ORIGINAL RESEARCH ARTICLE | UHD JOURNAL OF SCIENCE AND TECHNOLOGY

A Proposed Fully Homomorphic for Securing Cloud Banking Data at Rest

Zana Thalage Omar[1,2]*, Fadhil Salman abed[1,2]

[1]University of Human Development, College of Science and Technology, Department of Computer Science, Sulaymaniyah, Kurdistan Region of Iraq, Iraq, [2]University of Sulaimani, College of Science, Computer Department, Sulaymaniyah, Kurdistan Region of Iraq, Iraq

ABSTRACT

Fully homomorphic encryption (FHE) reaped the importance and amazement of most researchers and followers in data encryption issues, as programs are allowed to perform arithmetic operations on encrypted data without decrypting it and obtain results similar to the effects of arithmetic operations on unencrypted data. The first (FHE) model was introduced by Craig Gentry in 2009, and it was just theoretical research, but later significant progress was made on it, this research offers FHE system based on directly of factoring big prime numbers which consider open problem now, The proposed scheme offers a fully homomorphic system for data encryption and stores it in encrypted form on the cloud based on a new algorithm that has been tried on a local cloud and compared with two previous encryption systems (RSA and Paillier) and shows us that this algorithm reduces the time of encryption and decryption by 5 times compared to other systems.

Index Terms: Cloud Computing Security, Encryption, Decryption, Cloud Storage, Homomorphic Encryption

1. INTRODUCTION

Computing technology is seeing significant progress and significant interest, especially when the computation outsourcing has been outsourced to a third party as the cloud is the most frequently used form [1]. That is why many companies no longer trust to store their sensitive data in the cloud, which uses traditional unsecured encryption systems [2]. From this, the need to use homomorphic encryption for banking data is coming, which is a new approach that can help banks to increase data security and management [3]. There are two types of homomorphic cryptosystems: Partially homomorphic systems and fully homomorphic systems [4]. Partially homomorphic schemes support one of the additions or multiplication operations, these systems are divided into two parts according to the process that supports like the RSA, where it only supports the multiplication process and does not support the addition process, for example, if we have two numbers M1, M2 and they are encrypted by the RSA, then its value becomes C1, C2 and on obtaining the product of multiplying the two encrypted values C1 * C2 = C3 and then we decrypt the encrypted output C3, we will get a result similar to M1 * M2 = M3, but if we add the two values C1 + C2 = C4 and when decrypting the result C4 we do not get a result similar to M1 + M2 = M4. On the contrary, when the two values are encrypted using Paillier, we find that only the result of C1+C2=C5 is similar to M1+M2=M3 and C1*C2=C6 do not equal to M1+M2=M4. Therefore, we say that the two algorithms (RSA and Paillier) are not a fully homomorphic systems [5], [6]. The first FHE was given in 2009 by Craig Gentry [7]. Researchers first researched a (FHE) system in the late last century, specifically at the end of the seventies, and soon after, in 1987, RSA was published, the RSA algorithm became a leading approach by many researchers because at that time there was no idea of the public key cipher

Access this article online

DOI: 10.21928/uhdjst.v4n1y2020.pp87-95	E-ISSN: 2521-4217 P-ISSN: 2521-4209

Corresponding author's e-mail: Zana Thalage Omar, University of Human Development, College of Science and Technology, Department of Computer Science, Sulaymaniyah, Kurdistan Region of Iraq, Iraq, University of Sulaimani, College of Science, Computer Department, Sulaymaniyah, Kurdistan Region of Iraq, Iraq. E-mail: zana.th.omar@gmail.com

Received: 13-03-2020 Accepted: 10-05-2020 Published: 12-05-2020

that was presented during the RSA scheme for the first time [5]. Because this kind of encryption allows the key to decrypt the encrypted data, and thus one can read and know all the data, and for this reason, if one does not have the secret key, the data become useless. Therefore, a question and an issue were asked: Can mathematical operations apply to encrypted data without decrypting it, and from this, the idea of using fully homomorphic systems (FHE) was raised. After that, several attempts were made to develop these systems, but most of the research did not succeed as they received partially homomorphic schemes such as RSA and Goldwasser-Micali [8]. The algorithm that achieves the addition and multiplication properties can be considered as FHE, as it is regarded as a special algorithm that contains the feature of performing mathematical operations (addition and multiplication) on data without decrypting it and obtaining correct results [9]. FHE is an encryption technology that allows calculations to be performed on encrypted data without decrypting it, and this results in an encrypted result where when this result is decrypted we get a result similar to the result of the calculations on the data without encrypting it [9]. The world of computing is in constant progress, and the main challenge is to create a guarantee and trust among customers when storing their sensitive data on the cloud to ensure and respect their privacy. This is a new approach that cloud providers follow to encrypt users' data, upload it to the cloud, and perform operations on it without decrypting it to ensure the integrity of customer data [10]. This paper presents a fully homomorphic system (the correct numbers and texts) based on a new algorithm that will be explained later in this paper as this scheme relies on data encryption and operations performed on it without decrypting and reducing computational complications and the time used to encrypt and decrypt data and reduce energy consumption. Most of the previous research in this field deals with data when encrypting after converting it to the binary system and this means more time. As for our current research, data operations are encrypted without the need to convert them to binary representation and this reduces mathematical operations and there is a reduction in the time of encryption and encryption solution, as well as a mathematical model has been suggested that deals with the inverse calculation and the process of raising to the exponential and increases the complexity of attacking the new system.

2. LITERATURE REVIEW

C. Gentry *et al.* (2012), this paper introduces contrast/orientation techniques to transfer the elements of plain text across these vectors very efficiently so that they are able to perform general calculations in a batch way without the need to decrypt the text and also make some improvements that can accelerate the normal homomorphic, where you can make homogeneous evaluation of arithmetic operations using multi-arithmetic head only [11].

J. Fan *et al.* (2012), this paper concludes two copies of the redefinition that lead to a quick calculation of homogeneous processes using the parameter transformation trick, as this paper conveys Brakerski's fully homomorphic scheme based on the learning with errors (LWE) problem to the ring-LWE [12].

Z. Brakerski *et al.* (2012), this paper introduces squash and bootstrapping techniques to convert a somewhat symmetric encryption scheme into an integrated symmetric encryption scheme [13].

X. Cao *et al.* (2014), this paper presents a completely symmetric encryption scheme using only a basic unit calculation as it relies on the technique of using multiplication and addition instead of using ideal clamps on a polynomial loop [14].

C. Xiang *et al.* (2014), this paper presents an entirely symmetric encryption scheme on integers, as it reduces the size of the public key using the square model encryption method in public key elements instead of using a linear model based on a stronger variant of the approximate-GCD problem [15].

M. M. Potey *et al.* (2016), this paper presents a completely symmetric encryption system where it focuses on storing customer data on the cloud in an encrypted form so that customer data remain safe and when any data modification is made the system loads data on the customer's device and modifies it and then stores it again on the cloud in encrypted form [16].

K. Gai *et al.* (2018), this paper proposes a new solution for mixing real numbers on a novel tensor-based FHE solution that uses tensor laws to reduce the risk of unencrypted data storage [17].

S. S. Hamad *et al.* (2018), these heirs offer a completely symmetric encryption system, as it relies on the principle of encryption a number from the plain text with another number using a secret key without converting to binary format and then comparing the result with a DGHV and SDC system [18].

S. S. Hamad *et al* (2018), this paper presents a fully homomorphic encryption system based on Euler's theory and the time complexity has been calculated and compared with other systems with an encrypt key size up to 512 bits while the size of the key in our proposed scheme reaches more than 2048 bits and the encrypting process is done through more complex and powerful mathematical equations [19].

V. Kumar *et al* (2018), this paper presents fully homomorphic encryption system with probabilistic encrypting and relies on Euler's theory. The encrypting process is done through the following mathematical equation ($C=M^{k*\mu(n)+1} \bmod x$) while in our proposed scheme a more complex and difficult mathematical algorithm is used which helps to stand more against hacker attacks and deter them [20].

R. F. Hassan *et al.* (2019), this paper proposes a blueprint for building asymmetric cloud-based architecture to save user data in the form of unusual text. This pattern uses the elliptic curve to create the secret key for data encryption. This pattern is a new algorithm that reduces processing time and storage space [21].

3. STATEMENT OF THE PROBLEM

Cloud providers provide many services, including applications and storage many companies and users do not trust the providers of these services due to security concerns. Where the user does not upload his personal data to the cloud because the cloud providers are able to read and modify every bit loaded on the cloud and use it for personal purposes, and this thing does not comply with respecting the user's privacy. Furthermore, some cloud providers still use traditional security techniques that are not secure with low-security level to protect user privacy. Some of the cloud providers have started to use high-level technologies to protect the privacy of users and the security of their data, but there remains a problem that the provider of the cloud itself is still able to access user data, and this is not safe for users. This problem can be solved when following FHE systems when storing data on the cloud where these systems can encrypt the data and store it in the cloud in an encrypted form and thus the cloud provider or others cannot see the data and use it, so the privacy of users and the security of their data are protected.

4. PROPOSED FHE SYSTEM

The proposed scheme works as follows:
Generating the encryption key and then encrypting the numbers and texts and storing them in encrypted form on the cloud. In our work, we use a local cloud and experiment with the proposed scheme on it. The purpose of this process is to save the data encrypted on the cloud so that no one can view the data and use it for personal purposes Therefore, when the data owner needs to perform an amendment of the encrypted data on the cloud, an encrypted request is sent to the server, and the server performs mathematical operations on the encrypted data and returns an encrypted result where this encrypted result can only be decrypted through the private encryption key which is with the owner of data only so that he can decrypt the encrypted result and see his data. In this way, we have been able to maintain the privacy and security of the data when stored in the cloud. These procedures go through three stages. Generation the encryption key stage, the encryption stage, and the decryption stage. The model of the proposed scheme is given in Fig. 1, and the flowchart of the proposed scheme is given in Fig. 2. The proposed scheme performs several random examples with multiplication and addition as follows:

A. Key Generation:
1. Generate two large Prime number p and q
2. Compute $n = p*q$
3. Calculate $L=((P^{-1} \bmod q)*p)+((q^{-1} \bmod p)*q)$
4. Select r: Where r is a big random integer

B. B. Messages Encryption

The conditions:

$$(M_1\&M_2),\ (M_1+M_2),\ \text{and}\ (M_1*M_2) < n$$

Where M1 and M_2 are the Messages.

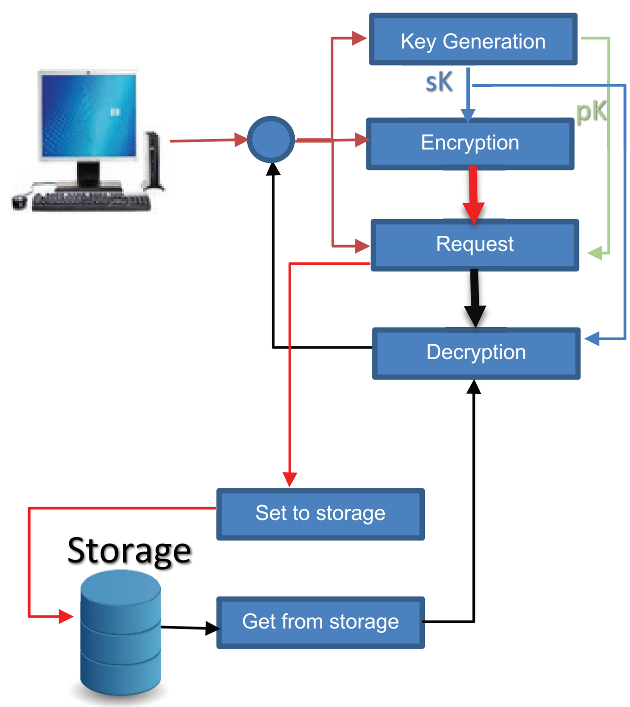

Fig. 1. Model of proposed FHE scheme.

The schema of message encryption is:

$$C = L * M^{r\,\mu(p)+1} \bmod n. \quad (1)$$

Where μ (p) = (p-1), Euler function and C, the ciphers text.

C. Message Decryption

The schema of cipher decryption is:

$$M=C \bmod p \quad (2)$$

Where M is the number or text that will be encrypt and C is the result of the encrypted number or text we named it cipher text

D. Euler's Theorem

All of us know that Euler's Theorem contains two-part they are:

1. $M^{\mu(p)} \equiv 1 (\bmod p)$, when p and m are prime to each other.
2. $M^{r*\mu(n)+1} \equiv M (\bmod n)$, when r is an integer, M<n and n=p*q where q and p are two primes number.

E. A simple example of how to make an amendment to encrypted data

We have two values M1 = 3, M2 = 5. We encrypt them through a simple encryption equation that is multiplied by each value, so we get C1 = M1 * 2 and C2 = M2 * 2, so C1 = 6, C2 = 10 when we add the two values C1 + C2 = C3 so C3 = 16 We decrypt C3 so we get the result 16/2 = 8 which is the same result when we add M1 + M2 = M3 where 3 + 5 = 8 as shown in Fig. 3.

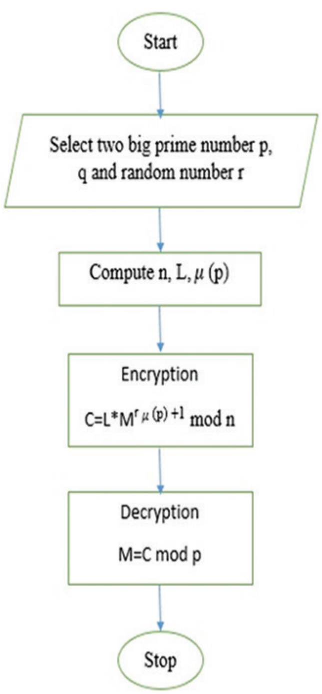

Fig. 2. Flowchart of proposed FHE scheme.

5. THE PROVE OF OUR SCHEMA IS FHE

We choose two numbers M_1, M_2 and encrypt them to get two encrypted or (ciphers) C_1 and C_2, respectively, and then we combine $C_1 + C_2$ to get a new ciphered result we name it C_3 then we decrypt C_3 and compare the result with M_3 which is the result of combine $M_1 + M_2$ we also multiply $C_1 * C_2$ to get C_4 and compare it to M_4, which is the result of $M_1 * M_2$.

A. The Prove of Additive Homomorphic

If the following condition is fulfilled, it becomes clear to us that the proposed scheme additive homomorphic:

$$M_1+M_2=\text{dec}\,[\text{enc}\,(M_1) + \text{enc}\,(M_2)] \quad (4)$$

Where dec is the decryption function and enc is the encryption function

Proof:

$C_1=L*(M_1^{r\,\mu(p)+1} \bmod n)$.
$C_2=L*(M_2^{r\,\mu(p)+1} \bmod n)$.
$C_1+C_2= L*(M_1^{r\,\mu(p)+1} \bmod n) + L*(M_2^{r\,\mu(p)+1} \bmod n)$.
$\text{dec}\,(C_1+C_2) = (C_1+C_2) \bmod p$
$= [L*(M_1^{r\,\mu(p)+1} \bmod n) + L*(M_2^{r\,\mu(p)+1} \bmod n)] \bmod p$
$= [(L \bmod p) + ((M_1^{r\,\mu(p)+1} \bmod n) \bmod p) + (L \bmod p) + ((M_2^{r\,\mu(p)+1} \bmod n) \bmod p)]$
$= [(M_1^{r\,\mu(p)+1} \bmod p) \bmod n + (M_2^{r\,\mu(p)+1} \bmod p) \bmod n]$

We know $M_1^{r\,\mu(p)+1} \bmod p = M_1$ and $M_2^{r\,\mu(p)+1} \bmod p = M_2$ by Euler's Theorem so

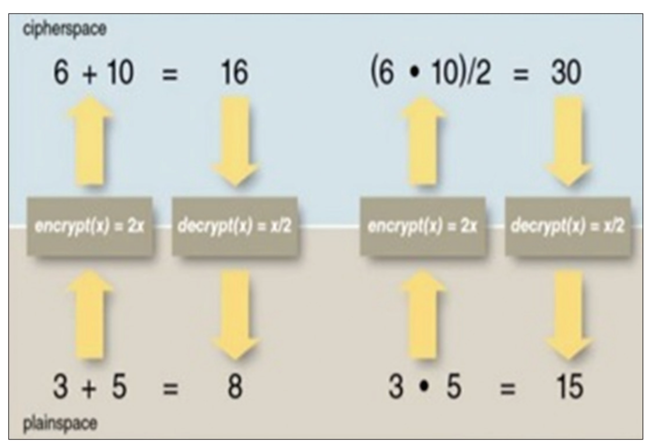

Fig. 3. A simple example of how modify encrypted data.

$= (M_1 \bmod n) + (M_2 \bmod n)$
$= (M_1+M_2) \bmod n$
Because M_1+M_2 less than $< (n)$
$= M_1+M_2$
$dec\ (C_1+C_2) = M_1+M_2$ so the condition is fulfilled

B. The Prove of Multiplicative Homomorphic

If the following condition is fulfilled, it becomes clear to us that the proposed scheme multiplicative homomorphic:

$$M_1*M_2=dec\ [enc\ (M_1) * enc\ (M_2)] \tag{5}$$

Where dec is the decryption function and enc is the encryption function

Proof:
$C_1=L*(M_1^{r\mu(p)+1} \bmod n).$
$C_2=L*(M_2^{r\mu(p)+1} \bmod n).$
$C_1*C_2= (L*(M_1^{r\mu(p)+1} \bmod n)) * (L*(M_2^{r\mu(p)+1} \bmod n)).$
$dec\ (C_1*C_2) = (C_1*C_2) \bmod p$
$= [(L*(M_1^{r\mu(p)+1} \bmod n)) * (L*(M_2^{r\mu(p)+1} \bmod n))] \bmod p$
$= [(L \bmod p) * ((M_1^{r\mu(p)+1} \bmod n) \bmod p) * (L \bmod p) * ((M_2^{r\mu(p)+1} \bmod n) \bmod p)]$
$= [(M_1^{r\mu(p)+1} \bmod p) \bmod n * (M_2^{r\mu(p)+1} \bmod p) \bmod n]$
We know $M_1^{r\mu(p)+1} \bmod p = M_1$ and $M_2^{r\mu(p)+1} \bmod p = M_2$ by Euler's Theorem so
$= (M_1 \bmod n) * (M_2 \bmod n)$
$= (M_1*M_2) \bmod n$
Because M_1*M_2 less than $< (n)$
$= M_1*M_2$
$dec\ (C_1*C_2) = M_1*M_2$ so the condition is fulfilled

6. REAL EXAMPLE

Let us choose two different number $M_1= 10$, M2 = 40, select two big prime numbers p=523, q=617, select random number r=100 and compute n, L where n = p*q and L = $((P^{-1} \bmod q)*p) + ((q^{-1} \bmod p)*q)$, as in Fig. 1, so n = 322691 and L = 322692 now we will compute C_1, C_2 as shown in Fig. 4 where

$C_1= L*(M_1^{r\mu(p)+1} \bmod n)$
$C_1= 322692 * (10^{100*(522)+1} \bmod 322691)$
$C_1= 84555952836$
$C_2= L*(M2^{r\mu(p)+1} \bmod n)$
$C_2= 322692 * (40^{100*(522)+1} \bmod 322691)$
$C_2= 70220360736$

A. Check the Additive Homomorphism

As shown in Fig. 5, Let us define C_3 is the result of C_1+C_2

$C_3= C_1+C_2$
$C_3= 84555952836+70220360736$
$C_3= 154776313572$
$M_3= C_3 \bmod p$
$M_3= 154776313572 \bmod 523$
$M_3= 50$, which is the same of $M_1+M_2 = 10 + 40 =50$

B. Check the Multiplication Homomorphism

As shown in Fig. 6, Let us define C_4 is the result of C_1*C_2
$C_4= C_1*C_2$
$C_4= 84555952836*70220360736$
$C_4= 5937549510520122247296$
$M_4= C_4 \bmod p$
$M_4= 5937549510520122247296 \bmod 523$
$M_4= 50$, which is the same of $M_1*M_2 = 10 * 40 =400$

7. RESULTS

Our proposed method has been applied in Java Language on a laptop that has these characteristics Intel (R) core (TM)

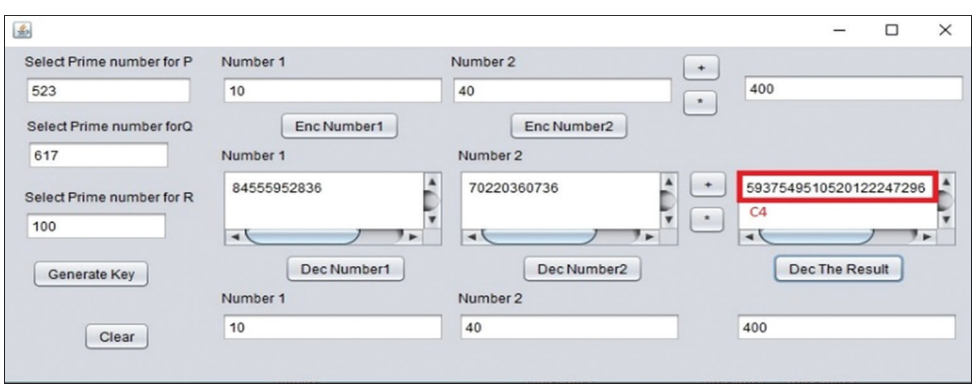

Fig. 4. Verification that the proposed scheme supports multiplicative homomorphic system.

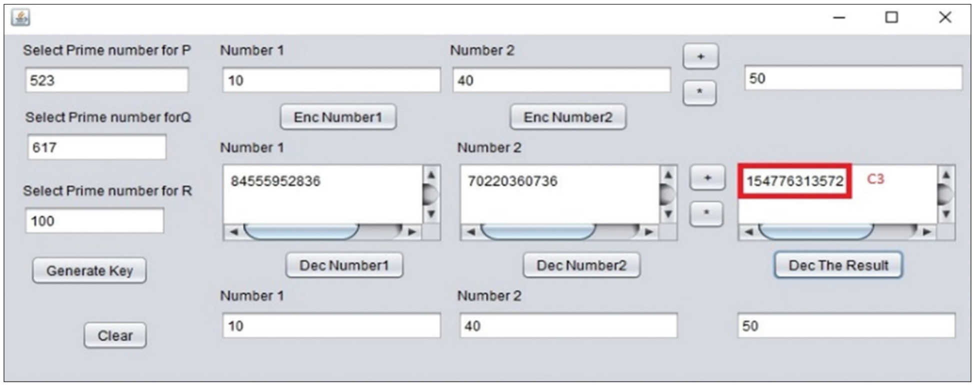

Fig. 5. Verification that the proposed scheme supports additive homomorphic system.

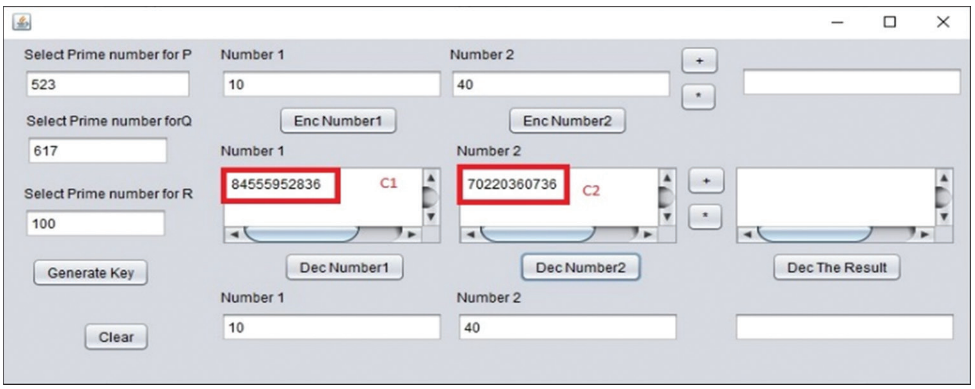

Fig. 6. A real example of generating an encryption key and encrypting two different numbers.

i7-8550U CPU @1.80GHz 2.00GHz, 8 GB Ram, 64-bit Operating System,x64-based processor, Windows 10 and Big Integer library of java is used. We have previously seen that in section 5 our scheme achieves the two properties (addition and multiplication) on the correct numbers when encrypt in contrast to the two systems (RSA and Pailler) that produce a single property either multiplication or addition, we took as an example of 2048 bit A message containing several languages: English, Kurdish, Arabic and Chinese, to indicate that our scheme works in all languages. The message was:

The language considered at the university is English

ئەو زمانەی كە لە زانكۆدا بە پەسەندە زمانی ئینگلیزییە

اللغة المعتبرة في الجامعة هي الانكليزية

大學考慮的語言是英語

P=16963600001744112018845147215300525136637986969
16067418627001848350512356831638156323463645139635
08417868627233690910742758025297248862671313854081
10011485075986472668526287650474047702785552627
71396058882448569487514849713569452550255019774552
24245023795329108774824924351450367586579265699393
7298380518114511

Q=142871047674167505698301457557184056182343464543
08656139478728164897664839386811178089812726550
57083043177998260407080109195135906300003355141230
45027277642869191854226880441763828684416539791520
25714562637864986328594073222682892055143305181
26876554660629020396776578921479709409043082934474
43717248390239707

R=365786518636912644776042065063644900157123708
34320716872902633782216844351341184895528113792015
87981908122293385001246745347846500741948845152271
827981943

N=24236073045746910110572201683394297457755108768
56825573501591215207476748560343071696011805583892
43234428533681091865082478949426971733704964018
75762586863569789414738414794462518054251740833 99

0064335766935428553190201256143433654205627755229
6773381542471828455818813480172653876483980783338
1523057539397742755088141082360135822895062302531
9405062251415063552873019444449238666440140085803
2829153319755489679960430558612883401366594381416
5468112883656495673094721811758386521739451237520
5070768701405826931878983152614067930454176175622
2924904444160392437762620644204922911348434700560
072718252562650911031994574848 57

L=2423607304574691011057220168339429745775510876856825573501591215207476748560343071696011805583892432344285336810918650824789494269717337049640187576258686356978941473841479446251805425174083399006433576693542855319020125614343365420562775522967733815424718284558188134801726538764839807833381523057539397742755088141082360135822895062302531940506225141506355287301944444923866644014008580328291533197554896799604305586128834013665943814165468112883656495673094721811758386521739451237520507076870140582693187898315261406793045417617562229249044441603924377626206442049229113484347005600727182525626509110319945748 4858

The Message after encryption (Cipher Text)

5414765588094097023913378967862065788913713058606910830470756667353053430101331634552013306003298182248768286499539195275662377358315787474955186618150060150943273735993240581405016376825239813660812634444029538789582250041028814049872452140851921304639686231620361327142189883458667338828289030279594385776771938589562521268936022433220023458229979036307501828080603296937268909738214290520221470582643052952450970177540992694753809680462018541391816247983013734786006845363919941350425392173047922834259284294384054149431149567318796039500765387170939679389180974764733554252834282574172152676629672180641049605636362181830441111512121224578713415756751582749861669965260065789688204024656012125845119782942985142685541255549956033755261323225746331454723599082347201330811438811210005203796747401988173414177618608268726913258172107683067656002371046588261012408315631146494925672581002557889746744145480628 25

TABLE 1: Computation encryption time of various schema

Key Size	Proposed method	RSA	Pailler
64 Bit	88 ms	59 ms	103 ms
128 Bit	139 ms	100 ms	182 ms
256 Bit	218 ms	149 ms	727 ms
512 Bit	1141 ms	397 ms	4212 ms
1024 Bit	6058 ms	2185 ms	55139 ms
2048 Bit	65876 ms	29820 ms	263303 ms
Average	12253 ms	5451 ms	53944 ms

TABLE 2: Computation decryption time of various schema

Key Size	Proposed Method	RSA	Pailler
64 Bit	31 ms	72 ms	193 ms
128 Bit	43 ms	116 ms	330 ms
256 Bit	50 ms	315 ms	1429 ms
512 Bit	60 ms	1450 ms	11441 ms
1024 Bit	131 ms	7829 ms	122628 ms
2048 Bit	260 ms	79609 ms	976289 ms
Average	95 ms	14898 ms	185385 ms

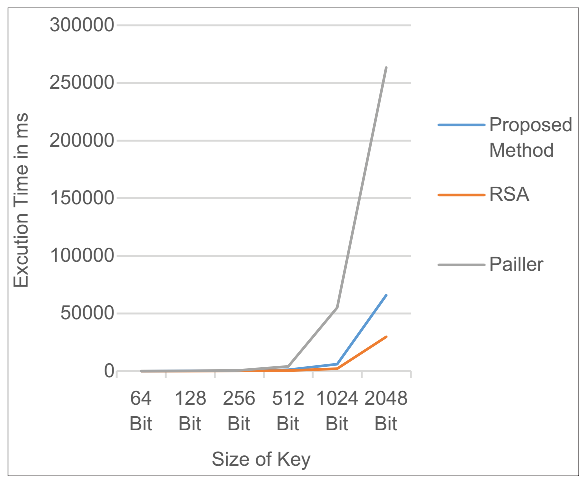

Fig. 7. Computation encryption time of various schema.

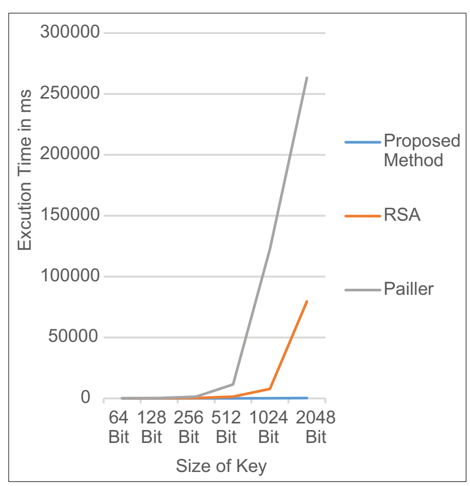

Fig. 8. Computation decryption time of various schema.

2892837234662839061292113517475051240283024822857039832150288766224552917207474813575344328669795611610517942144953845555647644200227716739046080269214238010929862344527946869053956142504282153978829913279741016316926632254422850165388901441231774699848323143262753973952981234912026166080929309059343417238287374915366024637186651090446025995427032842858325561760100762479528333312842498083186883479202672771030666642010819171038712008353233977703555339349166794021508312097161373132603109611666109415743990715529774034556154583520372539433462549143084673593882815487336243261126911242981325891250006136188595483928019489540285506606523583489298137160845149207589839264683606383242087579161421012774684088722206157675920392220322437888837467761391646974013621593727999527387894145533554657005640988111761561242777691841460412436817297935149248403437793923291041969716726718988314898193850389144973734527764417056337441280540889889965231589793043301722177856967321141588234755398782709859264037093772068861826447323185329396490549555672427762431169794565317121037175050358312647042605790539753324457714637549871900468942240262274576522420220671065577816480533078578928195481985808140541026441726779572492306966870609990207….etc. Due to the length of the encrypted text (Cipher Text), which reaches 67 pages, it has been truncated. Where the encrypted time was 1572 ms, and the decrypted time was 31 ms. We have also tested it on text with 8KB in its size and several different keys in terms of size, and we compared the results with the planners from. In terms of velocity, we obtained the following results: As shown in Tables 1 and 2, respectively, and the graph in Figs 7 and 8.

8. CONCLUSION

Our scheme relies on FHE on whole numbers, texts, and supports all languages such as English, Arabic, Kurdi, and Chinese and others. Very large prime numbers (up to 617 digits, 2048 bit) represent the strength for the attack of our scheme because the proposed system depends on the problem of factorization to the primary factors, which are considered mathematical problems under discussion at the present time when taking the time. We have come to the conclusion that our scheme is very effective in relation to the time when encrypt and decrypt numbers and texts in comparison with other techniques and approaches that are circulated and used at the present time.

REFERENCES

[1] L. A. Tawalbeh and G. Saldamli. "Reconsidering big data security and privacy in cloud and mobile cloud systems". *Journal of King Saud University Computer*, vol. 40, pp. 1-7, 2019.

[2] J. Domingo-Ferrer, O. Farràs, J. Ribes-González and D. Sánchez. "Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges". *Computer and Communications*, vol. 140-141, no. 2018, pp. 38-60, 2019.

[3] S. Sakharkar, S. Karnuke, S. Doifode and V. Deshmukh. "A research homomorphic encryption scheme to secure data mining in cloud computing for banking system". *International Journal for Innovative Research in Multidisciplinary Field*, vol. 4, no. 4, pp. 276-280, 2018.

[4] J. H. Cheon, A. Kim, M. Kim and Y. Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *Lecture Notes in Computer Science*. Vol. 10624. Springer Science+Business Media, Berlin, Germany, pp. 409-437, 2017.

[5] P. Sha and Z. Zhu. "The modification of RSA algorithm to adapt fully homomorphic encryption algorithm in cloud computing". In: *Proceeding 2016 4th IEEE International Conference Cloud Computing and Intelligence Systems*. pp. 388-392, 2016.

[6] L. Chen and Z. Zhang. "*Bootstrapping Fully Homomorphic Encryption with Ring Plaintexts Within Polynomial Noise*. Vol. 2. Conference Paper, pp. 285-304, 2017.

[7] C. Gentry. "*A Fully Homomorphic Encryption Scheme*". Dissertation, p. 169, 2009.

[8] V. Kumar and N. Srivastava. "Chinese Remainder Theorem based Fully Homomorphic Encryption over Integers". *International Journal of Applied Engineering Research*, vol. 14, no. 2, pp. 203-208, 2019.

[9] 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. 1, pp. 347-361, 2019.

[10] K. Gai, M. Qiu, Y. Li and X. Y. Liu. "Advanced fully homomorphic encryption scheme over real numbers". In: *Proceeding 4th IEEE International Conference Cyber Secur Cloud Computing CSCloud 2017 3rd IEEE Intertnational Conference Scalable Smart Cloud, SSC 2017*, pp. 64-69, 2017.

[11] C. Gentry, S. Halevi and N. P. Smart. "Fully homomorphic encryption with polylog overhead". In: *Lecture Notes in Computer Science*. Vol. 7237. Springer Science+Business Media, Berlin, Germany, pp. 465-482, 2012.

[12] J. Fan and F. Vercauteren. "Somewhat practical fully homomorphic encryption". In: *Proceeding 15th International Conference Practice Theory Public Key Cryptogr*, pp. 1-16, 2012.

[13] Z. Brakerski and V. Vaikuntanathan. "Fully homomorphic encryption from ring-LWE and security for key dependent messages". In: *Lecture Notes in Computer Science*. Vol. 6841. Springer, Berlin, Germany, pp. 505-524, 2011.

[14] X. Cao, C. Moore, M. O'Neill, N. Hanley and E. O'Sullivan. "High-speed fully homomorphic encryption over the integers". In: *Lecture Notes in Computer Science*. Vol. 8438. Springer, Berlin, Germany, pp. 169-180, 2014.

[15] C. Xiang and C. M. Tang. "Improved fully homomorphic encryption over the integers with shorter public keys". *International Journal of Security and its Applications*, vol. 8, no. 6, pp. 365-374, 2014.

[16] M. M. Potey, C. A. Dhote and D. H. Sharma. "Homomorphic encryption for security of cloud data". *Procedia Computer Science*,

vol. 79, pp. 175-181, 2016.

[17] K. Gai and M. Qiu. “Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers”. *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3590-3598, 2018.

[18] 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.

[19] S. S. Hamad and A. M. Sagheer. “Fully homomorphic encryption based on Euler’s theorem”. *The International Journal of Information Security*, vol. 9, no. 3, p. 83, 2018.

[20] V. Kumar, R. Kumar, S. K. Pandey and M. Alam. “Fully homomorphic encryption scheme with probabilistic encryption based on euler’s theorem and application in cloud computing”. In: *Advances in Intelligent Systems and Computing*. Vol. 654. Springer, Berlin, Germany, pp. 605-611, 2018.

[21] R. F. Hassan and A. M. Sagheer. “A proposed secure cloud environment based on homomorphic encryption”. *International Advanced Research Journal in Science, Engineering and Technology*, vol. 6, no. 5, pp. 166-175, 2019.