Kurdistan Journal of Applied Research (KJAR)
Print-ISSN: 2411-7684 | Electronic-ISSN: 2411-7706

Website: Kjar.spu.edu.iq| Email: kjar@spu.edu.iq

An improved Fully Homomorphic Encryption model based on N-Primes

Mohammed Anwar Mohammed
Computer Science Department
College of Science
University of Sulaimani
Sulaimani, Iraq
mohammed.anwar@univsul.edu.iq

Fadhil Salman Abed
Department of Information Technology
Kalar Technical Institute
Sulaimani Polytechnic University
Khanaqeen, Iraq
fadhil.abed@spu.edu.iq

Volume 4 - Issue 2
December 2019

DOI:
10.24017/science.2019.2.4

Received:
03 August 2019

Accepted:
01 October 2019

Abstract

Cloud computing is the provision of computing services over the internet, which provides unlimited computing capabilities to its users. Cloud Service Providers (CSP) in the distanced places helps the users such as businesses and individuals to use its software and hardware means. The physical distance between the users and providers allows third parties to be capable of accessing the data which threats the privacy of the users. Thus, its security is the main concern when it comes to transform data from a locally owned storage to cloud storage. Cloud providers are required to save an encrypted version of user's data on their storage. The traditional encryption schemes have been used for data encryption prior to sending them to the provider. Thought, the secret key has to be provided by the users to the server so as to decrypt the information prior to the requirement of calculations. Therefore, the traditional cryptographic schemes cannot be used to process cloud's data. After the encryption of the information data are revealed to calculation in clouds, so confidentiality is not guaranteed and this result in difficulty in using cloud. In Homomorphic Encryptions calculation on ciphertext can be performed with no need for decryption. This paper, develops and designs a new mathematical model to achieve the characteristics of the Fully Homomorphic Encryption. The proposed model's security depends on the problem of Factorization the integers to their primary numbers. In this paper, instead of dealing with two prime numbers it is expanded to deal with n prime numbers. The security of the presumptive algorithm to be more efficient in front of the security challenges facing cloud computing. What distinguishes this proposed system is that it deals with the

explicit text after converting it to the ASCII code instead of converting it to the binary system as it is in the existing systems, thus providing speed in the encryption process and returns the encryption.

Keywords: Storage protection, Cloud Computing Security, Cloud Storage, Fully Homomorphic Encryption, Privacy Protection

1. INTRODUCTION

Nowadays cloud computing plays an important role due to the fast progress of computer networks and big data, it enables individuals and enterprises to access services for instance storage or application on request [1]. It provides flexible remote storage and computing capabilities to its users. Nonetheless, saving private data on third parties storage is the main concern of cloud users, as they do not have full control over their data. Therefore, cloud computing is not fully trustable [2]. It is required from CSPs to store an encrypted version of user's data. Nevertheless, the traditional encryption technique requires encrypted data on the cloud to be decrypted before performing any operation on it. This still threaten the privacy of stored data. To overcome this concern, Homomorphic Encryption (HE) was introduced. HE allows performing calculations directly on ciphertext data without decrypting it. Additionally, operations on encrypted data results the same as it is performed on its corresponding plain text operation. Also, It is categorized into either Partial or Full, as Partially (PHE) supports either addition or multiplication operation, while, Fully (FHE) supports random number of both operations [3].Furthermore, interim of third party computation (FHE) appears to be more secure and efficient than (PHE), since it gets advantage of properties of both operations [4]. The rest of the paper is structured as follows; subsection 1.1 explains the motivation behind the work. In section 2 a brief on related study to the proposed work will be introduced. Then, section 3 provides the methods and materials used as the idea of (HE) and the details of the proposed algorithm. Additionally, section 4 presents a detailed explanation of the results and experiments gained from the work.Discussions of the proposed work its performance and comparing it to other existing techniques will be introduced in section 5. Lately, the conclusion of the current work and recommendation for future work is located in section 6.

1.1. *Reason of the study*

Nowadays, the demands of accessing private information whenever needed have increased rapidly. Enterprises and individuals are aiming at saving their private information on cloud storage. Nevertheless, migrating private information from a locally owned storage to a third party storage arises the challenge of addressing of extra amount of risks, such as maintaining confidentiality, integrity, authentication, data security and privacy. For example, the cyber-attacks on PlayStation network in 2011 leads to the breach of millions of user accounts leaking passwords, credit card information, physical addresses and other personal information. After the attacks, Sony announced that they could have taken special protection by encrypting the data on their network [5].Thus; it is required from CSPs to save an encrypted version of user's data on their storage. Several techniques can be used to perform encryption on user's data. Conversely, as the data resides on the cloud storage it required to be decrypted before performing any operation on the data. This might cause privacy and confidentiality issues to the stored data. In Homomorphic Encryption computations can be performed on encrypted data with no need to decryptit and the results of the computations are same as they were processed on the corresponding plaintext data. Therefore, HE solves the issue of privacy protection and confidentiality of cloud data.

2. RELATED LITERATURE

At first the idea of homomorphic encryption was suggested by [6] which was partially homomorphic encryption. Then multiplicative homomorphism introduced by RSA [7]. Subsequently, [8] [9] [10] [11] were presented a partially homomorphic encryption scheme. Afterward, [12] suggested a (FHE) scheme that performs calculation of any number of multiplication and addition. However, the suggested scheme was based on somewhat homomorphic encryption which increases the length and cipher-text's noise. Consequently, in [13] a (FHE) scheme has been introduced in which the scheme uses elementary modular arithmetic and converts SWHE to FHE by using Gentry's technique. Then, the authors of [14] announced an improved version of Smart-Vercauterenencryption scheme; the scheme decreased the ciphertext and keys lengths. In 2013 IBM released a software package named HElib which based on the use of Smart-Vercauterentechniques [15]. Moreover, in [16] the authors have depended on the hardness of large integer factorization to introduce a new homomorphic encryption scheme.The authors also show that how size of the key and time of computations reduced enough for practical deployment. Afterward, homomorphic encryption has been worked on by several authors and also examined it on cloud computing system. Furthermore, a study on different homomorphic encryption cryptosystems has been done by [17] the authors examined El-Gamal, Paillier, RSA and Gentry on a cloud computing environment. In [18] a new mechanism based on algebraic homomorphic encryption was introduced, this mechanism was aimed at better security and was based on Fermat's Little Theorem. Additionally, the authors of [19] proposed Gentry's encryption in parallel processing and were tested on a private cloud. Also, in [20] simplified and structured wide definitions in the homomorphic encryption discipline has been introduced, and raised the question of using homomorphic encryption as a solution to their problem.

3. METHODS AND MATERIALS

3.1. *The idea of Homomorphic Encryption (HE)*

This section introduces the basics of (HE) and its different categories. It is categorized into three different types as Fully Homomorphic Encryption (FHE), Somewhat Homomorphic Encryption and Partially Homomorphic Encryption (PHE).An encryption scheme is called homomorphic over the operation '*' if it supports the following equation [21]:

$$E(m_1) * E(m_2) = E(m_1 * m_2), \forall m_1, m_1 \in M \qquad (1)$$

(PHE) allows any number of either addition or multiplication. Examples of PHE are El-Gamal, RSA, Goldwasser-Micali, Paillier and Benaloh. Whereas, in (SWHE) some types of operations with limited number of times are allowed, such as, PollyCracker introduced by [22] and BNG introduced by [23]. In (FHE) unlimited number of both addition and multiplication can be performed, examples of FHE are FHE schemes Over Integers [24], NTRU-like FHE schemes [25], Ideal Lattice-based FHE schemes [26], LWE-based FHE schemes [27], Gen10 [28] and Simple FHE scheme [29].

3.2. *The proposed algorithm*

The proposed algorithm converts each plaintext character into ASII code and passes it to the encryption algorithm $ct = m + r * l * n$ where ct is the ciphertext, m is the plaintext message and $m \in [0, L-1]$, r is the noise added to the ciphertext, l is a prime big integer and $n = p_1 * p_2 * \ldots p_i$ is the multiplication of numerous prime numbers resulting in one ciphertext for each character in the plaintext.

Step 1: Key generation

Generate $l \in Z_n$

Generate l, where l is a big prime integer value

Generate n, where n is a multiplication of various prime numbers

Step 2: Encryption algorithm

$$ct = m + r * l * n \quad (2)$$

Where $m \in [0, L-1]$ and $n = p_1 * p_2 * ... p_i$

Step 3: Decryption algorithm

$$m = ct \bmod l \quad (3)$$

Step 4: Proof of additive homomorphism

$$m_3 = ct_1 + ct_2 = (m_1 + r_1 * l * n) + (m_2 + r_2 \times l * n) = (m_1 + m_2) + l * n(r_1 + r_2)$$

$m_3 = m_1 + m_2$ Since $l \times n(r_1 + r_2) \bmod l = 0$ then $m_3 = m_1 + m_2$

Step 4: Proof of multiplicative homomorphism

$$m_3 = ct_1 * ct_2 = (m_1 + r_1 * l * n) * (m_2 + r_2 * l * n)$$

$$m_3 = [\{m_1 * m_2 + m_1 * r_2 * l * n\} + \{(m_2 * r_1 * l * n) + (r_1 * l * n)(r_2 * l * n)\}]$$

$$m_3 = [m_1 * m_2 + m_1 * r_2 * l * n + r_1 * l * n * m_2 + r_1 * r_2 * l^2 * n^2]$$

$$m_3 = [m_1 * m_2 + m_1 * l * \{r_2 * n + r_1 * n * m_2 + r_1 * r_2 * l * n^2\}]$$ But

$[l * \{r_2 * n + r_1 * n * m_2 + r_1 * r_2 * l * n^2] \bmod l = 0$, since (multiple of $l \bmod l = 0$) then $m_3 = m_1 * m_2$

4. RESULTS

The proposed algorithm has been tested on a simulation using Java programming language and on a computer with Windows 10 64-bit operating system, Intel Core i7 processor and 16GB RAM. The following experiments demonstrate the generation of the secret key and its corresponding values, and to show how these values are used for encryption and decryption algorithm.

4.1. *Experiment 1*

Choose a prime number $l = 524287$, random numbers $r_1 = 687529, r_2 = 249685$, $n = 23 * 73 * 101 = 69579$ and two messages $m_1 = 65$ and $m_2 = 66$. Then calculate the ciphertexts ct_1 and ct_2.

$$ct_1 = m_1 + r_1 * l * n = 65 + 687529 * 524287 * 69579 = 25080621453027582$$

$$ct_2 = m_2 + r_2 * l * n = 66 + 249685 * 524287 * 69579 = 9108350293220571$$

Proof of additive

$$ct_3 = ct_1 + ct_2 = 25080621453027582 + 9108350293220571 = 341889717513248153$$

$$m_3 = ct_3 \bmod l = 341889717513248153 \bmod 524287 = 131$$

$m_3 = m_1 + m_2 = 65 + 66 = 131$

Proof of multiplicative

$ct_3 = ct_1 * ct_2 = 25080621453027582 * 9108350293220571$

$= 22844308581137967146740012778932e+32$

$m_3 = ct_3 \bmod l = 22844308581137967146740012778932e+32 \bmod 524287 = 4290$

$m_3 = m_1 * m_2 = 65 * 66 = 4290$

4.2. *Experiment 2*

In experiment 2 the proposed algorithm has been tested on a plaintext file that contains a message "Use new techniques of encryption for CSP in 2019." Choose a prime number $l = 6700417$, random number $r = 345957466$, $n = 31 * 53 * 71 = 116653$ then the encrypted file will contain:

270408569943805901351	270408569943805901381	270408569943805901367
270408569943805901298	270408569943805901376	270408569943805901367
270408569943805901385	270408569943805901298	270408569943805901382
270408569943805901367	270408569943805901365	270408569943805901370
270408569943805901376	270408569943805901371	270408569943805901379
270408569943805901383	270408569943805901367	270408569943805901381
270408569943805901298	270408569943805901377	270408569943805901368
270408569943805901298	270408569943805901367	270408569943805901376
270408569943805901365	270408569943805901380	270408569943805901387
270408569943805901378	270408569943805901382	270408569943805901371
270408569943805901377	270408569943805901376	270408569943805901298
270408569943805901368	270408569943805901377	270408569943805901380
270408569943805901298	270408569943805901333	270408569943805901349
270408569943805901346	270408569943805901298	270408569943805901371
270408569943805901376	270408569943805901298	270408569943805901316
270408569943805901314	270408569943805901315	270408569943805901323
270408569943805901312		

4.3. *Experiment 3*

In experiment 3 the proposed algorithm has been tested on a plaintext file that contains a message "Test your algorithm on any character and number: $#123" Choose a prime number $l = 6700417$, random number $r = 345957466$, $n = 31 * 53 * 71 = 116653$ then the encrypted file will contain:

270408569943805901350	270408569943805901367	270408569943805901381
270408569943805901382	270408569943805901298	270408569943805901387
270408569943805901377	270408569943805901383	270408569943805901380
270408569943805901298	270408569943805901363	270408569943805901374
270408569943805901369	270408569943805901377	270408569943805901380
270408569943805901371	270408569943805901382	270408569943805901370
270408569943805901375	270408569943805901298	270408569943805901377
270408569943805901376	270408569943805901298	270408569943805901363
270408569943805901376	270408569943805901387	270408569943805901298
270408569943805901365	270408569943805901370	270408569943805901363
270408569943805901380	270408569943805901363	270408569943805901365
270408569943805901382	270408569943805901367	270408569943805901380
270408569943805901298	270408569943805901363	270408569943805901376
270408569943805901366	270408569943805901298	270408569943805901376
270408569943805901383	270408569943805901375	270408569943805901364

270408569943805901367 270408569943805901380 270408569943805901324 270408569943805901298 270408569943805901302 270408569943805901301 270408569943805901315 270408569943805901316 270408569943805901317

4.4. ***Experiment4***

This time the proposed algorithm has been tested on a 911 Bytes file size, which was encrypted in 0.010 MS and decrypted in 0.025 MS.Also in this experiments small numbers were chosen as follows; choose a smallprime number $l = 3$, a random number $r = 5$, $n = 2*3*5 = 30$ then the encrypted file will contain:

533 567 565 562 551 560 550 555 565 545 551 482 565 551 550 482 565 551 559 562 551 564 482 552 551 558 555 565 496 482 519 566 555 547 559 482 559 547 566 586 555 565 482 559 547 553 560 547 482 559 555 494 482 565 567 565 549 555 562 555 566 482 567 558 528 547 559 549 561 564 562 551 564 482 566 551 558 568 567 565 482 551 567 555 565 559 561 550 482 565 551 550 496 482 515 551 560 551 547 560 482 549 561 560 553 567 551 482 565 549 551 558 551 564 555 565 563 567 551 482 558 555 553 567 558 547 482 555 550 482 565 561 550 547 558 551 565 496 482 517 558 547 565 595 482 547 562 566 551 560 566 482 566 547 549 555 566 555 482 565 561 549 555 561 565 563 567 482 547 550 482 558 555 566 561 564 547 482 566 561 564 563 567 551 560 566 482 562 551 564 482 549 561 560 567 548 555 547 482 560 561 565 566 564 547 494 482 562 551 564 482 555 560 549 551 562 566 561 565 482 554 555 559 551 560 547 551 561 565 496 482 528 567 560 549 482 565 551 559 482 558 551 549 566 567 565 494 482 553 564 547 568 555 550 547 482 547 549 482 550 567 555 482 560 561 560 494 482 562 554 547 564 551 566 564 547 482 562 561 565 567 551 564 551 482 558 551 561 496 482 527 547 551 549 551 560 547 565 482 558 547 549 567 565 482 558 555 548 551 564 561 494 482 552 547 549 555 558 555 565 555 565 482 551 566 482 551 558 555 566 482 568 555 566 547 551 494 482 549 561 559 519 561 550 561 482 552 547 549 555 558 555 565 555 565 482 565 551 559 496 482 536 555 568 547 559 567 565 482 555 550 482 560 555 565 558 482 560 567 558 528 547 496 482 523 560 566 551 553 551 564 482 547 566 482 559 547 570 555 559 567 565 482 550 567 555 496 482 535 566 482 547 482 566 555 560 549 555 550 567 560 566 482 558 561 564 551 559 496 482 536 555 568 547 559 567 565 482 568 555 566 547 551 482 558 555 553 567 558 547 482 568 551 558 482 558 547 549 567 565 482 549 567 564 565 567 565 482 549 561 560 550 555 559 551 560 566 567 559 496 482 530 554 547 565 551 558 578 567 565 482 563 567 555 565 482 559 547 567 564 555 565 482 558 561 548 561 564 566 555 565 494 482 552 555 560 555 548 567 565 482 558 561 564 551 559 482 555 560 494 482 568 567 558 562 567 566 547 566 551 482 551 570 496 482 526 561 564 551 559 482 555 562 565 567 559 482 550 561 558 561 564 482 565 555 566 482 547 559 551 566 494 482 549 561 560 565 551 549 566 551 566 567 564 482 547 550 555 562 555 565 549 555 560 553 482 551 558 555 566 496 482 533 551 550 482 552 547 567 549 555 548 567 565 482 547 558 555 563 567 547 559 482 559 551 566 567 565 494 482 563 567 555 565 482 568 547 564 555 567 565 482 551 558 555 566 482 562 561 564 566 466 555 566 561 564 482 555 550 496 482 536 555 568 547 559 567 565 482 550 555 553 560 455 565 531 585 559 482 565 561 558 458 555 549 555 566 567 550 555 560 482 565 549 551 558 551 564 555 565 563 567 551 496 482 527 561 564 548 555 482 566 555 560 549 555 550 567 560 566 494 482 550 561 558 561 564 482 563 567 555 565 482 568 551 554 555 549 567 558 547 482 549 561 560 565 551 563 567 547 566 494 482 550 567 555 482 550 555 547 559 482 549 561 560 550 555 559 551 560 566 567 559 482 560 567 560 549 494 482 568 555 566 547 551 482 565 549 551 558 551 564 555 565 563 567 551 482 561 550 555 561 482 558 555 548 551 564 561 482 560 551 549 482 558 555 553 567 558 547 496 482 536 551 565 566 555 548 567 558 567 559 482 547 560 566

551 482 555 562 565 567 559 482 562 564 555 559 555 565 482 555 560 482 552 547 567 549 555 548 567 565 482 561 564 549 555 482 558 567 549 566 567 565 482 551 566 482 567 558 566 564

5. DISCUSSION

The proposed algorithm has been tested on different file sizes and it indicates that the proposed algorithm has better performance for encrypting different file sizes including large files. The below figures 1, 2 and table 1 illustrates the encryption and decryption time measured in millisecond.

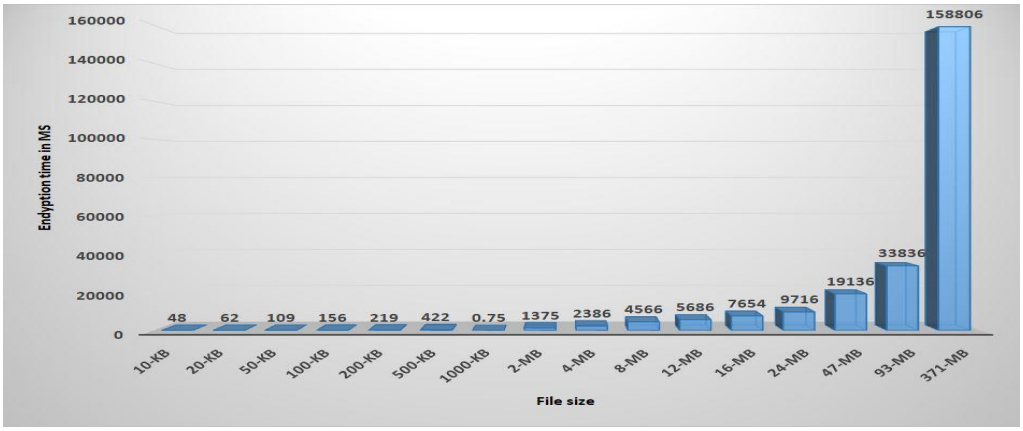

Figure 1: Encryption time measured in millisecond

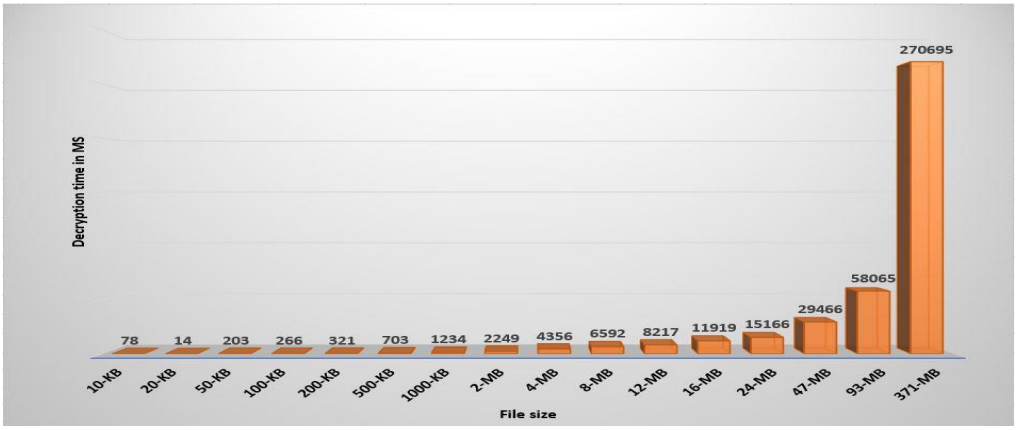

Figure 2: Decryption time measured in millisecond

Table 1: Testing the proposed algorithm on different file sizes

File size	Encryption (MS)	Decryption (MS)
10 KB	48	78
20 KB	62	14
50 KB	109	203
100 KB	156	266
200 KB	219	321
500 KB	422	703
1000 KB	750	1234

2 MB	1375	2249
4 MB	2386	4356
8 MB	4566	6592
12 MB	5686	8217
16 MB	7654	11919
24 MB	9716	15166
47 MB	19136	29466
93 MB	33836	58065
371 MB	158806	270695

5.1. *The proposed algorithm vs. DGHV and SDC schemes*

The proposed algorithm has been tested and compared to both DGHV and SDC schemes in terms of performance and the results shows that it has better performance than the other mentioned schemes, the below table 2 and figure 3 illustrates the comparison on different lengths of messages.

Table 2: Comparing the proposed algorithm with DGHV and SDC measured in second

Message length	**Proposed Algorithm**	**DGHV**	**SDC**
12 bytes	0.003	1.1	1.13
1800	0.02	113.071	117.2057
2400	0.35	560.4037	331.4788

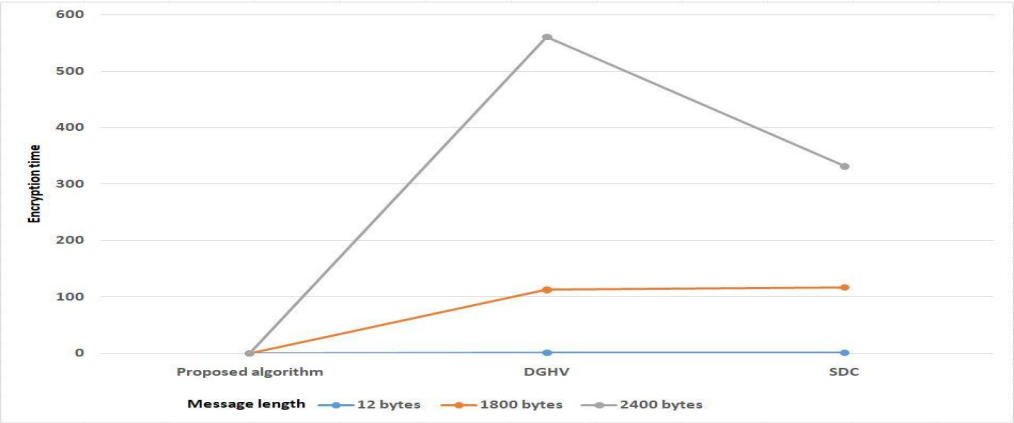

Figure 3: Execution times measured in second

In term of security we have used n as multiple of i-Prime Modular Operation which provides the security over the networks. In which we endeavored to get the quality that makes the cryptography easier to have a good use of i- prime numbers. Additionally, the proposed system provides easier selection of prime numbers and computationally difficult to solve by intruders.However,the DGHV and SDC schemes are using only two large prime numbers.

Furthermore, this paper uses the equation of $ct = m + r * l * n$ and it deals with massage by converting each character of plaintext into its corresponding ASCII value , where message $m \in [0, L-1]$,while, SDC Scheme handles the message after converting it to binary system and themessage $m \in [0,1]$ this increases the time it takes to encrypt a message, however,

selecting n a secret key in the proposed scheme gives the robust in the security than choosing any number. The reason for this belongs to that the i-prime numbers itself cost to the third party where, s/he must verify first if that the number is prime and then experiment it on the encryption equation. This requires further attempts to break the code; in addition, the prime numbers gives one probability of the solution. The non-prime numbers does not give all probabilities of the solution and it may give more than number of same solution.

5.2. *Research limitations*

This section explains the limitation of the proposed study. Ciphertext file size is one of the main points that should be tackled in the further works; as the size of the encrypted file is larger than its equivalent plaintext file. Therefore, the decryption process always takes further time than encryption process. The experiments are tested on private cloud and it is recommended to test them on another type of cloud server.

4. CONCLUSION

In this paper we have proposed a FHE algorithm to protect cloud data; this is by saving an encrypted version of user's data. The proposed algorithm works by converting each character of plaintext into corresponding ASCII value, then pass it to the encryption algorithm. The proposed model's security depends on the problem of factorization the integers to their primary numbers, and it deals with n prime numbers of the security of the presumptive algorithm, which make the proposed algorithm more efficient in facing of the challenge of cloud computing security. The proposed algorithm has been compared to other existing algorithms such as DGHV and SDC, and the results show thatit performs better in term of security and performance, which also works on encrypting large file sizes.Nevertheless, it is recommended as future work to focus on the decreasing the size of the decrypted file (ciphertext file), as it is larger than its corresponding plaintext file. Therefore, it requires more time to decrypting the ciphertext file than encrypting its corresponding plaintext file.

REFERENCE

[1] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani and S. U. Khan, "The rise of 'big data' on cloud computing: review and open research issues," Information Systems, vol. 47, pp. 98–115, 2015.

[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," IEEE Transactions on Cloud Computing, vol. 5, no. 3, pp. 523–536, 2017.

[3] 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

[4] B. VankudothandD. Vasumathi, "Homomorphic Encryption Techniques for securing Data in Cloud Computing: A Survey," International Journal of Computer Applications,Vol. 160. pp. 1-5, 2017. doi:10.5120/ijca2017913063.

[5] K. Sangani, "Sony securitylaidbare," in Engineering&Technology, vol. 6, no. 8, pp. 74-77, 2011. doi:10.1049/et.2011.0810

[6] R. L. Rivest, L. Adleman and M. L. Dertouzos,"On data banks and privacyhomomorphisms", Foundations of securecomputation, vol. 4, no. 11, pp. 169-180, 1978.

[7] R. L. 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.

[8] A. C. Yao,"Protocols for secure computations (extended abstract)," InIEEE 23rd Annual Symposium on Foundations of Computer Science (FOCS'82), pp. 160-164, 1982.

[9] G. Shafi 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, Advances in cryptology. Springer Berlin Heidelberg, 1985.

[11] P. Paillier,"Public-key cryptosystems based on composite degree residuosity classes,"International Conference on the Theory and Applications of Cryptographic Techniques,pp. 223-238, 1999.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proc. of the 41st ACM Symposium on Theory of Computing STOC'2009, pp. 169–178, 2009,

[13] M. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in Proc. EUROCRYPT'2010, pp. 24–43, 2010.

[14] N. Smart and F. Vercauteren, “Fully homomorphic encryption with relatively small key and cipher sizes,” in Proc. Public Key Cryptography PKC’2010, pp. 420–443, 2010.
[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. doi: 10.1109/ACCESS.2019.2911300
[16] L. Xiao, O. Bastani and I-L. Yen, “An efficient homomorphic encryption protocol for multi-user systems.”IACR Cryptology ePrint Archive, 2012.
[17] M. TEBAA and S. E. HAJII,”Secure Cloud Computing through Homomorphic Encryption”, International Journal of Advancements in Computing Technology(IJACT), vol. 5, no. 16, 2013.
[18] R.Alattas, K.Elleithy, ”Cloud Computing Algebra Homomorphic Encryption Scheme Based on Fermat’s Little Theorem”, In The American Society of Engineering Education, ASEE, 2016.
[19] R. Hayward andC.Chiangb, ”Parallelizing fully homomorphic encryption for a cloud environment”, Journal of Applied Research and Technology, vol. 13, pp. 245-252, 2015.
[20] F.Armknechtet. al., ”A Guide to Fully Homomorphic Encryption”, IACR Cryptology ePrint Archive, 2015.
[21] A. Acar, H. Aksu, A. S. Uluagac and M. Conti. “A Survey on Homomorphic Encryption Schemes,” ACM Computing Surveys, vol. 51, no. 4, pp. 1–35, 2018. doi:10.1145/3214303
[22] M. Fellows and N. Koblitz,“Combinatorial cryptosystems galore!,”2nd International conference, Finite fields: theory, applications, and algorithms, pp. 51-62, 1993.
[23] D.Boneh, E.Goh, and K.Nissim,“Evaluating 2-DNF formulas on ciphertexts,” In Theory of cryptography,pp. 325–341, 2005.
[24] M. V.Dijk, C. Gentry, S.Halevi, and V.Vaikuntanathan,“Fully homomorphic encryption over the integers,” In Advances in cryptology–EUROCRYPT 2010. pp. 24–43, 2010.
[25] J.Hoffstein, J.Pipher and J. H. Silverman,“NTRU: A ring-based public key cryptosystem,” In Algorithmic number theory, pp. 267–288, 1998.
[26] C. Gentry,“A fully homomorphic encryption scheme,” Ph.D. Dissertation, Stanford University, 2009.
[27] O.Regev, “On lattices, learning with errors, random linear codes, and cryptography,”Proceedings of the thirty-seventh annual ACM symposium on Theory of computing,pp. 84-93, 2009. doi:10.1145/1060590.1060603
[28] C. Gentry, ”Computing Arbitrary Functions of Encrypted Data,” Communications of the ACM, vol. 53 no. 3, pp. 97-105, 2010. doi:10.1145/1666420.1666444
[29] J. Li, D. Song, S. Chen and X. Lu, “A simple fully homomorphic encryption scheme available in cloud computing,” IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, pp. 214-217, 2012. doi: 10.1109/CCIS.2012.6664399