is a property of homomorphic encryption which means that output of $Evaluate_{\varepsilon}$ does not exposes any information the circuit that evaluates.
- ***Compact Homomorphic encryption:*** The scheme ε = *(KeyGen, Enc, Dec, Eval)* is compact if the size of c' is independent of either the number of inputs or the complexity of function$f$ depend only on thesize of the output of$f$.
- ***Fully Homomorphic Encryption:*** A scheme εconsidered fully homomorphic if it is both compact and homomorphic for the class of all arithmetic circuits over GF (2).
- ***Leveled fully homomorphic encryption:*** is a homomorphic scheme where *KeyGen* algorithm gets an additional input$l$ *as* $(pk, evk, sk) \leftarrow KeyGen_{\varepsilon}(\lambda, l)$ and the resulting scheme is homomorphic for all depth- l binary arithmetic circuit.

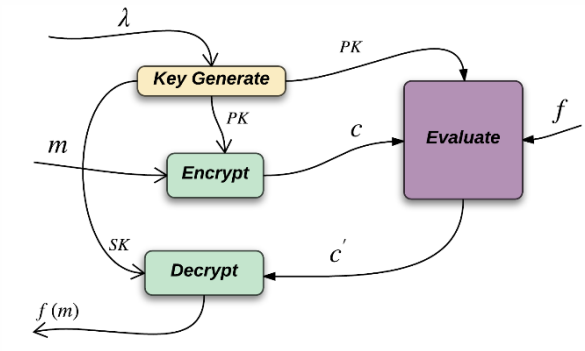

Figure 3. Asymmetric homomorphic encryption scheme.

VI. CONSTRUCT A FULLY HOMOMORPHIC ENCRYPTION

A. Gentry Homomorphic Scheme

Gentry announced to the cryptography nation his construction of first fully homomorphic encryption scheme based on the ideal lattice. Gentry's construction depends on serial of steps as illustrated in Figure 4:

1) Construct a somewhat homomorphic encryption that can evaluate low-degree polynomials to support a limited number of both additions and multiplications and because each ciphertext has noise component that is increasing by applying each homomorphic operation it reach reality that is inevitable which at a certain point the resulting ciphertext cannot correctly decryptanymore.
2) Squash the decryption circuit of the somewhat homomorphic encryption scheme by transforming it into the same homomorphic space but the decryption circuit could be depicting as low-degree of a polynomial in bits of ciphertext and *sk*
3) Bootstrapping encryption based on the idea of a somewhat homomorphic encryption scheme that can evaluate its owned decryption function; refreshing the ciphertext by running the decryption

algorithm on the Homomorphically using an encrypted evk key and the output is a refresh ciphertext with reduced noise.

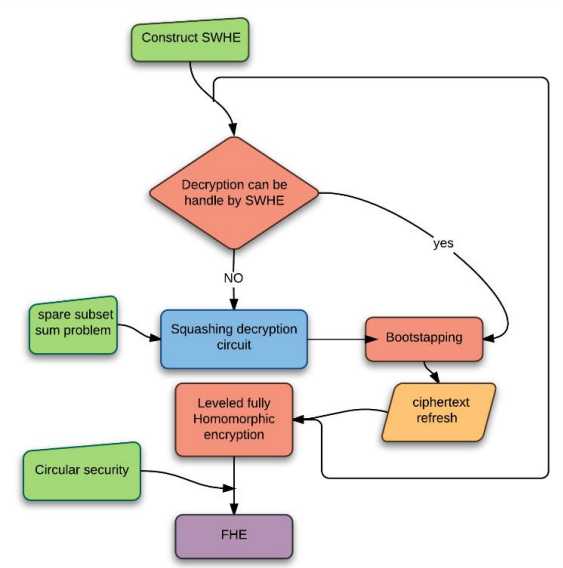

Figure 4. Algorithm of construct FHE.

- ***Bootstrappable Encryption Definition***: let $\varepsilon = (KeyGen, Enc, Dec, Eval)$ be a homomorphic encryption scheme, and for every value of the security parameter λ let $C_\varepsilon\ (\lambda)$ be a set of circuits with respect toε is correct. whereε is bootstrappable if $D_\varepsilon(\lambda) \subseteq C_\varepsilon\ (\lambda)$ hold every λ.[8]

B. *Somewhat Homomorphic Encryption over INTEGERS*

By using the same Gentry's algorithm of constructing [8], the second Fully homomorphic encryption scheme was developed but over integers instead of ideal lattices. The second scheme was better just in one matter in which plaintext consisted of integers rather than single bit nevertheless the public-key was in $\tilde{O}(\lambda^{10})$ which mean it is not suitable for any practical system.

- The construction

1) $\boldsymbol{KeyGen}_\varepsilon$: The key is an odd integer, chosen from some interval$p\epsilon[2^{\eta-1}2^{\eta}]$
2) $\boldsymbol{Encrypt}_\varepsilon(p,m)$: To Encrypt a bit $m\epsilon\{0,1\}$, set the ciphertext as an integer whose residue $mod\ p$ has the same parity as the plaintext. Namely, set $c = pq + 2r + m$Where the integers $q\ and\ r$ chose at random in some other prescribed intervals, such as $2r$is smaller than $p/2$ in absolute value.
3) $\boldsymbol{Decrypt}_\varepsilon$(p, c): m= $(c\ mod\ p) mod\ 2$.
4) $constraints: r \approx 2\sqrt{\eta}$and $q \approx 2^{\eta^3}$

C. *Fully Homomorphic Property*

Given two ciphertext as shown in Figure 5: $c_1 = pq_1 + 2r_1 + m_1$ and $c_2 = pq_2 + 2r_2 + m_2$ having

Distance to nearest multiple of $\boldsymbol{P}$

$$C_1 + C_2 = (q_1 + q_2)p + 2(r_1 + r_2) + (m_1 + m_2)$$

$$C_1.C_2 = (pq_1p_2 + 2q_1r_2 + m_1q_2 + m_2q_1)p + 2(2r_1r_2 + m_1r_2 + m_2r_1) + m_1m_2$$

When $r_1 + r_2 < p/2$

$$2r_1r_2 + m_1r_2 + m_2r_1 < p/2$$

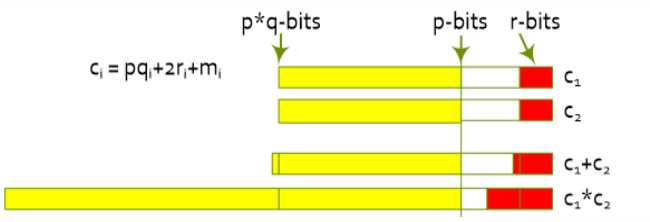

Figure 5. Ciphertext and noise size expansion.

D. *Bootstrappable Encryption*

- Somewhat Homomorphic encryption evaluates F to the depth-d as shown in Figure 6.
- $F = D_{sk}(m)$ is the decryption algorithm, that can be evaluated by somewhat homomorphic encryption at most depth$(d-1)$

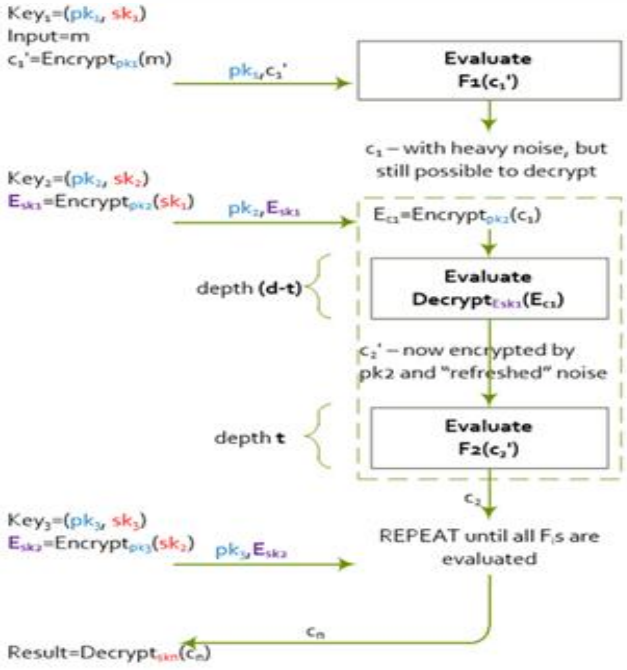

Figure 6. Bootstrappable encryption.

VII. FULLY HOMOMORPHIC ENCRYPTION SCHEMES LATTICE-BASED

A. *FHE with Relatively Small Key and Ciphertext Sizes*

Smart and Vercauteren [9] followed the same approach of Gentry's theory but based on the average-case of the small principal ideal problem. It considers the first attempt to implement Gentry's theory, but Unfortunately, it proves that Gentry's scheme was still not practical because they fail to implement boots- trappable due to the inability to develop

keys for the parameter size of N =212 which is essential for security purpose.

B. *Implementing Gentry's Fully Homomorphic Encryption Scheme*

Gentry and Halevi [10] follow the same methodology variant used in [9] but with optimization approach for all aspects of the scheme. For the key generation of the SomeWhat homomorphic encryption, it does not require full polynomial inversion. The implementation tested with four lattices dimensions classified into "toy" setting in dimension 512, to "small", "medium" and large setting of dimensions 2048,8192 and 32768, respectively. The public-key size for "small" setting is 70 Megabytes and for large setting is 2.3 Gigabytes. Nevertheless, the implementation was run on a high-end workstation (an IBM system x3500 server, featuring a b4-bit quad-core Intel Xeon E5450 processor running at 3GHz with 12MB L2 cache and 24GB of RAM), the time to run one bootstrapping operation takes for small setting 30 seconds and for large setting it took 30 minutes.

C. *Fully Homomorphic Encryption SIMD Operations*

Smart and Vercauteren [11] appoint the parameters setting of Gentry and Halevi's implementation to allow use SIMD operations for SomeWhat homomorphic encryption scheme and how to use SIMD operations in constructing Fully homomorphic encryption by implementing the re-encryption producer in parallel. The Experimental test was implemented in C++ programming language using NTL library and run on a high workstation machine with six Intel Xeon 2.4 GHz processors and 47 GB of RAM. The result was an improvement over standard Fully Homomorphic encryption scheme by 2.4 times faster, and the ciphertext sizes were reduced by a factor 1/72.

VIII. Fully Homomorphic Encryption Schemes Based on Ring-Learning by Errors.

A. *Efficient Fully Homomorphic Encryption from (Standard)LWE*

Fully homomorphic encryption schemes based on Gentry's theory by ideal lattices does not achieve the goal of effectiveness performance due to suffering from the massive size of keys and ciphertext. A series of new research works has started in the research community by attempt a new trend instead of lattice problems. Brakerski and Vaikuntanathan [12] introduce a (leveled) Fully homomorphic encryption based on the hardness of much more standard "learning by error" (LWE). In their scheme, the bootstrapping was constructed directing instead of squash phase in which this led to shortens the ciphertext and reduces the complexity of decryption procedure.

B. *Fully Homomorphic Encryption without Bootstrapping*

Brakerski, Gentry, et al., [13] show how they improve the [12] scheme to construct a leveled Fully Homomorphic encryption scheme but without Gentry's bootstrapping procedure. The constructed scheme based on computation quasi-linear in the security parameter.

C. *Design and Implementation of a Homomorphic Encryption library (HElib)*

IBM in 2013 [14] [15] has developed and implemented a library based on Brakerski, Gentry, et al. [13] homomorphic encryption scheme. The library implement using C++ programming language and NTL mathematical library. The library design to provide many optimizations to make homomorphic evaluation run faster with focusing on the Smart and Vercauteren [9] in ciphertext optimizations techniques. In 2015, IBM released a new software package library with a bootstrapping procedure [16].

IX. Conclusion.

The study of Homomorphic encryption has proved to the research community that it remains open research problem. From the theoretical perspective currently Fully homomorphic encryption scheme construction is still sophisticated processes for turning SWE to FHE depending on lattice-based or LWE, which led to the fact that there is no algebraically fully homomorphic cryptosystem. Also, almost all the introduced fully homomorphic encryption scheme supports single user setting notwithstanding that most application of homomorphic operation involves multiple data coming from the different user. Underlying demand for devising homomorphic operations is supporting multiuser system. From the practical perspective of the fully homomorphic implementation indispensable requirement for optimizations at an algorithmic level with effective a hardware design using GPU technology and the advantage of multicore computing to enhances parallelism's level forbringing the possibility of real-time implementation.

Reference

[1] R. Rivest, L. Adelman, and M. Dertouzous, "On data banks and privacy homomorphisms," Foundations of secure computation, vol. 4, no. 11, pp. 169-180., 1978.

[2] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126., 1978.

[3] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," ACM, 1982, pp. 365-377.

[4] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in Advances in Cryptology, Springer, 1984, pp. 10-18.

[5] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residue Classes," in In Advances in Cryptology—EUROCRYPT'99, Springer, 1999, pp. 223-238.

[6] D. Boneh, E.-J. Goh and K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts. In Theory of Cryptography," in Theory of Cryptography, Springer Berlin Heidelberg., 2005, pp. 325-341.

[7] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in STOC, vol. 9, 2009, pp. 169-178.

[8] M. V. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," Proceedings of Advances in Cryptology,EUROCRYPT'10, pp. 24-43, 2010.

[9] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in Public Key Cryptography-PKC 2010, vol. 6056, Springer, 2010, pp. 420-443.

[10] C. Gentry and S. Halevi, "Implementing Gentry's fully homomorphic encryption," in Advances in Cryptology-EUROCRYPT 2011, Springer, 2011, pp. 129-148.

[11] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," Cryptology ePrint Archive, Report 2011/133, 2011.

[12] Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (Standard) LWE," FOCS, 2011.

[13] Z. Brakerski, C. Gentry and V. Vaikuntanathan, "Fully Homomorphic Encryption without Bootstrapping," 2011.

[14] S. Halevi and V. Shoup, "Design and implementation of a homomorphic encryption library," IBM Research, 2013.

[15] S. Halevi and V. Shoup, "Algorithms in HELib," in Advances in Cryptology, Springer, 2014, pp. 554-571.

[16] S. Halevi and V. Shoup, "Bootstrapping for HELib," in Advances in Cryptology, Springer, 2015, pp. 641-670.