ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Fully Homomorphic Encryption Scheme for Securing Cloud Data

Mohammed Anwar Mohammed, Tara Nawzad Ahmad Al Attar

Department of Computer Science, College of Science, University of Sulaimani, Sulaimaniyah, Iraq

ABSTRACT

One of the pioneer and important fields in the computer science area is cloud computing. The data within cloud computing are usually transformed to it from local storage; therefore, the security of this data is an important issue. To solve this data security issue, it is important that cloud service providers (CSPs) store encrypted versions of user data. Before transmitting data to the cloud provider, it was encrypted using traditional encryption schemes. Nevertheless, for these schemes, the private key must be provided to the server to be used for the decryption on the other side before any calculations, yielding a security risk and issue for the cloud data. Homomorphic encryption provides a capable solution to this issue since it enables calculations on encrypted data with no need to be decrypted and the private encryption key is not compromised. A new fully homomorphic encryption scheme to protect cloud data is proposed in this paper, it is called NAZUZ. The NAZUZ scheme is based on prime modular operations and encrypts messages by operating on each character without converting them to binary. NAZUZ security relies on the difficulty of factoring large integer numbers and introduces noise complexity to the plaintext through the number of CSP users.

Index Terms: Cryptography, Cloud, Data security, Data privacy, Information security

1. INTRODUCTION

The significance of cloud computing raised due to the fast and rapid progress of computer networks as well as the spread of big data. Cloud computing provides means to store and process huge amounts of data [1]. It provides users with suitable access to remote storage and computational resources that are flexible and on-demand. There is little control over the data in the cloud environment; therefore, the security, confidentiality, and integrity of cloud computing became an issue. The presentation of data leaks and the protection of personal privacy is crucial for both individuals and enterprises that are planning to move their data to cloud storage [2]. To protect the privacy of cloud data, data protection is considered a vital mechanism. To protect the data, either traditional encryption methods and techniques or homomorphic encryption methods are used. For the traditional methods, no process or calculations on the cloud-encrypted data can be performed before decrypting it, and this will result in putting the data at risk and compromising its security. On the other hand, decrypting the data is not required in homomorphic encryption and any operation required can be directly performed on the encrypted version of the data, plus the fact that the results were the same when performing the same calculation on both the original dataset as well as on the encrypted data. Homomorphic encryption has two general categories, partial homomorphic encryption (PHE) and fully homomorphic encryption (FHE). Either addition or multiplication is permitted on the encrypted data for PHE [3]. While on the hand, FHE allows a random number of additions and multiplications on the encrypted data, so it is more comprehensive [4]. The organization of

Access this article online

DOI: 10.21928/uhdjst.v7n2y2023.pp40-49

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: Tara Nawzad Ahmad Al Attar, Department of Computer Science, College of Science, University of Sulaimani, Sulaimaniyah, Iraq . E-mail: tara.ahmad@univsul.edu.iq

Received: 19-07-2023 Accepted: 23-09-2023 Published: 10-10-2023

the remaining sections of this paper is as follows: Section II presents the motivation behind the proposed work. Next, in Section III, a comprehensive review of the relevant literature is presented. A brief explanation of homomorphic encryption is illustrated in Section IV. In Section V, the proposed algorithm named NAZUZ is introduced. All the results and analysis of the proposed work are presented in Section VI including several case studies. A comparative analysis of the proposed algorithm is provided in Section VII. Then, Section VIII discusses the limitations of the work, and finally, the conclusion of the paper is in Section IX.

2. MOTIVATION OF THE WORK

The need of accessing private information anytime and anywhere increases. Individuals and enterprises deploy their private information onto cloud storage, and this requires addressing of extra amount of risks, which makes it challenging to maintain the security outlines such as data security, confidentiality, integrity, authentication, and privacy. For instance, in 2011, the PlayStation Network was hacked causing the leak of information for millions of user accounts such as passwords, physical addresses, credit card information, and other personal information. Later, Sony announced that they could have taken special protection by encrypting the data on their network [5]. Therefore, to protect users' privacy, it is required from cloud service providers (CSPs) to save an encrypted version of the user's data. There are several techniques that perform encryption on users' data. However, as the data resides in cloud storage, it is required to be encrypted before performing any operation on the data. Privacy and confidentiality issues to the stored data might be caused. Homomorphic encryption allows performing computations on the encrypted data without decrypting it, and the results of the computations are the same as they were processed on the corresponding plaintext data. Thus, homomorphic encryption solves the problems of confidentiality and privacy of the stored data inside the cloud.

3. RELATED LITERATURE

The first homomorphic encryption was suggested by Rivest *et al.* [6] which was partially homomorphic encryption (PHA). Then RSA provided the multiplicative homomorphism [7]. Afterward, Yao, Goldwasser and Micali, Elgamal, Paillier [8]-[11] presented a PHA scheme. Subsequently, a FHE scheme suggested by Gentry [12] allows the calculation of any number of addition and multiplication and hence computing arbitrary functions of encrypted data. However, the scheme was based on somewhat homomorphic encryption (SWHE) which increases the length and noise of cipher text when calculation performs on the cipher text as shown in Fig. 1. Consequently, the authors of Dijk *et al.* [13] have introduced a FHE scheme that uses elementary modular arithmetic and use Gentry's techniques to convert a somewhat homomorphic cryptosystem to a FHE scheme. Afterward, in Smart and Vercauteren [14], an improved version of the Smart-Vercauteren encryption scheme was proposed, which allowed several times a decrease in the cipher text and key lengths. In 2013, HElib software package was released by IBM which implements homomorphic encryption, which made many optimizations to make homomorphic evaluation run faster, focusing mostly on the effective use of the Smart-Vercauteren cipher text packing techniques [15]. In addition, in Xiao *et al.* [16], the authors have proposed homomorphic encryption in which the security of a homomorphism depends on the hardness of large integer factorization using symmetric keys. The authors also show how key size and computational time are reduced enough for practical deployment. Then, several authors worked on homomorphic encryption and also examined it in cloud computing systems. In Maha and Said [17] the authors have studied different homomorphic encryption cryptosystems such as El-Gamal, Paillier, RSA, and Gentry in a cloud computing environment. In addition, the application of an algebraic homomorphic encryption mechanism was introduced in Reem and Khaled [18], this mechanism was aimed at better security and was based on Fermat's little theorem on cloud computing. In addition, in Hayward and Chiangb [19], the authors have proposed parallel processing for Gentry's encryption and were tested in a private cloud computing domain. Furthermore, the authors of Frederik *et al.* [20] have introduced the simplified and structured wide definitions in the homomorphic encryption discipline and questioned whether presently existing applications need homomorphic encryption supposed as an explainable solution to their problems, both in theoretical, along with practical approaches. In Hamad and Sagheer [21], the authors have implemented FHE over integers named SAM and show how the algorithm meets both additive and multiplicative homomorphism.

Recently, Xiao *et al.* [22] have proposed a privacy data protection method for industrial field equipment based on FHE scheme. In addition, the authors of Hashim and Benaissa [23] proposed their FHE accelerator on a hardware platform to speed up the encryption process so that practical encryption time could be achieved. They argued an optimization on digital signal processing utilization on modern field programmable gate array (FPGA), Virtex7.

Moreover, an efficient private database query proposal has been argued by Tan *et al.* [24], the protocol supports compound conditions with equality and order comparisons.

4. HOMOMORPHIC ENCRYPTION (HE)

This section will illustrate the basics of homomorphic encryption theory, then explain the different types of (HE). Homomorphic encryption can be categorized into three different types FHE, SWHE, and PHE. An encryption scheme is called homomorphic over an operation "+" if it supports the following equation:

$$E(Msg_1) + E(Msg_2) = E(Msg_1 + Msg_2), \quad \forall Msg_1, Msg_2 \in M \tag{1}$$

4.1. PHE

In PHE, either addition or multiplication is allowed regardless of the number of times. There are several useful examples of PHE such as RSA, Goldwasser-Micali, El-Gamal, Benaloh, and Paillier. While SWHE allows some types of operations with a limited number of times, some examples of SWHE are BNG by Dan *et al.* (Boneh-Goh- Nissim) [25] and Polly Cracker introduced by Fellows and Koblitz [26] in 1994. Nevertheless, FHE performs both addition and multiplication at the same time, and it can compute any operations, examples of FHE are Ideal Lattice-based FHE schemes [27], FHE schemes Over Integers [28], LWE-based FHE schemes [29], NTRU-like FHE schemes [30], Gen10 [31], and simple FHE scheme [32].

5. THE PROPOSED ALGORITHM (NAZUZ)

We have named our proposed algorithm NAZUZ; it is the nickname of the first author's mother who passed away during performing his research study. Unlike DGHV and SDC schemes, our proposed scheme instead of converting each plaintext character into binary values (0, 1). NAZUZ converts each plaintext character into ASCII code and passes it to the encryption algorithm as,

$$Enc = Msg + L(rK_s + Iter) \tag{2}$$

Where Enc is the ciphertext, Msg is the message $Msg \in [0, L-1]$, r is the noise, L is a big prime integer, K_s is the secret key generated by the key generation algorithm, and $Iter$ is a counter added as extra noise to the plaintext, these all are resulting in one ciphertext for each character in the plaintext, the rest of this section illustrates NAZUZ algorithm in details.

Secret key generation Ks:
Generate L, where L a big prime number is, then generate $r \in Z_n$, where r is the noise and ($r < L/4$ or $2r < L/2$), then calculate the secret key K_s as follows:

Choose multiple n prime numbers $p_1, p_2, p_3, \ldots p_n$ as secret keys, then calculate n as $n = p_1 \times p_2 \times p_3 \ldots \times p_n$, calculate M as $M = (p_1+1)(p_2+1)\ldots(p_n+1)$, then calculate $N_{sum} = \sum_{i=1}^{m} F_i$, where F_i = set of prime numbers up to M, and then calculate the average value of the sum of all prime numbers as $N_{avg\_sum} = \frac{N_{sum}}{M}$, then choose a random number R that satisfies $\gcd(R, N_{avg\_sum}) = 1$, then select U_s as it is the number of existing users of the cloud system $\{U_{s1}, U_{s2}, \ldots, U_{sn}\}$, where $U_s \geq 1$, calculate $\theta(n) = (p_1-1)\ldots(p_1-1)$, and calculate $Q = U_s \times (\theta(n) \bmod N_{sum})$ and finally calculate K_s as Flowchart shown in Fig. 2:

$$K_s = (R \times Q) \bmod 256 \tag{3}$$

Encryption Flowchart shown in Fig. 3:

$$Enc = Msg + L(rK_s + Iter) \tag{4}$$

Decryption flowchart shown in Fig. 4:

$$Msg = Enc \bmod L \tag{5}$$

Homomorphic evaluation:
Suppose there are two ciphertexts Enc_1 and Enc_2 as:

$$Enc_1 = Msg_1 + L(r_1 K_s + Iter)$$

$$Enc_2 = Msg_2 + L(r_2 K_s + Iter)$$

$Enc \bmod L = Msg$, where $Msg < L$, otherwise we must take ($Msg \bmod L$)

Additive Homomorphism:
First, we illustrate the sum of two ciphertexts Enc_1 and Enc_2 denoted by ($Enc^+ = Enc_1 + Enc_2$)

$$Enc^+ = Enc_1 + Enc_2 = (Msg_1 + Msg_2) + L(r_1 K_s + Iter) + L(r_2 K_s + Iter)$$

But

$$L(r_1 K_s + Iter) + L(r_2 K_s + Iter) = LK_s[(r_1 + r_2) + 2Iter] = 0$$

Then

$$Msg^{+} = \left(Enc_1 + Enc_2\right) \mod L = Msg_1 + Msg_2$$

Multiplicative Homomorphism:
To begin with, we illustrate the multiple of two ciphertexts Enc_1 and Enc_2 denoted by ($Enc^* = Enc_1 * Enc_2$)

$$Enc^{*} = [Msg_1 + L\left(r_1 K_s + Iter\right)] * [Msg_2 + L \left(r_2 K_s + Iter\right)]$$

$$Enc^{*} = [Msg_1 * Msg_2 + Msg_1 * L\left(r_2 K_s + Iter\right)] + L\left(r_1 K_s + Iter\right) * Msg_2 + L\left(r_1 K_s + Iter\right) * L\left(r_2 K_s + Iter\right)]$$

Since

$$L * [Msg_1 * \left(r_2 K_s + Iter\right) + \left(r_1 K_s + Iter\right) * Msg_2 + L * \left(r_1 K_s + Iter\right) * \left(r_2 K_s + Iter\right) \mod L] = 0$$ So that

$$Enc^{*} = Msg_1 \times Msg_2 + 0 = Msg_1 \times Msg_2$$

6. RESULTS AND ANALYSIS

To test the proposed algorithm NAZUZ, we have implemented a simulation using Java programming language and tested it on a computer with 16 GB RAM, an Intel Core i7 processor, and the Windows 10 64-bit operating system. In the following, various case studies will be presented to demonstrate the generation of the secret key and its corresponding values and to show how these values are used for encrypting and decrypting the plaintext value.

Case Study No.1
First, generate K_s as p_1=3, p_2=5, and p_1=7, assume U_s=10, $\theta(n)$=2×4×6=48, n=3×5×7=105, M=(3+1)(5+1)(7+1)=192, N_{sum}=102001, N_{avg_sum}=531 Q=10×(48mod102001)=480, R=5, K_s=(5×48)mod256=96, then choose a prime number L=457679, random numbers r_1=102235482 and r_2=782542926, assume $Iter$=3, and the messages are Msg_1=97 and Msg_2=98, then calculate the ciphertexts Enc_1 and Enc_2.

$$Enc_1 = Msg_1 + L\left(r_1 K_s + Iter\right)$$
$$= 97 + 457679 * (102235482 \times 96 + 3)$$
$$= 4491939185335822$$

$$Enc_2 = Msg_2 + L(r_2 K_s + Iter)$$
$$= 98 + 457679 * \left(782542926 \times 96 + 3\right)$$
$$= 34382732528933519$$

Proof of Additive Homomorphism:

$$Enc^{+} = Enc_1 + Enc_2 = 4491939185335822$$
$$+34382732528933519 = 3887467171426934$$

$$Msg^{+} = Enc^{+} \mod L =$$
$$38874671714269341 \mod 457679 = 195$$

$$Msg^{+} = Msg_1 + Msg_2 = 97 + 98 = 195$$

Proof of Multiplicative Homomorphism:

$$Enc^{*} = Enc_1 \times Enc_2 = 4491939185335822$$
$$*34382732528933519 =$$
$$154445143545637098259373527217 62e + 32$$

$$Msg^{*} = Enc^{*} \mod L =$$
$$154445143545637098259373527217 62e + 32 \mod$$
$$457679 = 9506$$

$$Msg^{*} = Msg_1 \times Msg_2 = 97 \times 98 = 9506$$

Case Study No.2
This time we have tested NAZUZ on a plaintext file. Choose a prime number L=457679, a random number r=102235482, which is generated as illustrated in case study number one, and a text file that contains the message Msg = Hello This is my text to be: -->Encrypted "HELLOOOOW WORLD" please keep my file in a safe location. After encrypting the plaintext file, the ciphertext file contains:

4491939183962772	4491939184420471
4491939184878151	4491939185335840
4491939185793436	4491939186251188
4491939186708877	4491939187166473
4491939187624229	4491939188081920
4491939188539510	4491939188997273

4491939189454937	4491939189912635
4491939190370310	4491939190827905
4491939191285668	4491939191743342
4491939192200942	4491939192658687
4491939193116369	4491939193574005
4491939194031671	4491939194489350
4491939194947046	4491939195404732
4491939195862452	4491939196320120
4491939196777814	4491939197235500
4491939197693170	4491939198150853
4491939198608517	4491939199066195
4491939199523806	4491939199981487
4491939200439204	4491939200896880
4491939201354566	4491939201812245
4491939202269927	4491939202727606
4491939203185285	4491939203642964
4491939204100651	4491939204558275
4491939205016009	4491939205473680
4491939205931362	4491939206389035
4491939206846706	4491939207304351
4491939207762028	4491939208219787
4491939208677462	4491939209135134
4491939209592809	4491939210050506
4491939210508171	4491939210965781
4491939211423535	4491939211881208
4491939212338887	4491939212796577
4491939213254176	4491939213711932
4491939214169623	4491939214627213
4491939215084962	4491939215542644
4491939216000326	4491939216457998
4491939216915608	4491939217373360
4491939217831044	4491939218288645
4491939218746389	4491939219204003
4491939219661765	4491939220119426
4491939220577110	4491939221034788
4491939221492398	4491939221950153
4491939222407835	4491939222865502
4491939223323179	4491939223780877
4491939224238545	4491939224696230
4491939225153908	4491939225611523

Case Study No. 3

This time we have also tested NAZUZ on a plaintext file with different values for the algorithm. Choose a prime number L=15485863, a random number r=882235482, which is generated as illustrated in case study number one, and a text file that contains the message *Msg*= Hello This is my text to be:-->Encrypted "HELLOOOOW WORLD" please keep my file in a safe location. After encrypting the plaintext file, the ciphertext file contains:

1311569069567132820	1311569069582618703
1311569069598104567	1311569069613590440
1311569069629076220	1311569069644562156
1311569069660048029	1311569069675533809
1311569069691019749	1311569069706505624
1311569069721991398	1311569069737477345
1311569069752963193	1311569069768449075
1311569069783934934	1311569069799420713
1311569069814906660	1311569069830392518
1311569069845878302	1311569069861364231
1311569069876850097	1311569069892335917
1311569069907821767	1311569069923307630
1311569069938793510	1311569069954279380
1311569069969765284	1311569069985251136
1311569070000737014	1311569070016222884
1311569070031708738	1311569070047194605
1311569070062680453	1311569070078166315
1311569070093652110	1311569070109137975
1311569070124623876	1311569070140109736
1311569070155595606	1311569070171081469
1311569070186567335	1311569070202053198
1311569070217539061	1311569070233024924
1311569070248510795	1311569070263996603
1311569070279482521	1311569070294968376
1311569070310454242	1311569070325940099
1311569070341425954	1311569070356911783
1311569070372397644	1311569070387883587
1311569070403369446	1311569070418855302
1311569070434341161	1311569070449827042
1311569070465312891	1311569070480798685
1311569070496284623	1311569070511770480
1311569070527256343	1311569070542742217
1311569070558228000	1311569070573713940
1311569070589199815	1311569070604685589
1311569070620171522	1311569070635657388
1311569070651143254	1311569070666629110
1311569070682114904	1311569070697600840
1311569070713086708	1311569070728572493
1311569070744058421	1311569070759544219
1311569070775030165	1311569070790516010
1311569070806001878	1311569070821487740
1311569070836973534	1311569070852459473
1311569070867945339	1311569070883431190
1311569070898917051	1311569070914402933
1311569070929888785	1311569070945374654
1311569070960860516	1311569070976346315

NAZUZ has also been tested on different file sizes including large text file size of 93 MB, and as it is illustrated in Table 1, Figs. 5 and 6, the proposed algorithm has a better performance for file encryption and decryption.

6.1. Noise Complexity and Security Enhancement

The term "noise complexity" relates to introducing unpredictable variations during mathematical operations in FHE. This introduces a layer of security, making it arduous for unauthorized parties to extract meaningful information from encrypted data. These variations, termed "noise," act as a barrier against decryption attempts without proper authorization.

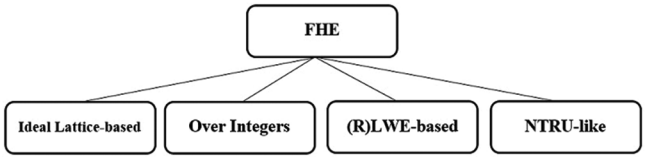

Fig. 1. Main fully homomorphic encryption schemes after Gentry's discovery [33]

6.2. Impact on Performance

While noise complexity significantly elevates security, it brings about a trade-off with performance. The inclusion of noise requires additional computational steps during decryption to handle its effects. This leads to increased processing time

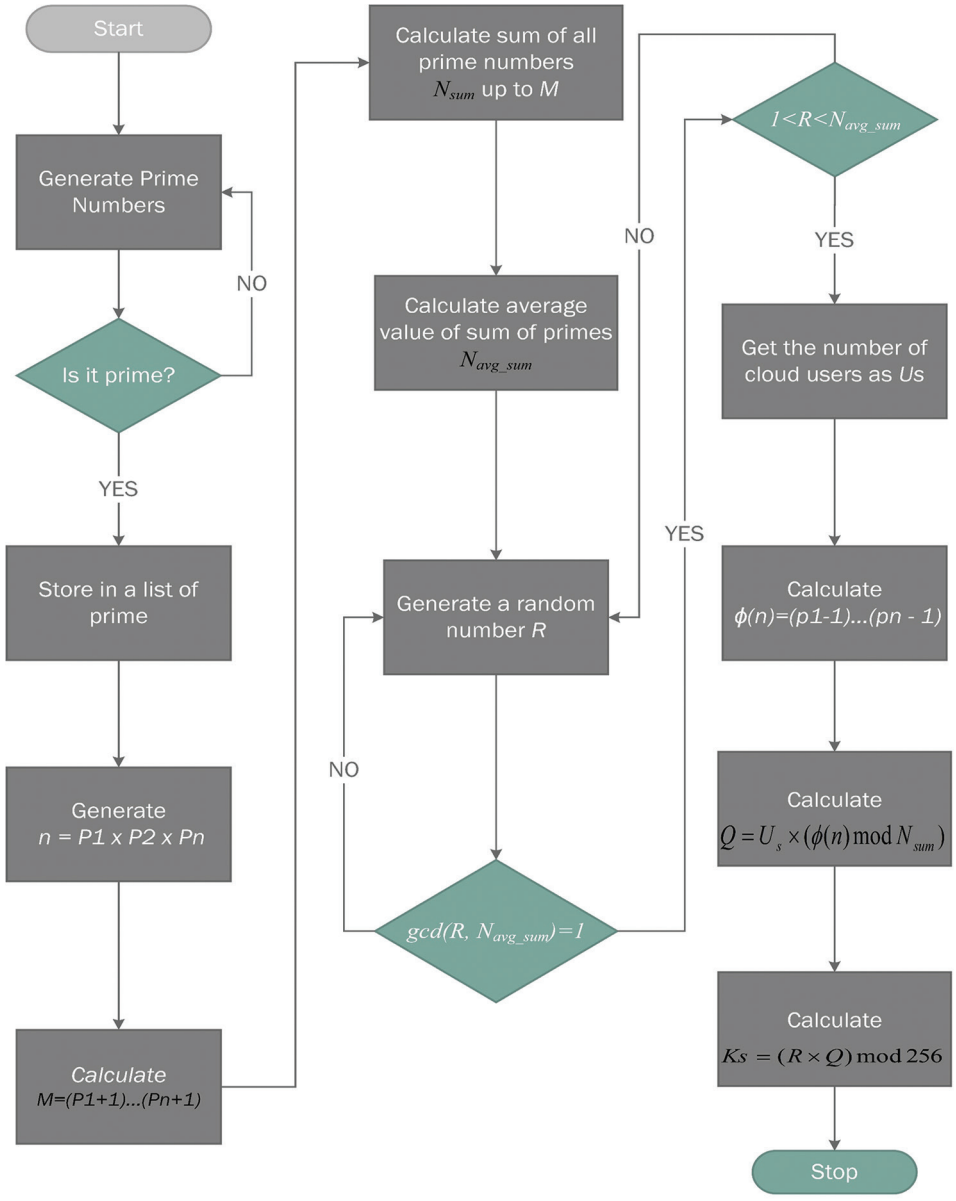

Fig. 2. Key generation flowchart

and resource utilization. This trade-off becomes particularly crucial when dealing with scenarios involving numerous users.

Table 1: NAZUZ performance on different file sizes

File size	Encryption time in seconds	Decryption time in seconds
10 KB	0.054	0.116
20 KB	0.116	0.201
50 KB	0.176	0.337
100 KB	0.263	0.456
200 KB	0.401	0.67
500 KB	0.783	1.288
1000 KB	1.419	2.322
2 MB	2.669	4.327
4 MB	4.744	6.686
8 MB	7.121	9.001
12 MB	9.783	12.451
16 MB	12.315	15.919
24 MB	15.460	25.158
47 MB	26.856	49.989
93 MB	56.968	98.212

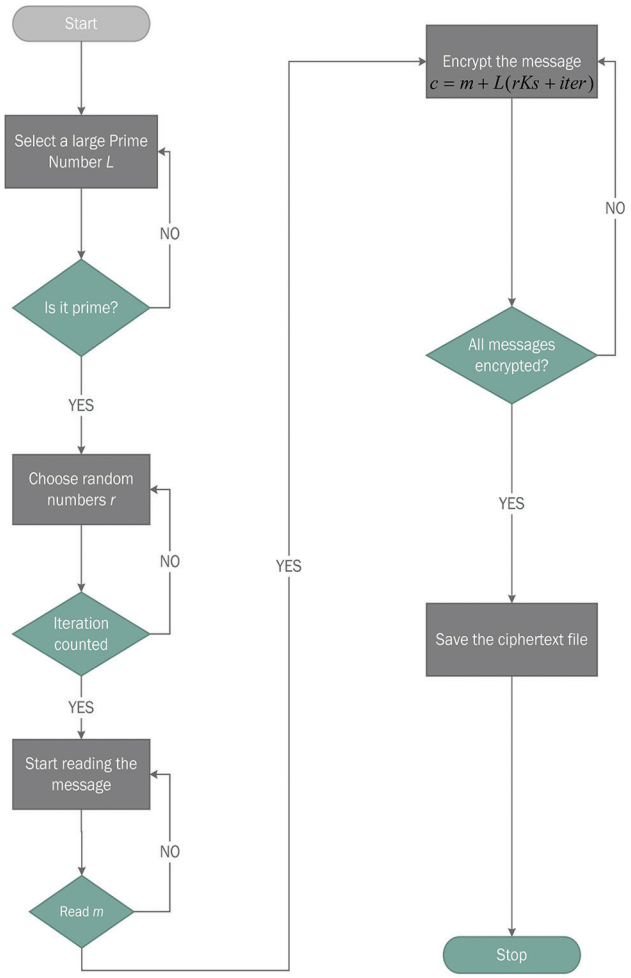

Fig. 3. Encryption flowchart

6.3. Strategies for Noise Management

To tackle the performance implications associated with noise complexity, several strategies are employed. These include monitoring noise levels, altering mathematical parameters, and employing controlled noise reduction techniques. These techniques are vital to ensure data integrity during decryption while maintaining acceptable performance levels.

6.4. Noise Complexity in Nazuz and Scalability Evaluation

The NAZUZ encryption approach integrates noise complexity intrinsically into the encryption process. As data undergo mathematical transformations, noise is introduced

Fig. 4. Decryption flowchart

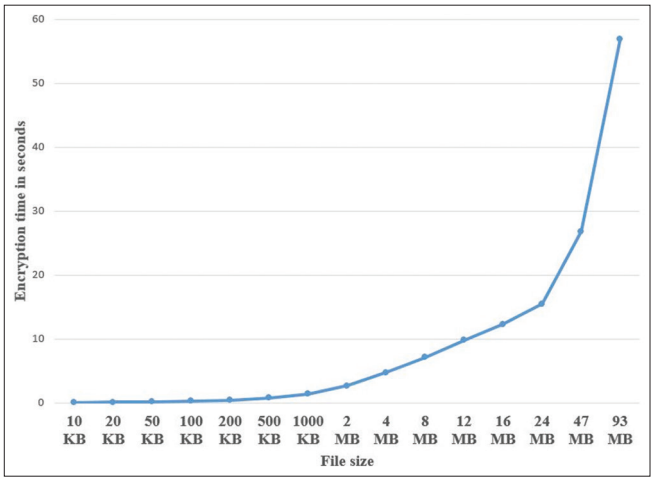

Fig. 5. Encrypting different file sizes measured in seconds

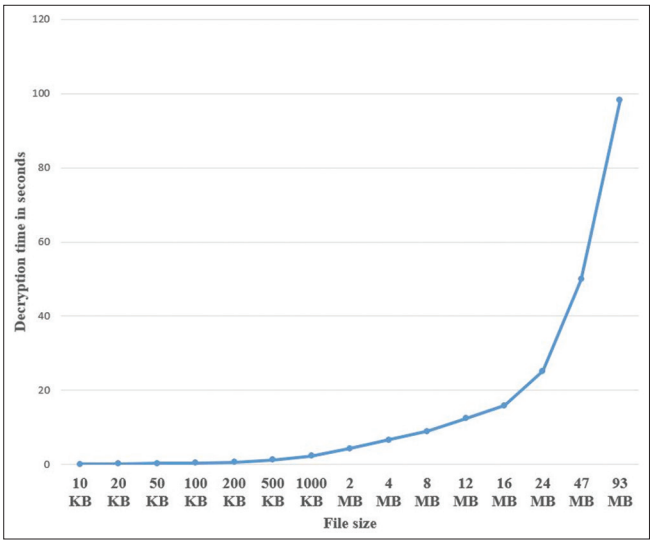

Fig. 6. Decrypting different file sizes measured in seconds

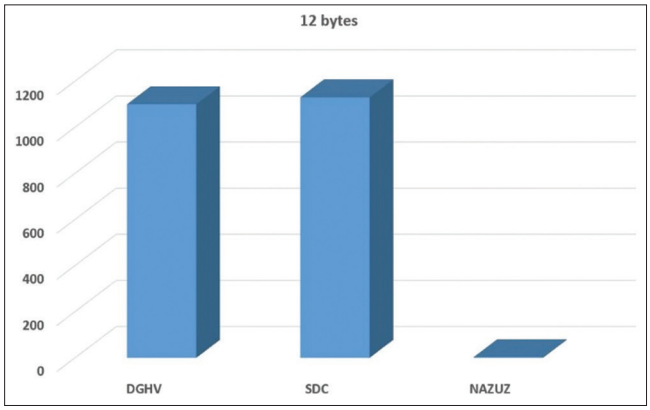

Fig. 7. Comparing NAZUZ to DGHV and SDC over a message of size 12 bytes

into the encrypted output. Our methodology encompasses noise management techniques, balancing security and performance considerations.

7. COMPARING NAZUZ TO DGHV AND SDC SCHEMES

The proposed algorithm (NAZUZ) has been compared to both DGHV and SDC schemes in terms of performance and it shows that the proposed algorithm works better than the other mentioned schemes, Table 2, Figs. 7-9 illustrate the comparison on different lengths of messages.

7.1. Limitations of the Work

The encryption process generates a ciphertext file that is larger than its corresponding plaintext file, and these results in

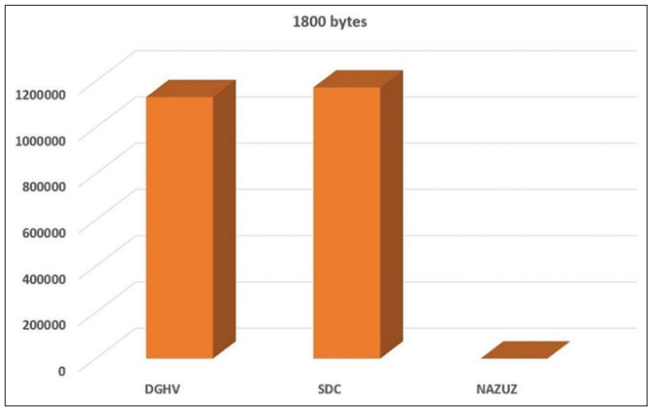

Fig. 8. Comparing NAZUZ to DGHV and SDC over a message of size 1800 bytes

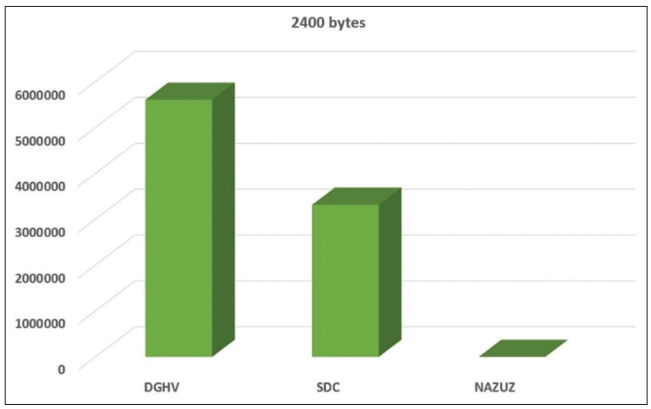

Fig. 9. Comparing NAZUZ to DGHV and SDC over a message of size 2400 bytes

Table 2: Comparing performance of DGHV, SDC, and NAZUZ measured in millisecond

Message length	DGHV (MS)	SDC (MS)	NAZUZ (MS)
12 bytes	1100	1130	3
1800 bytes	1130710	1172057	20
2400 bytes	5604037	3314788	35

the decryption process taking longer time than the encryption process.

8. CONCLUSION

A new dimension to cloud storage will be introduced using homomorphic encryption. The data will not be exposed at any stage so its confidentiality is guaranteed. Using FHE, the security of cloud computing will have a new concept and that is to perform calculations on encrypted data and produce the results without the knowledge of the original

data, and the confidentiality of data is respected. In this paper, we have proposed a FHE scheme named NAZUZ to protect cloud data at rest; this is by saving an encrypted version of the user's data. NAZAUZ works on converting each character of plaintext into a corresponding ASCII value, then passing it to the encryption algorithm. The results show that our proposed algorithm works better than other proposed algorithms in terms of security and performance, which works on encrypting large file sizes. NAZUZ provides very high security as it depends on the difficulty of factoring large integer numbers, which is still an open problem in mathematics. It also adds the complexity of noise to the plaintext, by taking the number of users of the cloud system. As well as the characteristic of FHE that allows performing calculations on ciphertext solves the problem of key management. It also generates different ciphertext even if the same character appeared more than once in the plaintext file. Therefore, it protects the plaintext file from being guessed or retrieved based on character repetition.

REFERENCES

[1] B. Alabdullah, N. Beloff and M. White. "*Rise of Big Data-Issues and Challenges*". Society National Computer Conference (NCC), Saudi Arabia, 2018.

[2] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman and D. Woods. "Cloud-trust-a security assessment model for infrastructure as a service (IaaS) clouds". *IIEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 523-536, 2017.

[3] R. Hamza, A. Hassan, A. Ali, M. Bashir, S. Al Qhutani, T. Tawfeeg and A. Yousif. "Towards secure big data analysis via fully homomorphic encryption algorithms". *Entropy*, vol. 24, no. 4, p. 519, 2022.

[4] B. Vankudoth and D. Vasumathi. "Homomorphic encryption techniques for securing data in cloud computing: A survey". *International Journal of Computer Applications*, vol. 160, no. 6, pp. 1-5, 2017.

[5] K. Sangani. "Sony security laid bare". *Engineering and Technology*, vol. 6, no. 8, pp. 74-77, 2011.

[6] R. L. Rivest, L. Adleman and M. L. Dertouzos. "On data banks and privacy homomorphisms. In: *Foundations of Secure Computation*". Academic Press, New York, 1978, pp. 169-180.

[7] M. Alia. "Combining public-key encryption with digital signature scheme. In: *Advances in Intelligent Systems and Computing*". Springer, Germany, 2016, pp. 870-878.

[8] A. C. Yao. "*Protocols for Secure Computations. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science* (*SFCS '82*)". IEEE Computer Society, 1982, pp. 160-164.

[9] S. Goldwasser and S. Micali. "Probabilistic encryption". *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270-299, 1984.

[10] T. Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469-472, 1985.

[11] P. Paillier. "Public-key cryptosystems based on composite degree residuosity classes. In: *Advances in Cryptology-Eurocrypt* '99". vol. 1592, Springer, Germany, 1999, pp. 223-238.

[12] C. Gentry. "*Fully Homomorphic Encryption Using Ideal Lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing* (*STOC '09*)", 2009, pp. 169-178.

[13] M. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan. "Fully homomorphic encryption over the integers. In: *Advances in Cryptology-Eurocrypt*". Springer, Germany, vol. 6110, pp. 24-43, 2010.

[14] N. Smart and F. Vercauteren. "Fully homomorphic encryption with relatively small key and ciphertext sizes. In: *Public Key Cryptography-PKC*". *2010*, Springer, Germany 2010, pp. 420-443.

[15] J. H. Cheon, H. Choe, D. Lee and Y. Son. "Faster linear transformations in HElib, revisited". *IEEE Access*, vol. 7, pp. 50595-50604, 2019.

[16] L. Xiao, O. Bastani and I. L. Yen. "*An Efficient Homorphic Encryption Protocol for Multi-User Systems*". IACR Cryptology, Bellevue, WA, 2012.

[17] T. Maha and E. L. H. Said. "Secure cloud computing through homomorphic encryption". *Computing Research Repository*, vol. 1409, 2014.

[18] A. Reem and E. Khaled. "*Cloud Computing Algebra Homomorphic Encryption Scheme Based on Fermat's Little Theorem*". The American Society of Engineering Education, United States, 2013.

[19] R. Hayward and C. Chiangb. "Parallelizing fully homomorphic encryption for a cloud environment". *Journal of Applied Research and Technology*, vol. 13, no. 2, pp. 245-252, 2015.

[20] A. Frederik, C. Boyd, C. Carr, K. Gjøsteen, A. J¨aschke, C. A. Reuter and M. Strand. "*A Guide to Fully Homomorphic Encryption*". vol. 1192. IACR Cryptology, Bellevue, WA, 2015.

[21] S. S. Hamad and A. M. Sagheer. "Design of fully homomorphic encryption by prime modular operation". *Telfor Journal*, vol. 10, no. 2, pp. 118-122, 2018.

[22] F. Xiao, J. Wang, M. Wei, C. Liu, V. Le and J. Xu. "*Privacy Data Protection Scheme of Industrial Field Equipment Based on Fully Homomorphic Encryption. In: 2023 International Conference on Information Networking* (*ICOIN*)". Bangkok, Thailand, 2023, pp. 236-241.

[23] S. Hashim and M. Benaissa. "*Fully Homomorphic Encryption Accelerator Using DSP Embedded Multiplier. In: 2023 15th International Conference on Computer Research and Development* (*ICCRD*)". Hangzhou, China, 2023, pp. 278-284.

[24] B. H. M. Tan, H. T. Lee, H. Wang, S. Ren and K. M. M. Aung. "Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields". *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2861-2874, 2021.

[25] B. Dan, G. Eu-Jin and N. Kobbi. "*Evaluating 2-DNF formulas on ciphertexts. In: Proceedings of Theory of Cryptography Conference*". vol. 3378, 2005, pp. 325-341.

[26] M. Fellows and N. Koblitz. "*Combinatorial Cryptosystems Galore!. In: Second International Conference on Finite Fields: Theory, Applications, and Algorithms August* (*Contemporary Mathematics*) *of Finite Fields: Theory, Applications, and Algorithms*, vol. 168, 1993, pp. 51-61

[27] G. Craig. "*A Fully Homomorphic Encryption Scheme. Ph.D. Dissertation*". Stanford University, 2009.

[28] L. Cardoso dos Santos, G. Rodrigues Bilar and F. Dacêncio

Pereira. "*Implementation of the fully homomorphic encryption scheme over integers with shorter keys*". *In: 2015 7th International Conference on New Technologies*, Mobility and Security (NTMS), Paris, 2015, pp. 1-5.

[29] R. Oded. "*On Lattices, Learning with Errors, Random Linear Codes, and Cryptography*. *In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing* (*STOC '05*)", 2005, pp. 84-93.

[30] H. Jeffrey, P. Jill and J. S. Joseph. "*NTRU: A Ring-Based Public Key Cryptosystem. In: International Algorithmic Number Theory Symposium*". Springer, Germany. pp. 267-288, 1998.

[31] G. Craig. "Computing arbitrary functions of encrypted data". *Communications of the ACM*, vol. 53, no. 3, pp. 97-105, 2010.

[32] J. Li, D. Song, S. Chen and X. Lu. "*A Simple Fully Homomorphic Encryption Scheme Available in Cloud Computing. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*". Hangzhou, 2012, pp. 214-217.

[33] A. Acar, H. Aksu, A. S. Uluagac and M. Conti. "A survey on homomorphic encryption schemes: Theory and implementation". *ACM Computing Surveys*. vol. 51, no. 4, pp. 1-35, 2018.