UDC 004.032.26

O. Belej, N. Nestor,
Lviv Polytechnic National University, DCAD,
N. Melnyk,
Lviv institute State University “The University of Banking”

APPLICATION HOMOMORPHIC CRYPTOGRAPHIC ALGORITHM FOR ENCRYPTING DATA IN THE CLOUD STORAGE

The article analyzes existing and perspective systems of homomorphic encryption and their practical application. The author considers some models of homomorphy cryptographic algorithms, which may be useful from a practical point of view. One of the interesting and practically valuable encryption schemes is cryptographic algorithm, which is constructed on matrix polynomials. Also considered are isolated cases of homomorphic encryption, a protected cloud database model based on a completely homomorphic encryption scheme.

Key words: homomorphic, cryptographic algorithm, encryption, cloud technologies, data warehouses.

ЗАСТОСУВАННЯ ГОМОМОРФНОГО КРИПТОАЛГОРИТМУ ДЛЯ ШИФРУВАННЯ ДАНИХ У ХМАРНОМУ СХОВИЩІ

Проаналізовано наявні та перспективні системи гомоморфного шифрування і їх практичне застосування. Автори розглянули деякі моделі гомоморфних криптографічних алгоритмів, що може бути корисно з практичного погляду. Однією з цікавих і практично цінних схем шифрування вважається алгоритм, побудований на матричних поліномах. Розглянуто також окремі випадки гомоморфного шифрування, модель захищеної хмарної бази даних, яка ґрунтується на повністю гомоморфній схемі шифрування.

Ключові слова: гомоморфний, криптоалгоритм, шифрування, хмарні технології, сховища даних.

Problem statement

Of considerable interest for analysis and practical use is homomorphic cryptography, or homomorphic encryption. Homomorphic encryption allows you to execute certain types of computations in encrypted text and receive encrypted results of calculations, which, when decoded, are consistent with the results of operations performed with open text. The theoretical and practical aspects of homomorphic encryption are closely related to the problem of cloud computing security. It is believed that the cloud computing ideology has become popular since 2007 due to the rapid development of communication channels and the rapidly growing needs of users.

The reasons are obvious for the growing popularity of cloud technologies, and the possibilities of their application are very diverse. The user saves both on maintenance and personnel, and on infrastructure. No need to buy software licenses, organize and maintain your own servers, hire experienced administrators. All of these problems are transferred to the service provider. In addition, this approach allows standardization of software even if different enterprise operating systems (Windows, Linux, MacOS) are installed on enterprise computers.

Problems of information security in cloud technologies have been actively analyzed late enough when the clouds were already actually implemented by technology. The practice of cloud computing has shown that there is not enough cryptographic means available to protect information. Let's explain it in this example. Assume that the cloud *S* contains a lot of users (clients) $p_1, \ldots, p_i, \ldots, p_n$.

Pi has confidential xi data stored in the cloud. This cloud service is called Storage aaS (repository as a service). The user of P_i can refer to the cloud requesting to calculate the value of some function *F*, which depends on confidential data. The request must consist of a description of the function *F*, the user ID and its public key pk_i. The cloud should check the credentials of p_i for calculating $F\,(x_i)$. Such verification can be implemented using the standard procedure of electronic digital signature (EDS). If the user has confirmed his rights to calculate the *F* function, then the cloud must calculate the value of $E\,(pk_i, F(x_i))$ and send it to the user. As ***E*** can to take the functions of encryption of some public key cryptosystem.

A user who places his confidential data in a repository and inquires for the calculation of the function *F* does not trust the cloud and must take appropriate measures and impose requirements to ensure their safety. Obviously, it would be much safer to transfer data in such a way that during the operations that are carried out over them, the information about these data has not been disseminated at all. Therefore, first of all, the data must be encrypted, and they must be sent to the server already in an encrypted form. This means that the encryption must be done by the user. Secondly, it is necessary to process this data without decrypting, as for the transmission and storage of a secret key, it is necessary to adhere to certain procedures, especially complex, if the information is processed in a distrust environment.

It turned out that the protection of information in cloud computing is much more complicated than those tasks for protecting information that are solved by known cryptographic means. Public key cryptosystems for solving this problem are not always appropriate. In 1978, the authors of the well-known RSA public key algorithm, Michael Dertuzos, Ronald Rivers, Leonard Adleman, first proved that the method that allows successful operations on encrypted data without distorting or decrypting them is the so-called homomorphic encryption [8]. In their work they described the concept of homomorphic encryption, as well as wondering if such an encryption is possible in principle and for which algebraic systems such a homomorphism exists.

One of the most is interesting and important tasks facing modern cryptography is the computation of encrypted data without their prior decoding. The question of the fundamental possibility of such calculations remained open for a long time, and it must be said that the authors of the RSA encryption scheme believed that such calculations were impossible in principle. The section of cryptography devoted to a scheme that allows computation over cipher text is called homomorphic cryptography, and the corresponding schemes are homomorphic. In this case, there are completely homomorphic and partially homomorphic encryption schemes. In a completely homomorphic encryption scheme, the operation of adding and multiplying cipher text is homomorphic. More precisely, the following relationships are performed:

$$D(E(m_1)\times E(m_2)) = m_1 \times m_2,\ D(E(m_1)+E(m_2)) = m_1 + m_2, \tag{1}$$

where $E(m)$ – encryption function, and $D(m)$ – decryption function.

If in some encryption scheme at least one of these conditions is fulfilled, then this scheme is called partially homomorphic. Examples of partially homomorphic cipher circuits are quite numerous. For example, the RSA scheme itself is homomorphic with respect to the multiplication operation, as well as the El-Gamal scheme. The RSA scheme and the El-Gamal scheme, for example, are partially homomorphic, namely the cipher text multiplication operation, is homomorphic [2].

In the article we consider a relatively new direction in the construction of cryptographic algorithms based on a homomorphic encryption system.

Analysis of recent research and publications

Homomorphic encryption is a form of encryption that allows you to perform a certain algebraic operation over the open text by executing an algebraic operation over the encrypted text. Let *E(k, m)* be an encryption function, where *k* is an encryption key and *m* is an open text. The function w is called homomorphic with respect to the operation over open texts, if there is an effective algorithm *M* (requires a polynomial number of resources and operates in polynomial time), which, upon receipt of any pair of encrypted text of the form $(E(k,m_1),E(k,m_2))$, yields an encrypted text $c=M(E(k,m_1),E(k,m_2))$ such that when decoding c will be open text $m_1 \times m_2$ [4].

As a rule, the following particular case of homomorphic encryption is considered. For this encryption function *E* and operation $*_1$ over open texts there is an operation $*_2$ over encrypted texts, such that from encrypted text $E(k,m_1), E(k,m_2)$, when decrypted, extracted open text $m_1 *_1 m_2$. In this case, it is necessary that $c, E(k,m_1), E(k,m_2)$, at the given, but with an unknown key, it would not be possible to effectively check that the encrypted text c received from $E(k,m_1)$, і $E(k,m_2)$.

Any standard encryption system can be described as three operations: key generation (*KeyGen*), encryption (Encrypt) and decrypt (Decrypt).

Homomorphic encryption system, in addition to the three above-mentioned operations, includes a calculation operation (*Eval*) [4]. Thus, homomorphic encryption is a sequence of four operations: key generation, encryption, computation, decryption (Fig. 1):

· generation of keys: the client generates an open key *pk* (public key) and secret key (secret key) for encrypting the open text;

· encryption: using the secret key *sk*, the client encrypts the plain text PT (plain text), creates E_{sk} (PT) and, together with the public key, *pk* sends the encrypted *CT* text (cipher text) to the server;

· computing: the server receives a function *F* for computing *CT*-encoded text and executes it according to the requirements of this function using *pk*;

· decryption: to get the desired result, the value *Eval(F(PT))* obtained during the calculation is decrypted by the client using its secret key *sk*.

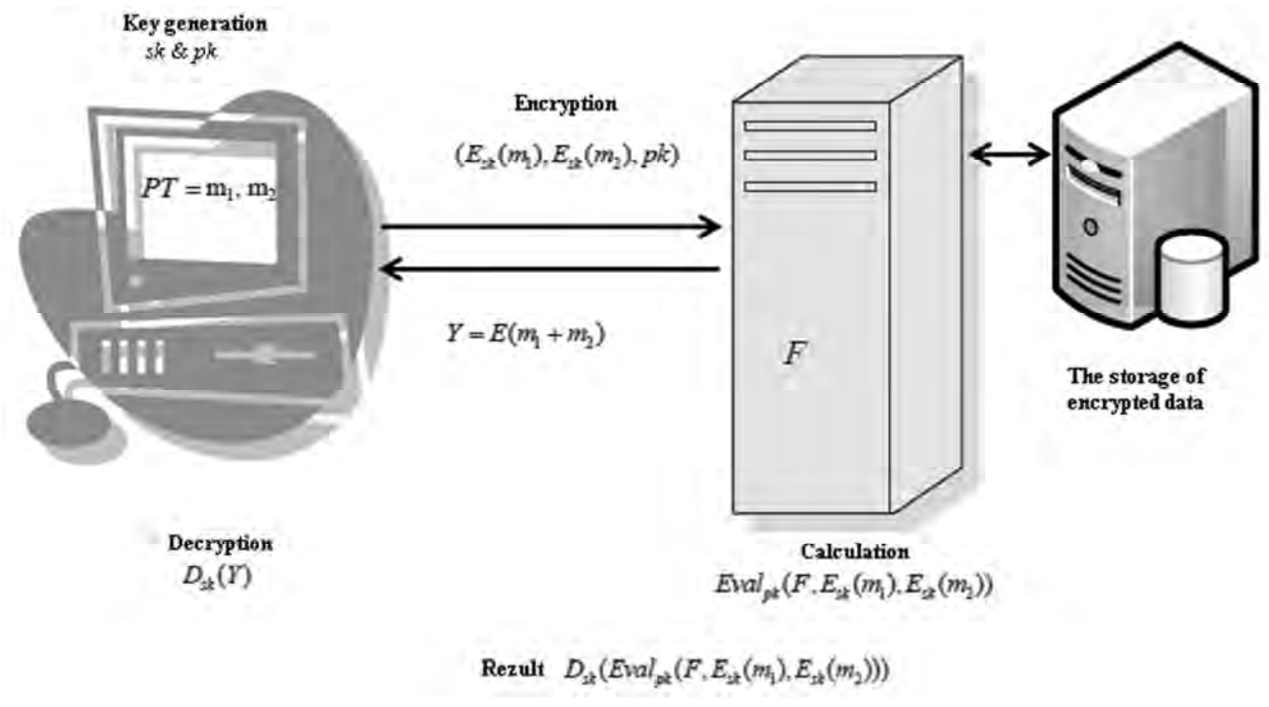

Fig. 1. An algorithm for homomorphic data encryption in cloud storage: own

If a cryptosystem with such properties can encrypt two bits, then since the addition and multiplication operations form a complete Turing basis over the bits, it becomes possible to compute any Boolean function, and hence any other computable function.

For more than 30 years the problem of completely homomorphic encryption has remained unsolved – the creation of a system that is homomorphic with respect to operations of adding and multiplying simultaneously. Only in 2009, a postgraduate student at Stanford University and a trainee at IBM, Craig Gentry, theoretically substantiated the fundamental possibility of creating such an encryption system. In the Gentry scheme [5], the properties of a homomorphism, both in terms of multiplication and addition, are fulfilled, that is, it is an algebraic homomorphic system. The proposed Gentry system can be used to ensure the confidentiality of data in any kind of processing in a non-trusted environment, such as cloud computing or distributed computing. However, the Gentry model was too impractical. With the increase in the number of operations generated over encrypted text, the complexity and size of the encrypted text increase with incredible speed. Despite the fact that in recent years, many improvements have been made to this scheme, it is still a quicker theoretical model, which cannot yet be applied in practice.

Formulating the whole of article

On the example of specific algorithms and schemes, we describe multiplicative, additive [1, 2, 3] and mixed [6, 7] properties of homomorphic encryption. The above algorithms and schemes are public, so let's give a detailed description of only some of them; with respect to others we restrict ourselves to a short listing of their main properties and areas of application.

The RSA encryption method (abbreviation for the names of the creators – Rivest, Shamir, Adleman) was proposed in 1977 as an implementation of the idea of the founders of cryptography with the public key of Diffie and Hellman [2].

Consider the situation when the open text is represented by the number *m*, so that $0 < m < N$. User *B* wants to send a secret message to him. To do this, he makes publicly available two *N* numbers (constituent module) and *e* (public key) that satisfy the following conditions: $N = pq$, where *p*, *q* are large prime numbers that *B* holds in secret; $p,q \geq 2^{256}$; *e* is chosen mutually simple with $\varphi(N) = (p-1)(q-1)$.

User *A*, who sent the message *m*, encrypts it as follows:

$$E(m) = m^{q} (\operatorname{mod} N) . \tag{2}$$

To decipher *B*, the number *d* is the same as that $1 \leq d \leq N-1$ & $ed = 1(mod\ \varphi(N))$. That comparison is solved in a unique way, since $(e, \varphi(N))=1$. To solve the equation $ed = 1(mod\ \varphi(N))$, user *B* must calculate $\varphi(N)$ that it is not difficult for him, as $\varphi(N)= \varphi(pq)= \varphi(p)\ \varphi(q)=(p-1)(q-1)$. Any other user who knows only *N* is forced to find *p* and *q*, that is, to decompose the number of *N* into simple factors, and this problem for large *p* and *q* has a significant computational complexity

Next, having available $y=E(m)=m^{\theta}(mod\ N)$, User *B* calculates the value $D(y)=E(m)=y^{d}(mod\ N)$, which is the open text *m*. Indeed, using the Euler theorem, we obtain:

$$D(y) = y^{d} = m^{qd} = m^{j\ (N)k+1} = (m^{j\ (N)})^{k} m = m(\operatorname{mod} N) . \tag{3}$$

The RSA cryptosystem is homomorphic with respect to the operation of multiplying open texts. For any two open texts m_1, m_2.

$$E(m_1)E(m_2) = m_1^{q} m_2^{q} (\operatorname{mod} N) = E(m_1 m_2) . \tag{4}$$

As for the El-Gamal cryptosystem, consider the cyclic group of order *G p and* g, which are the generating elements of the group. As a secret key, a random element of *d* group $Z_{(p-1)}$ is selected. The corresponding public key e is calculated by the formula $e=g^{d}$.

The encryption function for the message *m* looks like this:

$$E(e,m) = (e^{r} m, g^{r}) , \tag{5}$$

where *r* – a random element of the group $Z_{(p-1)}$.

Decryption of cryptograms (c_1, c_2) is performed as follows. is calculated:

$$c_2^d = g^{rd}, \tag{6}$$

whence:

$$m = \frac{c_1}{c_2}. \tag{7}$$

El-Gamal's cryptosystem is homomorphic to the operation of multiplying open texts. If $E(e,m_1) = (e^{r1}m_1, g^{r1})$ і $E(e,m_2) = (e^{r2}m_2, g^{r2})$, then:

$$E(e,m_1m_2) = (e^{r1}e^{r2}m_1m_2, g^{r1}g^{r2}) = E(e,m_1)E(e,m_2). \tag{8}$$

Peier's Cryptosystem is based on the probabilistic asymmetric transformation algorithm and is used in cryptographic protocols with a key.

Let *p and q* be two prime numbers, *n = pq, λ = mod(p-1, q-1)*. Choose a random number *g* of $Z^k_{n^2}$ and calculate $\mu = (L(g^\lambda \bmod n^2))^{-1} (\bmod n)$, where $L(u) = \frac{(u-1)}{u}$.

The open key is a pair *(n, g)*, and the private key is a pair *(λ, μ)*.

To encrypt plaintext $m \in Z_n$ is selected a random number $r \in Z^k_{n^2}$ and calculate $E(m) = c = g^m r^n (\bmod n^2)$.

Decryption is performed according to the formula $m = L(c^\lambda (\bmod n^2))\mu(\bmod n)$.

The property of a homomorphism will have the following form:

$$E(m_1)E(m_2) = (g^{m_1} r_1^n)(g^{m_2} r_2^n) = g^{m_1+m_2}(r_1r_2)^n = E(m_1+m_2)(\bmod n). \tag{9}$$

Let's consider already known Gentry scheme for completely homomorphic encryption on an example of calculations in Z_2. First we generate keys. Choose an arbitrary odd integer $p = 2k - 1$. This number p is a secret key. Let's try to encrypt the bit $m \in (0,1)$. To do this, we generate a number $z = 2r + m$ where *r* is an arbitrary integer. This means that: $z = m(\bmod 2)$.

Encryption is that for each number *m* the matching number is assigned $c = pq + z$, where *q* – an arbitrary integer. So, $E(m) = c = 2r + m + (2k+1)q = 2(r+kq) + m + q$. The exact number is calculated *c*.

To decipher, we take the numbers *c, p, q*, where *p, q* are known. We will decrypt using a secret key *p*:

$$\begin{aligned} &c(\bmod p) = (z + pq)(\bmod p) = z(\bmod p) + pq(\bmod p) = z(\bmod p) = \\ &= (2r + m)(\bmod p) = 2(r(\bmod p)) + m(\bmod p) \end{aligned} \tag{10}$$

Then we calculate: $(C(\bmod p))(\bmod 2) = 2(r(\bmod p))(\bmod 2) = m(\bmod 2) = m$.

Encryption is homomorphic with respect to adding and multiplying operations. Consider a couple of bits $m_1, m_2 \in (0,1)$ and compare for them: $z_1 = 2r_1 + m_1$, $z_2 = 2r_2 + m_2$. Choose a secret key $p = 2k+1$. Then $E(m_1) = c_1 = z_1 + pq_1$, $E(m_2) = c_2 = z_2 + pq_2$ – encrypted texts for m_1 and m_2 respectively.

The operation of adding over the encrypted numbers will look like:

$$E(m_1) + E(m_2) = c_1 + c_2 = z_1 + z_2 + p(q_1 + q_2) = 2(r_1 + r_2) + m_1 + m_2 + (q_1 + q_2). \tag{11}$$

The multiplication operation over the encrypted numbers will look like:

$$\begin{aligned} &E(m_1)E(m_2) = c_1c_2 = z_1z_2 + p(z_1q_2 + z_2q_2) + p^2q_1q_2 = \\ &= (2r_1 + m_1)(2r_2 + m_2) + p(z_1q_2 + z_2q_2) + p^2q_1q_2 = \\ &= 4r_1r_2 + 2(r_1m_2 + r_2m_1) + m_1m_2 + p(z_1q_2 + z_2q_2) + p^2q_1q_2 \end{aligned} \tag{12}$$

And when decoding we get:

$$D(E(m_1)+E(m_2))=((c_1+c_2)(\mathrm{mod}\,p))(\mathrm{mod}2)=m_1+m_2\,. \tag{13}$$

$$D(E(m_1)E(m_2))=((c_1c_2)(\mathrm{mod}\,p))(\mathrm{mod}2)=m_1m_2\,. \tag{14}$$

An essential disadvantage of this scheme is that computing leads to an accumulation of error and, after it exceeds p, decrypt the message becomes impossible. One solution to this problem is to decrypt data after a certain number of operations, but this option reduces the computing performance and requires constant access to the secret key. The stability of the Gentry scheme on the basis of ideal lattices (lattices with properties of the ideal on a certain number of numbers) reduces to the NP-complete problem of finding the shortest vector. There are many works aimed at the development of ideas proposed in it and the elimination of shortcomings. In particular, the scheme of BGV (Brakerski, Gentry, Vaikuntanathan) was proposed. The authors presented an alternative to completely homomorphic encryption based on LWE (Learning with Errors) [5], which made it possible to reduce the complexity of constructing a cryptosystem, but inherited the main drawbacks of the Gentry scheme: the existence of a growing error in the encrypted text; increase the size of the encrypted text.

Depending on the circumstances, the property of homomorphism can be regarded as both dignity and a cryptosystem as a defect. This applies, for example, to the RSA cryptosystem, in which the decryption function is used in the EDS scheme. The signature of the message m is calculated by the formula $s=m^d(\mathrm{mod}\,N)$, where d is the secret key component. Obviously, this is a homomorphism transformation with respect to the multiplication operation. So, we can offer the following way of forgery of signatures. If s_1, s_2 messages are known m_1, m_2, then the message m_1m_2 of the message will be s_1s_2, respectively. However, in practice, this vulnerability does not pose a threat to the stability of the EDS scheme, since they are not signed by their own messages, but the value of the hash functions of the messages. However, the homomorphism of the signature generating function imposes an additional requirement on the hash function, which does not follow from standard definitions of the cryptographic hash function.

Most algorithms of homomorphic encryption, the stability of which is based on the complexity of discrete logarithms in the finite field, is quite easy to transfer to the case of elliptic curves. Crypto systems based on elliptic curves outperform other public key systems in two important parameters: the degree of protection of the calculation for each key bits and the speed of the software implementation. This is explained by the fact that for calculation of inverse functions on elliptic curves, only algorithms with increasing labor intensity are known, whereas sub exponential methods are proposed for ordinary systems. When providing the same stability of the cryptographic protocols, the calculation is performed approximately 20 % faster in the group of points of the elliptic curve, than for the groups of the final field.

The most voluminous operation in cryptosystems based on matrix polynomials is the multiplication of cipher text with matrix polynomials, so the computational complexity of the entire scheme will directly 48depend on this operation. In turn, the multiplication of matrix polynomials depends on two algorithms: the matrix multiplication algorithm and the polynomial multiplication algorithm.

The algorithm for multiplying polynomials, suitable for the scheme described, has an asymptotic complexity of operations $O(d^{\log_2 3})=O(d^{1,5849...})$ over the coefficients of polynomials, and where d is the largest of the degrees of polynomials. The algorithm for multiplying two matrices $(N\times N)$ has the asymptotic complexity of elementary operations $O(N^{2,373...})$.

If $N=O(\lambda)$ and the degree of the cipher text is the equal of polynomials, we find that the total number of operations on the elements Z_p has an asymptotic complexity $\approx O(\lambda^{3,76})$.

At present, the best estimate of computational cost in the calculation of homomorphism is $g(\lambda)=O(\lambda^{3,5})$. Such computational complexity has the scheme described in [5].

Here it is worth noting that when evaluating the computational complexity of the encryption scheme based on matrix polynomials, it was considered the multiplication of matrices $(N \times N)$ from dimension requires operations $O(N^{2,373...})$. Indeed, there is an algorithm capable of multiplying two matrices at a given time, and this is the Coppersmith-Grape algorithm improved by Williams [6]. However, in practice, the Coppersmith-Vineyard algorithm cannot be used at present, since it has a very large constant of proportionality and begins to gain in performance in comparison with other known algorithms only for matrices whose size exceeds the memory of modern computers

On the other hand, the well-known Strassen hypothesis states that for an arbitrarily small algorithm $\varepsilon > 0$, with a sufficiently large natural n, guarantees the multiplication of two size matrices $n \times n$ for operations $O(n^{2+s})$.

The cryptosystem on matrix polynomials inferior to the asymptotic estimations of the system described in [6], but from the practical point of view, it now represents value, since it allows for wide parallelization. In particular, an experiment was conducted with the use of technology for mass parallel computing by CUDA of firm Nvidia, which showed an advantage over the time of computing cryptosystems on matrix polynomials.

The best estimate of the computations in homomorphic calculations belongs to the Gentry cryptosystem [6]. IBM has implemented the so-called homomorphic encryption library called “Helib” (https://github.com/shaih/HElib). This library contains the implementation of a cryptosystem, which is considered today asymptotically better than computing costs among homomorphic encryption systems.

For a comparative analysis of the cryptosystem described in the article and the modified Gentry cryptosystem [6], a cryptosystem matrix on matrix polynomials was implemented in which the CUDA parallel computing technology was used to multiply the matrices and polynomials, and the modified Gentry system was taken from the HElib library.

A number of experiments with different parameters of crypto stability were carried out. We evaluated the time required for the following operations: encryption; decryption; multiplication.

Table 1

Evaluation of cryptosystem performance on matrix polynomials using parallel computations

Parameter λ	Encryption	Decryption	Multiplication of cipher text
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 performance of the modified Gentry cryptosystem

Parameter λ	Encryption	Decryption	Multiplication of cipher text
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

Based on our performance estimates in Tables 1 and 2, it is clear that in practice the cryptosystem based on matrix polynomials does not yield to the improved Gentry encryption model, and even benefits from the use of parallel computing technologies. We conducted an experiment for the maximum value of the parameter $l = 64$ and already at the value $l = 32$ we have obtained that the Gentry model, which is considered to be asymptotically the most “fast”, yields the model on matrix polynomials

On the basis of the foregoing, it can be argued that in practice, cryptosystems designed to apply homomorphic encryption must satisfy at least the following requirements:

· A set of supported mathematical functions should cover the everyday needs of programmers;

· The accuracy and speed of computations should not degrade during computations;

· The stability of the algorithm should exclude the attack by a complete override.

Presenting main material

Fully homomorphic encryption can solve the problem of using a cloud database for storing information in an open form by the fact that all data that will be stored in cloud storage will be encrypted, moreover, access to decrypted data will only be the owner of the repository. Also, third parties accessing this cloud storage will not be able to receive information about requests sent by the owner to the cloud, as well as the results of relevant queries. The fully homomorphous encryption model for implementing a reliable cloud storage is described below.

Relational database is represented by a set of rectangular pages. For simplicity, without loss of dependencies, we can assume that the database consists of one table. The table has attributes $a_1, a_2, ..., a_m$ and consists of a multitude of records $\{R_i\}_{i=1}^{n}$, where $R_i = \{w_{i,j}\}_{j=1}^{m}$ – he value of the record R_i in the attribute a_j.

Consider the case when the customer needs to do two types of query requests:

$$\text{SELECT * FROM db WHERE } (a_{t1} = v_1) \text{ OR } (a_{t2} = v_2) \text{ OR ... OR } (a_{tk} = v_k) \quad (15)$$

$$\text{SELECT * FROM db WHERE } (a_{t1} = v_1) \text{ AND } (a_{t2} = v_2) \text{ AND ... AND } (a_{tk} = v_k) \quad (16)$$

Queries type (15) will be called disjunctive, and queries type (16) – conjunctive. Now let's consider how it is possible to build a reliable cloud data warehouse, having a completely homomorphic encryption scheme.

Let *S* be some cloud server that stores the base $\mathrm{db} = \{R_i\}_{i=1}^{n}$ belonging to client *K*. Periodically, client *K* refers to the server *S* with requests. As a result, he must obtain a list of records that correspond to the ***WHERE*** condition on the client request, and it is necessary for the server *S* to know nothing about the value v_i of the condition of the ***WHERE*** request of the client, as well as the records in the database that satisfy the query condition.

One approach to solving this problem is as follows:

· In the first step, the client *K* receives the entry numbers $i_1, i_2, ..., i_t$ that match the query. This step must be implemented so that *S* recognize it does not $i_1, i_2, ..., i_t$;

· The client *K* extracts from the database with the numbers $i_1, i_2, ..., i_t$ one by one so that *S* does not recognize the value of the indices.

In order for the server *S*, as well as third-party users not to be able to access the client base *K*, is stored in encrypted form. In fact, *S* stores the value of each attribute for each entry in an encrypted form. Let encryption use a completely homomorphic encryption scheme *E*, and *sk* – a secret key.

How does a customer get secured indexes? Customer *K* wants to hide the value $v_i, 1£i£k$, so he needs to encrypt them. On server *S* client *K* transfers bets $a_{z1}, E(v_i), 1£i£k$.

In turn, *S* should hold the following calculations for each entry $R_i = \{E(w_{i,j})\}_{j=1}^{m}, i = 1,...,n$ carry out the following calculations:

1) for everyone $z_j = 1,...,k$ to calculate $e_k = f\ E(w_{i,zj}), E(v_k)$, where *f* is a function that checks for equality $w_{i,zj}$ v_k and is homomorphic, that is $e_i = E(1)$, in the case of equality, $w_{i,zj}$, w_j and $e_k = E(0)$ in another case;

2) depending on the type of request, perform the following:

· conjunctive inquiry: $\grave{e}_i = H_{AND}(e_1, e_2, ..., e_t)$, where $H_{AND}(e_1, e_2, ..., e_t) = e_1, e_2, ..., e_t$;

· disjunctive quest: $\grave{e}_i = H_{xor}(e_1, e_2, ..., e_t)$, where $H_{xor}(\ )$ – a function that calculates *XOR* from arguments;

· the server *S* sends the vector *K* to the client $R_{es} = (\grave{e}_1, \grave{e}_2, ..., \grave{e}_n)$.

To remove entries from the database by their indexes, the client K can use the standard secret information reception protocol. At the moment a large number of different secret information protocols have been developed, but there are very few protocols for open data with the ability to protect them from unauthorized access at the system level.

Conclusion

Based on the above, it can be argued that in the near future methods of homomorphic encryption will have a significant impact on the market of cloud services and the appearance of modern information technology. However, until there are effective algorithms for fully homomorphic encryption that provides a level of performance suitable for practical use, and even more so for real-time applications, no effective algorithms have been created. All proposed schemes cannot be implemented in practice and are not ready for implementation in real systems, as they result in accumulation of errors and rapid increase of encrypted texts. In this case, partially homomorphic systems (in terms of addition or multiplication operations) are successfully used in cloud computing, electronic voting, secure information retrieval, feedback systems and others.

It should be borne in mind that some homomorphic cryptosystems can be subjected to intentional external influences (vulnerable to attack with adaptively picked cipher text), and therefore not always suitable for secure data transfer. An assessment of the crypto stability of homomorphic systems requires a separate study.

Unlike lightened cryptography for homomorphic encryption, the corresponding international standards have not yet been developed; however, work is actively under way to create acceptable solutions that allow the safe processing of confidential data in clouds and other applications.

1. Ahmad, I. Survey: homomorphic encryption schemes / I. Ahmad, D. Adiga // Proc. of 9th IRF Intern. Conf. – Pune, India, 2014. – P. 89–94. 2. D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. In Proc. of Crypto '08, LNCS 5157, pages 108–125.. 3. Cash D., Jaeger J., Jarecki St., Jutla Ch., Krawczyk H., Rosu M.-C., Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries // Advances in Cryptology – CRYPTO 2013 / R. Canetti, J. A. Garay, eds. – Berlin: Springer, 2013. – (Lect. Notes Comp. Sci. Security Cryptology; Vol. 8042). – P. 353–373. 4. Gentry C. A Fully Homomorphic Encryption Scheme: Ph. D. thesis. – Stanford Univ., 2009. 4. Song D., Wagner D., Perrig A. Practical techniques for searches on encrypted data // SP'00 Proc. 2000 IEEE Symp. Security and Privacy. – Berkeley: Univ. California, 2000. 5. Gentry, C. A Fully homomorphic encryption using ideal lattices / C. Gentry // Symposium on the Theory of Computing (STOC). – Bethesda, USA, 2009. – P. 169–178. 6. Gentry, C. Homomorphic Encryption from

Learning With Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based / C. Gentry, A. Sahai, B. Waters // Advances in cryptology – CRYPTO-2013, 33rd Annual Cryptology Conf. – Santa Barbara, CA, USA, 2013. – Part 1. – P. 73–93. 7. Stehle D., Steinfeld R. Faster fully homomorphic encryption // Advances in Cryptology – ASIACRYPT 2010: 16th Int. Conf. on the Theory and Application of Cryptology and Information Security, Singapore, December 5–9, 2010. Proc. – Berlin: Springer, 2010. – (Lect. Notes Comp. Sci.; Vol. 6477). – P. 377–394. 8. Rivest, R. L. On data banks and privacy homomorphisms / R. L. Rivest, L. Adleman, M. L. Dertouzos // Foundations of secure computation. – 1978. – Vol. 32, no. 4. – P. 169–178. 7. Survey of various homomorphic encryption algorithms and schemes / P. V. Parmar [et al.] // Intern. J. of Computer Applications. – 2014. – Vol. 91, no. 8. – P. 26–32. 8. M. Prabhakaran and M. Rosulek. Homomorphic Encryption with CCA Security. In Proc. of ICALP '08. Springer, 2008. Mikolaj Karpinski, Vitalii Chyzh, Stepan Balaban, Joanna Gancarczyk, Paweł Fałat. Increasing the level of information security in wireless sensor networks by improving their geometric models // SGEM 2018 : 18th International Multidisciplinary Scientific Geoconference : Informatics. Informatics, go informatics and remote sensing. Issue 2.1 : conference proceedings. Volume 18. – Sofia : STEF92 Technology, 2018. – S. 469–476. – P-ISSN: 1314-2704 ; p-ISBN: 978-619-7408-39-3. – DOI: 10.5593/sgem2018/2.10000019098