STM JOURNALS

Journal of
Telecommunication, Switching Systems and Networks

ISSN: 2454-6372
Volume 12, Issue 1, 2025
January–April
DOI (Journal): 10.37591/JoTSSN

https://journals.stmjournals.com/jotssn

Review JoTSSN

A Practical Approach for Privacy Preserving in Cloud Computing Using Fully Homomorphic Encryption Schemes BFV and CKKS

Sunita Godara[1,*], Simran Choudhary[2]

Abstract

Cloud computing is a new type of computing architecture whereby data may be accessed over the Internet along with other services linked to its scalable data centers in the cloud. The risk associated with computing is increased since it provides essential services that are typically provided to any third party, making it more difficult to enable data security, privacy, confidentiality, integrity, and authentication. To reduce security risks, most users choose to save their data in the cloud in encrypted format. Nevertheless, the cloud must first decrypt the data before it can be used for any server-side operations. This operation may result in difficult problems like cloud storage secrecy and privacy of sensitive data. In this study homomorphic encryption (HE) is presented, which also addresses issues with data privacy and confidentiality stored on the cloud. Homomorphic encryption (HE) is a type of encryption technology that allows users to perform computations on the cipher text itself. This produces an unoriginal or encrypted output, which, when decrypted, resembles the outcome of operations performed on the plain text. There are two main types of homomorphic encryption: fully homomorphic encryption (FHE) and partial homomorphic encryption (PHE). Since FHE benefits from both additive and multiplicative homomorphism, it is thought to be more secure and effective when it comes to third-party calculations. In this study focus is on giving a short overview of FHE, its types, and practical implementation of BFV and CKKS along with their performance analysis.

Keywords: Cloud computing, FHE, data security, privacy, BFV, CKKS

INTRODUCTION

Nowadays, the majority of people on the planet own multiple digital devices with constrained local storage. Services that make it simple for users to save and retrieve personal files are therefore becoming more and more necessary. The process of converting paper-based data to a digital file is underway. Certain data, however, must remain confidential and should not be shared with the general public. A number of cloud services enable the safe uploading of encrypted private data; but, once the data is posted, the cloud service frequently has the ability to decode it. It achieves this because, after calculations are made on the data, the majority of encryption techniques no longer yield the accurate decrypted value. This is not the situation with homomorphic encryption. Information can be encoded using encryption, which aims to maintain confidentiality by allowing access to only those who

***Author for Correspondence**
Sunita Godara
E-mail: Smart.sunita83@gmail.com

[1]Ph.D. Research Scholar, Department of Computer Science and Engineering, Mugneeram Bangur Memorial University, Jodhpur, Rajasthan, India
[2]Assistant Professor, Department of Computer Science and Engineering, Mugneeram Bangur Memorial University, Jodhpur, Rajasthan, India

Received Date: January 15, 2025
Accepted Date: January 20, 2025
Published Date: January 26, 2025

Citation: Sunita Godara, Simran Choudhary. A Practical Approach for Privacy Preserving in Cloud Computing Using Fully Homomorphic Encryption Schemes BFV and CKKS. Journal of Telecommunication, Switching Systems and Networks. 2025; 12(1): 1–8p.

are permitted. Different encryption systems are available, and they can be either symmetric or asymmetric. When a secure channel has already been established, the symmetric setting, in which the same key is used for both encryption and decryption, is frequently employed. Each party in the asymmetric setup has a public key that is used for encryption and a private key that is used for decryption. In order for only the owner of the secret key to decipher a communication encrypted with the matching public key, the public key and private key are communicated between parties. Soon after public key cryptography was discovered in 1978, Rivest *et al*. were the first to notice that encrypted data might be meaningfully altered in addition to being stored and retrieved [1]. HE is a special type of encryption with many uses that permits actions on encrypted data. The result is a ciphertext that, upon decryption, corresponds to the output of the intended computation carried out on the plaintext. Homomorphically encrypted secured data at cloud is shown in Figure 1. They posed the following query: Is it possible to do random calculations on ciphertext material without ever having to decrypt it? Which requests the capability of doing calculations on encrypted data without having the data "visible"? Users can use this feature to encrypt their sensitive information locally and transfer it to the cloud service along with the computations.

Properties of Homomorphic Encryption

Homomorphic encryption mainly consists of three properties, which are as follows: Let $m1$ and $m2$ be plaintext and $c1$ and $c2$ are ciphertext.

Additive Homomorphic Encryption

Additive homomorphism is the property of an encryption function (Encrypt) that is achieved only when the encrypted result of addition performed on plaintext equals the result of addition calculated on ciphertext. As seen in Eq. (1), E (m1+m2), which is the encryption of sum of m1 and m2, may be computed from ciphertext of m1 and ciphertext of m2 without the need to disclose these two messages [2].

$$E(m1)+E(m2)=E(m1+m2) \quad (1)$$

$$E(m1+m2)=c1+c2 \; or$$

$$E(m1)+E(m2)=c1+c2$$

$$Decrypt\ (c1+c2)=m1+m2.$$

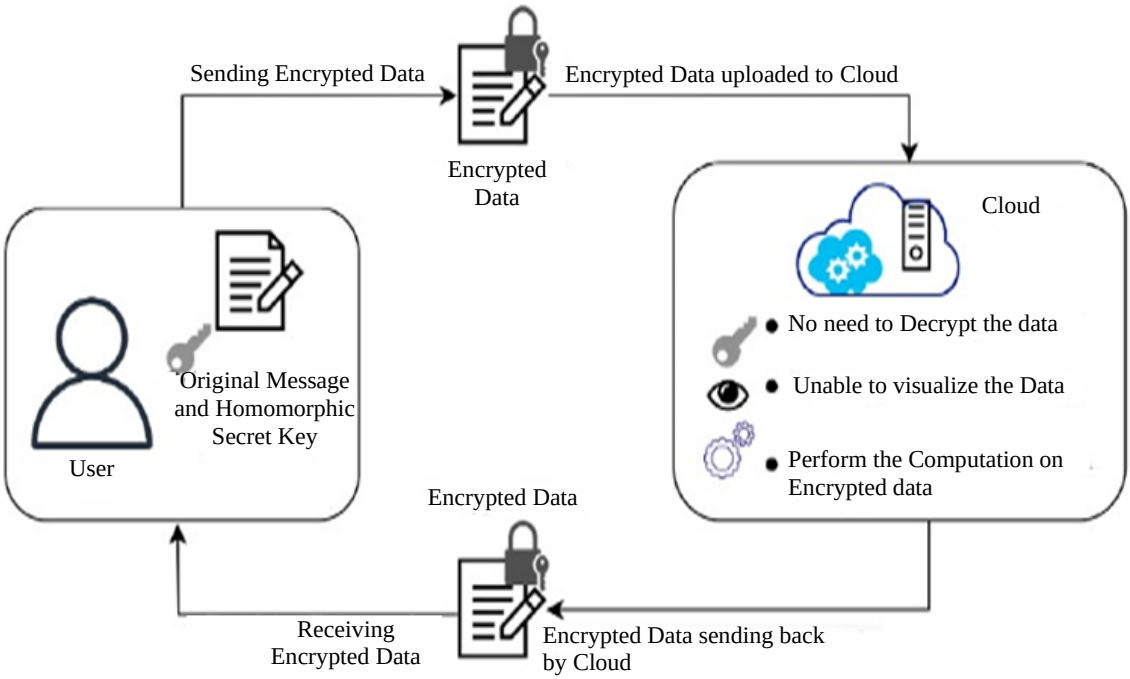

Figure 1. Homomorphically encrypted secured data at cloud [3].

Multiplicative Homomorphic Encryption

Multiplicative homomorphism, or the encryption function Encrypt, is used when the result of a product operation calculated on ciphertext is the same as the encrypted result of multiplication performed on original text. Eq. (2) illustrates how E (m1×m2), which is the encryption of product of m1 and m2, can be calculated using E(m1) and E(m2) without the need for m1 and m2.

$$E(m1)\times E(m2)=E(m1\times m2) \quad (2)$$
$$E(m1\times m2)=c1\times c2 \; or$$
$$E(m1)\times E(m2)=c1\times c2$$
$$Decrypt\ (c1\times c2)=m1\times m2.$$

Mixed Homomorphic Encryption

The term refers to the result of either multiplication or addition operations calculated on encrypted text, which is comparable to the encrypted results of multiplication or addition operations performed on original text, regardless of order or depth. For example, E (m1×m2) and E (m1+m2) can be computed from E(m1) and E(m2) without requiring the provision of m1 and m2, as stated in Eq. (3).

$$E(m1)\times E(m2)=E(m1\times m2) \text{ and}$$
$$E(m1)+E(m2)=E(m1+m2) \quad (3)$$

The HE schemes also provides a one-to-many relationship that means one plaintext can be encrypted in many ciphertexts, as shown in Eq. (4).

$$Encrypt1\ (m1)\neq Encrypt2\ (m1) \text{ but}$$
$$Decrypt\ (Encrypt1\ (m1)) = Decrypt\ (Encrypt2\ (m1)) \quad (4)$$

RELATED WORK

The literature review is a vital element of any research work that helps to have a complete understanding of current research going on in a specific field. It helps to learn about the previous research that had been done already and what is still unknown. Survey work is shown in Figure 2. Analysis and evaluation of literature help to solve a particular set of problems. The complete survey work has been divided into three different phases as shown in Figure 2.

History of Homomorphic Encryption

The concept of privacy homomorphism, which refers to the ability to perform computation over encrypted text, was initially presented by Rivest, Shamir, and Adleman in RSA algorithm cryptographic method. This algorithm later evolved into the concept known as Homomorphic Encryption. HE enables computing over encrypted data. A partially multiplicative homomorphic technique is the RSA algorithm. The difficulty of large integer number factorization determines the security feature of the RSA. The algorithms deterministic concept makes it particularly insecure since, as is typical, the generated ciphertext can be deduced from its plaintext; the corresponding ciphertext yields the equivalent plaintext.

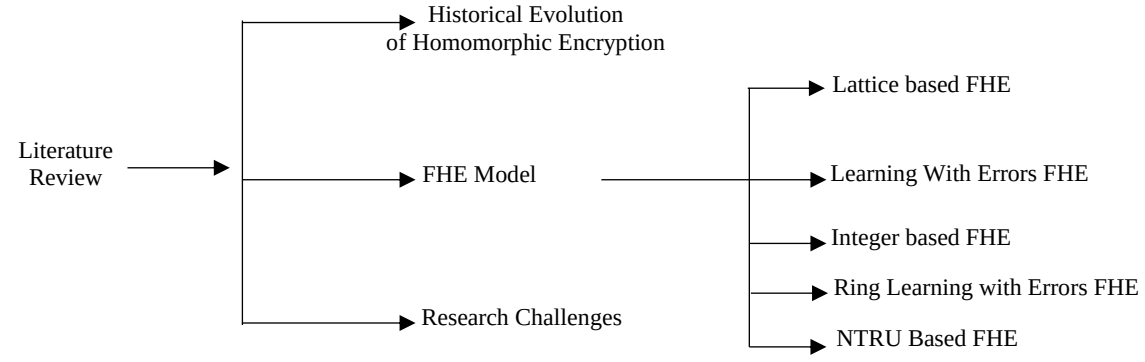

Figure 2. Survey work.

Additionally, there is no practical use for RSA's multiplicative property, and any modifications made to the basic RSA calculation would typically cause it to lose its homomorphic property [1]. In 2018, Aganya and Sharma also developed the symmetric Homomorphic Encryption scheme for big size plaintext, concept designed on polynomial rings using integers [4]. The plaintext is a string of bits instead of a single bit that limits the size of the plaintext for easy handling of big files having data. The scheme was more secure and faster but the ciphertext increased with polynomial size. The author used pseudo-random numbers to decrease the public key size. In 2020, Aubry *et al.* proposed an improved multiplicative depth of Boolean circuits, as performance and ciphertext size depend on it [5]. The author used the new circuit rewriting operators (cone rewrite operators) and the heuristic method, based on the priority function helps to improve the circuit depth [6]. The implementation of the BGV scheme with the proposed concept improved the scheme in terms of smaller ciphertext, better computational performances, and the overall execution time. The proposed scheme can also be extended for arithmetic numbers not even for binary numbers. In 2023, the research by Mittal *et al*. applied fully homomorphic encryption (FHE) methods in a comparison of privacy protection in cloud applications [3]. Many approaches are examined to reach a conclusion regarding their performances. The experimental study demonstrates the efficacy of the different frames in terms of security performance and accuracy when creating a cloud application for safe analytics. Mahato *et al*. discussed the security concerns affecting cloud computing, evaluated the effectiveness of conventional and cutting-edge security measures, and concentrated on the difficulties in putting the technology into practice [7]. The HE Scheme permits computation in the encrypted state without requiring the data to be decrypted to its original form, in contrast to traditional encryption schemes like RSA, ElGamal, and Pailler that prohibit operations on the encrypted data. As a result, both security and the requirement to disclose the secret key are guaranteed.

In 2016, Dasgupta and Pal proposed a symmetric FHE model, called SWHE due to increase in noise after certain calculations that made FHE by a refresh procedure [8]. It was based on polynomial over integer rings. Large ciphertexts were updated after a certain number of homomorphic processes to ensure proper decoding. In addition, polynomial addition required a high computational cost. In their study, Agrawal *et al*. provided a detailed description of the CKKS method [9]. Because it performs real number operations, the CKKS scheme is the one that is recommended for machine learning applications that protect privacy. Although the CKKS method allows homomorphic calculations using real numbers, it was not possible to use fundamental optimization techniques based on the Number Theoretic Transform (NTT) and the Residue Number System (RNS) decomposition in its implementation. As a result, Cheon *et al*. suggested a RNS version of the CKKS scheme, which is best suited for standard computer system [10].

Classification of Homomorphic Encryption

Depending on the types and count of computations performed over encrypted text, homomorphic methods can be of three types. These are Partial, Somewhat and Fully Homomorphic Encryption.

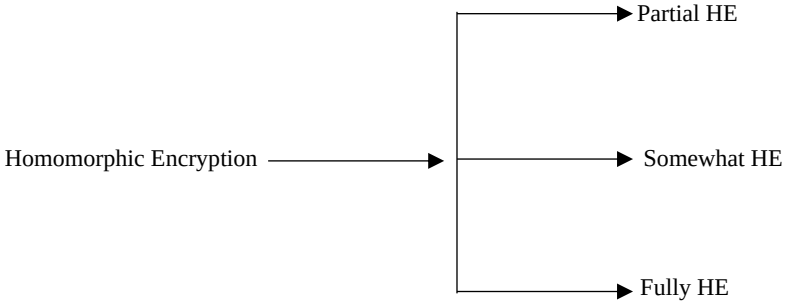

Figure 3. Homomorphic encryption [11].

Partially Homomorphic Encryption (PHE)

In Partial Homomorphic Encryption, either one of the operation addition or multiplication exists. It is an encryption scheme, in which only arbitrary additive or multiplicative operations can be performed on the ciphertext [12]. Due to this, it is only suitable for applications whose algorithm requires only one operation (addition or multiplication). Homomorphic Encryption is shown in Figure 3. Some popular partial homomorphic encryption schemes are RSA [1], Goldwasser and Micali, ElGamal [13], Benaloh, and Paillier. RSA and ElGamal encryption schemes are multiplicative whereas Paillier, and Benaloh encryption methods based on additive homomorphic technique.

Somewhat Homomorphic Encryption (SWHE)

A Somewhat Leveled Homomorphic Encryption scheme allows for computing the addition and multiplication operations either individually or in combinations for specific depth of circuits i.e., limited computations on the ciphertext. SWHE is having very less use in real life because there is a limit on the number of operations computed on ciphertext as encrypted text size increases with every homomorphic operation. Some popular SWHE schemes are Yao's, SYY's, BGN [14], Brakerski's encryption scheme [11], and Polly cracker. Only Boneh, Goh, and Nissim's encryption scheme allows a random addition and one multiplication, with no increase in ciphertext size.

Fully Homomorphic Encryption (FHE)

Unlike PHE and SWHE, there is no limit on the count of operations on an encrypted text. Fully Homomorphic Encryption includes the computation of an arbitrary number of addition and multiplication operations on the ciphertext. In 2009, the first FHE scheme was developed by Gentry [13]. After this some variants were developed like one was based on lattice [7], another integer method [4], and learning error method.

PRACTICAL APPROACH FOR COMPARISON OF BFV AND CKKS

Practical work for fully homomorphic encryption systems, namely: BFV (Brakerski-Fon-Vercauteren) and CKKS (Cheon-Kim Kim-Song) are demonstrated in this research work [14, 15].

METHOD

Python is a versatile programming language that is now at the top of the field for machine learning and data science since it allows for quick prototypes. First, we select two schemes BFV and CKKS and one Python library to start the FHE implementation process. These libraries selections are based on their language's readability and simplicity, which makes it perfect for prototyping [16].

RESULT AND DISCUSSION

In this research work we provide a comparison of addition and multiplication operations using the BFV and CKKS methods. The practical work was executed on E5LQ353V ASUS laptop running Windows 10 64-bit, equipped with an Intel(R) Core (TM) i5 1035G1, CPU 1.00 GHz, 1.19 GHz processor, 8 GB RAM. We implement both CKKS and BFV encryption schemes using the Visual Studio Code Python IDE [17]. A comparative study of the differences in executing times is performed by applying each method to two homomorphic operations: addition and multiplication. We provide addition and multiplication runtimes for the CKKs and BFV schemes. The systems make use of three primary HE parameters:

- n is polynomial degree and it is considered as the number of slots of the input plaintext vectors.
- p is plaintext modulus, and it is considered as the modulus of the plaintext space in BFV method.
- q is size of plaintext or known as ciphertext modulus): before decryption fails how much noise can be accumulated, q can determine.

Table 1 shows the BFV and CKKS vector addition using two vectors and the time is recorded in millisecond (ms) for each of the encryption and decryption operations [18]. While keeping p and q

constant when we increase the ciphertext dimension n, there is an increase in the execution time for each operation for both schemes. The following observations are done:

- When the polynomial degree n increases, the addition operation in the CKKS method becomes faster compared to the BFV method but slower within the CKKS method itself.

Next execution time for BFV and CKKS method for Multiplication is calculated with the same parameters for encryption decryption operations. The observation of results has been shown in Table 2.

From Figure 4 it is observed that the performance of CKKS scheme for ciphertext addition operation is getting improved as the value of n, polynomial degree reaches 16384.

Table 1. BFV and CKKS addition.

Scheme	Polynomial Degree (n)	Addition Encryption Time (ms)	Addition Decryption time (ms)	Total Time (sec)
BFV	4096	0.013619	0.001029	14.648
	8192	0.016636	0.002000	18.636
	16384	0.012528	0.016357	28.885
CKKS	4096	0.000001	0.001183	1.184
	8192	0.009647	0.001817	11.464
	16384	0.004775	0.020341	25.116

Table 2. BFV and CKKS multiplication.

Scheme	Polynomial Degree (n)	Multiplication Encryption Time (ms)	Multiplication Decryption time (ms)	Total Time (sec)
BFV	4096	0.001999	0.001596	3.595
	8192	0.004721	0.002961	7.682
	16384	0.025308	0.017342	42.65
CKKS	4096	0.001035	0.001002	2.037
	8192	0.005016	0.005588	10.604
	16384	0.008334	0.024844	33.178

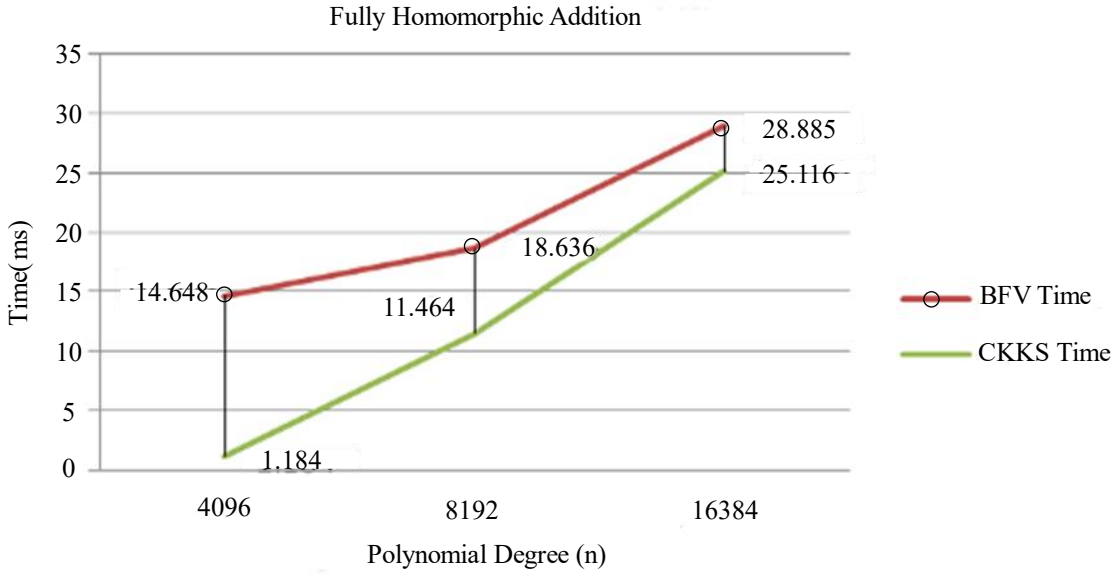

Figure 4. FHE Addition for BFV and CKKS.

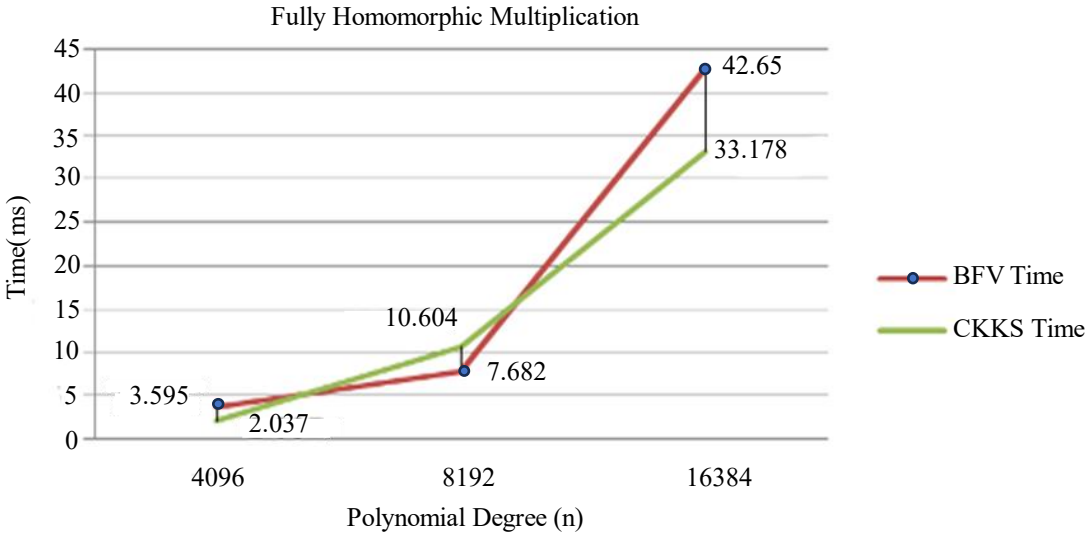

Figure 5. FHE Multiplication for BFV and CKKS.

From Figure 5 it is observed that for FHE multiplication operation, BFV performs better for n<16384 and for n up to 16384, CKKS performs better.

CONCLUSION AND FUTURE WORK

There is a lot of focus on strengthening security on several levels in the domain of third-party computations and outsourced computing, including infrastructure, network, host, application, and data. One important function of the Homomorphic Encryption approach is to make calculations easy on encrypted data inside the cloud. This suggests that the data can be processed without having to be converted to plain text. The FHE technique was introduced by Craig Gentry in 2010, which was a significant advancement in the realm of cryptography. The main objective of the research work is to provide detailed study of fully homomorphic encryption schemes and its types. It also provides a practical approach to implement BFV and CKKS as an FHE scheme and a comparative study of these two schemes with addition and multiplication operations on different vector sizes with their execution times and finally came to the conclusion that CKKS performs better for large vector size.

REFERENCES

1. Rivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. In: Foundations of Secure Computation. New York: Academic Press; 1978 Oct 16; 169–80.
2. Kodanda Ramaiah GN, Harkude Savita A. Elliptic Curve Cryptography Based Homomorphic End-to-End Encryption Security in Cloud Computing. Mathematical Statistician and Engineering Applications. 2022; 71(3s): 64–75. https://doi.org/10.17762/msea.v71i3s.8
3. Mittal Sonam, Ramkumar KR. Research perspectives on fully homomorphic encryption models for cloud sector. J Comput Secur. 2021; 29(2): 135–160.
4. Aganya K, Sharma I. Symmetric Fully Homomorphic Encryption Scheme with Polynomials Operations. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India. 2018; 1954–1957. doi: 10.1109/ICECA.2018.8474729.
5. Pascal Aubry, Sergiu Carpov, Renaud Sirdey. Faster Homomorphic Encryption is not Enough: Improved Heuristic for Multiplicative Depth Minimization of Boolean Circuits. In Topics in Cryptology (CT-RSA 2020): The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings. Berlin, Heidelberg: Springer-Verlag; 2020; 345–363. https://doi.org/10.1007/978-3-030-40186-3_15
6. Biksham V, Vasumathi D. A lightweight fully homomorphic encryption scheme for cloud security. Int J Inf Comput Secur. 2020; 13(3–4): 357–371.

7. Mahato Ganesh Kumar, Swarnendu Kumar Chakraborty. A comparative review on homomorphic encryption for cloud security. IETE J Res. 2023; 69(8): 5124–5133. 10.1080/03772063.2021. 1965918.
8. Dasgupta S, Pal SK. Design of a Polynomial Ring Based Symmetric Homomorphic Encryption Scheme. Perspect Sci. 2016; 8(1): 692–695. doi: 10.1016/j.pisc.2016.06.061.
9. Agrawal Rashmi, Ajay Joshi. On architecting fully homomorphic encryption-based computing systems. Cham: Springer; 2023. DOI: 10.1007/978-3-031-31754-5.
10. Archer D, Chen L, Cheon JH, Gilad-Bachrach R, Hallman RA, Huang Z, Jiang X, Kumaresan R, Malin BA, Sofia H, et al. Applications of homomorphic encryption. Homomorphic Encryption. org, Redmond WA: Tech Rep. 2017. https://www.researchgate.net/publication/320976976_ APPLICATIONS_OF_HOMOMORPHIC_ENCRYPTION
11. Acar A, Aksu H, Uluagac AS, Conti M. A survey on homomorphic encryption schemes: Theory and implementation. ACM Comput Surv. 2018; 51(4): 1–35.
12. Poteya Manish M, Dhoteb CA, Sharmac Deepak H. Homomorphic Encryption for Security of Cloud Data, Computing and Virtualization. 7th International Conference on Communication, Procedia Comput Sci. 2016; 79: 175–181.
13. Gentry C, Boneh D. A fully homomorphic encryption scheme. Stanford University ProQuest Dissertations & Theses. Stanford: Stanford university; 2009.
14. Smart NP, Vercauteren F. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen PQ, Pointcheval D, editors. Public Key Cryptography – PKC 2010. PKC 2010. Lecture Notes in Computer Science. Vol. 6056. Berlin, Heidelberg: Springer; 2010. https://doi.org/10.1007/978-3-642-13013-7_25.
15. Brakerski Z. Fundamentals of fully homomorphic encryption-a survey. In Electronic Colloquium on Computational Complexity (ECCC). 2018; 25: 125.
16. Parmar PV, Padhar SB, Patel SN, Bhatt NI, Jhaveri RH. Survey of various homomorphic encryption algorithms and schemes. Int J Comput Appl. 2014; 91(8): 26–32.
17. Alloghani M, Alani MM, Al-Jumeily D, Baker T, Mustafina J, Hussain A, Aljaaf AJ. A systematic review on the status and progress of homomorphic encryption technologies. J Inf Secur Appl. 2019; 48: 102362.
18. Atayero AA, Feyisetan O. Security issues in cloud computing: The potentials of homomorphic encryption. Journal of Emerging Trends in Computing and Information Sciences. 2011; 2(10): 546–552.

