DOI 10.1007/s10958-019-04162-8
Journal of Mathematical Sciences, Vol. 237, No. 3, March, 2019

APPLIED HOMOMORPHIC CRYPTOGRAPHY: EXAMPLES

G. G. Arakelov, A. V. Gribov, and A. V. Mikhalev UDC 004.056.55

Abstract. This paper is devoted to the application aspects of homomorphic cryptography. It provides a description of a fully homomorphic matrix polynomial-based encryption scheme. It also gives the results of practical comparison of fully homomorphic encryption schemes. We consider some special cases of homomorphic encryption allowing computations of a limited number of functions.

Introduction

One of the most interesting and important tasks of modern cryptography is computations on encrypted data without its preliminary decryption. The issue of such computations' possibility in principle has remained open for a long time, and it should be mentioned that the authors of the RSA encryption scheme supposed that such computations were impossible in principle. The branch of cryptography devoted to the schemes allowing computations on ciphertexts is commonly referred to as homomorphic cryptography, and the corresponding schemes as homomorphic schemes. A distinction is made between fully homomorphic and partially homomorphic encryption schemes. In a fully homomorphic encryption scheme, the operations of addition and multiplication of ciphertexts are homomorphic. To be more exact, the following equalities hold:

$$D\big(E(m_1)\cdot E(m_2)\big) = m_1\cdot m_2, \tag{1}$$

$$D\big(E(m_1)+E(m_2)\big) = m_1+m_2, \tag{2}$$

where $E(\cdot)$ is the encryption functions and $D(\cdot)$ is the decryption function.

If a certain encryption scheme meets at least one of the two conditions, such a scheme is referred to as partially homomorphic. Partially homomorphic schemes are quite widespread. For example, the RSA scheme itself is homomorphic relative to the multiplication operation as well as Elgamal scheme. The RSA scheme and Elgamal scheme are partially homomorphic, i.e., homomorphic relative to the operation of ciphertext multiplication.

The first model of a fully homomorphic encryption system was suggested by Craig Gentry in 2009. The work by Gentry proved the possibility of building up fully homomorphic encryption systems in principle. The model suggested by Gentry requires significant computation costs for its practical implementation. And, in spite of the fact that after the work by Gentry was issued a lot of other works appeared to suggest various improvement of the Gentry model, no fully homomorphic encryption system that can be widely used in practice has appeared yet. Apart from fully homomorphic encryption systems, there are schemes allowing homomorphic computations in certain cases.

Building up cryptographic schemes based on nonassociative structures is a relatively new area in cryptography. An encryption system retaining homomorphism is suggested in [10], where a cryptographic system is built on the basis of a quasigroup ring. In [11], the authors use nonassociative groupoids for implementation of the open key distribution.

This paper touches upon certain models of homomorphic encryption schemes, which are useful from the practical point of view. One of the interesting encryption schemes having practical value was suggested in [3]; it is based on matrix polynomials. The benefit of this model is that all the computations on the

Translated from Fundamentalnaya i Prikladnaya Matematika, Vol. 21, No. 3, pp. 25–38, 2016.

encrypted data are limited to addition and multiplication of matrices, and such operations are known to allow wide paralleling. The possibility of computation paralleling improves the practical importance of an encryption scheme.

The first part of this paper reviews a matrix polynomial-based cryptographic scheme. It provides results of experiments for comparison of such a scheme vs. an improved version of the Gentry cryptographic system described in [14]. The second and third parts of the paper feature some special cases of homomorphic encryption allowing computations of a limited number of functions. The fourth part reviews a protected relational database model based on a fully homomorphic encryption scheme.

1. Matrix Polynomial-Based Cryptographic System

1.1. Basic Notions and Definitions. An interesting model of fully homomorphic encryption was suggested in [3]. The suggested model is based on a ring of matrix polynomials. The open texts are elements of residue field $\mathbb{Z}_p$. Even though it ranks slightly below the model suggested in [11] in asymptotic estimate of computational complexity, the analysis conducted shows its practical value.

Let us recall the key definitions to be used.

A matrix polynomial over a ring of $\mathbb{Z}_p$ $N \times N$ square matrices is a matrix sequence

$$F = \{A_0, A_1, A_2, \ldots\}, \quad A_i \in \mathbb{Z}_p^{N\times N},$$

such that among A_i there are only a finite number of nonzero matrices.

Let $F = \{F_1, F_2, \ldots\}$ and $G = \{G_1, G_2, \ldots\}$ be matrix polynomials over the ring $\mathbb{Z}_p$. The addition and multiplication operations on matrix polynomials are determined as follows:

$$F + G = \{F_0 + G_0, F_1 + G_1, \ldots\},$$

$$F \cdot G = \{F_0 \cdot G_0, F_0 \cdot G_1 + F_1 \cdot G_0, F_0 \cdot G_2 + F_1 \cdot G_1 + F_2 \cdot G_0 \ldots\}, \text{ i.e., } [F \cdot G]_k = \sum_{i+j=k} F_i \cdot G_j.$$

The set of matrix polynomials over the ring of $N \times N$ matrices over $\mathbb{Z}_p$ with the addition and multiplication operations is itself a ring. This ring will be designated as $\mathbb{Z}_p^{N\times N}[X]$

A reduced matrix polynomial is a matrix polynomial whose leading coefficient is the identity matrix.

Let

$$F = A_n X^n + A_{n-1} X^{n-1} + \cdots + F_0$$

be some matrix polynomial. We can search for such a matrix K, $K \in \mathbb{Z}_p^{N\times N}$, where after the substitution of K for X we obtain

$$F(K) = F_n \cdot K^n + F^{n-1} \cdot K^{n-1} + \cdots + F_0 = 0,$$

where 0 means the zero matrix.

Note that the equation of $F(X) = 0$ may have more roots than its degree or may not have any roots at all.

For any matrix polynomial $C(X)$ and an arbitrary reduced polynomial $K(X)$ such that $\deg\big(C(X)\big) > \deg\big(K(X)\big)$, there exists a unique representation in the form of $C(X) = K(X)Q(X) + R(X)$, where $\deg\big(R(X)\big) < \deg\big(K(X)\big)$ (see, e.g., [8]). To ensure complete presentation, let us give the algorithm of division by a reduced matrix polynomial.

Algorithm of division by a reduced matrix polynomial. Let us divide an arbitrary matrix polynomial $C(X) \in \mathbb{Z}_p^{N\times N}$ by a reduced matrix polynomial $K(X) \in \mathbb{Z}_p^{N\times N}$. In this case, we need to act as follows.

(1) Assign the value of 0 to $Q(X)$, i.e., all the coefficients of the polynomial will be equal to zero: $Q(X) := 0$.

(2) Multiply $K(X)$ by $X^{\deg(C(X))-\deg(K(X))}$ and by a matrix $A \in \mathbb{Z}_p^{N\times N}$ that makes the leading coefficients of the polynomials $C(X)$ and $A \cdot K(X) \cdot X^{\deg(C(X))-\deg(K(X))}$ equal. Assign a new value to $Q(X)$: $Q(X) := Q(X) + A \cdot X^{\deg(C(X))-\deg(K(X))}$.

(3) Assign a new value to $C(X)$: $C(X) := C(X) - A \cdot K(X) \cdot X^{\deg(C(X))-\deg(K(X))}$.

(4) If $\deg\big(K(X)\big) > \deg\big(C(X)\big)$, then we return the polynomial $Q(X) \cdot K(X) + C(X)$ as the result, otherwise go to step (2).

As the matrix polynomial $K(X)$ is reduced, i.e., its leading coefficient is the identity matrix, step (2) will always be uniquely performable while $\deg\big(K(X)\big) < \deg\big(C(X)\big)$.

1.2. Building up a Cryptographic System. Let us denote the parameter setting the cryptoresistability level by $\lambda \in \mathbb{N}$. The plaintext space is $\mathbb{Z}_p$, where p is a prime number. The ciphertext space is $\mathbb{Z}_p^{N\times N}[X]$, i.e., ciphertexts are matrix polynomials, where $N = O(\lambda)$. The secret key space is $\mathbb{Z}_p^{N\times N}[X] \times \mathbb{Z}_p^N$, i.e., a secret key is a pair $(K(X), k)$, where k is an N-dimensional vector of integers modulo p. Apart from the secret key, this scheme also uses a superencryption key. It is some matrix polynomial $\mathrm{rk} \in \mathbb{Z}_p^{N\times N}[X]$ transmitted to the computation side to reduce the size of ciphertexts. Now we can switch to description of the cryptographic scheme algorithms.

1.2.1. Secret key generation.

(1) One generates a rootless reduced polynomial $K(X) \in \mathbb{Z}_p^{N\times N}[X]$ such that $\deg\big(K(X)\big) = O(\lambda)$.
(2) A vector $k \in \mathbb{Z}_p^N$ is generated. As $\mathbb{Z}_p$, where p is a prime number, is a field, each component of the vector k is invertible.
(3) The pair $(K(X), k)$ is retained as a secret key.

1.2.2. Superencryption key generation. A superencryption key is used to prevent an increase in the size of ciphertexts. After multiplying ciphertexts the result is provided modulo the superencryption key.

(1) A reduced matrix polynomial $R'(X) \in \mathbb{Z}_p^{N\times N}[X]$ is generated such that $\deg\big(R'(X)\big) = O(\lambda)$.
(2) A polynomial $\mathrm{rk}(X) = R'(X) \cdot K(X)$ is retained as an encryption key.

1.2.3. Encryption.

(1) The plain text $m \in \mathbb{Z}_p$ is associated with a matrix $M \in \mathbb{Z}_p^{N\times N}$ such that $M \cdot \vec{k} = m \cdot \vec{k}$ and $M \cdot K(X) = K(X) \cdot M$, i.e., the matrix M has its own vector $\vec{k}$ with its own value m and commutes with the matrix polynomial $K(X)$.
(2) A matrix polynomial $R(X) \in \mathbb{Z}_p^{N\times N}[X]$ is generated, where $\deg\big(R(X)\big) = O(\lambda)$ and $\deg\big(R(X)\big) < \deg\big(R'(X)\big)$.
(3) The ciphertext is computed: $C(X) = R(X)K(X) + M$.

1.2.4. Decryption.

(1) The matrix $M = C(X) \mod K(X)$ is computed for the ciphertext $C(X)$.
(2) Any invertible component of the vector $\vec{k}$ is selected, let it be k_i, and $m = \big(k_i^{-1}(M \cdot \vec{k})\big)_i$ is computed.

Here we will not provide the proof that this scheme is correct. This proof is provided in detail in [3]. We are interested in homomorphic properties of the described cryptographic system. Let us verify that we are really dealing with full homomorphism, i.e., that homomorphism exists for both addition and multiplication of ciphertexts.

1.3. Additive and Multiplicative Homomorphism.

Lemma 1. *Matrix polynomial-based encryption is fully homomorphic, i.e., the following equalities hold*:

$$D\big(E(m_1) + E(m_2)\big) = m_1 + m_2, \tag{3}$$

$$D\big(E(m_1) \cdot E(m_2)\big) = m_1 \cdot m_2, \tag{4}$$

where m_1 and m_2 are plain texts, $E(\cdot)$ is the encryption function, and $D(\cdot)$ is the decryption function.

Proof. Let us show that both additive and multiplicative homomorphism take place. Let $C_1(X) = R_1(X) \cdot K(X) + M_1$ and $C_2(X) = R_2(X) \cdot K(X) + M_2$ be two ciphertexts that refer to the texts m_1 and m_2, respectively.

Let us consider the sum of ciphertexts:

$$C_1(X) + C_2(X) = R_1(X) \cdot K(X) + M_1 + R_2(X) \cdot K(X) + M_2. \tag{5}$$

From (5) we see that

$$C_1(X) + C_2(X) = \big(R_1(X) + R_2(X)\big) \cdot K(X) + (M_1 + M_2). \tag{6}$$

Let us see what we will obtain after application of the decryption algorithm to the right-hand side of Eq. (6). By division of $C_1(X) + C_2(X)$ by $K(X)$ we obtain $M_1 + M_2$. After multiplication by the vector $\vec{k}$, we obtain

$$(M_1 + M_2) \cdot \vec{k} = M_1 \cdot \vec{k} + M_2 \cdot \vec{k} = m_1\vec{k} + m_2 \cdot \vec{k}. \tag{7}$$

Then, multiplying the vector obtained in (7) by k_i^{-1}, where i is selected arbitrarily, we obtain

$$\big(m_1\vec{k} + m_2 \cdot \vec{k}\big) \cdot k_i^{-1} = m_1\vec{k} \cdot k_i^{-1} + m_2 \cdot \vec{k} \cdot k_i^{-1}. \tag{8}$$

As a result, the ith position of the vector obtained in the expression (8) will correspond to the sum of $m_1 + m_2$, which is required to be demonstrated, whence it follows that the right-hand side of Eq. (6) is a ciphertext for the sum of plain texts $m_1 + m_2$ and after decryption will give the value of $m_1 + m_2$, which proves additivity of the described cryptographic scheme.

Now let us consider the product of ciphertexts:

$$C_1(X) \cdot C_2(X) = (R_1(X) \cdot K(X) + M_1) \cdot (R_2(X) \cdot K(X) + M_2). \tag{9}$$

In Eq. (9), $R'(X)K(X)$ is a superencryption key. From Eq. (9) we obtain

$$R_1(X) \cdot K(X) \cdot R_2(X) \cdot K(X) + R_1(X) \cdot K(X) \cdot M_2 + M_1 \cdot R_2(X) \cdot K(X) + M_1 \cdot M_2. \tag{10}$$

On the right-hand side of Eq. (10), we put the polynomial $K(X)$ and obtain

$$C_1(X) \cdot C_2(X) = (R_1(X) \cdot K(X) \cdot R_2(X) + R_1(X) \cdot M_2 + R_2(X) \cdot M_1) \cdot K(X) + M_1 \cdot M_2. \tag{11}$$

Let us see what we will obtain after polynomial decryption from Eq. (11). At the first step of decryption, after dividing $C_1(X) \cdot C_2(X)$ by the polynomial $K(X)$, we obtain a product of two matrices $M_1 \cdot M_2$. Then, after application of the next step of the decryption algorithm we obtain

$$(M_1 \cdot M_2) \cdot \vec{k} = M_1 \cdot \big(M_2 \cdot \vec{k}\big) = M_1 \cdot (m_2\vec{k}) = m_2\big(M_1 \cdot \vec{k}\big) = m_1 \cdot m_2 \cdot \vec{k}. \tag{12}$$

From (12) it follows that

$$D\big(E(m_1) \cdot E(m_2)\big) = m_1 \cdot m_2.$$

We have proved that addition and multiplication of ciphertexts have the property of homomorphism, i.e., the described encryption scheme is fully homomorphic. □

1.4. Computation Costs. The most expensive operation in this encryption scheme is multiplication of ciphertexts that are matrix polynomials and, therefore, the computational complexity of the whole system will directly depend on this operation. In turn, multiplication of matrix polynomials depends on two algorithms:

(1) matrix multiplication algorithm;

(2) polynomial multiplication algorithm.

The algorithm of polynomial multiplication suitable for the described scheme has the asymptotic complexity of operations on polynomial coefficients $O(d^{\log_2 3}) = O(d^{1.5849\ldots})$, where d is the highest degree of the polynomials. The algorithm of multiplication of two $N \times N$ matrices has the asymptotic complexity of elementary operations $O(N^{2.373\ldots})$.

Provided that $N = O(\lambda)$ and the polynomial degrees are equal to $O(\lambda)$, we obtain that the total number of operations on the elements of $\mathbb{Z}_p$ has the asymptotic complexity $\approx O(\lambda^{3.76})$.

Currently, the best estimate of the computation costs for homomorphic computation is $g(\lambda) = O(\lambda^{3.5})$. The scheme described in [14] has such computational complexity. It should be mentioned here that in the estimation of computational complexity of an encryption scheme based on matrix polynomials it was believed that multiplication of $N \times N$ matrices requires $O(N^{2.373\ldots})$. Indeed, there is an algorithm able to provide for multiplication of two matrices in the specified time, and this is the Coppersmith–Winograd algorithm improved by Williams. However, in practice the Coppersmith–Winograd algorithm may not be currently used, as its proportionality constant is too large and starts showing higher speed than other known algorithms only for matrices whose size exceeds the memory of modern computers.

On the other hand, the known Strassen hypothesis argues that for an arbitrarily small $\varepsilon > 0$ there is an algorithm ensuring large enough n multiplication of two $n \times n$ matrices in $O(n^{2+\varepsilon})$ operations.

The Strassen hypothesis is still one of the unsolved problems of linear algebra and if we believe it is true, then the cryptographic system described here suggested by F. B. Burtyka is currently the "fastest" known schemes of fully homomorphic encryption in terms of computations. A cryptographic system based on matrix polynomials is inferior to the system described in [14] in terms of asymptotic estimates, but, in terms of practice, it is of value, as it allows wide paralleling. For example, an experiment was conducted using the CUDA technology (Nvidia) for massively parallel computing to show the superiority of computations of a matrix polynomial cryptographic system in terms of time.

1.5. Use of Parallel Computations in Implementation. The encryption system described in [14] provides the best evaluation of computational costs in the case of homomorphic computations. IBM implemented the so-called homomorphic encryption library known as HElib (https://github.com/shaih/HElib). The library implements a cryptosystem which is currently considered to be asymptotically best among homomorphic encryption systems in terms of computational costs.

For the purpose of contrastive analysis of the encryption system described in the paper and modified Gentry's cryptosystem [14], the pilot matrix polynomial-based cryptographic system was implemented where the Nvidia CUDA parallel computing technology was used for multiplication of matrices and polynomials, and the modified Gentry's cryptosystem was obtained from the HElib library.

We performed a series of experiments involving various parameters of encryption strength. The time spent was estimated for the following operations:

(1) encrtyption;
(2) decryption;
(3) multiplication.

The results are listed in Tables 1 and 2. The mentioned performance evaluation results show that in practice the matrix polynomial-based encryption system is highly competitive with the advanced Gentry's encryption model and, furthermore, it compares favorably in the case of parallel computations. We have conducted an experiment for the maximum value of the parameter $\lambda = 64$, and even when $\lambda = 32$ we see that Gentry's model, believed to be the asymptotically best, is actually below Burtyka's model.

Table 1. Performance evaluation of the matrix polynomial-based encryption system with the use of parallel computing technology.

Parameter λ	Encryption	Decryption	Multiplication
16	4 ms	13 ms	8 ms
24	79 ms	13 ms	15 ms
32	1.5 s	14 ms	22 ms
64	2 min	20 ms	1 s

Table 2. Evaluation of the modified Gentry's cryptosystem performance.

Parameter λ	Encryption	Decryption	Multiplication
16	2 ms	6 ms	5 ms
24	40 ms	11 ms	12 ms
32	1 s	15 ms	50 ms
64	5 min	200 ms	10 s

2. Searchable Encryption

First, we considered the schemes that allow for keyword searching. The Song, Wagner, and Perrig (2000) and Curtmola, Garay, Kamara, and Ostrovsky (2006) schemes are the best option in terms of complexity and security [13]. A scheme with search by Boolean expressions suggested by Cash, Jarecki, Jutla (2013) can also be considered [4]. One of the latest studies resulted in a scheme that is already used in a number of applications in practice: Cash, Jaeger, Jarecki, Jutla, Krawczyk, Rosu, and Steiner (2014) [5].

Let us consider the key principle of such schemes through the example of a cryptosystem using the symmetric encryption system. The algorithm involves two sides: the client side and the server side. Suppose that $D = (D_1, \ldots, D_n)$ is an array, e.g., a set of text files, DB is a database containing correlation between all keywords w from D and corresponding identifiers $\mathrm{DB}[w]$, i.e., all files containing the word w. Then the encryption scheme consists of three algorithms $(\mathrm{Setup}, \mathrm{Token}, \mathrm{Search})$, where the algorithm $\mathrm{Setup}\colon (1^k, \mathrm{DB}) \to (K, \mathrm{EDB})$ generates a private key K and encrypts the database DB. The client side algorithm $\mathrm{Token}\colon (w, K) \to \mathrm{tk}_w$ generates a specific token for the given keyword, and the algorithm $\mathrm{Search}\colon (\mathrm{EDB}, \mathrm{tk}_w) \to \mathrm{DB}[w]$ is activated on the algorithm's server side and returns a range of identifiers containing the keyword w.

The algorithm Search uses the symmetric encryption system $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, pseudorandom function $F\colon \{0,1\}^k \times W \to \{0,1\}^k$, and pseudorandom permutation $P\colon \{0,1\}^k \times W \to \{1, |W|\}$. First, random keys K_t and K_f are generated for the respective transforms, and the storage space for T and RAM_1 arrays is allocated. For each word $w \in W$ and all $1 \le i \le |\mathrm{DB}[w]|$ the following array of elements is generated

$$N_{w,i} = \langle \mathrm{id}_{w,i}, \mathrm{ptr}_1(w, i+1) \rangle,$$

where $\mathrm{id}_{w,i}$, is the ith identifier in $\mathrm{DB}[w]$ and $\mathrm{ptr}_1(w, i+1)$ is an address in the array RAM_1. Then values from RAM_1 are inserted to random positions in the array RAM_2, and each value from RAM_2 is encrypted. For this purpose, we create a new array RAM_3, where for each $w \in W$ and all $1 \le i \le |\mathrm{DB}[w]|$ the following is performed:

$$\mathrm{RAM}_3[\mathrm{addr}_2(N_{w,i})] = \mathrm{Enc}_{K_w}(\mathrm{RAM}_2[\mathrm{addr}_2(N_{w,i})]),$$

where $K_w = F_{K_f(w)}$ and addr_2 is a function that returns the address of the argument in the array RAM_2. Now for all $w \in W$ keywords

$$T[P_{K_T}(w)] = \mathrm{Enc}_{K_w}\big(\mathrm{addr}_3(N_{w,1})\big),$$

where addr_3 is a function that returns the address of the argument in the array RAM_3. The final step on the client side is the generation of $\mathrm{EDB} = (T, \mathrm{RAM}_3)$. Such array is stored on the server side and used for searching keywords in the array. Now, in order to find a keyword w in the array EDB stored on the server side, it is required to calculate a token

$$\mathrm{tk} = (\mathrm{tk}_1, \mathrm{tk}_2) = \big(P_{K_T}(w), F_{K_f(w)}\big)$$

on the client side. Server calculates the value $c = T[\mathrm{tk}_1]$, deciphers it, and gets the address $a_1 = \mathrm{Dec}_{\mathrm{tk}_2}(c)$. Then for each i, while $a_i = \perp$, the server deciphers the values $\big(N_{w,1}, \ldots, N_{w,|\mathrm{DB}[w]|}\big)$ calculating

$$(\mathrm{id}, a_{i+1}) \leftarrow \mathrm{Dec}_{K_F}(\mathrm{RAM}_3[a_i]).$$

The final step is the finding of the documents with $(\mathrm{id}_1, \ldots, \mathrm{id}_{|\mathrm{DB}[w]|})$ identifiers. Search complexity for this scheme is $O(|\mathrm{DB}[w]|)$ thus enabling one to use it for practical applications. Knowledge of $\mathrm{EDB} = (T, \mathrm{RAM}_3)$ gives the adversary an opportunity to get the number of keywords as the value T and $\sum\limits_{w \in W} |\mathrm{DB}[w]|$ as the size of the array RAM_3. However, the adversary cannot obtain any nontrivial information such as keyword frequency.

3. Functional Encryption

Similar to homomorphic encryption, functional encryption enables one to calculate various functions above the encrypted information. However, it restricts the list of possible functions and information that would become publicly available in the case of computing. This may become possible due to special private key applicable to each function. The first research in this field was the paper of Boneh, Sahai, and Waters (2012) [2]. Further development is related to the search for the practical implementation of functional encryption for any random function. One of these approaches includes the possibility to use multi-party computation in [9]. Formal definition of such kind of schemes is given in [2]. The algorithms $(\mathrm{Setup}, \mathrm{Keygen}, \mathrm{Enc}, \mathrm{Dec})$ are called functional encryption scheme if they satisfy the conditions as follows:

- $(\mathrm{pk}, \mathrm{mk}) \leftarrow \mathrm{Setup}(1^k)$ is the generation of public key and private key pair;
- $\mathrm{sk} \leftarrow \mathrm{Keygen}(\mathrm{mk}, x)$ is the generation of a private key for k;
- $c \leftarrow \mathrm{Enc}(\mathrm{pk}, x)$ is the encryption of the message x;
- $y = F(k, x) \leftarrow \mathrm{Dec}(\mathrm{sk}, c)$ is the calculation of $F(k, x)$ from c using the key sk.

Examples of functional encryption schemes are the schemes providing for

- predicate encryption (for a variety of applications the pair $(\mathrm{ind}, m) \in I \times M$ is a plaintext, where ind is the index of the message m; for example, the index may represent the name of a recipient and m a message in the email system);
- IBE (identity-based encryption) (allows encrypting a message without a public key; the encryption of an email sent to the recipient who has no public-key certificate generated may serve as an example of the use of such a scheme);
- ABE (attribute-based encryption) (ciphertext serves as a decryption key).

Let us consider the IBE scheme suggested by Sahai and Waters. Suppose that G_1 is a bilinear group of pth order and parent element g and $e\colon G_1 \times G_1 \to G_2$ is a bilinear mapping. Identities belong to a set U of elements from $\mathbb{Z}_p^*$.

Setup. The following set of elements represents an open parameter: $T_1 = g^{t_1}, \ldots, T_{|U|}^{t_{|U|}}$, $Y = e(g, g)^y$, while the following set is a private master key: $t_1, \ldots, l_{|U|}, y$.

Keygen. To generate a private key for an individual value of $w \subset U$, the polynomial q of order $d - 1$ is generated, where d is a scheme parameter given that $q(0) = y$. Then a private key consists of the component $(D_i) = g^{q(i)/t_i}$ for each $i \in w$.

Enc. The following set represents ciphertext for the message $M \in G_2$ for the value w': $E = (w', E' = MY^s, \{E_i = T_i^s\}_{i \in w'})$, where s is a random element from $\mathbb{Z}_p$.

Dec. Lagrangian coefficient

$$\Delta_{i,S}(x) = \prod_{j \in S,\ j \neq i} \frac{x - j}{i - j}$$

is used for decryption, and the following actions should be taken:

$$E' / \prod_{i \in S} (e(D_i, E_i)^{\Delta_{i,S}(0)} = Me(g, g)^{sy} / \prod_{i \in S} \left(e(g^{q(i)/t_i}, g^{st_i})\right)^{\Delta_{i,S}(0)} = Me(g, g)^{sy} / \prod_{i \in S} \left(e(g, g^{\mathrm{sq}(i)})\right)^{\Delta_{i,S}(0)} = M.$$

The last equation is correct as the polynomial $\mathrm{sq}(x)$ of degree $d - 1$ may be found based on the d values.

4. Homomorphic Encryption of Cloud Databases

Cloud technologies are rapidly developing representing a state-of-the-art multi-layered field of science and engineering. One of the examples of the application of cloud technologies is a cloud database. According to a number of national legislations including but not limited to the laws of the Russian Federation, in some cases it is not allowed to use a cloud database to store information in the public domain. Fully homomorphic encryption can solve the problem due to the fact that all data to be stored in the cloud drive will be encrypted. Furthermore, decrypted information will be available only to the owner of the cloud storage. Unauthorized persons having access to such cloud storage will not be able to obtain information about queries sent to the cloud by the owner, nor about any results of appropriate queries. A case model of using fully homomorphic encryption for implementation of secure cloud storage is given below.

Database Model Based on Homomorphic Encryption. Let us consider the case of relational database described in [3]. A relational database is represented by a set of rectangular pages. For the sake of simplicity, without loss of generality, we may assume that the database consists of one table. The table includes attributes $a_1, a_2, \ldots, a_m$ and consists of a set of records $\{R_i\}_{i=1}^n$, where $R_i = \{w_{i,j}\}_{j=1}^m$ is the value of the record R_i in the attribute a_j. Let us consider a case where a client needs to make two types of queries to the database:

$$\texttt{SELECT * FROM db WHERE } (a_{t_1} = v_1) \texttt{ OR } (a_{t_2} = v_2) \texttt{ OR } \ldots \texttt{ OR } (a_{t_k} = v_k), \tag{13}$$

$$\texttt{SELECT * FROM db WHERE } (a_{t_1} = v_1) \texttt{ AND } (a_{t_2} = v_2) \texttt{ AND } \ldots \texttt{ AND } (a_{t_k} = v_k). \tag{14}$$

We may call the queries of the type (13) disjunctive, and the queries of the type (14) conjunctive. Now let us consider how to build secure cloud storage with the use of fully homomorphic encryption scheme. Suppose that we have a cloud server S, where the db $= \{R_i\}_{i=1}^n$ owned by the client K is stored. The client K makes quires to the server S from time to time. As a result, he should obtain a list of records satisfying the `WHERE` condition from the client's query. In addition, it is necessary that the server S knows nothing about the values v_i from the `WHERE` condition of the client's query, as well as about any records in the database that satisfy the query criterion. One of approaches to task solution is as follows:

- in step one the client K gets the numbers of $i_1, i_2, \ldots, i_t$ it records that satisfy the client's query. This step must be implemented in such a way that S is be kept from knowing $i_1, i_2, \ldots, i_t$;
- from the database, the client K extracts records with the numbers $i_1, i_2, \ldots, i_t$ one by one in such a way so that S does not become aware of the values of indices.

For the purpose of not providing access to the client K database to the server S and any third-party users, such a database is stored in encrypted form, i.e., S stores the encrypted value of each attribute for each record. Suppose that for the purpose of encrypting we use a completely homomorphic encryption scheme E, and sk is a private key.

Protected Generation of Indices. The client K wants to conceal the values v_i, $1 \le i \le k$; thus he needs to encrypt them. The client K provides the pairs $\big(a_{z_i}, E(v_i)\big)$, $1 \le i \le k$, to the server S. The serever S, in turn, should perform the following calculations for each record $R_i = \{E(w_{i,j})\}_{j=1}^m, i = 1, \ldots, n$:

(1) for all $z_j = 1, \ldots, k$, the server S calculates $e_k = f\big(E(w_{i,z_j}), E(v_k)\big)$, where f is a function comparing for equality of w_{i,z_j} and v_j homomorphically, i.e., $e_j = E(1)$ in the case of equality of w_{i,z_j} and w_j and $e_k = E(0)$ otherwise;

(2) depending on the type of query, S performs the following actions:

- a conjunctive query:

$$e'_i = H_{\text{AND}}(e_1, e_2, \ldots, e_t),$$

where $H_{\text{AND}}(e_1, e_2, \ldots, e_t) = e_1 \cdot e_2 \cdot e_3 \cdots e_t$;

- a disjunctive query:

$$e'_i = H_{\mathrm{XOR}}(e_1, e_2, \ldots, e_t),$$

where $H_{\mathrm{XOR}}()$ is the function calculating XOR of the arguments;

(3) the server S sends to the client K the vector $\mathrm{Res} = (e'_1, e'_2, \ldots, e'_n)$.

To extract records from the database using their indices, the client K may use a standard private information retrieval protocol (PIR). Currently, there is a great number of various PIR protocols [1].

REFERENCES

1. D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in: M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, eds., *Applied Cryptography and Network Security: 11th Int. Conf., ACNS 2013, Banff, AB, Canada, June 25–28, 2013. Proc.*, Lect. Notes Comp. Sci. Security Cryptology, Vol. 7954, Springer, Berlin (2013), pp. 102–118.
2. D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," *Theory Cryptography*, 253–273 (2011).
3. F. B. Burtyka, "Symmetric fully homomorphic encryption using irreducible matrix polynomials," *Izv. Yuzhn. Federal. Univ. Tekhn. Nauki*, 107–122 (2014).
4. D. Cash, J. Jaeger, St. Jarecki, Ch. Jutla, H. Krawczyk, M.-Cat. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in: R. Canetti and J. A. Garay, eds., *Advances in Cryptology — CRYPTO 2013: 33rd Annual Cryptology Conf., Santa Barbara, CA, USA, August 18–22, 2013. Proc., Pt. 1*, Lect. Notes Comp. Sci. Security Cryptology, Vol. 8042, Springer, Berlin (2013), pp. 353–373.
5. D. Cash, J. Jaeger, St. Jarecki, Ch. Jutla, H. Krawczyk, M.-Cat. Rosu, and M. Steiner, *Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation*, Cryptology ePrint Archive: Report 2014/853.
6. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in: *Proc. 13th ACM Conf. Computer Communication Security*, ACM, New York (2006).
7. C. Gentry, *A Fully Homomorphic Encryption Scheme*, Ph.D. thesis, Stanford Univ. (2009).
8. M. M. Glukhov, V. P. Elizarov, and A. A. Nechaev, *Algebra* [in Russian], Lan, St. Petersburg (2015).
9. S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in: R. Safavi-Naini and R. Canetti, eds., *Advances in Cryptology — CRYPTO 2012* Lect. Notes Comp. Sci., Vol. 7417, Springer, Berlin (2012), pp. 162–179.
10. A. V. Gribov, P. A. Zolotykh, and A. V. Mikhalev, "Constructing algebraic cryptosystems over quasigroup ring," *Math. Probl. Cryptography*, **1**, No. 4, 23–32 (2010).
11. S. Y. Katyshev, V. T. Markov, and A. A. Nechaev, "The use of non-associative groupoids for the implementation of public key distribution procedure," *Discrete Math.*, **26**, No. 3, 45–64 (2014).
12. A. S. Kuzmin, V. T. Markov, A. A. Mikhalev, A. V. Mikhalev, and A. A. Nechaev, "Cryptographic algorithms on groups and algebras," *Fundam. Prikl. Mat.*, **20**, No. 1, 205–222 (2015).
13. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in: *SP '00 Proc. 2000 IEEE Symp. Security and Privacy*, Univ. California, Berkeley (2000).
14. D. Stehle and R. Steinfeld, "Faster fully homomorphic encryption," in: *Advances in Cryptology — ASIACRYPT 2010: 16th Int. Conf. on the Theory and Application of Cryptology and Information Security, Singapore, December 5–9, 2010. Proc.*, Lect. Notes Comp. Sci., Vol. 6477, Springer, Berlin (2010), pp. 377–394.

G. G. Arakelov, A. V. Gribov, and A. V. Mikhalev
Moscow State University, Moscow, Russia
E-mail: g.g.arakelov@gmail.com