Integer Matrices Homomorphic Encryption and Its application

Yanan Bai Jingwei Chen Yong Feng
Wenyuan Wu
Chongqing Key Lab Automated Reasoning & Cognit,
Chongqing Institute of Green and intelligent Technology,
Chinese Academy Sciences,
University of Chinese Academy of Sciences.
e-mail:baiyanan@cigit.ac.cn

October 22, 2018

Abstract

We construct an integer matrices encryption scheme based on binary matrices encryption scheme proposed in [9], and support homomorphic addition and multiplication operations, we prove the correctness and analyze the security. Besides, we implement four encryption schemes including public-key and symmetric-key binary matrices encryption schemes from [9], and public-key and symmetric-key integer matrices encryption schemes from this work. The experimental results show that the running time of homomorphic multiplycation just costs 0.32sec for 40×40 integer matrices, it provides a promising prospect for application. Finally, we apply integer matrices encryption into graph theory to homomorphiclly solve a problem that the number of length-k walks between any two vertices, the algorithm shows the effectiveness.

1 Introduction

Homomorphic encryption is a ciphertext computation technology, which allows us to evaluate functions over encryped texts, and can get the same results as evaluating over the corresponding plaintexts. With the development of cloud computing and global growth of data, private and sensitive information have received increasingly concern, ciphertext computing becomes an urgent demand, Homomorphic encryption can meet this requirement [17] [16], [12] [11].
Homomorphic encryption arises from privacy homomorphism proposed by Rivest et.al [14] in 1978, in the fellowing thirty years there is no deteministic solutions. Until the year 2009, Gentry constructed the frist fully homomorphic encryption scheme [7] [6], which based on ideal lattice with low efficiency. In the year

of 2012, Brakerski, Gentry and Vaikuntanathana made a newly leveled fully homomorphic encryption scheme [3], emplying module switch and key switch technologies [4] [2]. In order to make homomorphic operations more natural, GSW, as the third generation homomorphic encryption system was proposed [8] by Gentry, Sahai and Waters in the year 2013. This scheme used eigenvector to construct encryption scheme, and can encrypt binary bits and integers. At present, this scheme is the fastest and simplest fully homomorphic encryption scheme compared with the first and the second generation. Later, Khedr et.al introduced SHIELD [10], which based on ring LWE setting, The impressed feature of this scheme is that it can conduct addition and multiplication homomorphic operations on encrypted polynomials. [9] constructed the first fully homomorphic encryption scheme for binary matrices, and optimized the bootstrapping procedure of Alprin-Sheriff and Peikert [1]. As the homomorphic operations are limited to binary matrices, most of applications have to design complex circuit to achieve more homomorphic operations, besides, for some application scenarios, efficiency is more significant than achieving full homorphism. Therefore, it is meaningful to design the integer matrices encryption scheme. In this paper, there are four aspects to our work.

- We propose the novel encryption and decryption algorithms for the integer matrices , and also realize the homomorphic addition and multiplication operations in the form of integer matrices, rather than the traditional binary form.
- We demonstrate the correctness of the proposed scheme in theory, and analyze the security.
- With implementing the binary matrices and integer matrices encryption schemes, we evaluate the efficiency of proposed integer matrices encryption schemes.
- We apply the proposed scheme in the graph theory, which calculates the number of length-k walks between any two vertices in the encrypted network. The experimental result shows the effective of proposed method.

2 Preliminaries

2.1 The Learning with Errors(LWE) Problem

The learnnig with errors (LWE) was introduced by Regev in 2005 [13], the definition is:

Definition 2.1. For security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $\chi = \chi(\lambda)$ is a Gasussian distribution over $\mathbb{Z}$, $q = q(\lambda)$ be an integer. The $DLWE_{n,\chi,q}$ problem is to distinguish the following two distributions:
1. Sample (a_i, b_i) uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.
2. One fristly draws $s \xleftarrow{U} \mathbb{Z}_q^n$ uniformly, and then sample (a_i, b_i) by sampling

$a_i \overset{U}{\longleftarrow} \mathbb{Z}_q^n$, $e_i \overset{R}{\longleftarrow} \chi$, and setting $b_i =< a_i, s > +e_i$.
The $DLWE_{n,q,\chi}$ assumption is that the $DLWE_{n,q,\chi}$ problem is infeasible.

Gadget matrix G: Let $l = \lceil log_2 q \rceil$ and identify $\mathbb{Z}_q$ as $0, 1, ..., q-1$. Then each $m \in \mathbb{Z}_q$ can be represented as

$$m = \sum_{i=0}^{l-1} x_i 2^i$$

where $x_i \in \{0,1\}$. Let row vector $g^T = (1, 2, ..., 2^{l-1})$, and column vector $x = (x_0, x_1, ...x_{l-1})^T \in \mathbb{Z}^l$ such that $g^T x \equiv m(modq)$. Define

$$G = g^T \otimes I_{(n+1)} = \begin{bmatrix} g^T & & & \\ & g^T & & \\ & & \ddots & \\ & & & g^T \end{bmatrix} \in \mathbb{Z}^{(n+1)\times((n+1)l}$$

Define operation $G^{-1} : \mathbb{Z}_q^{n+1} \longrightarrow \mathbb{Z}^{(n+1)l}$ describs below:
For any $m \in \mathbb{Z}_q^{n+1} \to \mathbb{Z}^{(n+1)l}$ and short vector $x \in \mathbb{Z}^{(n+1)l}$, such that $Gx \equiv m(modq)$ so we define $G^{-1}(m) = x$.
G^{-1} maps each vector $m \in \mathbb{Z}_q^{n+1}$ to a short vector $x \in \mathbb{Z}^{(n+1)l}$ and $Gx \equiv m(modq)$ will satisfied.

2.2 GSW Fully Homomorphic Encryption scheme

The GSW fully homomorphic encryption scheme use approximate eigenector method to construct ciphertext C to make secrect key s be an approximate $mod-q$ eigenvector of C. The scheme was modified by Alperin-Sheriff [1]. We describe the scheme below.
GSW.Setup($1^\lambda, 1^L$): This step produce related parameters. Let $l = \lceil \log_2 q \rceil$, and $m \geqslant nl$
GSW.keyGenSec($params$): Choose $\bar{s} \overset{U}{\longleftarrow} \mathbb{Z}_q^n$, and output $sk = (1, -\bar{s}) \in Z_q^{n+1}$
GSW.keyGenPub($params$): Choose a $n \times m$ random matrix $A \in \mathbb{Z}_q^{n\times m}$, and pick $e_i \overset{R}{\longleftarrow} \chi^m$ at random.Compute

$$b^T = \bar{s}^T A + e^T \in \mathbb{Z}_q^m$$

Then the public key is $B = \begin{pmatrix} b^T \\ \overline{A} \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$, Observe that $sB = e^T modq$
GSW.Enc($params, pk, m$): Let $m \in \mathbb{Z}_q$ be an integer, To encrypt m, choose a random short matrix $R \in \{0,1\}^{m\times(n+1)l}$, Compute

$$C = mG + BR \in \mathbb{Z}_q^{(n+1)\times(n+1)l}$$

G is the Gadget matrix, B is the public key. If $m \in \{0,1\}$, take the same encryption method.

GSW.DEC(C, sk): This algorithm can recover plaintext $m \in \{0,1\}$. Let c be the penultimate column of ciphertext matrix C, the decryption algorithm output plaintext $m = \lfloor < s, c > \rceil_2$, where $\lfloor \cdot \rceil_2$ donates that $\mathbb{Z}_q \longrightarrow \{0,1\}$ whether the inner product is closer to 0 or $q/4$. As $s^T c = ms^T G + s^T BR = ms^T G + e^T R (mod q)$. We need $e^T R$ to be small, let $noise = e^T R$, if $\|noise\|_\infty < q/8$, then we can recover m. We denote notation $\|x\|_\infty$ as Maximum norm.
GSW.MPDEC(C, sk): This algorithm can recover plaintext $m \in \mathbb{Z}_q$, we will show the decryption process below. Taking the frist $l-1$ rows of Cs, Let

$$Cs = \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{l-2} \end{pmatrix} = m \cdot \begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{l-2} \end{pmatrix} + noise \in \mathbb{Z}_q^{l-1}$$

we set plaintext $m = \sum_{i=0}^{l-2} x_i 2^i$ where $x_i \in \{0,1\}$. Taking the last entity is

$$\gamma_{l-2} = (x_0 + 2x_1 + ...2^{l-2} x_{l-2}) \cdot 2^{l-2} + e_{l-2}$$

so

$$x_0 = \frac{\gamma_{l-2} - e_{l-2}}{2^{l-2}} - (2x_1 + 2x_2 + ...2^{l-2} x_{l-2})$$

$$\frac{\gamma_{l-2}}{2^{l-2}} - \frac{\gamma_{l-2} - e_{l-2}}{2^{l-2}} = \frac{e_{l-2}}{2^{l-2}}$$

as long as $e_{l-2} < q/8$, and set $q = 2^{l-1}$, so $\frac{e_{l-2}}{2^{l-2}} < 1/4$, so we can proof that

$$round\left(\frac{\gamma_{l-2} - e_{l-2}}{2^{l-2}}\right) = round\left(\frac{\gamma_{l-2}}{2^{l-2}}\right)$$

so we can get

$$x_0 = round\left(\frac{\gamma_{l-2}}{2^{l-2}}\right) mod 2$$

we can recover the left bits in this way,

$$x_1 = round\left(\frac{\gamma_{l-3} - 2^{l-3} x_0}{2^{l-2}}\right) mod 2$$

$$x_2 = round\left(\frac{\gamma_{l-4} - (2^{l-4} x_0 + 2^{l-3} x_1)}{2^{l-2}}\right) mod 2$$

$$\vdots$$

$$x_{l-2} = round\left(\frac{\gamma_0 - (2^0 x_0 + 2x_1 + ... + 2^{l-3} x_{l-3})}{2^{l-2}}\right) mod 2$$

Finally, we can get plaintext m by $m = \sum_{i=0}^{l-2} x_i 2^i$.

2.3 binary matrices encryption scheme

Ryo, Masayuki and Tatauaki [9] proposed a fully homomorphic encryption scheme that encrypts binary matrices and supports homomorphic matrix addition and multiplication. They use the multilinear maps to construct ciphertext matrix to satisfy decryption equation. For a secret matrix $S \in Z_q^{r\times(n+r)}$, The ciphertext $C \in \mathbb{Z}_q^{(n+1)\times(n+1)\cdot l}$ of matrix $M \in \{0,1\}^{r\times r}$ satisfy that $SC = MS + noise$. The scheme constructs the preimage of $MS + noise$ for the function $f_S(x) = Sx(modq)$, as $S[BR + \left(\frac{MS}{0}\right) G] = ER + MSG$, so the ciphertext C is a preimage of $BR + \left(\frac{MS}{0}\right) G$ for the function f_G, But the pliantext space is $M \in \{0,1\}^{r\times r}$, In this paper, We extend this scheme to encrypt integer matrices.

3 Integer Matrices Homomorphic Scheme

In this section, we firstly present a modifield scheme which can encrypt matrices with integer element, basing on [9] encrypting binary matrix. We revised the encyption algorithm and redesign the decryption algorithm, the shceme can proceed homomorphic operation of Addition and multiplication, then give the correctness and security analysis.

3.1 midifield scheme

The integer matrices encryption scheme can be split into two shemes including symmetric and public-key encryption systems, the key generation algorithms and the encryption algorithms are slightly different, so we present the schemes respectly.

3.1.1 public-key encryption scheme

-Setup:(1^λ): Our scheme is parameterized by an integer lattice dimension n, Let λ be security parameter, q is an integer modulus, a distribution χ over $\mathbb{Z}$, the parameter above depends on λ. Let $l := \lceil \log_2 q \rceil$, the message space is $\mathbb{Z}_q^{r\times r}$, $m := O((n+r)\log q)$, $N := (n+r)\cdot l$, The ciphertext space is $\mathbb{Z}_q^{(n+r)\times N}$, $g^T = (1, 2, ...2^{l-1})$, and $G = g^T \otimes I_{n+r}$,This algorithm outputs parameters above.
The key generation procedure is described as two steps: secret key generation and public key generation.

-KeyGenSec($params$): The input is the parameters generated from the Setup procedure, sample $\bar{S}$ from Gassian distribution $\chi^{r\times n}$, I_r is the identity matrix with r order, we concatenate I_r and $\bar{S}$, output secret key $S := [I_r \parallel -\bar{S}]$ and $\bar{S}$.

-KeyGenPub($params, \bar{S}$): This step get the input including parameters, $\bar{S}$, A is a random matrix sampled uniformly $A \xleftarrow{U} \mathbb{Z}_q^{r\times(n+r)}$, $\mathbb{Z}_q$ is defined as

$\mathbb{Z} \cap [-q/2, q/2)$,the noise matrix $E \xleftarrow{R} \chi^{r\times m}$, Let

$$B := \begin{pmatrix} \bar{S}A + E \\ \hline A \end{pmatrix} \in \mathbb{Z}_q^{(n+r)\times m}$$

Set $M_{(i,j)}$ is a $r \times r$ matrix with 1 in the $(i,j)-th$ position and 0 in the others, and $R_{(i,j)}$ is a random $\{0,1\}$ matrix, $R_{(i,j)} \xleftarrow{U} \{0,1\}^{m\times N}$ and compute

$$P_{(i,j)} := BR_{(i,j)} + \begin{pmatrix} M_{(i,j)}S \\ \hline 0 \end{pmatrix} G \in \mathbb{Z}_q^{(n+r)\times N}$$

output public key $pk := (P_{(i,j),i,j\in r}, B)$.

-PubIntEnc($params, pk, M$**)**: Plaintext matrix $M \in \mathbb{Z}_q$, $M[i][j]$ donotes the $(i,j) - th$ element value in M, sample $R_{(i,j)} \xleftarrow{U} \{0,1\}^{m\times N}$, output the ciphtext matrix

$$C := BR + \sum_{i,j\in[r]} M[i][j] \cdot P_{(i,j)} \in \mathbb{Z}_q^{(n+r)\times N} \tag{3.1}$$

-IntDec($params, C, S$**)**: The algorithm inputs parameters, ciphtext and secret key, observe that $SC = MSG + noise$, The frist $r \cdot l$ rows of $(MSG)^T$ is

$$\begin{bmatrix} m_{00} & m_{10} & ... & .... & m_{r0} \\ 2\cdot m_{00} & 2\cdot m_{10} & ... & ... & 2\cdot m_{r0} \\ ... & ... & ... & ... & ... \\ 2^{(l-1)}m_{00} & 2^{(l-1)}m_{10} & ... & ... & 2^{(l-1)}m_{r0} \\ m_{01} & m_{11} & ... & .... & m_{r1} \\ 2\cdot m_{01} & 2\cdot m_{11} & ... & ... & 2\cdot m_{r1} \\ ... & ... & ... & ... & ... \\ 2^{(l-1)}m_{01} & 2^{(l-1)}m_{11} & ... & ... & 2^{(l-1)}m_{r1} \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ m_{0r} & m_{1r} & ... & .... & m_{rr} \\ 2\cdot m_{0r} & 2\cdot m_{1r} & ... & ... & 2\cdot m_{rr} \\ ... & ... & ... & ... & ... \\ 2^{(l-1)}m_{0r} & 2^{(l-1)}m_{1r} & ... & ... & 2^{(l-1)}m_{rr} \end{bmatrix}$$

whose the frist column and the frist l rows is correspondence to the frist column and the frist l rows of $(SC)^T$, when the noise is very small. This have the same form as Cs in algorithm GSW.MPDec, so we can call GSW.MPDec to recover every bit of m_{00}. Besides, the following l rows of $(MSG)^T$ is correspondence to the respect $(SC)^T$, so we can call GSW.MPDec again to recover every bit of m_{01}, in this way every element in matrix M will be got along rows.

Homomorphic addition is $C_1 \bigoplus C_2 = C_1 + C_2$.

Homomorphic multiplication is $C_1 \bigodot C_2 = C_1 \cdot G^{-1}(C_2)$.

3.1.2 asymmetric encryption scheme

The scheme contains four algorithms: Setup, KeyGenSec, SecEnc and IntDec. The Setup, KeyGenSec and IntDec are the same as the public-key enryption scheme.
-SecEnc($M, params, S, \bar{s}$): The input is M, $parameters$, secret key S and $\bar{S}$, sample a random matrix $\bar{A} \xleftarrow{U} \mathbb{Z}_q^{n\times N}$ and $E \xleftarrow{R} \chi^{r\times N}$, the output is:

$$C := \begin{pmatrix} \bar{S}\bar{A} + E \\ A \end{pmatrix} + \begin{pmatrix} MS \\ 0 \end{pmatrix} G \in \mathbb{Z}_q^{(n+r)\times N}$$

Homomorphic addition is $C_1 \bigoplus C_2 = C_1 + C_2$.
Homomorphic multiplication is $C_1 \bigodot C_2 = C_1 \cdot G^{-1}(C_2)$.

3.2 correctness and security analysis

The correctness of decryption will be guaranteed by the following lemma.

Lemma 3.1. *If a ciphertext C in a public-key encryption scheme encrypts a plaintext matrix $M \in \mathbb{Z}_q^{r\times r}$ and let $\sum_{i,j\in[r]} M[i][j] = \mu$, and the noise matrix E as long as such that $\|E\|_\infty \cdot (\mu + 1) < q/8$, then $IntDec_{sk}(C) = M$.*

Proof. Put equation (3.1) into SC, as $SB = E$, so

$$\begin{aligned} SC &= S(BR + \sum_{i,j\in[r]} M[i][j] \cdot P_{(i,j)}) \\ &= S(BR + \sum_{i,j\in[r]} M[i][j] \cdot \left(BR_{(i,j)} + \frac{M_{(i,j)}S}{0}\right) G) \\ &= ER + \sum_{i,j\in[r]} M[i][j] \cdot ER_{(i,j)} + \sum_{i,j\in[r]} M[i][j] \cdot (1, -\bar{S}) \left(\frac{M_{(i,j)}S}{0}\right) G \\ &= E(R + \sum_{i,j\in[r]} M[i][j] \cdot R_{(i,j)}) + \sum_{i,j\in[r]} M[i][j] \cdot M_{(i,j)}SG \\ &= E(R + \sum_{i,j\in[r]} M[i][j] \cdot R_{(i,j)}) + MSG \end{aligned}$$

Let $noise = E(R + \sum_{i,j\in[r]} M[i][j] \cdot R_{(i,j)})$, if $\|noise\|_\infty < q/8$, i.e. $\|E(R + \sum_{i,j\in[r]} M[i][j] \cdot R_{(i,j)})\|_\infty < q/8$, so $\|E\|_\infty \cdot (\mu + 1) < q/8$, according to the GSW scheme, the decryption will be correct.

□

The symmetric encryption scheme is similar with public- key encryption scheme, so no more detailed description here.

The security of the two schemes holds from $DLWE_{n,q,\chi}$ problem directly, and circular security definition, the detailed come from [9] Lemma 4. In the public-key scheme, the algorithm encrypts the bases of every element in plaintext matrix in the public key generation step. According to circular security definition in [9], lake nothing to adversary, so the security of the scheme can be hold.

4 Implementation

We have accomplished four homomorphic encryption schemes including binary matrices public-key and symmetry encryption scheme from literature [9], and integer matrices public-key and symmetry encryption scheme from section 3. We show the results of the implementation of these schemes and analyze the efficiency.

4.1 experimental platform

We implemented the schemes in python 2.7 using pycharm community 2018.1.3, and run the algorithms on computer with 64-bit double cores(i7-7500U) at 2.7GHz and 2.9GHz, and 8GB RAM. The parameters chosen as follows:

Table 1: Parameters selection in encryption schemes

Parameter	this work
q	2^{30}
n	16
l	30
r	32
m	1440
N	1440
var	10

The four schemes contain four algorithms respectly, each algorithm runs 20 times and takes the average as the result, table 2 shows the running time of binary matrices encrypion scheme, and table 3 shows the running time of integer matrices encryption scheme.

Table 2: running time of binary matrices encryption scheme(sec)

Scheme Type	Setup	KeyGen	enc	dec
public-key encryption	0.000331	76.313	0.274	0.00436
symmetric encryption	0.0002402	0.0960	0.0961	0.00474

Table 3: runing time of integer matrices encryption scheme running time(sec)

Scheme Type	Setup	KeyGen	enc	dec
public-key encryption	0.00029	143.104	0.764	0.117
symmetric encryption	0.000285	0.0894	0.0984	0.129

In the symmetirc encryption schemes, binary matrices and integer matrices have almost the same running time in setup, KeyGen and enc algorithms, but in dec algorithm, the latter is slower than the former.
In the public-key encyption schemes, key generation step consumes the most time about 76s in binary matrices and 143s in integer matrices, it costs the most time in two types of schemes. Figure 2 shows in public-key integer matrices encryption scheme, how the time of key generation grows with the size of matrices changes. Besids, for $r = 32$ we computed the space occupied in memory of the secret key, public key and ciphertext, table 4 shows the details. We conclude that key generation takes a lot of time and space in public-key integer matrices scheme, which is a problem to research in future work.

Table 4: key and ciphertext size $r = 32$ (MB)

type	sksize	pksize	ctsize
size	6	877.5	67.5

Table 5 shows the homomorphic operations time consuming in public-key integer matrices encryption scheme. For a plaintext matrices 32×32, the multiplication costs 0.25sec, and addition just costs 0.09sec, as the homorphic operation just natural matrices computations, the results confrim our theory above. Futhermore, Figure 1 shows that homomorphic multiplication time grows over the size of matrices grows. We can see that for $r = 8$, the multiplication just takes 0.06sec and for $r = 40$, the time takes 0.32sec, the time increases approximately linearly with the size of matrices, which provides applications for homomorphic evaluations.

Table 5: Homomorphic operation running time in public-key integer matrices encryption scheme(sec)

Operation	Setup	KeyGen	enc	dec	homomorphic
Multiplication	0.000446	125.711	1.425	0.115	0.251
Addition	0.000286	129.758	1.342	0.1197	0.0917

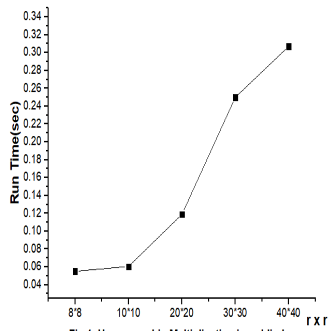

Fig.1. Homomorphic Multiplication in public-key integer matrices scheme

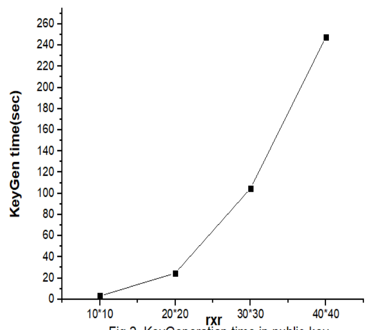

Fig.2. KeyGeneration time in public-key integer matrices scheme

5 Application

Matrices encryption schemes have a lot of applications, such as graph theory, one of the application scenarios is to homomorphicly compute the number of legth-k walks between any two vertexes, given an undirected graph. In graph theory, the plaintext operation to get the solution of this problem is described in [15] [5] through the powers of adjacent matrix. As an application of integer matrices, we present the algorithm using integer matrices public-key encryption in encrypted graph to homomorphiclly compute the number of walks of length less than $k(=3)$ between any two vertexes. The detail is in Algorithm 1.

Algorithm 1 homomorphiclly compute the number of walks of length less than $k(=3)$ between any two vertexes

```
Input: Number of vertices $r$
Output: The Matrix $\bar{A}$, denotes the number of walks of length less than 3
    between any two vertexes
 1: Generate randomly an adjacent matrix $A$ with $r$ vertexes;
 2: Call algorithmn Setup, KeyGen in sequence;
 3: Call algorithmn PubIntEnc to get $E_{pk}(A)$;
 4: Call homorphic multiplication algorithmn $E_{pk}(A^2) = mult(E_{pk}(A), E_{pk}(A))$
    and $E_{pk}(A^3) = mult(E_{pk}(A^2), E_{pk}(A))$;
 5: Call homomorphic addition algorithmn $E_{pk}(A') = add(E_{pk}(A^2), E_{pk}(A^3))$;
 6: Compute $A^2 = A \cdot A$, $A^3 = A^2 \cdot A$, $A^{'} = A^2 + A^3$ in sequence;
 7: Call algorithmn IntDec to get $\bar{A} = IntDec(params, E_{pk}(A'), S)$
 8: Compare matrices $(\bar{A})$ and $A^{'}$;
 9: if $(\bar{A}) == A^{'}$ then
10:     return $\bar{A}$
11: end if
```

6 conclusion

In this paper, we extend binary matrices encryption schemes to integer matrices for improving the efficiency of encryption algorithm with homomorphic operations. We proved the correctness and analyze the security of the schemes, and implement four encryption schemes to show the efficiency of the schemes. Finally, as an application, design an algorithm to solve a encryped graph theory problem.
The fellowing work has three aspects, reduce the cost of space and time in key generation to improve the efficiency of the public-key encryption scheme. Analyze the growth of noise to make the scheme fully homomorphic. Design specific application scenarios using the matrices encryptiion schemes to make homomorphic encryption more practical.

References

[1] J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *International Cryptology Conference*, pages 297–314, 2014.

[2] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Cryptology Conference on Advances in Cryptology — CRYPTO*, pages 868–886, 2012.

[3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *Acm Transactions on Computation Theory*, 6(3):1–36, 2014.

[4] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Foundations of Computer Science*, pages 97–106, 2011.

[5] A. Duncan. Powers of the adjacency matrix and the walk matrix. *The Collection*, pages 9, 4–11, 2004.

[6] Gentry and Craig. Fully homomorphic encryption using ideal lattices. *Stoc*, 9(4):169–178, 2009.

[7] C. Gentry. *A fully homomorphic encryption scheme.* Stanford University, 2009.

[8] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Cryptology Conference*, pages 75–92, 2013.

[9] R. Hiromasa, M. Abe, and T. Okamoto. *Packing Messages and Optimizing Bootstrapping in GSW-FHE.* Springer Berlin Heidelberg, 2015.

[10] A. Khedr, G. Gulak, and V. Vaikuntanathan. Shield: Scalable homomorphic implementation of encrypted data-classifiers. *IEEE Transactions on Computers*, 65(9):2848–2858, 2016.

[11] J. Liu, N. Asokan, and B. Pinkas. Secure deduplication of encrypted data without additional independent servers. In *ACM Sigsac Conference on Computer and Communications Security*, pages 874–885, 2015.

[12] Naehrig, Michael, Lauter, Kristin, Vaikuntanathan, and Vinod. Can homomorphic encryption be practical? *Proc Ccsw*, pages 113–124, 2011.

[13] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the Acm*, 56(6):1–40, 2009.

[14] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–179, 1978.

[15] D. B. West. *Introduction to graph theory, 2nd edition.* Prentice-Hall Inc, 1996.

[16] G. Xu, Y. Ren, H. Li, D. Liu, Y. Dai, and K. Yang. Cryptmdb: A practical encrypted mongodb over big data. In *IEEE International Conference on Communications*, pages 1–6, 2017.

[17] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. vsql: Verifying arbitrary sql queries over dynamic outsourced databases. In *Security and Privacy*, pages 863–880, 2017.