A new fully Homomorphic Cryptosystem based on a Super-increasing Sequence

Mohammed Anwar Mohammed and Fadhil Salman Abed

***Abstract* — Cloud computing is a powerful computing paradigm that provides a variety of computing services to its users. An example is storage, which allows individuals and enterprises to outsource their files to remote storage. However, saving private information onto third-party storage increases the security issues of data and privacy protection concerns. For this reason, cloud service providers (CSPs) are required to save an encrypted version of user data. In this paper, a novel encryption technique based on the use of Fully Homomorphic Encryption is presented. The technique uses a super-increasing sequence to derive the key and works on encrypted data with no need for decryption; this yields the same results as performing it on plaintext data. In the proposed technique, the characters are converted to their corresponding ASCII code values, which differs from the binary values produced by other existing techniques.**

***Keywords* — Cryptography, Cloud Computing Security, Fully Homomorphic Encryption, Information Security, Privacy Protection.**

I. Introduction

Cloud computing provides its users with a wide variety of computing services, such as servers and storage spaces. These allow its users to benefit from the advantages of these services on demand. One of the crucial services is data storage outsourcing, which enables individuals and enterprises to outsource their data to remote storage, which reduces the load on local storage. Nevertheless, outsourcing data to remote storage raises the issue of privacy protection, as the stored data contains credential information and is loaded onto third-party storage; therefore, the outsourced data requires privacy protection. One possible technique to address the privacy issue is simply encrypting the data by utilizing standard cryptographic algorithms before sending the data to the cloud servers. If outsourced data were to be encrypted using some traditional encryption algorithms, it would be impossible for users to directly process their encrypted outsourced data from the cloud servers. The only way for the users to process their data would be to firstly download their data, then decrypt the data, and finally locally process the decrypted data; this is an over-complicated process. Several schemes such as remote data possession checking protocol [1] or outsourced attribute-based encryption schemes [2], [3] have been proposed to ensure the security of outsourced data on the remote cloud servers. Fortunately, an important cryptographic technique known as Homomorphic Encryption (HE) was introduced. HE allows calculations to be performed directly on ciphertext data without decrypting it; operations on encrypted data produce the same results as those obtained with its corresponding plaintext operation. HE is categorized into either partial or full forms: Partially Homomorphic Encryption (PHE) supports addition or multiplication operations, while Fully Homomorphic Encryption (FHE) supports a random number of both operations [4]. It encrypts data and meanwhile supports arbitrary computations on encrypted data, and hence is well suited for cloud storage outsourcing scenarios [5]. Furthermore, interim of third-party computation (FHE) appears to be more secure and efficient than PHE, since it benefits from the advantages of properties of both operations [6].

The idea of FHE was firstly introduced by [7] in 1978, but to date the most significant ideas emerged with Gentry's breakthrough work thirty years later in 2009 [8], [9], which was based on ideal lattices. After that, several different frameworks were introduced to design more efficient FHE schemes. The frameworks were classified based on the underlying hard problems, such as learning with errors (LWE) [10], approximate greatest common divisor (AGCD) [11] and learning with errors over rings (RLWE) [12]. The most important schemes were proposed by [13] and [14], which both had a crucial effect on efficiency improvement. The former offered a novel FHE scheme that scaled down the ciphertext after every multiplication. Its noise growth is linear with multiplicative depth instead of exponential, which improves performance dramatically. The latter presented a method to improve the efficiency of the FHE scheme. Additionally, it made the FHE scheme asymptotically faster by using matrix addition and multiplication for homomorphic addition and multiplication, respectively.

Noise management is a crucial factor that affects the efficiency of FHEs. In order to improve the security of ciphertext, noise should be added to ciphertext before encrypting plaintext. Nevertheless, during the homomorphic evaluation process these noises increase significantly; therefore, the decryption algorithm will not decrypt the ciphertext correctly. Consequently, the HE scheme in the noisy framework is basically a somewhat homomorphic encryption scheme (SWHE). For this reason,

Paper received September 14, 2019; revised April 30, accepted May 26, 2020. Date of publication July 31, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Miroslav Lutovac.

Mohammed Anwar Mohammed is a PhD student at the Department of Computer Science, College of Science, University of Sulaimani, Iraq. (phone: 964-770-1575224; e-mail: mohammed.anwar@univsul.edu.iq).

Fadhil Salman Abed, is a professor at the Department of Information Technology, Kalar Technical Institute, Sulaimani Polytechnic University, Iraq; (phone: 964-770-5012864, e-mail: fadhil.abed@spu.edu.iq).

bootstrapping was launched in order to convert any SWHE to an FHE scheme. In addition, several experiments were conducted to introduce noise-free FHE schemes. However, all the presented techniques failed to provide a demonstrable secure result. In 2010 a new FHE scheme was introduced by [15], based on the approximate greatest common divisor (GCD) assumption and using a simple algebraic structure to simplify the computation. In [16] the authors introduced a public key compression scheme, which maintained security and reduced the public key size. The authors have recently proposed a new public key compression scheme with correction and further reduction of the public key size [17]. The authors of [18] introduced batch-processing technology based on the decisional approximate GCD to enhance the DGHV scheme. Furthermore, in [19] an improved (FHE) scheme was presented. The scheme is based on approximate GCD in which plaintext bits are simultaneously encrypted. Moreover, a new SWHE scheme for integers is proposed by [20], with the authors claiming that the public key size of SWHE is reduced. Furthermore, in 2019 the authors of [21] used a lookup table (LUT) to propose a protocol to evaluate any function using FHE. In [22] Li et al. constructed an efficient symmetric FHE scheme and utilized it to design a privacy-preserving-outsourced association rule mining scheme. Their proposal allows multiple data owners to jointly mine some association rules without sacrificing data privacy. The security of the HE scheme against the known plaintext attacks was established by examining the difficulty of solving nonlinear systems. However, the authors of [23] illustrated that the security of Li et al's HE is overvalued. They presented the retrieval and the second part can also be retrieved using an Euclidean algorithm to address the GCD problem of the first part of the secret key. Nonetheless, the essentials of HE and its various classifications were provided by [24] in 2019, with the authors analyzing different categories of HE. They mainly focused on FHE and investigated different FHE schemes.

The authors of [25] used the idea of packed ciphertexts to construct a multi-bit FHE with a short public key on the basis of the learning with errors (LWE) problem. Others worked on evaluating the existing techniques, with [26] presenting a detailed explanation of both DGHV and BGV schemes, showing their efficient techniques including bootstrapping and modulus switching. Moreover, [27] presented ways to reduce the computation complexity of encrypted data by adopting the concept of aggregate plaintext and proposing an efficient scheme to handle the comparison and swap operation, which is commonly used for sorting and searching in cloud computing. The authors of [28] presented two hardware architectures optimized for accelerating the encryption and decryption operations of the BFV HE scheme with high-performance polynomial multipliers. In late 2019, the authors of [29] introduced FHE as an antidote to the challenges of security and privacy of cloud data computation; they also provided insight into future research directions in the field of FHE. Also, [30] proposed an improved FHE based on N-primes, where the proposed model's security depends on the problem of Factorization the integers to their primary numbers. Additionally, in 2020, [31] presented a private comparison algorithm on encrypted integers using FHE, which scales efficiently for the length of input integers, applying techniques from finite field theory. While, [32] proposed a novel framework and an algorithm for securing cloud data at rest. The proposed framework guarantees users' privacy protection as they are communicating with an intermediary rather than with the cloud server directly.

This paper will present a novel encryption technique by using FHE; the technique uses a super-increasing sequence to derive the key and works on encrypted data with no need to decrypt it. The proposed technique works on converting each plaintext character into its corresponding ASCII code value, which contrasts with the binary values used in other existing techniques.

The remainder of this paper is organized as follows: section II explains the motivation behind the work. Section III presents the proposed algorithm in terms of its structure and proves its full homomorphism. Section IV provides the results obtained from different experiments and section V discusses the results and compares the proposed algorithm to other existing techniques, before illustrating the limitations of the study. Finally, section VI concludes the paper.

II. Reason of the Study

As the needs to remotely access private information increase, individuals and enterprises are outsourcing their private information to cloud storage. This requires addressing an extra layer of risks, which make it challenging to maintain security aspects such as data security, confidentiality, integrity, authentication and privacy. For instance, the PlayStation network was hacked in 2011 and millions of user accounts were breached, leaking 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 [33]. Thus, CSPs are required to save an encrypted version of user data on their storage. Several techniques can be used to encrypt users' data. Conversely, as the data resides on the cloud storage it needs to be decrypted before performing any operation on it. This might cause privacy and confidentiality issues in relation to the stored data. HE computations can be performed on encrypted data with no need to decrypt it and the results of the computations are the same as those processed on the corresponding plaintext data. Therefore, HE solves the issue of privacy protection and confidentiality of cloud data.

III. The Proposed Algorithm

The proposed algorithm converts each plaintext character into ASCII code and passes it to the encryption algorithm $c = S*(q*2*r)+m$ where c is the ciphertext, m is the plaintext message and $m \in [0, L-1]$, r as a random number is the noise added to the ciphertext.

Key generation

- Let $A = \{a_1, a_2, ..., a_n\}$ be a super-increasing sequence that satisfies $a_2 > a_1$, $(a_1 + a_2 + ...a_n) > (a_1 + a_2 + a_{n-1})$

- Calculate E as $E = a_1 + a_2 + ...a_n$
- Choose S and W, where $S > E$ and $2 \le W < S$, $\gcd(S,W) = 1$
- Calculate $b_i = W * a_i \bmod S$
- Calculate $B = \{b_1, b_2, ..., b_n\}$
- Randomly choose multiple numbers from b_i and compute q as it is equal to the sum of chosen numbers from b_i
- Choose a random number r, which is the noise added to the ciphertext

Encryption

$$c = S * (q * 2 * r) + m \quad (1)$$

Decryption

$$m = c \bmod S \quad (2)$$

Additive homomorphism

Let c^+ denote the additive homomorphic properties

$c^+ = c_1 + c_2 = (2*S*q*r_1 + m_1) + (2*S*q*r_2 + m_2)$

$= 2*S*q(r_1 + r_2) + (m_1 + m_2)$

But $2*S*q*(r_1 + r_2) \bmod S = 0$,

[since multiply of S modular S] Then $m^+ = m_1 + m_2$

Multiplicative homomorphism

Let c^+ denote the multiplicative homomorphic properties

$c^* = c_1 * c_2 =$

$(2*S*q*r_1 + m_1)*(2*S*q*r_2 + m_2)$

$= (2*S*q*r_1)*(2*S*q*r_2) + (2*S*q*r_1)$

$*m_1 + m_2 * (2*S*q*r_2) * m_1 + m_2$

$= 4*S^2*q^2*r_1*r_2 + 2*S*q*r_1*m_2 + m_1$

$*(2*S*q*r_2) + m_1 * m_2$

But

$[(4*S^2*q^2*r_1*r_2 + 2*S*q*r_1*m_2 + m_1 *$

$(2*S*q*r_2)) \bmod S = 0]$

[since multiply of S modular S] Then $m^* = m_1 * m_2$

IV. Result and Analysis

The proposed algorithm was tested and evaluated on a simulation developed using the Java programming language and tested on a computer with the following properties: Windows 10 64-bit operating system, processor Intel Core i7 and 16GB of RAM. The following case studies demonstrate the generation of the secret key and its corresponding values, and show how these values are used by the encryption and decryption algorithm.

A. First case study

Assume $A = \{15, 29, 108, 279, 563, 2243, 4468\}$ is a super-increasing sequence, then $E = 7705$ Let $S = 9291$ and $W = 2393$

Calculate $b_i = W * a_i \bmod S$ as

$b_1 = 2393 * 15 \bmod 9291 = 8022$ and so on.

$B = \{8022, 4360, 7587, 7986, 64, 6592, 7274\}$.

Then choose q by calculating the sum of two random numbers from B, $q = 8022 + 4360 = 12382$, then let $r_1 = 2$ and $r_2 = 3$, then choose two messages $m_1 = 30$ and $m_2 = 53$, and then calculate the two ciphertexts c_1 and c_2.

$c_1 = S*(q*2*r_1) + m =$
$9291*(12382*2*2) + 30$, $c_1 = 460164678$

$c_2 = S*(q*2*r_2) + m =$
$9291*(12382*2*3) + 53$, $c_2 = 690247025$

Additive homomorphism

$c_3 = c_1 + c_2 = 460164678 + 690247025$

$= 1150411703$

$m_3 = c_3 \bmod S = 1150411703 \bmod 9291 = 83$

$m_3 = m_1 + m_2 = 30 + 53 = 83$

Multiplicative homomorphism

$c_3 = c_1 * c_2 = 460164678 * 690247025$

$= 317627299999582950$

$m_3 = c_3 \bmod S$

$= 317627299999582950 \bmod 9291 = 1590$

$m_3 = m_1 * m_2 = 30 * 53 = 1590$

B. Second case study

This time, different larger values for the super-increasing sequence were selected in order to perform the algorithm on larger numbers. Let

$A = \{61, 75, 250, 977, 3987, 11235, 35659\}$ be a super-increasing sequence so $E = 52244$, and then choose $S = 68927$ and $W = 55235$, then calculate $b_i = W * a_i \bmod S$ as $b_1 = 55235 * 61 \bmod 68927 = 60839$ and so on.

$B = \{60839, 7005, 23350, 63681, 180, 15444, 35840\}$, then choose q by calculating the sum of two random numbers from B, $q = 7005 + 180 = 7185$. Let $r_1 = 5$ and $r_2 = 7$, then choose two messages $m_1 = 107$ and $m_2 = 109$, then calculate the two ciphertexts c_1 and c_2.

$c_1 = S*(q*2*r_1) + m = 68927*(7185*2*5) + 107$,

$c_1 = 4952405057$

$c_2 = S*(q*2*r_2) + m = 68927*(7185*2*7) + 109$,

$c_2 = 6933367039$

Additive homomorphism

$c_3 = c_1 + c_2 = 4952405057 + 6933367039$

$= 11885772096$

$m_3 = c_3 \bmod S = 11885772096 \bmod 68927 = 216$

$m_3 = m_1 + m_2 = 107 + 109 = 216$

Multiplicative homomorphism

$c_3 = c_1 * c_2 = 4952405057 * 6933367039$

$= 34336841985980716223$

$m_3 = c_3 \bmod S$

$= 34336841985980716223 \bmod 68927 = 11663$

$m_3 = m_1 * m_2 = 107 * 109 = 11663$

C. Third case study

In this case study, the proposed algorithm was tested on a plaintext message consisting of the text "**Please encrypt my information securely**". The values from the second case study were used to encrypt the message, and the generated ciphertext is:

4952405030 4952405058 4952405051 4952405047 4952405065
4952405051 4952404982 4952405051 4952405060 4952405049
4952405064 4952405071 4952405062 4952405066 4952404982
4952405059 4952405071 4952404982 4952405055 4952405060
4952405052 4952405061 4952405064 4952405059 4952405047
4952405066 4952405055 4952405061 4952405060 4952404982
4952405065 4952405051 4952405049 4952405067 4952405064
4952405051 4952405058 4952405071

D. Fourth case study

In this case study, the proposed algorithm was tested on a plaintext file of 272 bytes, which was encrypted in 0.006 seconds and decrypted in 0.025 seconds. The values from the second case study were also used in this experiment, and the encrypted output is as follows:

4952405015 4952405051 4952405060 4952405051 4952405047
4952405060 4952404982 4952405049 4952405061 4952405060
4952405053 4952405067 4952405051 4952404982 4952405065
4952405049 4952405051 4952405058 4952405051 4952405064
4952405055 4952405065 4952405063 4952405067 4952405051
4952404982 4952405058 4952405055 4952405053 4952405067
4952405058 4952405047 4952404982 4952405055 4952405050
4952404982 4952405065 4952405061 4952405050 4952405047
4952405058 4952405051 4952405065 4952404996 4952404982
4952405017 4952405058 4952405047 4952405065 4952405063
4952404982 4952405047 4952405062 4952405066 4952405051
4952405060 4952405066 4952404982 4952405066 4952405047
4952405049 4952405055 4952405066 4952405055 4952404982
4952405065 4952405061 4952405049 4952405055 4952405061
4952405065 4952405063 4952405067 4952404982 4952405047
4952405050 4952404982 4952405058 4952405055 4952405066
4952405061 4952405064 4952405047 4952404982 4952405066
4952405061 4952405064 4952405063 4952405067 4952405051
4952405060 4952405066 4952404982 4952405062 4952405051
4952405064 4952404982 4952405049 4952405061 4952405060
4952405067 4952405048 4952405055 4952405047 4952404982
4952405060 4952405061 4952405065 4952405066 4952405064
4952405047 4952404994 4952404963 4952404960 4952404982
4952405062 4952405051 4952405064 4952404982 4952405055
4952405060 4952405049 4952405051 4952405062 4952405066
4952405061 4952405065 4952404982 4952405054 4952405055
4952405059 4952405051 4952405060 4952405047 4952405051
4952405061 4952405065 4952404996 4952404982 4952405028
4952405067 4952405060 4952405049 4952404982 4952405065
4952405051 4952405059 4952404982 4952405058 4952405051
4952405049 4952405066 4952405067 4952405065 4952404994
4952404982 4952405053 4952405064 4952405047 4952405068
4952405055 4952405050 4952405047 4952404982 4952405047
4952405049 4952404982 4952405050 4952405067 4952405055
4952404982 4952405060 4952405061 4952405060 4952404994
4952404982 4952405062 4952405054 4952405047 4952405064
4952405051 4952405066 4952405064 4952405047 4952404982
4952405062 4952405061 4952405065 4952405067 4952405051
4952405064 4952405051 4952404982 4952405058 4952405051
4952405061 4952404996 4952404963 4952404960 4952404982
4952405027 4952405047 4952405051 4952405049 4952405051
4952405060 4952405047 4952405065 4952404982 4952405058
4952405047 4952405049 4952405067 4952405065 4952404982
4952405058 4952405055 4952405048 4952405051 4952405064
4952405061 4952404994 4952404982 4952405052 4952405047
4952405049 4952405055 4952405058 4952405055 4952405065
4952405055 4952405065 4952404982 4952405051 4952405066
4952404982 4952405051 4952405058 4952405055 4952405066
4952404982 4952405068 4952405055 4952405066 4952405047
4952405051 4952404994 4952404982 4952405049 4952405061
4952405059 4952405039 4952405061 4952405050 4952405061
4952404982 4952405052 4952405047 4952405049 4952405055
4952405058 4952405055 4952405065 4952405055 4952405065
4952404982 4952405065 4952405051 4952405059 4952404996
4952404963 4952404960

V. Discussion

The previous case studies show the functionality of the FHE of the proposed algorithm. This section presents the test results for the operation of this algorithm on different file sizes. The results from Tables 1 and 2, and Figures 1 and 2, illustrate that the proposed algorithm performs well in encrypting different file sizes. We also compared our proposed algorithm against other techniques, and the results show that it has a better performance than other presented techniques.

Table 1: The proposed algorithm tested on small file sizes measured in milliseconds

File size	*Encryption*	*Decryption*
10 KB	46	94
20 KB	62	156
50 KB	94	203
100 KB	156	296
200 KB	203	406
500 KB	359	688
1000 KB	641	1107

Table 2: The proposed algorithm tested on large file sizes measured in seconds

File size	*Encryption*	*Decryption*
2 MB	1	2
4 MB	2	4
8 MB	3	6
12 MB	5	9
16 MB	7	14
24 MB	10	18
47 MB	20	37
93 MB	37	61

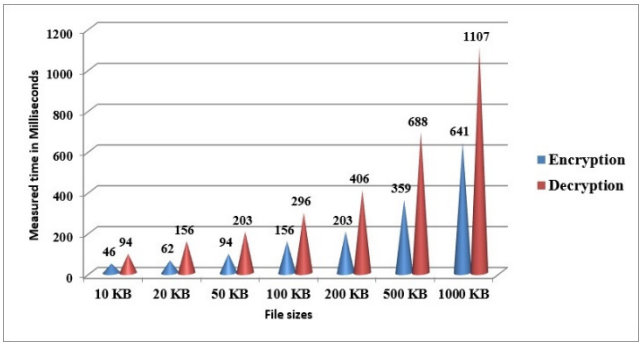

Fig. 1. Encryption and decryption performance on small file sizes.

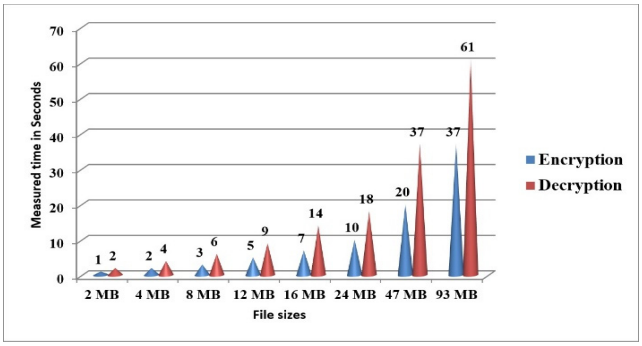

Fig. 2. Encryption and decryption performance on large file sizes.

A. Detailed security analysis of the proposed scheme

Our new algorithm aims to build a strong encryption system based on the property of Full Homomorphism, by building mathematical models that rely on randomness to strengthen and hide the elements of the secret key and protects against attacks on the ciphertext. This is accomplished through the process of hiding the secret key features as follows:

- The choice of a very large series of numbers satisfies the super-increasing condition (n=200 or more):

$$\sum_{i=1}^{n+1} a_i \geq \sum_{i=1}^{n} a_i$$

- Calculate $E = \sum_{i=1}^{n} a_i$
- Choose two random numbers S, W that satisfy $S > W$, $\gcd(W, S) = 1$
- Calculate a series of numbers based on $b_i = W * a_i \bmod S$ and $b_i = \{b_1, b_2, ..., b_n\}$
- Randomly choose multiple numbers from b_i and compute q as it is equal to the sum of chosen numbers from b_i
- Then add noise to the ciphertext through selecting a big integer value r

The above steps show that the process of building the secret key relies on several steps that further complicate the analysis and cracking of the secret key.

B. Results of NIST statistical tests on the generated secret keys

The randomness of this novel proposal is evaluated by the well-known NIST test suite. Table 3 shows the test results of the proposed algorithm from the NIST statistical tests, demonstrating that the best statistical performance was obtained with this algorithm.

TABLE 3: NIST SP 800-22 TEST RESULTS

Tests	*P-value*
Frequency (Monobits)	0.347218
Block Frequency	0.826728
Cumulative Sums (Cusum)	0.557894
Runs	0.955402
Longest Run of Ones	0.885923
Rank	0.862457
Discrete Fourier Transform	0.783087
Non-Overlapping Template Matching	0.967432
Overlapping Template Matching	0.959450
Universal Statistical	0.228486
Approximate Entropy	0.047855
Random Excursions	0.297181
Random Excursions Variant	0.153252
Serial	0.957312
Linear Complexity	0.959422
PASSED TESTS	**15/15**

C. The proposed algorithm in comparison to other proposed techniques

This section provides a comparison between the proposed algorithm and some previously presented techniques. The results prove that it performs better than other presented techniques in terms of file encryption and decryption. It was tested and compared to both SDC and DGHV schemes and the results presented in Table 4 and 5, and Figure 3 and 4, show the better efficiency of the proposed algorithm.

TABLE 4: ENCRYPTION TIME MEASURED IN SECONDS BETWEEN THE PROPOSED ALGORITHM AND SDC

Message length	*Proposed algorithm*	*SDC*
12 bytes	0.001	1.13
1800 bytes	0.022	117
2400 bytes	0.031	331

TABLE 5: ENCRYPTION TIME MEASURED IN SECONDS BETWEEN THE PROPOSED ALGORITHM AND DGHV

Message length	*Proposed algorithm*	*SDC*
12 bytes	0.001	1.1
1800 bytes	0.022	113
2400 bytes	0.031	560

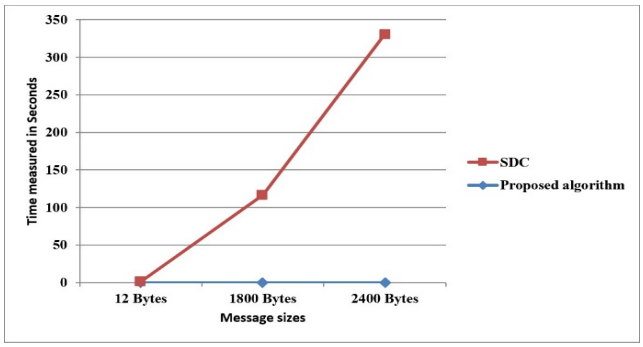

Fig. 3. Comparing the proposed algorithm to SDC.

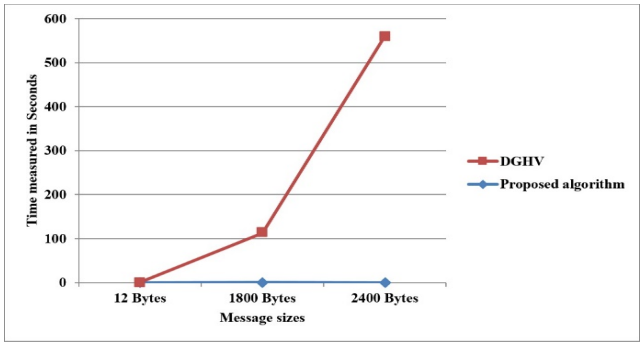

Fig. 4. Comparing the proposed algorithm to DGHV.

D. Limitation of the work

The size of the ciphertext file is one of the main points that should be addressed in future work, as the size of the encrypted file is larger than its equivalent plaintext file. Therefore, the decryption process always takes a longer amount of time than the encryption process. Also, the experiments were tested on a private cloud and it is recommended to test them on another type of cloud server.

VI. CONCLUSION

Homomorphic encryption will bring a new dimension to cloud storage. It provides confidentiality to the data as at no stage is data exposed in plaintext form. Security of cloud computing based on FHE is a new security concept, which enables the results of calculations on encrypted data to be provided without knowing the raw entries on which the calculation was carried out, thus respecting the confidentiality of data. In this paper, we have proposed a FHE scheme to protect cloud data at rest; this operates by saving an encrypted version of user data. The proposed algorithm works by converting each plaintext character into its corresponding ASCII value, and then passes it to the

encryption algorithm. The results show that our proposed algorithm works better than other proposed algorithms in terms of security and performance, and is able to encrypt larger file sizes. It also provides very high security as it depends on the difficulty of using a super-increasing sequence with a subset sum problem. It attempts to hide this sequence with modular multiplication to generate the key and uses it in FHE, which creates a robust security encryption system that is more difficult to analyze to access the plaintext and to obtain the keys. Additionally, it adds the complexity of noise to the plaintext, as well as the characteristic of FHE, which allows calculations to be performed on ciphertext and solves the problem of key management.

REFERENCES

[1] H. Yan, J. Li, Y. Zhang, and J. Han, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.

[2] J. Li, X. Lin, Y. Zhang and J. Han, "KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage," *IEEE Trans. on Services Computing*, vol. 10, no. 5, pp. 715–725, 2017. DOI: 10.1109/TSC.2016.2542813

[3] H. Yan, J. Li, J. Han and Y. Zhang, "A Novel Efficient Remote Data Possession Checking Protocol in Cloud Storage," *IEEE Trans on Information Forensics and Security*, vol. 12, no. 1, pp. 78–88, 2017. DOI: 10.1109/TIFS.2016.2601070

[4] T. Shen, F. Wang, K. Chen, K. Wang and B. Li, "Efficient Leveled (Multi) Identity-Based Fully Homomorphic Encryption Schemes," *IEEE Access*, vol. 7, pp. 79299–79310, 2019. DOI: 10.1109/ACCESS.2019.2922685

[5] K. Gai, M. Qiu, Y. Li and X. Liu, "Advanced Fully Homomorphic Encryption Scheme Over Real Numbers," *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 64–69. DOI: 10.1109/CSCloud.2017.61

[6] B. Vankudoth and D. Vasumathi, "Homomorphic Encryption Techniques for securing Data in Cloud Computing: A Survey," *Int. J. of Comp. App.*, vol. 160. pp. 1–5, 2017. DOI: 10.5120/ijca2017913063.

[7] R. L. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, USA, pp. 169–178, 2009.

[9] C. Gentry, "Computating arbitrry functions of encrypted data," *Commun. ACM*, vol. 53, pp. 97–105, 2010.

[10] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in Proc. *IEEE 52nd Annu. Symp. Found. Comput. Sci.*, CA, USA, 2011, pp. 97–106.

[11] J. Cheon and D. Stehlé, "Fully Homomophic encryption over the integers revisited," in *Advances in Cryptology-EUROCRYPT*, vol. 9056. Berlin, Germany, pp. 513–536, 2015.

[12] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from Ring-LWE and security for key dependent messages," in *Advances in Cryptology-CRYPTO*. Berlin, Germany, pp. 505–524, 2011.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. ACM 3rd Innov. Theor. Comput. Sci. Conf.*, pp. 309–325, 2012.

[14] C. Gentry, A. Sahai, and B.Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attributebased," in *Advances in Cryptology–CRYPTO.* Berlin, Germany, pp. 75–92, 2013.

[15] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology-EUROCRYPT*, vol. 6110, Berlin, Germany, pp. 24–43, 2010.

[16] J. S. Coron et al., "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology-CRYPTO*, vol. 6841. CA, USA, pp. 487–504, 2011.

[17] J. S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *Advances in Cryptology-EUROCRYPT*, vol. 7237. Cambridge, U.K., pp. 446–464, 2012.

[18] J. H. Cheon et al., "Batch fully homomorphic encryption over the integers," in *Advances in Cryptology-EUROCRYPT*, vol. 7881. Athens, Greece, pp. 315–335, 2013.

[19] R. Lin, J. Wang, and H. Du, "Improved fully homomorphic encryption over integers," *Appl. Res. Comput.*, vol. 30, no. 5, pp. 1515–1519, 2013.

[20] H. M. Yang, Q. Xia, X. F.Wang, and D. H. Tang, "A new somewhat homomorphic encryption scheme over integers," in *Proc. Int. Conf. Comput. Distrib. Control Intell. Environ. Monitor.*, pp. 61–64, 2012.

[21] R. Li, Y. Ishimaki and H. Yamana, "Fully Homomorphic Encryption with Table Lookup for Privacy-Preserving Smart Grid," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, Washington, DC, USA, 2019, pp. 19-24.

[22] L. Li, R. Lu, K. R. Choo, A. Datta and J. Shao, "Privacy-Preserving-Outsourced Association Rule Mining on Vertically Partitioned Databases," in *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1847-1861, Aug. 2016.

[23] B. Wang, Y. Zhan and Z. Zhang, "Cryptanalysis of a Symmetric Fully Homomorphic Encryption Scheme," in *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1460-1467, June 2018.

[24] P. Chaudhary, R. Gupta, A. Singh and P. Majumder, "Analysis and Comparison of Various Fully Homomorphic Encryption Techniques," in *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, NCR New Delhi, India, 2019, pp. 58-62.

[25] X. Song, Z. Chen and L. Chen, "A Multi-Bit Fully Homomorphic Encryption With Shorter Public Key From LWE," in *IEEE Access*, vol. 7, pp. 50588-50594, 2019.

[26] K. Hariss, M. Chamoun and A. E. Samhat, "On DGHV and BGV fully homomorphic encryption schemes," in *2017 1st Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, 2017, pp. 1-9.

[27] J. Ji and M. Shieh, "Efficient Comparison and Swap on Fully Homomorphic Encrypted Data," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, 2019, pp. 1-4.

[28] A. C. Mert, E. Öztürk and E. Savaş, "Design and Implementation of Encryption/Decryption Architectures for BFV Homomorphic Encryption Scheme," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 353-362, Feb. 2020.

[29] A. M. Jubrin, I. Izegbu and O. S. Adebayo, "Fully Homomorphic Encryption: An Antidote to Cloud Data Security and Privacy Concems.," in *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria, 2019, pp. 1-6.

[30] M. A. Mohammed and F. S. Abed, "An improved Fully Homomorphic Encryption model based on N-Primes", *Kurdistan Journal of Applied Research*, vol. 4, no. 2, pp. 40-49, Oct. 2019. DOI: https://doi.org/10.24017/science.2019.2.4.

[31] B. H. M. Tan, H. T. Lee, H. Wang, S. Q. Ren and A. M. M. Khin, "Efficient Private Comparison Queries over Encrypted Databases using Fully Homomorphic Encryption with Finite Fields," in *IEEE Transactions on Dependable and Secure Computing*.

[32] M. A. Mohammed and F. S. Abed, "A symmetric-based framework for securing cloud data at rest", *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 1, pp. 347-361, Jan. 2020. DOI: 10.3906/elk-1902-114

[33] K. Sangani, "Sony security laid bare," *Engineering & Technology*, vol. 6, no. 8, pp. 74–77, 2011. DOI:10.1049/et.2011.0810.