Indonesian Journal of Electrical Engineering and Computer Science
Vol. 34, No. 3, June 2024, pp. 1989~1998
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v34.i3.pp1989-1998

Development Paillier's library of fully homomorphic encryption

Temirbekova Zhanerke Erlanovna[1,2], Tynymbayev Sakhybay[2], Abdiakhmetova Zukhra Muratovna[1], Turken Gulzat[1]
[1]Faculty Information Technology, Kazakh National University Named After Al-Farabi (KazNU), Almaty, Kazakhstan
[2]Faculty of Computer Technology and Cybersecurity, International IT University (IITU), Almaty, Kazakhstan

Article Info

Article history:

Received Jan 17, 2024
Revised Feb 21, 2024
Accepted Mar 10, 2024

Keywords:

Cryptography
Homomorphic encryption
Library
Paillier cryptosystem
Security

ABSTRACT

One of the new areas of cryptography considered-homomorphic cryptography. The article presents the main areas of application of homomorphic encryption. An analysis of existing developments in the field of homomorphic encryption carried out. The analysis showed that existing library implementations only allow processing bits or arrays of bits and do not support division and subtraction operations. However, to solve applied problems, support for performing integer operations are necessary. Because of the analysis, the need to implement the homomorphic division and subtraction operations identified, as well as the relevance of developing our own implementation of a homomorphic encryption library over integers. The ability to perform four operations (addition, difference, multiplication and division) on encrypted data will expand the areas of application of homomorphic encryption. A homomorphic division and subtraction methods proposed that allows the division operation performed on homomorphically encrypted data. An architecture for a library of fully homomorphic operations on integers is proposed. The library supports basic homomorphic operations on integers, as well as homomorphic division method. The article also provides measurements of the time required to perform certain operations on encrypted data and analyzes the efficiency of the developed implementation of the library.

Corresponding Author:

Temirbekova Zhanerke Erlanovna
Faculty Information Technology, Kazakh National University Named After Al-Farabi (KazNU)
71/10 Farabi, Almaty, Kazakhstan
Email: temyrbekovazhanerke2@gmail.com

1. INTRODUCTION

Throughout history, cryptography has played a crucial role in ensuring the secure transmission of information in environments vulnerable to security breaches, preserving the confidentiality of transmitted data [1], [2]. This discipline has undergone continuous evolution, with a relatively recent advancement being homomorphic cryptography. What distinguishes homomorphic cryptography is its capacity to manipulate encrypted data without requiring prior decryption [3], [4]. This unique attribute ensures that the outcomes of operations performed on encrypted data, once decrypted, replicate the results of operations conducted on unencrypted data. This innovation addresses a fundamental challenge in cryptography: ensuring the secure generation, storage, and distribution of shared session keys. This progress significantly bolsters data security, enabling servers to receive encrypted data, process it, and return encrypted results, all while maintaining open data and encryption keys within a secure segment during network interactions [5]-[8]. For instance, consider a scenario where a hospital holds a substantial amount of private and sensitive patient information; it can homomorphically encrypt the data and transmit it to a third party for analysis. The third party can conduct computations on the encrypted data and send back the results (also encrypted) to the hospital, which can then decrypt the data using a private key to view the outcomes. Homomorphic encryption schemes are classified based on the number of operations permitted on encrypted data. For a cryptosystem to be fully homomorphic encryption (FHE), it must support an unlimited number of arbitrary computations. Examples of fully homomorphic schemes include the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [9] and the chandler good government index CGGI scheme [10]. However, in practice, fully homomorphic schemes incur significant overhead and are computationally expensive. Somewhat homomorphic encryption (SWHE) schemes are more practical, but they only allow certain operations on encrypted data and restrict the number of computations due to the ciphertext size increasing with each step because of noise [11], [12]. Examples of SWHE schemes include various ones. Partially homomorphic encryption (PHE) schemes [13]-[16], such as others, permit only one type of operation (either addition or multiplication) on encrypted data any number of times, in contrast to somewhat homomorphic schemes that support both. The development of such schemes remains an active area of research, and the establishment of standards for homomorphic encryption has recently gained momentum, as discussed in various sources.

Compared to other homomorphic encryptions, the Paillier cipher is a more attractive option in a number of ways. The Paillier cipher has a unique feature that allows both addition and multiplication operations on encrypted data, unlike some PHEs which are limited in this regard. This makes it more flexible and versatile for solving various problems. In addition, the Paillier cipher has less significant computational and hardware overhead compared to FHE, which makes it more practical for implementation in real systems. It should also be noted that the Paillier cipher provides better performance and lower resource requirements compared to some PHEs, making it a more attractive choice for many applications. Overall, the Paillier cipher is an efficient and powerful solution for homomorphic encryption, given its ability to support both basic operations and its moderate resource requirements.

At the moment, there are already several implementations of libraries for homomorphic encryption [17]-[19]. Among them, the two most significant implementations available for general use are:

i) The HElib library, created by Shai Haveli and Victor Shoup, implements the BGV cryptosystem with GHS optimization.
ii) The FHEW library, developed by Leo Douglas and Daniel Micchianakio, is a combination of Regev's error-learning cryptosystem and Alperin-Sheriff and Peukert's flexible circuit design technique.

In this study, it was studied that both libraries have high speed and good optimization, and are also implemented in the C++ programming language. Although earlier research has examined application needs for homomorphic encryption, these implementations are of limited practical value because they can only handle bits (or arrays of bits) rather than integers. They clearly did not pay attention to the fact that these implementations do not support division and subtraction operations on encrypted data. Based on the analysis, it was concluded that it is necessary to develop a library for FHE that allows you to operate with integers and perform all mathematical operations on them (addition, subtraction, multiplication and division).

Section 2 of this study will highlight our specific contribution, focusing on adding the mathematical operation of subtraction and division to encrypted data. Subsequently, a full analysis of the obtained data will be carried out in section 3. This will be followed by a conclusion summarizing the important findings and insights gained from this study.

2. METHOD

The Paillier cryptosystem [20]-[22] is an asymmetric algorithm rooted in public key cryptography, relying on the challenge of computing the nth residue class, a known computationally challenging problem. This cryptosystem operates as an additive homomorphic system and has found widespread application in electronic cash transactions and secure electronic voting. However, the computational demands of the Paillier homomorphic encryption and decryption processes pose a significant challenge for devices with limited processing resources [23]-[25]. Therefore, there is a need for an improved and faster encryption and decryption version that supports devices with limited processing resources. An improved and fast decryption version used for Paillier homomorphic encryption and a library created. The main transformations that characterize this cryptosystem are as follows:

Key generation:

$$p, q. Gcd\big(pq, (p-1)(q-1)\big) = 1,$$
$$n = p * q \text{and } \lambda = lcm(p-1)(q-1),$$
$$\mu = \Big(L\big(g^{\lambda} mod n^2\big)\Big)^{-1} mod n,$$

where $L(u) = \left(\frac{u-1}{n}\right)$
$publickey(n)$;
$privatekey(p, q, n)$.
Encryption:

$$r \epsilon Z_n\,,\quad gcd(r, n) = 1,$$

$$\text{c} = g^m * r^n mod n \tag{1}$$

Decryption:

$$m_p = \left(\frac{c^{p-1} mod p^2 - 1}{p}\right) * h_p mod p\,,\qquad m_q = \left(\frac{c^{q-1} mod q^2 - 1}{q}\right) * h_q mod q \tag{2}$$

$$h = \left[L\left(g^{\varphi/n} mod n^2\right)\right]^{-1} mod n;$$
$$h_p = \left(\frac{g^{p-1} mod p^2 - 1}{p}\right)^{-1} mod p\,,\ h_q = \left(\frac{g^{q-1} mod q^2 - 1}{q}\right)^{-1} mod q;$$
$$m = L\left(c^{\varphi/n} mod n^2\right) * h mod n;$$

Homomorphic property:
a) Homomorphic addition of plain texts:

$$D(E(m_1)*E(m_2) mod n^2) = m_1 + m_2 mod n \tag{3}$$

b) Homomorphic multiplication of plain texts:

$$(E(m_1, r_1)^{m_2} mod n^2) = m_1 * m_2 mod n \tag{4}$$

c) Homomorphic subtraction of plaintexts:

$$D(E(m_1)*E(m_2)^{-1} mod n^2) = (m_1 - m_2) mod n \tag{5}$$

d) Homomorphic division of plaintexts:

$$D(E(m_1, r_1)^{m_2^{-1}} mod n^2) = m_1/m_2 mod n \tag{6}$$

An example of how the system works:
Key generation:
a. $p = 43, q = 37. gcd\big(pq, (p-1)(q-1)\big) = 1, m_1 = 180, m_2 = 90,$
b. $n = 1591 \qquad \lambda = 252,$
c. $\mu = 1168, g = 1592,\ r_1 = 63,\quad r_2 = 19$
Encryption:
$c_1 = 242783, c_2 = 347602.$
Decryption:
$m_1 = 180, m_2 = 90.$
a) Homomorphic addition of plain texts:

$$242783 * 347602 mod 2531281 = 1479107 \quad D = \frac{1479107^{252} mod 2531281 - 1}{1591} * 1168 mod 1591 = 1218 * 1168 mod 1591 = 270;$$

b) Homomorphic multiplication of plain texts:

$$242783^{90} mod 2531281 = 2174010 \qquad D = \frac{2174010^{252} mod 2531281 - 1}{1591} * 1168 mod 1591 = 406 * 1168 mod 1591 = 290;$$

c) Homomorphic subtraction of plaintexts:

$242783 * 347602^{-1} mod 2531281 = 1746509$ $D = \frac{1746509^{252} mod 2531281 - 1}{1591} * 1168 mod 1591 = 504 * 1168 mod 1591 = 90;$

d) Homomorphic division of plain texts:

$242783^{90^{-1}} mod 2531281 = 42190$ $D = \frac{42190^{252} mod 2531281 - 1}{1591} * 1168 mod 1591 = 406 * 1168 mod 1591 = 2.$

The developed library is implemented in the C++ programming language. To provide the ability to perform multiple operations on encrypted numbers and minimize computational inaccuracies (for example, rounding values), the library uses the NTL large number library. When developing this library, the following tasks were set: providing the ability to process integers; implementation of FHE; supports all mathematical operations, including division. To support the division operation, the library architecture is based on the homomorphic division method.

Development of a software module for the FHE algorithm within Paillier's cryptosystem.
Encryption procedure:
1. Input of initial plaintext data. Encryption files are incorporated, and data is extracted from these files.
2. In the "Parameters" section, p and q are randomly sample until $gcd(pq, (p-1)(q-1))$ is satisfied. $n = p * q, \lambda = LCM(p-1, q-1), g = n+1, \mu = \left(\frac{g^{\lambda} mod n^2 - 1}{n}\right)^{-1} mod n$ keys are calculated, where n, g– public, μ, λ–private key.
3. The encryption block is implemented using mathematical operations.
3.1 Randomly select r (r is an integer)
3.2 $c = g^m * r^n mod n^2$ based on which open data is encrypted.
4. Mathematical operations on encrypted data are performed as:
4.1 Add operation: $\frac{(c_1 * c_2 \% n^2)^{\lambda} \% n^2 - 1}{n} * \mu \% n;$
4.2 Subtraction operation: $\frac{(c_1 * c_2^{-1} \% n^2)^{\lambda} \% n^2 - 1}{n} * \mu \% n;$
4.3 Multiplication operation: $\frac{(c_1^{m_2} \% n^2)^{\lambda} \% n^2 - 1}{n} * \mu \% n;$ $\frac{(c_2^{m_1} \% n^2)^{\lambda} \% n^2 - 1}{n} * \mu \% n;$
4.4 Division operation: $\frac{\left(c_1^{m_2^{-1}} \% n^2\right)^{\lambda} \% n^2 - 1}{n} * \mu \% n;$ $\frac{\left(c_2^{m_1^{-1}} \% n^2\right)^{\lambda} \% n^2 - 1}{n} * \mu \% n.$
5. Decrypts data encrypted using the formula. $m = \left(\frac{c^{\lambda} mod n^2 - 1}{n}\right) * \mu mod n$
The pseudo code of the performed operations illustrated in Figure 1.

3. RESULTS AND DISCUSSION

A C++ static library was developed to enable Paillier FHE. This library utilizes the Boost library to handle extensive numerical operations, ensuring precision in calculations involving encrypted numbers. Throughout the modification process, key objectives included support for integer processing, complete homomorphic encryption, and the inclusion of all mathematical operations, including division. In order to facilitate division operations, the library's architecture devised based on the homomorphic division method mentioned earlier. The library structured around cryptographic and mathematical classes, along with a class responsible for fundamental information, as illustrated in Figure 2.

The secret key class is responsible for managing data related to the secret key used in the cryptographic algorithm. It offers functionalities such as key generation, random key generation, and key utilization. The present implementation of the library employs a standard library's random number generator with automatic randomization based on the current time for key and polynomial coefficient generation. However, it is important to note that the pseudo-random number generator from the standard library considered cryptographically insecure due to its use of a linear congruence method. To address this security concern, an interface introduced to enable the utilization of external random generators. By mutual agreement, the recommended choice is the random number generator from the Boost library, known for its non-deterministic random number generation and cryptographic security.

```
Algorithm 1
  $size \leftarrow RANDOM$
  $numbers1[size], numbers2[size]$
  $c1[size], c2[size]$
  $p, q \leftarrow PrimeNumber$
  if $GCD(pq, (p-1)(q-1)) \neq 1$ then
      $Incorrect.GCD(pq, (p-1)(q-1))$ must be 1 and p and q must be prime
  numbers
  else
      $n \leftarrow p \times q$
      $\lambda \leftarrow LCM(p-1)(q-1)$
      $g \leftarrow n + 1$
      $\mu \leftarrow ((g^{\lambda} mod n^2 - 1)/n)^{-1} \ mod \ n$
      $ENCRYPTION$
      for $i \leftarrow 0$ to size do
          $r \leftarrow RANDOM$
          $c1[i] \leftarrow g^{numbers1[i]} \times r^n \ mod \ n^2$
          $c2[i] \leftarrow g^{numbers2[i]} \times r^n \ mod \ n^2$
      end for
      $GOMOMORPHISM$
      $+$
      for $i \leftarrow 0$ to size do
          $buf \leftarrow c1[i] \times c2[i] \ mod \ n^2$
          $numebers1[i] + numbers2[i] mod n \leftarrow ((buf^{\lambda} mod n^2 - 1)/n) \times \mu \ mod \ n$
      end for
      $*$
      for $i \leftarrow 0$ to size do
          $buf \leftarrow c1[i]^{numbers2[i]} \ mod \ n^2$
          $numebers1[i] \times numbers2[i] mod n \leftarrow ((buf^{\lambda} mod n^2 - 1)/n) \times \mu \ mod \ n$
      end for
      $-$
      for $i \leftarrow 0$ to size do
          $buf \leftarrow c1[i] \times c2[i]^{-1} \ mod \ n^2$
          $numebers1[i] - numbers2[i] mod n \leftarrow ((buf^{\lambda} mod n^2 - 1)/n) \times \mu \ mod \ n$
      end for
  end if
```

Figure 1. Operations performed in the homomorphic Paillier cryptosystem

The encrypted data module primarily deals with a homomorphically encrypted number as its main data type. Within the encryption class, functionalities are implemented for creating new ciphertexts through plaintexts and secret keys (data encryption operation), as well as extracting clear data from encrypted data based on the encryption key (data decryption operation). Homomorphism is employed to execute fundamental mathematical operations, including subtraction, addition, division, and multiplication on encrypted data. Encryption and decryption can be performed using pre-generated keys or by providing secret parameters. Moreover, the class encompasses all the necessary mathematical operations for polynomials, such as division, multiplication, subtraction, and addition.

Decrypted data, on the other hand, reverses encrypted data using a private key and homomorphism. The verification of homomorphism involves several operations. For addition, an array named resBuf is opened, and its size is determined based on the length of the larger polynomial. All elements of resBuf are initially set to 0, and a loop is used to add index polynomial values corresponding to each index. The sum value is calculated using the variable resu=0 and resu+=resbuff[0]*x0, with similar steps performed for the remaining parts of the polynomial.

For subtraction, the algorithm involves placing the numbers of the first polynomial into the resBuf array and subtracting the numbers of the second polynomial from it, similar to the addition-checking algorithm. Multiplication operation verification includes opening an array called Kob, with a size of [(size of first polynomial)×(size of second polynomial)]. This array is divided into two columns, where the first column contains coefficients in front of x, and the second column contains the exponents of x. The resu array is then opened, and the same ranks are added to it. The division operation utilizes the horner scheme on encrypted data. When dividing one polynomial by another, the algorithm divides and takes the remainder. An architecture for Paillier homomorphic encryption has been developed, and the constructed architecture is depicted in Figure 2. Following this architecture, a static library for the Paillier encryption algorithm has been created.

Secret key: this category manages information pertaining to the secret key employed in the cryptographic algorithm. It facilitates key generation, random key generation, and provides access to both the public and private keys. Encryption: this process involves taking public data values from the ASCII table and encrypting the data using the public key. Homomorphism: this functionality executes various mathematical operations on encrypted data. DecryptedData: following operations on encrypted data, this allows for reverse encryption using a secret key, enabling retrieval of results from the performed operations. Once the data is reverse-encrypted, it can be used in its clear text form, and operations can be conducted on the data following homomorphic transformations. Development of a software module for the FHE algorithm within Paillier's cryptosystem. The generated software is depicted in Figure 3. In Table 1 present Paillier library implementation functions. In Table 2 shows execution time of the developed library. Figure 4 shows the dependence of the execution time of the algorithm on the encryption parameters.

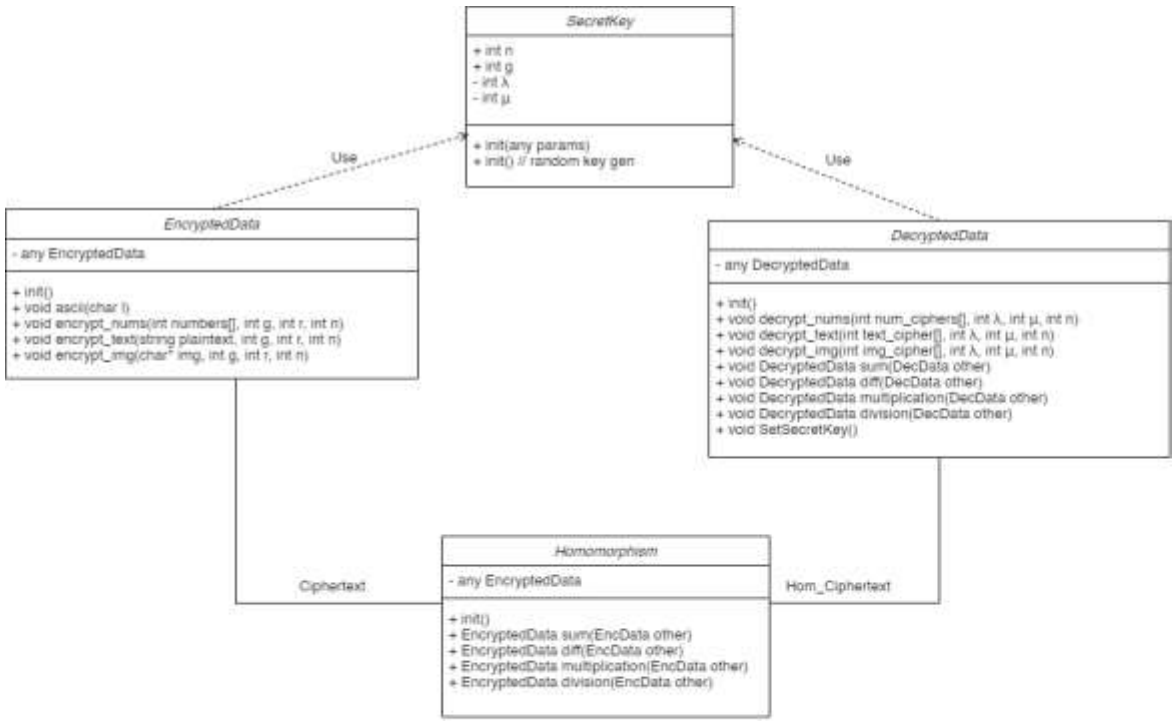
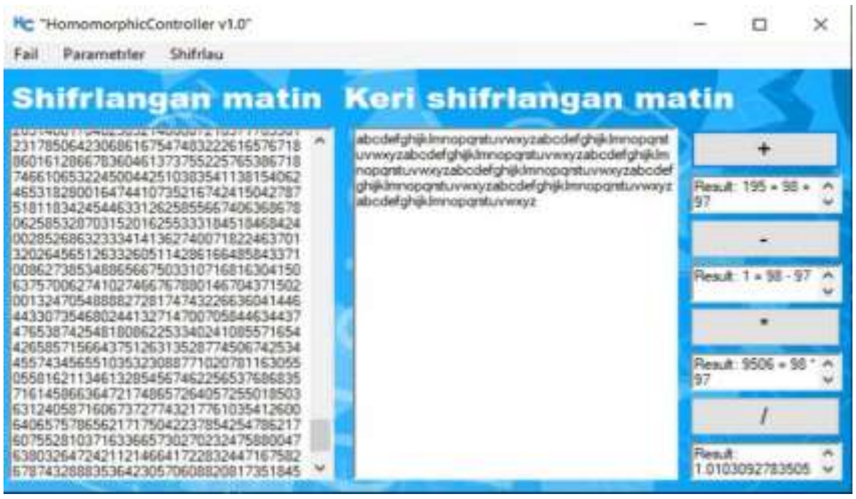

SecretKey
+ int n
+ int g
- int λ
- int µ
+ init(any params)
+ init() // random key gen
Use
Use
EncryptedData
- any EncryptedData
+ init()
+ void ascii(char l)
+ void encrypt_nums(int numbers[], int g, int r, int n)
+ void encrypt_text(string plaintext, int g, int r, int n)
+ void encrypt_img(char* img, int g, int r, int n)
DecryptedData
- any DecryptedData
+ init()
+ void decrypt_nums(int num_ciphers[], int λ, int µ, int n)
+ void decrypt_text(int text_cipher[], int λ, int µ, int n)
+ void decrypt_img(int img_cipher[], int λ, int µ, int n)
+ void DecryptedData sum(DecData other)
+ void DecryptedData diff(DecData other)
+ void DecryptedData multiplication(DecData other)
+ void DecryptedData division(DecData other)
+ void SetSecretKey()
Ciphertext
Hom_Ciphertext
Homomorphism
- any EncryptedData
+ init()
+ EncryptedData sum(EncData other)
+ EncryptedData diff(EncData other)
+ EncryptedData multiplication(EncData other)
+ EncryptedData division(EncData other)

Figure 2. Architecture of Paillier homomorphic encryption



```
2317850642306861675474832226165576718
8601612866783604613737552257653867 18
7466106532245004425103835411381540 62
4653182800164744107352167424150427 87
5181183424544633126258556674063686 78
0625853207031520162553331845184684 24
0028526863233341413627400718224637 01
3202645651263326051142861664858433 71
0086273853488656675033107168163041 50
6375700627410274667678801467043715 02
0013247054888827281747432266360414 46
4433073546802441327147007058446344 37
4765387425481808622533402410855716 54
4265857156643751263135287745067425 34
4557434565510353230887710207811630 55
0558162113461328545674622565376868 35
7161458663647217486572640572550185 03
6312405871606737277432177610354126 00
6406575786562171750422378542547862 17
6075528103716336657302702324758800 47
6380326472421121466417228324471675 82
6787432888353642305706088208173518 45
```

```
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrst
uvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklm
nopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdef
ghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

```
Result: 195 = 98 + 97
Result: 1 = 98 - 97
Result: 9506 = 98 * 97
Result: 1.0103092783505
```

Figure 3. The result of the FHE algorithm of the Paillier cryptosystem

Multiplication tests: in Figure 4, the charts illustrate the correlation between data processing time using the library and the number of sequential operations involved in the multiplication operation. Notably, the multiplication operation exhibits exponential complexity due to a rapid surge in noise during its execution. With each multiplication operation, the ciphertext volume approximately doubles, contributing to the observed exponential growth. Addition tests: in the same figure, Figure 4 portrays the relationship between data processing time using the library and the number of sequential operations for the addition operation. The graphs indicate that the addition operation demonstrates linear complexity relative to the number of consecutive addition operations. This linearity is attributed to the comparatively slow expansion of ciphertext size resulting from the addition operation.

Table 1. Paillier library implementation functions

Homomorphism over two numbers		
The name of the function	The signature of the function	Description
Add	Add(long long a, long long b);	Calculating the sum of two large integers. a and b are integers.
Modpow	Modpow(long long a, long long b, long long c);	Obtaining the remainder when dividing a certain power of a number by the next number. a, b, c are integers ($a^b\%c$).
ReverseMod	ReverseMod(long long a, long long b)	Obtain the remainder of the power of -1 of a number divided by the next number. a and b are integers ($a^{-1}\%b$).
PrintSome	PrintSome(long long A[], long long n);	A [] displays the data set (n is the length of the set).
IsPrime	IsPrime(int n);	Determines whether a given number is prime or not. n is an integer.
NOD	NOD(long long int a, long long int b);	It is used to find the greatest common divisor (GCD) of two numbers a and b are integers.
Nok	Nok(int a, int b);	Nok Nok(int a, int b); used to find the least common multiple (LCM) of two numbers a and b are integers.
FindG	FindG(long long g, long long ly, long long n2, long long n, int size);	
Gomomorph	Gomomorph(long long x, long long ly, long long n, long long n2);	The homomorphic property is proven by performing mathematical operations on encrypted data. x is the value obtained as a result of operations, ly is a secret key, n is an integer.
EncryptionPeye	EncryptionPeye(long long int C[], long long int M[], long long int r[], long long g, long long n, long long n2, long long size);	Searches the ASCII table for each character in the text. The values from the table are encrypted. C[] is a set of returned ciphers, M[] is a set of input data, r[] are integers used for each encryption process, g,n are public keys, n2 = n^2
DecryptionPeye	DecryptionPeye(long long int C[], long long int M2[], long long ly, long long mu, long long n, long long n2, long long size);	Performs the decryption of encrypted values. The outcome of the inverse encryption involves referencing the numbers in the ASCII table and recording the corresponding symbols. C[] represents a collection of ciphers for the reverse encryption process, M2[] denotes numbers acquired through reverse encryption, ly and mu are secret keys, n is an integer, and size indicates the length of the dataset.

The developed subtraction and division operations are of significant practical importance in the context of the use of homomorphic encryption in real applications. For example, computer security: adding subtraction and division operations expands the use of FHE to protect data privacy across systems. This is especially important in areas where arithmetic operations on encrypted data are required, such as financial transactions and processing of personal data. Processing sensitive data: subtraction and division operations enable a wider range of analytical tasks to be performed on encrypted data, which can be useful for organizations processing sensitive data, such as medical research where maintaining patient confidentiality is important. Research in cryptography: the development of new operations for homomorphic encryption represents an important step in the development of cryptographic methods for ensuring data confidentiality. This could lead to more efficient and secure encryption methods, which in turn will improve overall information security.

Table 2. Execution time of the development library

2-digit, different even numbers	Key generation, encryption			
	decryption, add, ms	decryption, subtraction, ms	decryption, multiplication, ms	decryption, division, ms
100	6.65	6.7	9	10.85
200	9.65	9.73	12.87	15.42
300	12.65	12.79	17.09	20.34
400	15.68	15.86	21.3	24.45
500	18.69	18.9	25.4	28.75

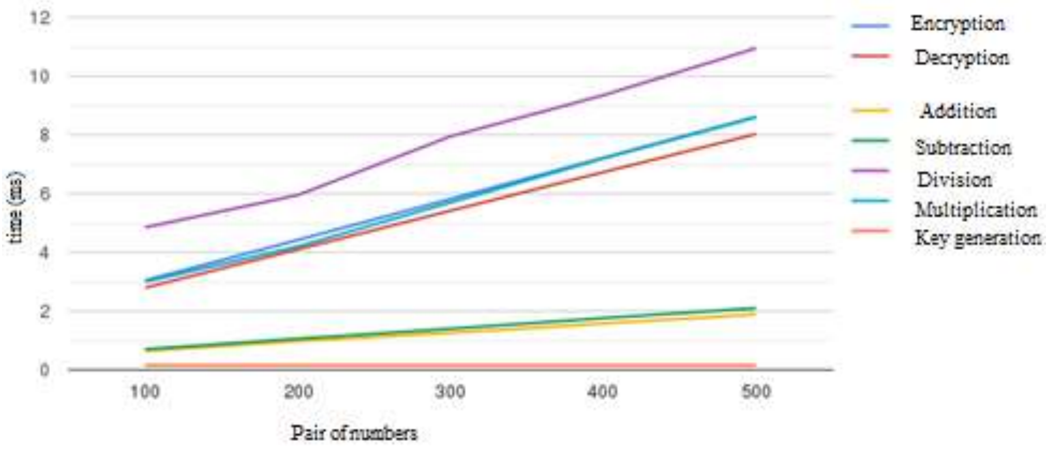


Figure 4. Algorithm execution time dependence on encryption parameters

4. CONCLUSION

Through the undertaken investigation, a comprehensive analysis of homomorphic encryption was conducted, unveiling both its vulnerabilities and strengths, while also elucidating potential avenues for practical application. A scrutiny of existing homomorphic encryption libraries revealed a gap in implementations capable of effectively handling integers with a complete mathematical framework. This assessment underscores the imperative to design and develop a proprietary library dedicated to FHE of integers. The envisioned library aims to facilitate accurate cryptographic transformations on integer data, encompassing essential mathematical operations such as addition, multiplication, subtraction, and division. Notably, the absence of a fully realized implementation for homomorphic encryption of integers prompted the identification of a crucial need in the field. Consequently, the research culminated in the formulation and proposal of a method specifically tailored to enable homomorphic division, ensuring its applicability across diverse fully homomorphic algorithms. This proposed solution addresses a critical gap in existing implementations, paving the way for more versatile and comprehensive homomorphic encryption practices. This scientific article has high practical significance in the field of cryptography and information security, making FHE more accessible and applicable in various fields. Our findings provide conclusive evidence that this phenomenon helps improve data security and privacy in the modern information world.

REFERENCES

[1] A. Y. U. Pyrkova and Z. H. E. Temirbekova, "Compare encryption performance across devices to ensure the security of the IOT," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 20, no. 2, pp. 894–902, 2020, doi: 10.11591/ijeecs.v20.i2.pp894-902.

[2] Z. Kasiran, S. Abdullah, and N. M. Nor, "An advance encryption standard cryptosystem in iot transaction," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 17, no. 3, pp. 1548–1554, 2020, doi: 10.11591/ijeecs.v17.i3.pp1548-1554.

[3] W. A. Awadh, A. S. Alasady, and M. S. Hashim, "A multilayer model to enhance data security in cloud computing," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 32, no. 2, pp. 1105–1114, 2023, doi: 10.11591/ijeecs.v32.i2.pp1105-1114.

[4] Q. Zhang, "An overview and analysis of hybrid encryption: the combination of symmetric encryption and asymmetric encryption," *Proceedings - 2021 2nd International Conference on Computing and Data Science, CDS 2021*, pp. 616–622, 2021, doi: 10.1109/CDS52072.2021.00111.

[5] W. A. Awadh, A. S. Alasady, and A. K. Hamoud, "Hybrid information security system via combination of compression, cryptography, and image steganography," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 6, pp. 6574–6584, 2022, doi: 10.11591/ijece.v12i6.pp6574-6584.

[6] M. N. S. Perera, T. Nakamura, M. Hashimoto, H. Yokoyama, C. M. Cheng, and K. Sakurai, "A survey on group signatures and ring signatures: traceability vs. anonymity," *Cryptography*, vol. 6, no. 1, p. 3, 2022, doi: 10.3390/cryptography6010003.

[7] A. Sakhi, S. E. Mansour, and A. Sekkaki, "Enhancing security mechanisms for robot-fog computing networks," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 33, no. 3, pp. 1660–1666, 2024, doi: 10.11591/ijeecs.v33.i3.pp1660-1666.

[8] J. Wei, W. Liu, and X. Hu, "Secure data sharing in cloud computing using revocable-storage identity-based encryption," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1136–1148, 2018, doi: 10.1109/TCC.2016.2545668.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[10] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol.* 10624LNCS, 2017. doi: 10.1007/978-3-319-70694-8_14.

[11] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10031 LNCS, pp. 3–33, 2016, doi: 10.1007/978-3-662-53887-6_1.
[12] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” in *Proceedings of the 15th international conference on Practice and Theory in Public Key Cryptography*, 2012, pp. 1–16, [Online]. Available: https://eprint.iacr.org/2012/144.
[13] A. C. Yao, “Protocols for secure computations.,” in *Annual Symposium on Foundations of Computer Science - Proceedings*, 1982, pp. 160–164, doi: 10.1109/sfcs.1982.38.
[14] D. Boneh, E. J. Goh, and K. Nissim, “Evaluating 2-DNF formulas on ciphertexts,” in *Theory of Cryptography*, vol. 3378, Springer, 2005, pp. 325–341.
[15] Y. Ishai and A. Paskin, “Evaluating branching programs on encrypted data,” in *Theory of Cryptography*, vol. 4392 LNCS, Springer, 2007, pp. 575–594.
[16] S. Halevi and V. Shoup, “Bootstrapping for HElib,” *Journal of Cryptology*, vol. 34, no. 7, pp. 641–670, 2021, doi: 10.1007/s00145-020-09368-7.
[17] V. Vaikuntanathan, “Computing blindfolded: new developments in fully homomorphic encryption,” in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2011, pp. 5–16, doi: 10.1109/FOCS.2011.98.
[18] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6110 LNCS, pp. 24–43, 2010, doi: 10.1007/978-3-642-13190-5_2.
[19] R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things,” *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, Jul. 2013, doi: 10.1016/j.comnet.2012.12.018.
[20] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in internet of things: the road ahead,” *Computer Networks*, vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.
[21] K. Lauter, M. Naehrig, and V. Vaikuntanathan, “Can homomorphic encryption be practical?,” in *Proceedings of the ACM Conference on Computer and Communications Security*, 2011, pp. 113–124, doi: 10.1145/2046660.2046682.
[22] K. Hussain, N. Z. Jhanjhi, H. M. ur-Rahman, J. Hussain, and M. Hasan Islam, “Using a systematic framework to critically analyze proposed smart card based two factor authentication schemes,” *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 4, pp. 417–425, 2021, doi: 10.1016/j.jksuci.2019.01.015.
[23] U. Musa, M. O. Adebiyi, F. Bukie Osang, A. Aduragba Adebiyi, and A. Ariyo Adebiyi, “An improved secured cloud data using dynamic rivest-shamir-adleman key,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 33, no. 1, p. 433, Jan. 2024, doi: 10.11591/ijeecs.v33.i1.pp433-441.
[24] M. Kaur and V. Kumar, “A comprehensive review on image encryption techniques,” *Archives of Computational Methods in Engineering*, vol. 27, no. 1, pp. 15–43, Jan. 2020, doi: 10.1007/s11831-018-9298-8.
[25] C. L. Stergiou, A. P. Plageras, K. E. Psannis, and B. B. Gupta, “Secure machine learning scenario from big data in cloud computing via internet of things network,” in *Handbook of Computer Networks and Cyber Security*, Cham: Springer International Publishing, 2020, pp. 525–554.

BIOGRAPHIES OF AUTHORS

Temirbekova Zhanerke Erlanovna was born in Jambyl Region, Kazakhstan in 1989. She received the B.S. degree from the Kazakh National University named after al-Farabi in 2011 and the M.S. degree from the Kazakh National University named after al-Farabi in 2013, both in computer science. She is senior lector at Faculty of Information Technology of Al-Farabi Kazakh National University. She holds a Ph.D. degree in Computer Engineering with specialization in Computer Science. Her research interests include cryptography, microcontroller security, microcontroller protection algorithms. She can be contacted at email: temyrbekovazhanerke2@gmail.com.

Tynymbayev Sakhybay holds the academic title of Professor and earned his Candidate of Technical Sciences degree. He has been honored with several state awards from the Republic of Kazakhstan, including the Medal "For Labor Distinction" (1984) and the Medal "Veteran of Labor of the USSR" (1989). Recognized for his outstanding achievements, he received the Breastplate of the USSR Ministry of Higher Education and a Certificate of Honor from the Ministry of Education and Science of the Republic of Kazakhstan (2009). With over 62 years of total work experience, he has spent more than 55 years in the industry. Noteworthy accomplishments include being the first graduate of the Faculty of Automation and Computer Science at Kazakh Polytechnic Institute named after V.I. Lenin in 1964, and successfully defending his dissertation for the degree of Candidate of Technical Sciences in 1971. Tynymbayev has served as a professor in the Departments of Computer Science and Information Security at KazPTI named after V.I. Lenin from 1970 to 2017. He is an author and co-author of more than 200 scientific works, including 2 monographs and 10 textbooks. Additionally, he holds 37 patents for his inventions. He can be contacted at email: s.tynym@mail.ru.

Abdiakhmetova Zukhra Muratovna was born in East Kazakhstan Region in 1987. She received the B.S. degree from the Kazakh National University named after al-Farabi in 2009 and the M.S. degree from the Kazakh National University named after al-Farabi in 2011, both in computer science. She is senior lector at Faculty of Information Technology of Al-Farabi Kazakh National University. She holds a Ph.D. degree in Computer Engineering with specialization in Computer science. Her research interests include cryptography, microcontroller security, neural networks, artificial intelligent. She can be contacted at email: zukhra.abdiakhmetova@gmail.com.

Turken Gulzat was born in China in 1988. She received the B.S. degree from the Al-Farabi Kazakh National University in 2008 and the M.S. degree from the Al-Farabi Kazakh National University in 2014, both in Computer Science. She is senior lecturer at Faculty of Information Technology of Al-Farabi Kazakh National University. Her research interests include cryptography, neural networks, artificial intelligent, database management system and data warehouse. She can be contacted at email: turken.gulzat@gmail.com.