

# 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 29, 2018

## Abstract

We construct an integer matrices encryption scheme based on binary matrices encryption scheme proposed in Hiromasa et al.(PKC 2015). Our scheme supports 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 Hiromasa et al.(PKC 2015), and public-key and symmetric-key integer matrices encryption schemes from this work. The experimental results show that the running time of homomorphic multiplication just costs 0.32sec for integer matrix with  $40 \times 40$  entry size. It provides a promising prospect for applications. Finally, we apply integer matrices encryption to homomorphically solve a problem that computes the number of length- $k$  walks between any two vertices of a graph. The algorithm shows the effectiveness.

## 1 Introduction

Homomorphic encryption is a ciphertext computation technology, which allows us to evaluate functions over encrypted 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 following thirty years there is no deterministic solutions. Until the year 2009, Gentry constructed the first 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], employing 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 2013. This scheme used eigenvector to construct encryption scheme, and can encrypt binary bits and integers. At present, this scheme is one of the fastest and simplest fully homomorphic encryption scheme compared with previous ones. 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. Hiromasa et al. [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, for most of applications we have to design complex circuit to achieve more homomorphic operations. Besides, for some application scenarios, efficiency is more significant than achieving full homomorphism. Therefore, it is meaningful to design the integer matrices encryption scheme. In this paper, our constructions are the following.

- We propose a novel encryption and decryption algorithms for the integer matrices encryption scheme, which supports homomorphic addition and multiplication operations in the form of integer matrices, rather than the traditional binary form.
- We prove 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 [13], the definition is:

**Definition 2.1.** For security parameter  $\lambda$ , 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 firstly draws  $s \xleftarrow{U} \mathbb{Z}_q^n$  uniformly, and then sample  $(a_i, b_i)$  by sampling  $a_i \xleftarrow{U} \mathbb{Z}_q^n$ ,  $e_i \xleftarrow{R} \chi$ , and setting  $b_i = \langle a_i, s \rangle + e_i$ .  
The  $DLWE_{n,q,\chi}$  assumption is that the  $DLWE_{n,q,\chi}$  problem is infeasible.

**Gadget matrix  $G$ :** Let  $l = \lceil \log q \rceil$  and identify  $\mathbb{Z}_q$  as  $0, 1, \dots, q-1$ . 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  $g^T = (1, 2, \dots, 2^{l-1})$ , and column vector  $x = (x_0, x_1, \dots, x_{l-1})^T \in \mathbb{Z}^l$  such that  $g^T x \equiv m \pmod q$ . 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  $G^{-1} : \mathbb{Z}_q^{n+1} \rightarrow \mathbb{Z}^{(n+1)l}$  describes below:

For any  $m \in \mathbb{Z}_q^{n+1} \rightarrow \mathbb{Z}^{(n+1)l}$  and short vector  $x \in \mathbb{Z}^{(n+1)l}$ , such that  $Gx \equiv m \pmod q$  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  $Gx \equiv m \pmod q$  will be satisfied.

## 2.2 GSW Fully Homomorphic Encryption scheme

The GSW fully homomorphic encryption scheme use approximate eigenvector method to construct ciphertext  $C$  such that secret key  $s$  is an approximate 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  $m \geq nl$

**GSW.keyGenSec**( $params$ ): Choose  $\bar{s} \xleftarrow{U} \mathbb{Z}_q^n$ , and output  $sk = (1, -\bar{s}) \in \mathbb{Z}_q^{n+1}$

**GSW.keyGenPub**( $params$ ): Choose a  $n \times m$  random matrix  $A \in \mathbb{Z}_q^{n \times m}$ , and pick  $e_i \xleftarrow{R} \chi^m$  at random. Compute

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

Then the public key is  $B = \begin{pmatrix} b^T \\ A \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times m}$ , observe that  $sB = e^T \pmod q$

. **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

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

where  $G$  is the Gadget matrix and  $B$  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 outputs plaintext  $m = \lfloor \langle s, c \rangle \rfloor_2$ , where  $\lfloor \cdot \rfloor_2$  denotes that  $\mathbb{Z}_q \rightarrow \{0, 1\}$  if the inner product is closer to 0, the result is 0, otherwise the inner product is closer to  $q/4$ , the result is 1. As  $s^T c = ms^T G + s^T BR = ms^T G + e^T R \pmod q$ , We need  $e^T R$  to be small, let  $noise = e^T R$ . If  $\|noise\|_\infty < q/8$ , then we can recover  $m$ , where  $\|x\|_\infty$  is the maximum norm of a vector  $x$ .

**GSW.MPDEC**( $C, sk$ ): This algorithm can recover plaintext  $m \in \mathbb{Z}_q$ , we will show the decryption process below. Taking the first  $l - 1$  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

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

so

$$\begin{aligned} x_0 &= \frac{\gamma_{l-2} - e_{l-2}}{2^{l-2}} - (2x_1 + 2^2x_2 + \dots 2^{l-2}x_{l-2}) \\ &\equiv \frac{\gamma_{l-2} - e_{l-2}}{2^{l-2}} \pmod 2 \end{aligned}$$

Since  $e_{l-2} < q/8$ , and as  $q = 2^{l-1}$ ,  $\frac{e_{l-2}}{2^{l-2}} < 1/4$ , which implying

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

so we have

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

for  $i = 1, 2, \dots, l - 2$ , we can recover the rest of the bits in this way,

$$x_i = \text{round}\left(\left(\frac{\gamma_{l-2-i} - \sum_{j=1}^i 2^{l-i+j-3}x_{j-1}}{2^{l-2}}\right)\right) \pmod 2$$

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

### 2.3 Binary matrices encryption scheme

Hiromasa et al. [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  $S \in \mathbb{Z}_q^{r \times (n+r)}$ , 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  $f_S(x) = Sx \bmod q$ , as  $S[BR + (\frac{MS}{0})G] = ER + MSG$ , so that the ciphertext  $C$  is a preimage of  $BR + (\frac{MS}{0})G$  for the function  $f_G$ . But the plaintext 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 modified scheme which can encrypt matrices with integer elements, based on the binary encryption scheme in [9]. We revise the encryption algorithm and redesign the decryption algorithm, so that the scheme can proceed homomorphic operation of addition and multiplication, then give the correctness and security analysis.

#### 3.1 Midfield scheme

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

##### 3.1.1 Public-key encryption scheme

**-Setup:**( $1^\lambda$ ): Our scheme is parameterized by an integer lattice dimension  $n$ . Let  $\lambda$  be security parameter. Let  $q$  be an integer modulus,  $\chi$  a distribution over  $\mathbb{Z}$ . The parameter above depends on  $\lambda$ . Let  $l := \lceil \log_2 q \rceil$ . The message space is  $\mathbb{Z}_q^{r \times r}$ . Let  $m := O((n+r) \log q)$ ,  $N := (n+r) \cdot l$ . The ciphertext space is  $\mathbb{Z}_q^{(n+r) \times N}$ . Let  $g^T = (1, 2, \dots, 2^{l-1})$ , 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 Gaussian distribution  $\chi^{r \times n}$ . Let  $I_r$  is the identity matrix with  $r$  order. We concatenate  $I_r$  and  $\bar{S}$ , and output secret key  $S := [I_r \parallel -\bar{S}]$  and  $\bar{S}$ .

**-KeyGenPub**( $params, \bar{S}$ ): This step gets the input including parameters and  $\bar{S}$ . Let  $A$  be 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

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

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

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

Output public key  $pk := \{(P_{(i,j)}, B) | 1 \leq i, j \leq r\}$ .

**-PubIntEnc**( $params, pk, M$ ): Plaintext matrix  $M \in \mathbb{Z}_q^{r \times r}$ ,  $M[i][j]$  denotes the  $(i, j)$  element value of  $M$ . Sample  $R_{(i,j)} \xleftarrow{U} \{0, 1\}^{m \times N}$ , and output the ciphertext matrix

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

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

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

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

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

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

### 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 encryption 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  $E \xleftarrow{R} \chi^{r \times N}$ , and 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  $C_1 \oplus C_2 = C_1 + C_2$ .

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

### 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 the public-key encryption scheme encrypts a plaintext matrix  $M \in \mathbb{Z}_q^{r \times r}$  with  $\sum_{i,j \in [r]} M[i][j] = \mu$ , and the noise matrix  $E$  satisfying  $\|E\|_\infty \cdot (\mu + 1) < q/8$ , then  $\text{IntDec}_{sk}(C) = M$ .*

*Proof.* Substituting  $C$  by (3.1) and combining  $SB = E$ , we have

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

Let  $\text{noise} = E(R + \sum_{i,j \in [r]} M[i][j] \cdot R_{(i,j)})$ . Since  $\|\text{noise}\|_\infty < q/8$ , 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 [8], the decryption is correct.  $\square$

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 in section 2.1, and circular security definition in [9] section 2.3 Definition 2. The security can be guaranteed by [9] [Lemma 4].

**Lemma 3.2.** [9] [Lemma 4]. *Let  $B, M_{(i,j)}, R_{(i,j)}, R_{(i,j)}, P_{(i,j)}, (i, j = 1, \dots, r)$  be the matrices generated in  $\text{KeyGen}$ , and  $R$  be the matrix generated in  $\text{PubEnc}$ . For every  $i, j = 1, 2, \dots, r$ , if the scheme is circular secure with respect to  $f_M$*

and  $DLWE_{n,q,\chi}$  holds, then the joint distribution  $(B, BR_{(i,j)}, P_{(i,j)}, BR)$  is computationally indistinguishable from uniform over  $\mathbb{Z}_q^{(n+r) \times m} \times \mathbb{Z}_q^{(n+r) \times N} \times \mathbb{Z}_q^{(n+r) \times N} \times \mathbb{Z}_q^{(n+r) \times N}$ .

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], leak 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 schemes from literature [9], and integer matrices public-key and symmetry encryption schemes 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 respectively. Each algorithm runs 20 times and takes the average as the result. Table 2 shows the running time of binary matrices encryption scheme. 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 encryption schemes, the key generation step consumes 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 increases. Besides, 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

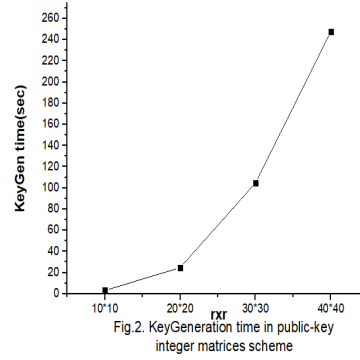
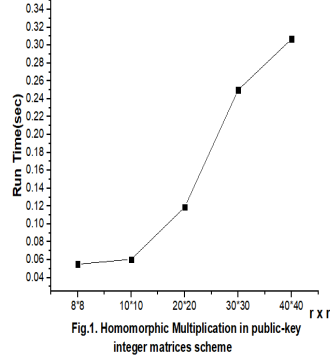
Table 5 shows the homomorphic operations time consuming in public-key integer matrices encryption scheme. For a plaintext matrices  $32 \times 32$ , the multiplication costs 0.25sec, and addition just costs 0.09sec. As the homorphic operation are just natural matrices computations, the results confirm our theoretical analysis 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, 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

## 5 Application

Matrices encryption schemes have a lot of applications, such as graph theory. One of the application scenarios is to homomorphically compute the number of



length- $k$  walks between any two vertices, 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, we present an algorithm using integer matrices public-key encryption in encrypted graph to homomorphically compute the number of walks of length less than  $k(= 3)$  between any two vertices. The detail is in Algorithm 1.

---

**Algorithm 1** Homomorphically compute the number of walks of length less than  $k(= 3)$  between any two vertices

---

Input: an graph  $G$  with  $r$  vertices

Output: The Matrix  $\bar{A}$ , denotes the number of walks of length less than 3 between any two vertices

- 1: Find the adjacency matrix  $A$  of graph  $G$ ;
  - 2: Call algorithmn Setup, KeyGen in sequence;
  - 3: Call algorithmn PubIntEnc to get  $E_{pk}(A)$ ;
  - 4: Call homomorphic 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: Call algorithmn IntDec to get  $\bar{A} = IntDec(params, E_{pk}(A'), S)$
  - 7: return  $\bar{A}$
- 

## 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, we design an algorithm to solve a encrypted graph theory problem.

The future work has three aspects: reducing the cost of space and time in key generation to improve the efficiency of the public-key encryption scheme; Analyzing the growth of noise to make the scheme fully homomorphic; Designing specific application scenarios using the matrices encryption 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.