 http://dx.doi.org/10.14500/aro.10590

Cloud Storage Protection Scheme Based on Fully Homomorphic Encryption

Mohammed A. Mohammed[1] and Fadhil S. Abed[2]

[1]Department of Computer Science, College of Science, University of Sulaimani, Sulaymaniyah, Kurdistan Region – F.R. Iraq

[2]Department of Information Technology, Kalar Technical Institute, Sulaimani Polytechnic University, Khanaqeen, Kurdistan Region – F.R. Iraq

***Abstract*— Cloud computing allows enterprises and individuals to have a less physical infrastructure of software and hardware. Nevertheless, there are some concerns regarding privacy protection which may turn out to be a strong barrier. Traditional encryption schemes have been used to encrypt the data before sending them to the cloud. However, the private key has to be provided to the server before any calculations on the data. To solve this security problem, this paper proposes a fully homomorphic encryption scheme for securing cloud data at rest. The scheme is based on prime modular operation, its security depends on factoring multiple large prime numbers ($p_1, p_2,...p_n$) up to n, which is formed from very large prime numbers up to hundreds of digits as this is an open problem in mathematics. In addition, the elements of the secret key are derived from a series of mathematical operations and the calculation of an Euler coefficient within the modular of integers. Furthermore, it adds the complexity of noise to the plaintext using the number of users of the Cloud Service Provider. Moreover, its randomness is evaluated by the National Institute of Standards and Technology statistical tests, and the results demonstrating that the best statistical performance was obtained with this algorithm.**

***Index Terms*— Cloud Computing Security, Cryptography, Fully Homomorphic Encryption, Information Security.**

I. Introduction

Cloud computing plays an important role in storing and processing huge amounts of data since the fast progress of computer networks and big data (Hashem, et al. 2015). It provides flexible and on-demand remote storage and computing capabilities to its users. Nevertheless, as Gonzales et al. (2017) stated that cloud computing is not fully trustable since its users do not have full control over their data. Privacy protection and data leakage are the main risks for individuals and enterprises when it comes to migrating their data to cloud storage. The encryption techniques that require encrypted data on the cloud to be decrypted before performing any computation is still portend the privacy of stored data. Whereas, in Homomorphic Encryption operations can be performed directly on encrypted data without decrypting it. In addition, the result of the operation on encrypted data is equivalent to the result of its corresponding plaintext operation. This paper attempts to add an extra value to the privacy protection of cloud's data through proposing a new FHE scheme based on prime modular operation, which security depends on factoring multiple large prime numbers ($p_1, p_2,...,p_n$) up to n, which is formed from very large prime numbers up to hundreds of digits as this is an open problem in mathematics. Moreover, the randomness of the proposed work is evaluated by the well-known National Institute of Standards and Technology (NIST) test suite, which is widely used as a standard battery of tests to test randomness. The results of the proposed algorithm in the NIST statistical tests show that it produces the best statistical performance through passing all the tests.

II. Problem Statement

Nowadays, individuals and enterprises are seeking to access their private information anytime and anywhere. This leads them to deploy it onto cloud storage. However, they will be facing an extra amount of risks, which makes it challenging to maintain the security of outsourced data such as confidentiality, integrity, authentication, and privacy. For example, the hacking attack on PlayStation network in 2011 led it to leak millions of user accounts' passwords, physical addresses, credit card information, and other personal information. Later, the company stated that they could have encrypted the data on their network (Sangani, 2011). In addition, as reported by the Identity Theft Resource Center on May 31, 2018, thousands of FedEx customer records were exposed due to an unsecured server; some of the documents were passports, driving licenses, and security IDs (CyberScout, 2018). Therefore, Cloud Service Providers (CSPs) are required to keep an encrypted version of user's information on their storage. There is a variety of different techniques used for data encryption. Nevertheless, as the

ARO-The Scientific Journal of Koya University
Vol. VIII, No.2 (2020), Article ID: ARO.10590, 08 pages
DOI:10.14500/aro.10590
Received: 14 November 2019; Accepted: 16 November 2020
Regular research paper: Published: 06 December 2020
Corresponding author's e-mail: mohammed.anwar@univsul.edu.iq

data resides on the cloud storage, it required to be decrypted before performing any operation on the data. This might cause privacy and confidentiality problems to the stored data. Whereas, homomorphic encryption allows performing computations on the encrypted data without decrypting it. Thus, HE solves the problems of confidentiality and privacy of the stored data inside the cloud. Therefore, this paper presents a new FHE scheme based on multiple large prime modular operation which is formed from very large prime numbers up to hundreds of digits. Hence, it makes the secret key very complicated which is difficult to retrieve it and resistance to different types of attacks.

III. Literature Review

Rivest et al. (1978) were proposed the first homomorphic encryption scheme and were partially homomorphic encryption (PHE). Then, Yao (1982) was also presented a PHE scheme. After that, RSA which was a multiplicative homomorphism introduced by Rivest et al. (1983). Afterward, several authors such as Goldwasser and Micali (1984), Elgamal (1985), and Paillier (1999) were also presented their PHE scheme. Subsequently, a fully homomorphic encryption (FHE) scheme suggested by Gentry (2009), which allows calculating of any number of addition and multiplication, hence compute arbitrary functions of encrypted data. Nevertheless, the scheme was based on Somewhat Homomorphic Encryption (SWHE), which increases the length and noise of ciphertext when calculation performs on the ciphertext. Consequently, van Dijk et al. (2010) have introduced FHE scheme that used elementary modular arithmetic and used Gentry's techniques to convert SWHE cryptosystem to FHE scheme. In addition, Smart and Vercauteren (2010) have presented an improved version of Smart-Vercauteren encryption scheme, the scheme was allowed several times decrease the ciphertext and keys lengths. In addition, IBM has released a software package named HElib in 2013, the company has implemented HE with further optimizations (Cheon, et al. 2019). Moreover, a HE scheme which is security dependent on the hardness of large integer factorization has been proposed by Xiao et al. (2012). Afterward, homomorphic encryption scheme has been worked on and improved by numerous authors, they have also tested it in a cloud computing system. Alattas and Elleithy (2013) have presented the application of algebraic homomorphic encryption mechanism and it was aiming at enhancing its security. In addition, several HE schemes such as RSA, Paillier, El-Gamal, and Gentry have been examined on a cloud computing environment by Tebaa and El Hajii (2014). In addition, Hayward and Chiang (2015) have improved Gentry's encryption in parallel processing and they have tested it in a private cloud domain. Furthermore, structured and simplified definitions in the homomorphic encryption discipline have been proposed by Armknecht et al. (2015). Moreover, SAM which is an FHE scheme over integers has been implemented by Shihab and Makki (2018). Furthermore, Li et al. (2016) 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, Wang et al. (2018) 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 a Euclidean algorithm to address the GCD problem of the first part of the secret key. Whereas, in 2019 (Li et al.) used a lookup table to propose a protocol to evaluate any function using FHE.

Moreover, Ji and Shieh (2019) 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. In late 2019, the authors of Jubrin et al. 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. Furthermore, Mohammed and Abed (2019) 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. Mert et al. (2020) presented two hardware architectures optimized for accelerating the encryption and decryption operations of the BFV/HE scheme with high-performance polynomial multipliers. In addition, in 2020, Tan et al. 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. Whereas, Mohammed and Abed (2020) 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.

Despite all the works presented previously, the randomness and robustness of the secret keys remain an open problem in the area of FHE. Therefore, this paper presents a new algorithm in which the elements of the secret key are derived from a series of mathematical operations and the calculation of an Euler coefficient within the modular of integers. Furthermore, it adds the complexity of noise to the plaintext by using the number of users of the CSP. Moreover, the proposed algorithm's randomness tests prove the best statistical performance was obtained with this algorithm. Furthermore, the algorithm works on encrypting and decrypting different languages such as Kurdish, English, and Arabic.

IV. Homomorphic Encryption

In this section, HE scheme and its categories will be presented. Homomorphic encryption is divided into different categories, which are SWHE, FHE, and PHE. An encryption scheme is said to be homomorphic over an operation "+" if it supports the following equation, where ms is the plaintext message given to the encryption algorithm E:

$$E(ms_1) + E(ms_2) = E(ms_1 + ms_2),\ \forall ms_1, ms_2 \in M$$

http://dx.doi.org/10.14500/aro.10590

Algorithm 1: Key generation

Procedure

Input : n prime numbers $p_1, p_2, p_3 \dots p_n$

for i = 1 to n

Pr = Pr × p_i

L = L(p_i+1)

end for

for i = 1 to L

$M_s = M_s + p_i$

end for

Calc : $M_{avg} = M_s$ DivisibleBy L

rand := a random number $R_n \rightarrow \gcd(R_n, M_{avg}) = 1$

R_n not equal ZERO AND smaller than M_{avg}

for i = 1 to n

(Pr) = (Pr)(p_i−1)

end for

Input : U_{sr} where its ≥ 1

Calc : Q = U_{sr} ×((Pr) mod M_s)

Calc : $K_{sp} = (R_n \times Q) \bmod 256$

Output : K_{sp} as the secret key

End procedure

Algorithm 2: Encryption

Procedure

Input: N as big prime number

Input: ms the plaintext message

rand:= a random number r

for i = 1 to length(ms)

cph = ms + N(r K_{sp}+i)

end for

Output: cph as ciphertext file

End procedure

Algorithm 3: Decryption

Procedure

for i = 1 to length(cph)

ms = cph mod N

end for

Output: ms as plaintext file

End procedure

Somewhat homomorphic encryption allows addition and multiplication operations, however, both operations can be performed in a limited number Fellows and Koblitz (1994) and BNG by (Boneh-Goh-Nissim) (Dan, et al. 2005). Whereas, PHE allows one type of operation, either addition or multiplication, that is, Paillier, Goldwasser-Micali, Benaloh, El-Gamal, and RSA. On the other hand, FHE allows an unlimited number of both addition and multiplication on the ciphertext. It can be considered as ring homomorphism. As in mathematics, a ring is a set R equipped with two operations, "+" and "×" satisfying the eight axioms, known as the ring axioms. Examples of FHEs are FHE schemes Over Integers (dos Santos, et al. 2015), Simple FHE scheme (Li, et al. 2012), LWE-based FHE schemes (Regev, 2005), ideal lattice-based FHE schemes (Gentry, 2009), and NTRU-like FHE schemes (Hoffstein, et al. 1998). Fig. 1 presents the popular schemes proposed after the Gentry's discovery.

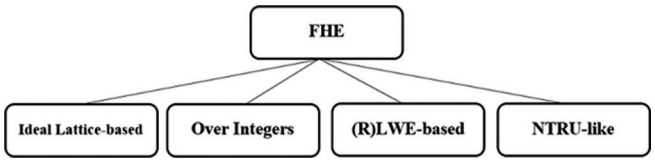

Fig. 1. Main FHE schemes after Gentry's discovery (Acar, et al. 2018).

V. The Proposed Scheme

The proposed scheme works on converting each plaintext character into its corresponding Unicode and then encrypts the derived Unicode by passing it to the encryption algorithm. In addition, the scheme also works on encrypting plaintexts in several languages such as Kurdish, English, and Arabic languages. In addition, the algorithm uses two different noises *r* as the first noise is added to make the ciphertext more digestive, whereas the counter *i* works on converting repeated characters in the text into different ciphertext values. The detailed notations used in the key generation, encryption, and decryption algorithms are presented in Table I. Subsequently, the working flow of the algorithms is illustrated in pseudocode.

Generating the Secret Key K_{sp}

At first choose multiple *P* prime numbers $p_1, p_2, p_3 \cdots p_n$ as secret keys, then calculate *P* as $P = p_1 \times p_2 \times \ldots \times p_n$, calculate *L* as $L = (p_1 + 1)(p_2 + 1)\ldots(p_n + 1)$, then calculate $M_s = \sum_{i=1}^{m} F_i$, where F_i = set of prime numbers up to *L*, and then calculate the average value of sum of all prime numbers as $M_{avg} = \frac{M_s}{L}$, then choose a random number R_n that satisfies $\gcd(R_n, M_{avg}) = 1$, $1 < R_n < M_{avg}$, then select U_{sr} as it is the number of existing users of the cloud system $\{U_{sr1}, U_{sr2}, \ldots, U_{srn}\}$, where $U_{sr} \geq 1$, calculate $\theta(P) = (p_1 - 1)\ldots(p_n - 1)$, and calculate $Q = U_{sr} \times (\theta(P) \bmod M_s)$ and finally calculate K_{sp} as:

$$K_{sp} = (R_n \times Q) \bmod 256 \quad (1)$$

Mod 255 is taken as this is because the secret key values are derived from a series of mathematical calculations that are within a certain scale between 1 and 255 so that the values resulting from the equation are not very large and prevent any slowness in the calculation process.

Encryption algorithm

$$cph = ms + N(rK_{sp} + i) \quad (2)$$

Decryption algorithm

$$\text{ms} = cph \bmod N \quad (3)$$

TABLE I
LIST OF NOTATIONS

Notations	Details
cph	Ciphertext
ms	The plaintext message
r	Noise added to the ciphertext
N	Big prime integer
K_{sp}	Secret key
i	Counter added as extra noise to the ciphertext works on converting repeated character into different ciphertext value.
$p_1, p_2, p_3 \dots p_n$	Multiple prime numbers
P	Is the multiplications of prime numbers
L	Is the multiplication of each prime number plus one
M_s	The summation of prime numbers up to *L*
M_{avg}	Average of all prime numbers
R_n	Is a random number where gcd(R_n, M_{avg}) = 1
U_{sr}	Is the number of CSP's users

CSP: Cloud service provider

A. Proof of Homomorphism

This subsection will illustrate the homomorphism of the proposed scheme, assume there are two ciphertexts cph_1 and cph_2 where $cph_1 = ms_1 + N(r_1 K_{\sup} + i)$, $cph_2 = ms_2 + N(r_2 K_{\sup} + i)$, and $cph \bmod N \equiv ms$, where $ms < N$, otherwise, we must take $(ms \bmod N)$.

Homomorphism (Addition)

Assume that the sum of two ciphertexts cph_1 and cph_2 is denoted by $(cph^+ = cph_1 + cph_2)$ so $cph^+ = cph_1 + cph_2 = (ms_1 + ms_2) + N(r_1 K_{sp} + i) + N(r_2 K_{sp} + i)$, nonetheless $N(r_1 K_{sp} + i) + N(r_2 K_{sp} + i) = NK_{sp}\begin{bmatrix}(r_1 + r_2)\\ +2i\end{bmatrix} = 0$ Then $ms^+ = (cph_1 + cph_2) \mod N = ms_1 + ms_2$

Homomorphism (Multiplication)

Assume that the sum of two ciphertexts cph_1 and cph_2 is denoted by ($cph^* = cph_1 * cph_2$) $cph^* = [ms_1 + N(r_1 K_{sp} + i)] \times [ms_2 + N(r_2 K_{sp} + i)]$ $cph^* = [ms_1 \times ms_2 + ms_1 \times N(r_2 K_{sp} + i)] + N(r_1 K_{sp} + i) \times ms_2 + N(r_1 K_{sp} + i) \times N(r_2 K_{sp} + i)]$ Then $N \times [ms_1 \times N(r_2 K_{sp} + i) + (r_1 K_{sp} + i) \times ms_2 N(r_1 K_{sp} + i) \times N(r_2 K_{sp} + i) \bmod N] = 0$ So that, $cph^* = ms_1 \times ms_2 + 0 = ms_1 \times ms_2$.

VI. Result and Analysis

In this section, the results gained from the proposed scheme will be presented through numerous tests on English, Kurdish, and Arabic languages. To test the proposed scheme, it is implemented with Java programming language and processed on a computer with the following features: Intel Core i7 processor, HDD hard drive, 16 GB RAM, and Windows 10 64-bit. At first, the generation of the secret key is illustrated then it will be used for all the tests presented in this section.

Secret key generation

Choose a set of prime numbers for as $p_1 = 31, p_2 = 59$ and $p_3 = 73$ then $\theta(P) = 30 \times 58 \times 72 = 125280$, $P = 31 \times 59 \times 73 = 133517$. Then, calculate $L = (31+1)(59+1)(73+1) = 142080$, so $M_s = 129548351731$, $M_{avg} = 911798$, assume $U_{sr} = 35$ and $R_n = 15$ then $Q = 4384800$, finally $K_{sp} = 224$.

A. Test on English Language

The proposed algorithm will be tested on an English language text of "Hello world; this is a new Fully Homomorphic algorithm." For this test, the secret key will be $K_{sp} = 224$ as generated previously, and $N = 524287$ which is big prime number, then choose a random number as $r = 62598$ the ciphertext of the given text after applying the proposed algorithm on it will be:

7351527148296 7351527672612 7351528196906
7351528721193 7351529245483 7 3 5 1 5 2 9 7 6 9 6 9 1
7351530294065 7351530818344 7 3 5 1 5 3 1 3 4 2 6 3 4
7351531866915 7351532391194 7 3 5 1 5 3 2 9 1 5 4 1 3
7351533439784 7351533964059 7 3 5 1 5 3 4 4 8 8 3 4 7
7351535012644 7351535536848 7 3 5 1 5 3 6 0 6 1 2 0 8
7351536585505 7351537109709 7 3 5 1 5 3 7 6 3 4 0 6 1
7351538158283 7351538682648 7 3 5 1 5 3 9 2 0 6 9 2 6
7351539731231 7351540255431 7 3 5 1 5 4 0 7 7 9 7 5 6
7351541304090 7351541828368 7 3 5 1 5 4 2 3 5 2 6 5 5
7351542876955 7351543401153 7 3 5 1 5 4 3 9 2 5 4 8 0
7351544449806 7351544974091 7 3 5 1 5 4 5 4 9 8 3 8 0
7351546022665 7351546546954 7 3 5 1 5 4 7 0 7 1 2 4 4
7351547595529 7351548119808 7 3 5 1 5 4 8 6 4 4 0 9 6
7351549168377 7351549692597 7 3 5 1 5 5 0 2 1 6 9 4 9
7351550741247 7351551265529 7 3 5 1 5 5 1 7 8 9 8 2 4
7351552314114 7351552838392 7 3 5 1 5 5 3 3 6 2 6 9 0
7351553886965 7351554411257

Table II and Fig. 2 illustrate the performance of the proposed algorithm tested on different file sizes that contain plaintext written in the English language.

B. Test on Kurdish Language

This test illustrates the proposed algorithm tested on a Kurdish language text of "سڵاو ئەمە ئەڵگۆریزمەکەمە بۆ تاقیکردنەوە," and the same values used for testing English language text and the ciphertext will be:

7351527149811 7 3 5 1 5 2 7 6 7 4 2 2 8
7351528198373 7351528722693 7 3 5 1 5 2 9 2 4 5 4 0 4
7351529771233735153029569 5 7 3 5 1 5 3 0 8 1 9 8 3 8
7351531344269 7351531866839 7 3 5 1 5 3 2 3 9 2 6 6 8
7351532917130 7351533441272 7 3 5 1 5 3 3 9 6 5 6 6 6
7351534489976 7351535014114 7 3 5 1 5 3 5 5 3 8 5 5 6
7351536062689 7351536586995 7 3 5 1 5 3 7 1 1 1 4 2 6
7351537635669 7351538160000 7 3 5 1 5 3 8 6 8 4 1 4 3
7351539208574 7351539731144 7 3 5 1 5 4 0 2 5 6 9 7 5
7351540781420 7351541304005 7 3 5 1 5 4 1 8 2 9 8 3 8
7351542354122 7351542878436 7 3 5 1 5 4 3 4 0 2 8 6 1
7351543927113 7351544451280 7 3 5 1 5 4 4 9 7 5 5 6 5
7351545499875 7351546024305 7 3 5 1 5 4 6 5 4 8 4 5 1
7351547072879

Table III and Fig. 3 illustrate the performance of the proposed algorithm tested on different file sizes that contain plaintext written in the Kurdish language.

http://dx.doi.org/10.14500/aro.10590

TABLE II
TESTING THE PROPOSED ALGORITHM ON DIFFERENT FILE SIZES WRITTEN IN THE ENGLISH LANGUAGE

File sizes	Encryption (ms)	Decryption (ms)
10 KB	378	414
20 KB	397	448
40 KB	413	499
80 KB	459	570
160 KB	486	625
320 KB	529	710
500 KB	599	831
1 MB	748	1081
2 MB	981	1597
4 MB	1698	2703
8 MB	2531	4843
16 MB	4234	8432

TABLE III
TESTING THE PROPOSED ALGORITHM ON DIFFERENT FILE SIZES WRITTEN IN THE KURDISH LANGUAGE

File sizes	Encryption (ms)	Decryption (ms)
10 KB	375	406
20 KB	391	438
40 KB	421	500
80 KB	437	562
160 KB	485	688
320 KB	578	766
500 KB	594	814
1 MB	734	1109
2 MB	1031	1625
4 MB	1702	2609
8 MB	2848	4582
16 MB	5471	9018

C. Test on Arabic Language

This time the proposed algorithm will be tested on an Arabic text of "مرحبا نقدم لكم خوارزمية جديدة," also the values from the first test will be used and the ciphertext is as follow:

7351527149829 7351527674096 7351528198379
7351528722661 7351529246947 7 3 5 1 5 2 9 7 6 9 6 9 1
7351530295552 7351530819835 7 3 5 1 5 3 1 3 4 4 1 0 3
7351531868412 7351532391126 7 3 5 1 5 3 2 9 1 6 9 8 5
7351533441271 7351533965560 7 3 5 1 5 3 4 4 8 8 2 7 4
7351535014111 7351535538424 7 3 5 1 5 3 6 0 6 2 6 7 8
7351536586975 7351537111263 7 3 5 1 5 3 7 6 3 5 5 6 9
7351538159861 7351538684115 7 3 5 1 5 3 9 2 0 6 8 5 7
7351539732692 7351540256982 7 3 5 1 5 4 0 7 8 1 2 9 6
7351541305556 7351541829837

The previous tests presented that the proposed algorithm can be performed on different languages, and it produces different cipher-values for all plaintext values and also for the repeated character within the same text. In addition, the rest of this section will present the performance of the proposed algorithm performed on different file sizes written in English, Kurdish, and Arabic languages. Table IV and Fig. 4 illustrate the performance of the proposed algorithm tested on different file sizes that contain plaintext written in the Arabic language.

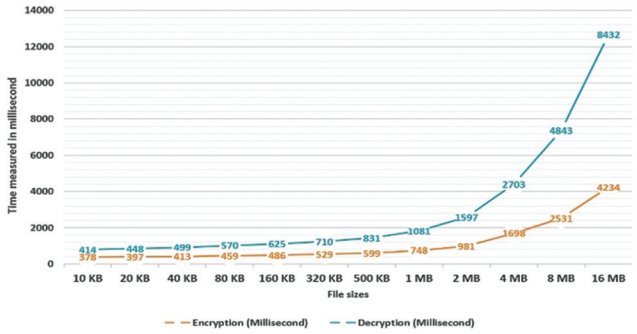

Fig. 2. Encryption and decryption time on English language plaintext file.

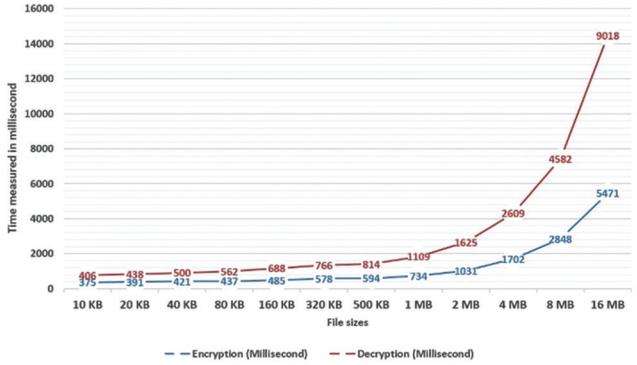

Fig. 3. Encryption and decryption time on Kurdish language plaintext file.

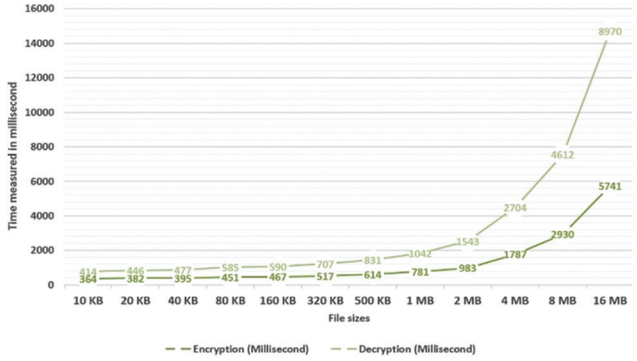

Fig. 4. Encryption and decryption time on Arabic language plaintext file.

The results of the previous tests show that the proposed algorithm is capable on encrypting plaintexts written in different languages efficiently regardless of the file size. In addition, it can be observed from the results that the algorithm performs almost the same performance on the same file sizes of various languages. Table V and Fig. 5 present a comparison of the previous tests gained from the proposed algorithm. As it is illustrated, the encryption and decryption time for all three languages are vary and almost works the same. Such as the encryption time of 20 KB Arabic text-file requires less time than the encryption time on its corresponding English and Kurdish text-file. Whereas, the encryption time of 2 MB English text-file takes less time than Kurdish and Arabic text-files.

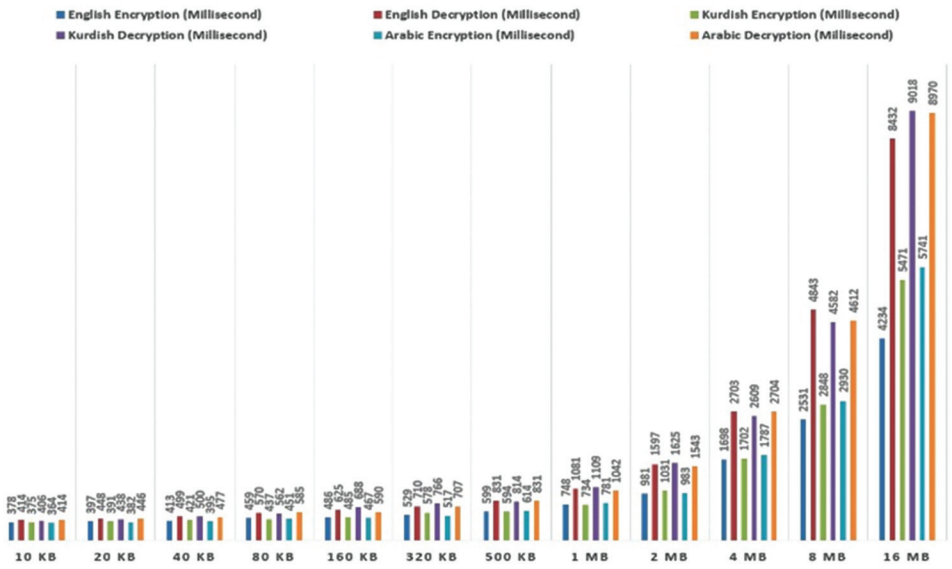

Fig. 5. Encryption and decryption time on English, Kurdish, and Arabic language plaintext file.

TABLE IV
TESTING THE PROPOSED ALGORITHM ON DIFFERENT FILE SIZES WRITTEN IN THE ARABIC LANGUAGE

File sizes	Encryption (ms)	Decryption (ms)
10 KB	364	414
20 KB	382	446
40 KB	395	477
80 KB	451	585
160 KB	467	590
320 KB	517	707
500 KB	614	831
1 MB	781	1042
2 MB	983	1543
4 MB	1787	2704
8 MB	2930	4612
16 MB	5741	8970

TABLE V
TIME COMPLEXITY OF THE BASIC ARITHMETIC OPERATIONS

Operation	Time complexity of binary integers of size n	Time complexity of decimal digits of size n
Addition x + y	O(n)	O(log(n))
Subtraction x – y	O(n)	O(log(n))
Multiplication x × y	$O(n^2)$	$O((\log(n^2))$
Division and Modular	$O(n^2)$	$O((\log(n^2))$
Inverse x^{-1}	$O(n^2\log(n))$	$O(\log(n))^3)$
Modular exponentiation x^n	$O(n^2\log(n))$	$O(\log(n))^3)$

D. Big O Notation (Time Complexity)

Searching for the "best" in algorithms is the main concern of algorithms' designers, and this can be achieved through using O-notation. The aim of studying the time complexity of an algorithm is to determine whether the algorithms' running time is *O(f(N))* for some function *f()* or not. Table VI illustrates the complexity of the basic arithmetic operations in Z_n (Sagheer, 2012).

The input numbers of encryption and decryption algorithms should be analyzed at first before performing any calculation of the time complexity. The input numbers are either binary integers or decimal digits, whereas the time complexity of the first mentioned is *O(n)* and the time complexity of decimal digits is *O(log(n))*, this excluding constant number whose complexity is *O(1)*. Since, n is the size of input numbers.

1) Time complexity of DGHV scheme

Let n be the size of input message unit.

Encryption function:

$cph = ms + 2r + p \times q$

Then: $T(cph) = O(n) + T(2r) + O(n^2)$

$T(2r) = O(n)$, by shift operation

$T(cph) = O(2n) + O(n^2) \equiv O(n^2)$ bit operation.

Decryption function:

$ms = (cph \bmod p) \bmod 2$

Then: $T(ms) = O(n^2)$ bit operation

2) Time complexity of SDC scheme

Let n be the size of input message unit.

Encryption function:

$cph = ms + p + r \times p \times q$

Then: $T(cph) = O(n) + O(n) + O(2(n^2))$

$T(cph) = O(2(n)) + O(2(n^2)) \equiv O(n^2)$ bit operation.

Decryption function:

$ms = cph \bmod p$

Then: $T(ms) = O(n^2)$ bit operation.

3) Time complexity of the proposed algorithm

Let n be the size of input message, n is decimal digit.

Encryption function:

$cph = ms + N(rK_{sp} + i)$

Then: $T(cph) = O(2(\log(n))) + O(2(\log(n))^2)$

 http://dx.doi.org/10.14500/aro.10590

TABLE VI
COMPARING THE RESULTS OF THE ALGORITHM GAINED FROM ENCRYPTING AND DECRYPTING ENGLISH, KURDISH, AND ARABIC LANGUAGE'S PLAINTEXTS

File sizes	English		Kurdish		Arabic	
	Encryption (ms)	Decryption (ms)	Encryption (ms)	Encryption (ms)	Encryption (ms)	Encryption (ms)
10 KB	378	414	375	406	364	414
20 KB	397	448	391	438	382	446
40 KB	413	499	421	500	395	477
80 KB	459	570	437	562	451	585
160 KB	486	625	485	688	467	590
320 KB	529	710	578	766	517	707
500 KB	599	831	594	814	614	831
1 MB	748	1081	734	1109	781	1042
2 MB	981	1597	1031	1625	983	1543
4 MB	1698	2703	1702	2609	1787	2704
8 MB	2531	4843	2848	4582	2930	4612
16 MB	5741	8970	5471	9018	5741	8970

$T(cph) \equiv O(\log(n)^2)$

Decryption function:

$ms = cph \bmod N$

Then: $T(ms) = O((\log(n))^2)$, Where, $(\log_2 n)$ is the number of bits of n

E. Resistance to Attacks

In this section, the resistance of the proposed algorithm to different types of attacks such as Key Generation and Character Repetition, Brute Force Attack, and Mathematical Attack are illustrated.

4) Key generation and character repetition

The proposed algorithm encrypts each file with a different key, and it depends on a variable that is different for every cloud user. In addition, the algorithm encrypts the repetition of each character into different values. Thus, the attacker cannot analyze character repetition in the file. Consequently, the combination of different keys for each file and different values for the same character allows our proposed algorithm to provide a strong encryption method.

5) Brute force attack

In the proposed algorithm, the strength of large prime numbers depends on the multiplication of n prime numbers p_1, p_2,... p_n. Thus, it is difficult to break the large prime number into multiple primes as compared to the existing algorithms. Furthermore, the multiple prime numbers increase the level of difficulty to break the security of the algorithm. In addition, the use of the addition noises makes it more difficult to break.

6) Mathematical attack

This kind of attack occurs when the attacker determines the values of p and q. In our proposed algorithm, it is reduced as the algorithm uses multiple numbers of primes, and it is hard to derive any of those primes from the multiplication result.

F. Results of NIST Statistical Tests

The randomness of this novel proposal is evaluated by the well-known NIST test suite. Table VII 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 VII
NIST SP 800-22 TEST RESULTS FOR THE NAZUZ ALGORITHM

Tests	***P***-value	Result
Frequency (Monobits)	0.997743	Success
Block frequency	0.999936	Success
Cumulative sums (Cusum)	0.983782	Success
Runs	0.982544	Success
Longest run of ones	0.993900	Success
Rank	0.999594	Success
Discrete Fourier transform	0.074478	Success
Non-overlapping template matching	0.999975	Success
Overlapping template matching	0.856322	Success
Universal statistical	0.999620	Success
Approximate entropy	0.999961	Success
Random excursions	0.997529	Success
Random excursions variant	0.837424	Success
Serial	0.999995	Success
Linear complexity	0.999438	Success

VII. CONCLUSION

It has been said that homomorphic encryption is the change point of cryptography, as it protects data regardless of its situation, whether the data are in transit or at rest. This helped CSPs to use this new technique for data protection. This paper is proposed a new FHE scheme based on prime modular operation. The scheme performs encryption and decryption on plaintext values regardless of the written language of the plaintext English, Kurdish, Arabic, or any other languages as well as special characters. In addition, the scheme encrypts repeated characters of the plaintext into different ciphertext values which increases the security of the ciphertext. The randomness of the proposal scheme is evaluated by the well-known NIST test suite (widely used as a standard battery of tests to test randomness). The results of the proposed algorithm in the NIST statistical tests show that it produces the best statistical performance through passing

all the tests. Moreover, the proposed scheme demonstrates good security for the stored data on the cloud.

References

Acar A, Aksu H., Uluagac A.S. and Conti, M., 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4), pp.1-35.

Alattas, R. and Elleithy, K., 2013. *Cloud Computing Algebra Homomorphic Encryption Scheme Based on Fermat's Little Theorem*. The American Society of Engineering Education, Northfield, VT, USA.

Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jaschke, A., Reuter, C. and Strand, M., 2015. A guide to fully homomorphicencryption. *IACR Cryptology ePrint Archive*, 2015, 1192.

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

CyberScout., 2018. *Data Breach Reports*. Identity Theft Resource Center, Berkeley, CA, USA.

Dan, B., Eu-Jin, G. and Kobbi, N. 2005. Evaluating 2-DNF formulas on ciphertexts. In: Proceedings of Theory of Cryptography Conference. Vol. 3378. Springer, Berlin. pp.325-341.

dos Santos, L.C., Bilar, G.R. and Pereira, F.D., 2015. Implementation of the Fully Homomorphic Encryption Scheme Over Integers with Shorter Keys. In: *2015 7th International Conference on New Technologies, Mobility and Security* (*NTMS*), Paris, France.

Elgamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), pp.469-472.

Fellows, M. and Koblitz, N., 1994. Combinatorial cryptosystems galore! In: *Finite Fields: Theory, Applications, and Algorithms*. American Mathematical Society, Providence, Rhode Island. pp.51-61.

Gentry, C., 2009. *A Fully Homomorphic Encryption Scheme, PhD*. Stanford University, United States.

Gentry, C., 2009. Fully Homomorphic Encryption Using Ideal Lattices. In: *Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing STOC '09*, Bethesda, Maryland, USA.

Goldwasser, S. and Micali, S., 1984. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2), pp.270-299.

Gonzales, D., Kaplan, J., Saltzman, E., Winkelman, Z. and Woods, D., 2017. Cloud-trust a security assessment model for infrastructure as a service (IaaS) clouds. *IEEE Transactions on Cloud Computing*, 5(3), pp.523-536.

Hashem, I., Yaqoob, I., Anuar, N., Mokhtar, S., Gani, A. and Khan, S.U., 2015. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47, pp.98-115.

Hayward, R. and Chiang, C., 2015. Parallelizing fully homomorphic encryption for a cloud environment. *Journal of Applied Research and Technology*, 13(2), pp.245-252.

Hoffstein, J., Pipher, J. and Silverman, J., 1998. NTRU: A ring-based public key cryptosystem. In: *Lecture Notes in Computer Science*. Springer Science+Business Media, Berlin, Germany. pp.267-288.

Ji, J. and Shieh, M., 2019. Efficient comparison and swap on fully homomorphic encrypted data. In: *2019 IEEE International Symposium on Circuits and Systems* (ISCAS), Sapporo, Japan, pp.1-4.

Jubrin, A.M., Izegbu, I. and Adebayo, O.S., 2019. 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, pp.1-6.

Li, J., Song, D., Chen, S. and Lu, X., 2012. A Simple Fully Homomorphic Encryption Scheme Available in Cloud Computing. In: *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, Hangzhou, China.

Li, L., Lu, R., Choo, K.R., Datta A. and Shao J., 2016. Privacy-preserving-outsourced association rule mining on vertically partitioned databases. *IEEE Transactions on Information Forensics and Security*, 11(8), pp.1847-1861.

Li, R., Ishimaki, Y. and Yamana H., 2019. Fully homomorphic encryption with table lookup for privacy-preserving smart grid. In: *2019 IEEE International Conference on Smart Computing* (SMARTCOMP), Washington, DC, USA, pp.19-24.

Mert, A.C., Öztürk E. and Savaş, E., 2020. *Design and Implementation of Encryption/Decryption Architectures for BFV Homomorphic Encryption Scheme*. Vol. 28. IEEE Transactions on Very Large Scale Integration Systems, pp.353-362.

Mohammed, M.A. and Abed, F.S., 2019. An improved fully homomorphic encryption model based on N-primes. *Kurdistan Journal of Applied Research*, 4(2), pp.40-49.

Mohammed, M.A. and Abed, F.S., 2020. A symmetric-based framework for securing cloud data at rest. *Turkish Journal of Electrical Engineering and Computer Sciences*, 28(1), pp.347-361.

Paillier, P., n.d. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology Eurocrypt*, 99, pp.223-238.

Regev, O., 2005. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In: *Proceedings of the 37th annual ACM Symposium on Theory of Computing STOC '05*, Baltimore, Maryland, USA.

Rivest, R., Shamir, A. and Adleman, L., 1983. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 26(1), pp.96-99.

Rivest, R.L.; Adleman, L. and Dertouzos, M.L., 1978. On data banks and privacy homomorphisms. In: *Foundations of Secure Computation*. Academia Press, Cambridge, Massachusetts. pp.169-179.

Sagheer, A.M., 2012. Elliptic Curves Cryptographic Techniques. In: *2012 6th International Conference on Signal Processing and Communication Systems*, Gold Coast, QLD, pp.1-7.

Sangani, K., 2011. Sony security laid bare. *Engineering and Technology*, 6(8), pp.74-77.

Shihab, H. and Makki, S., 2018. Design of fully homomorphic encryption by prime modular operation. *Telfor Journal*, 10(2), pp.118-122.

Smart, N. and Vercauteren, F., 2010. Fully homomorphic encryption with relatively small key and ciphertext sizes. *Public Key Cryptography*, 2010, pp.420-443.

Tan, B.H.M., Lee, H.T., Wang, H., Ren, S.Q. and Khin, A.M.M., 2020. Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. *IEEE Transactions on Dependable and Secure Computing*, p.1.

Tebaa, M. and El Hajii, S., 2014. Secure cloud computing through homomorphic encryption. *Computing Research Repository*, 5, 1409.

van Dijk, M., Gentry, C., Halevi, S. and Vaikuntanathan, V., 2010. Fully homomorphic encryption over the integers. *Advances in Cryptology Eurocrypt*, 2010, pp.24-43.

Wang, B., Zhan, Y. and Zhang, Z., 2018. Cryptanalysis of a symmetric fully homomorphic encryption scheme. *IEEE Transactions on Information Forensics and Security*, 13(6), pp.1460-1467.

Xiao, L., Bastani, O. and Yen, I., 2012. *An Efficient Homorphic Encryption Protocol for Multi-user Systems*. IACR Cryptology ePrint Archive, Lyon, France.

Yao, A., 1982. Protocols for Secure Computations. In: *23rd Annual Symposium on Foundations of Computer Science* (*sfcs 1982*), Washington, DC.