DOI: 10.15439/2015F140
ACSIS, Vol. 6

Fully Homomorphic Encryption for Secure Computations in Protected Database

Darya Chechulina, Kirill Shatilov, Sergey Krendelev
Department of Information Technology, Novosibirsk State University,
Novosibirsk, Russia
Email: chechulina, shatilov, krendelev@ccfit.nsu.ru

***Abstract*—Outsourced computations and, more particularly, cloud computations, are widespread nowadays. That is why the problem of keeping the data security arises. Multiple fully homomorphic cryptosystems were proposed in order to perform secret computations in untrusted environments. But most of the existent solutions are practically inapplicable as they require huge computation resources and produce big (∼1Gb) keys and ciphertexts. Therefore, we propose the undemanding fully homomorphic scheme with practically acceptable (∼few Kb) keys and output data. Our solution uses modular arithmetic in order to avoid the increase in data size. We have validated our approach through the implementation of the proposed cryptosystem. The details of used algorithms and the results of security evaluation are covered in this paper.**

I. Introduction

NOWADAYS Information Technologies and, particularly, computations over various data are the important part of our living and business processes. Modern trend to outsource computations to third-parties has aroused a problem of keeping the security of one's data. Cloud computing and other cases of giving the access to the personal data are affected by threat of exposing vulnerable data to unauthorized parties. Using a fully homomorphic encryption (FHE) scheme in secure computations helps to avoid the data leakage.

Originally a conception of FHE was introduced by Rivest, Adleman and Dertouzous in their paper [5]. Since people wanted to be able to perform the computations over the encrypted data, the problem of privacy homomorphism became very actual one in cryptography at whole. The first attempt of proposing FHE scheme belongs to Gentry [1]. After publication of the scheme's idea he introduced the implementation of his algorithm in conjunction with Halevi [2]. Then a lot of improvements of Gentry's work were proposed. But all of them were criticized as they required significant computing resources due to the usage of complex mathematical tools and produced big sizes of keys and output data [6].

Most of the proposed encryptions suffer from inefficiency to the practical use; therefore the problem of computations security is still actual [4]. That is why our ultimate goal for FHE developing is researching for the previously not used but efficient mathematical technics to make practical implementation. As a result, we introduce a new fully homomorphic scheme that doesn't require massive computation resources and provide acceptable sizes of encryption keys and output values.

This research was performed in Novosibirsk State University under support of the Ministry of education and science of Russia (contract no. 02.G25.310054)

The next section of this paper features the mathematical bases of our approach as it gives some fundamental definitions. Section 3 describes the properties of computations over encrypted data. After it, in Section 4, we show the core components of the proposed fully homomorphic encryption cryptosystem. Section 5 covers the evaluation of our scheme's security. Then, Section 6 discusses possible applications of the developed homomorphic encryption including the implemented one and summarizes our achievements.

II. Mathematical Foundation

This section gives essential mathematical bases. Let us discuss what a fully homomorphic cryptosystem means. Formally, such a scheme allows performing computation over the encrypted data without their decryption. In other words, an encryption algorithm E and a decryption D should satisfy the following conditions:

$$c_1 = E(a_1), c_2 = E(a_2)$$

$$D(f(c_1, c_2)) = f(a_1, a_2)$$

where c_1, c_2 are the ciphertexts and f is an arbitrary, efficiently computed function. In order to avoid the increase in data size we use modular arithmetic in our approach. Thus, the main idea of the proposed solution is as follows: we have a set of relatively prime numbers $(m_1, m_2, \dots, m_k)$. The plaintext P we associate with the set $P = (P_1, P_2, \dots, P_k)$ where $P_i = P \ mod \ m_i, i = 1, \dots, k$. This set is encrypted with proposed algorithm that will be described in details in the following section. The approach has the only constraint: the result of all the mathematical operations can't exceed the number $M = m_1 \cdot \ \dots \ \cdot m_k$.

Firstly we consider the simplified algorithm for the ring Z_m and the modulus m only. To encrypt P we select a secret vector $x = (x_1, \dots, x_n), x_i \in Z_m$. Then we construct a vector $a = (a_1, \dots, a_n), a_i \in Z_m$ as follows:

$$(a, x) = P \ mod \ m$$

It is worth noting that in general every number m can be represented as $m = p_1^{\alpha_1} \cdot \ \dots \ \cdot p_s^{\alpha_s}$, where $p_1, \dots, p_s$ are prime numbers. Thus, it is enough to make all the necessary steps of the algorithm for the power of prime number only, or, in the simplest case, only for the prime number.

So, let m be a prime number. Now we will describe some mathematical details of the proposed approach.

The scalar product can be considered as a linear function:

$$h(x_1, \dots, x_n) = (u, x), u, x \in Z_m^n$$

$$x = (x_1, \dots, x_n)$$

$$u = (u_1, \dots, u_n)$$

Thus, a linear function is completely determined by the vector u.

The secret point is defined as a vector $x = (x_1, \dots, x_n)$. Thereby, to represent the number P, we must construct a vector $v \in Z_m^n$ as follows:

$$(v, x) = P \ mod \ m$$

This task belongs to the standard linear algebra and can be easily solved. Thus, v is called a ciphertext for P.

III. Computations over Encrypted Data

As it was previously mentioned, the proposed encryption allows performing the computations over ciphertexts.

A. Addition

Addition of vectors is equivalent to addition of their components with given modulus m according to the properties of the scalar product and the modular arithmetic. So, if we have representations for two numbers P_1 and P_2:

$$(v, x) = P_1 \ mod \ m$$

$$(u, x) = P_2 \ mod \ m$$

and in general the sum of the simple linear functions is defined as:

$$(v, x) + (u, x) = (v + u, x)$$

thus:

$$(v + u, x) \ mod \ m = [(v, x) + (u, x)] \ mod \ m =$$

$$= P_1 \ mod \ m + P_2 \ mod \ m = (P_1 + P_2) \ mod \ m$$

Let us note that addition keeps the size of vectors. That means the resulting vector has the same length as the initial ciphers.

B. Multiplication

Multiplication of vectors v and u in common way leads to the increase in the result's length almost n times:

$$w = v \cdot u = (v_1 u_1, v_1 u_2, \dots, v_n u_n)$$

In order to prevent vectors' length growth, we define a specific kind of multiplication. Let the secret vector satisfy the following condition:

$$x_i x_j = \sum_{k=1}^{n} \gamma_{ijk} x_k \ mod \ m \tag{1}$$

Also generally, the result of two vectors' multiplication can be written as:

$$(v, x)(u, x) = \sum_{i=1}^{n} v_i x_i \sum_{j=1}^{n} u_j x_j = \sum_{i,j=1}^{n} v_i u_j x_i x_j \tag{2}$$

One can see that the right part of the expression is a quadratic function. Let us associate this function with the linear one according to rule:

$$x_i x_j = \sum_{k=1}^{n} \gamma_{ijk} x_k$$

So, let us rewrite (2):

$$\sum_{i,j=1}^{n} v_i u_j x_i x_j = \sum_{i,j=1}^{n} v_j u_j \sum_{k=1}^{n} \gamma_{ijk} x_k =$$

$$= \sum_{k=1}^{n} \left(\sum_{i,j=1}^{n} v_i u_j \gamma_{ijk} \right) x_k$$

This function can be represented as (w, x) and the components of vector w can be defined using the components of initial vectors v and u as follows:

$$w_k = \sum_{i,j=1}^{n} v_i u_j \gamma_{ijk} \tag{3}$$

In other words, we describe the specific kind of vectors' multiplication. According to (1), (2):

$$(v, x)(u, x) = (w, x)$$

Let us call the rule (3) the multiplication table and γ_{ijk} - the structural constants.

Such a determination of multiplication table is similar to the definition of algebra. But there is an important difference: the structural constants have no constraints such as commutativity, associativity and presence of "unit".

In order to avoid the evident question whether we can find the structural constants that satisfy (1) for every secret vector or not, let us indicate the method of its construction. Let us represent the structural constants as a set of vectors:

$$\gamma_{ij} = (\gamma_{ij1}, \dots, \gamma_{ijn})$$

Thus, rewritten (1) looks as follows:

$$x_i x_j = (\gamma_{ij}, x) \ mod \ m, i, j = 1, \dots, n \tag{4}$$

If we consider (4) as a set of linear equations with a given left part $x_i x_j$ and n^3 variables γ_{ijk}, these unknown variables are found ambiguous for every equation due to its non-trivial kernel.

The problem is the fact that in order to produce the real computations we need to disclose the structural constants. It is unobvious whether it is possible in this case to determine the secret vector with given constants. This question is equivalent to the question whether we can find a solution of the following system (if the coefficients γ_{ijk} are given):

$$x_i x_j = \sum_{k=1}^{n} \gamma_{ijk} x_k \ mod \ m \tag{5}$$

On the one hand, it is considered that solving the equation (2) in a finite field is a difficult task. But on the other hand, the system (5) consists of n^2 equations in reference to n variables, i.e. highly overdetermined. For highly overdetermined systems of equations it is expected that the solution is unique. There is one more argument to justify the complexity of the problem: the prime number is a secret, therefore, it is still unknown what modulus should be used in order to solve the system.

Thus, let us prove the following.

Theorem 3.1: The secret vector $x = (x_1, \dots, x_n)$ and the structural constants γ_{ijk} can be selected so that the system of equations (5) has at least n solutions.

In order to prove this theorem, let us give the construction of such a set of the structural constants. Let S be an arbitrary $n \times n$ matrix with the only constraint - it should be invertible by given modulus m. Then, choose two arbitrary columns of the matrix with i and j indexes (i and j may be the same). These columns match with two vectors - s_i and s_j respectively.

As it was mentioned, componentwise multiplication of the vectors $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$ is defined by the following rule:

$$u \cdot v = (u_1 v_1, u_1 v_2, \dots, u_n v_n)$$

Let us define vector γ_{ij} as a solution of the equation

$$s_i \cdot s_j = S\gamma_{ij}$$

According to the invertibility of the matrix S, the solution can be rewritten:

$$\gamma_{ij} = S^{-1}\big(s_i \cdot s_j\big) \ mod \ m$$

All the columns of the matrix S satisfy the equation (5), where the structural constants are obtained as it is described above. As matrix S has n rows, we finally get n different solutions of the equation (5).

Remark 3.1: This construction is appropriate for any finite fields.

Remark 3.2: Since n different secret vectors correspond to the same set of structural constants, we can produce n secure computations simultaneously.

IV. Proposed Cryptosystem

In this section we consider the description of the proposed encryption that is based on modular arithmetic.

A. Basics

Firstly, let original message P be an integer number - we impose the only constraint: $P < M, M \sim 2^{64}$ in order to perform all the computations correctly. Then, let us define the encryption's secret key as a triple $(mods, \alpha, x)$, where

- $mods = (m_1, \dots, m_k)$ - a set of k moduli, m_i is prime $\forall i = 1, \dots, k$;
- $\alpha = (\alpha_1, \dots, \alpha_k)$ - a set of k arbitrary vectors needed for generating secret vectors x;
- $x = (x_1, \dots, x_k)$ - a set of k vectors with length n.

Thus, to encrypt P we should represent it as a set of residues $(P_1, \dots, P_k) : P_i = P \ mod \ m_i$ and after that construct a vector c_i for every P_i such that it satisfies the following condition:

$$(c_i, x_i) \ mod \ m_i = P_i$$

A set of vectors $C = (c_1, \dots, c_k)$ is a ciphertext for the initial number P.

B. Multiplication table

Before presenting the essence of the proposed encryption algorithm, let us describe the special multiplication table $T = (\gamma_{ijk})$ introduced in the previous section. Matrix T is used for the computations over ciphertexts in order to avoid the increase in the data lengths. We can work with the only multiplication table for all the moduli, but also we can generate k different tables for k different moduli. Let us consider this method for the chosen modulus m_i and fix the index i for all used terms; so then, we work simply with modulus m.

In order to generate such a table we need matrix S described in Section 2. Thus, computing the constants γ_{ijk} for every couple of i and j we get the specific multiplication table $T = (\gamma_{ijk})$ for the fixed modulus $m = m_i$.

Let us note one more feature of the multiplication table. If we construct matrix T as a non-symmetric matrix, we will get different results while computing $(a_i \cdot a_j) \cdot a_k$ and $a_j \cdot (a_i \cdot a_k)$. It means that the operation of multiplication has no associative and commutative properties. Also this fact invokes a non-deterministic character of the proposed encryption scheme.

C. Cryptosystem

Our fully homomorphic cryptosystem consists of three algorithms $(KeyGen, Enc, Dec)$, where

- $KeyGen$ - the probabilistic key generation algorithm that constructs the key;
- Enc - the encryption algorithm that takes initial message P, $mods$ - a part of the secret key and the multiplication tables T as the input parameters and returns a ciphertext C;
- Dec - the decryption algorithm that uses the secret key and the ciphertext C, returns the original message P.

1) Key Generation: As it was previously mentioned, the encryption key is secret and consists of the set of the relatively prime moduli and two sets of vectors. Let us consider the way of key generation in details.

Step 1. Let S be an arbitrary $n \times n$ matrix with non-zero determinant $det(S)$. Then we choose k relatively prime moduli $(m_1, \ldots, m_k)$ with a condition: $gcd(m, det(S)) = 1$. It is necessary in order to provide the invertibility of S by each modulus. Thus, matrix S for each modulus will be computed as follows:

$$S_i = (s_{ij}) \ mod \ m_i$$

Step 2. Then we should construct an arbitrary vector $\alpha = (\alpha_{i1}, \ldots, \alpha_{in})$ associated with modulus m_i using a rule:

$$\forall \alpha_{ij} \ \exists \alpha_{ij}^{-1} : \alpha_{ij} \cdot \alpha_{ij}^{-1} = 1 \ mod \ m_i$$

This rule means that every element of vector α_{ij} is invertible by chosen modulus m_i.

Besides we should provide the existence of at least two relatively prime elements in vector α_i in order to solve diophantine equations in the Enc algorithm.

A set of vectors α_i is also a part of the secret key.

Step 3. At the last step of key generation we compute x_i from the equation:

$$\alpha_i = Sx_i$$

Due to the fact that matrix S_i is invertible by modulus m_i:

$$x_i = (S_i^{-1} \alpha_i) \ mod \ m_i \ \forall m_i$$

Therefore, after key generation process we have k moduli $(m_1, \ldots, m_k)$ and the set of k secret vectors $x = (x_1, \ldots, x_n)$ constructed using the set of α_i. It is worth noting that the generation method is probabilistic due to the arbitrariness of S and α_i selection.

2) Encryption: The input parameters for this algorithm are the original message P - an integer number that satisfies the following constraint: $P < M, M \sim 2^{64}$, the secret key and the set of multiplication tables $(T_1, \ldots, T_k)$.

Step 1. Let us start with the computing the set $(P_1, \ldots, P_k)$ as follows:

$$P_i = P \ mod \ m_i \ \forall i = 1, \ldots, k$$

Step 2. Using vectors of the secret key $(\alpha_1, \ldots, \alpha_k)$, consider the equation:

$$P_i = (\alpha_i, y_i) = \alpha_{i1} y_{i1} + \cdots + \alpha_{in} y_{in} \qquad (6)$$

Then compute the set of y_i as a result of the diophantine equation. Let us describe the way of solving such an equation in details. Due to the existence of two relatively prime components in every vector α_i the solution of this equation can be found as follows: let the position of two coprime integers be r and s, then choose random values for the coefficients $y_{iq} : q = 1, \ldots, n, q \neq r, q \neq s$ and substitute them into the equation (6). Thus, we get a linear diophantine equation with only two variables:

$$P_i - \sum_{q=1, q \neq r,s}^{n} \alpha_{iq} y_{iq} = \alpha_{ir} y_{ir} + \alpha_{is} y_{is} \qquad (7)$$

The equation (7) can be solved, because the coefficients α_{ir} and α_{is} are relatively prime. Therefore, the values of the components y_{ir} and y_{is} can be computed using the Euclidean algorithm.

Also we can use the multiplication table T_i in order to solute such an equation. In this case we should only substitute $x_i x_j$ in the formula (5) with P_i.

Step 3. Compute a cipher $C = (c_1, \ldots, c_k)$ using the following rule:

$$c_i = (y_i \cdot S_i) \ mod \ m_i$$

The result of the encryption algorithm is the ciphertext C that consists of k vectors of length $n : (c_1, \ldots, c_k)$. Thus, cipher C is a $k \times n$ matrix.

3) Decryption: The algorithm's input parameters are the ciphertext C, described previously, and the secret key.

Step 1. Compute a set $(P_1, \ldots, P_k)$ as follows:

$$P_i = (c_i, x_i) \ mod \ m_i \qquad (8)$$

Let us prove the correctness of the equation (8) using previously given formulas of the encryption algorithm and the properties of the standard linear algebra:

$$(c_i, x_i) \ mod \ m_i = (y_i S_i, x_i) \ mod \ m =$$
$$= (y_i, \alpha_i) \ mod \ m = P_i$$

Step 2. As we have the set of P_i, apply the Chinese remainder theorem [3] and get the original integer number P that satisfies the next condition:

$$P \equiv \begin{cases} P_1 (mod \ m_1) \\ \vdots \\ P_k (mod \ m_k) \end{cases}$$

Let us consider the modification of the algorithm that provides the probabilistic character of the encryption in order to improve its security. Let C be a ciphertext for the initial number P. First of all, we compute a ciphertext corresponding to zero - C_0, then multiply it by an arbitrary coefficient θ. After that we add the result $\theta \cdot C_0$ to the ciphertext C:

$$C' = C + \theta \cdot C_0$$

Then C' is called a new ciphertext for the number P. As our encryption is fully homomorphic we may be sure the ciphertext C' is appropriate for P. So, to get the original message P, we should decrypt C' only. Thus, the proposed modification improves the complexity of the encryption algorithm. Such a modification is considered as a primary encryption algorithm. Its security evaluation will be discussed in the following section.

To conclude, in this section the details of the proposed fully homomorphic scheme were given. Briefly, let us mention the main features of this scheme again. The secret key is a triple $(mods, \alpha, x)$. We decided to perform all of the secure computations using modular arithmetic in order to avoid growth of the integers' size. Also the specific kind of vectors' multiplication that allows performing arithmetical operations over ciphertexts without the increase in the resulting vectors' length was proposed. Then the probabilistic modification of our FHE scheme was described.

V. Encryption Security Evaluation

In order to analyze the complexity of the proposed FHE scheme, we provide some information about its efficiency:

- $O(k \cdot n^2)$ is the complexity of key generation algorithm;
- $O(n^3)$ is the complexity of multiplication table generation process;
- $O(n^2)$ is encryption algorithm's complexity;
- $O(k^2 \cdot n)$ is decryption algorithm's complexity.

In previous section it was mentioned that we might consider the proposed fully homomorphic encryption as a probabilistic one. The probabilistic encryption algorithm means that we get different ciphertexts if we encrypt the same plaintext more than once. Obviously such a modification prevents our scheme from common attacks, i.e. chosen ciphertext or plaintext attacks.

VI. FHE Applications

The proposed homomorphic encryption can be used in a multiple applications due to its practical allowance and acceptable data overhead. It's main purpose - as it was stated previously - to perform mathematical operations over encrypted data in untrusted and non-interactive environments without access to the encryption keys or initial data. So, the proposed solution can be practically used in the following cases of the secure computations.

A. Computation in Database

Databases and cloud databases, as a special case of cloud services, are affected by the same problem of keeping data confidentiality. Such a problem arises when a customer does not trust a database provider and/or an administrator or is not sure about security of connection between end user machine and database server [13]. Analogically, Fully Homomorphic and Order Preserving encryptions (OPE) can be applied to solve problem of keeping confidentiality of database entries. Properties of FHE and OPE allow users to perform any kind of computations (of course, with corresponding limitations) inside DBMS and the end user should decrypt only the result of selected data. Such an approach was implemented in MIT CryptoDB [14] and was positively acclaimed by the academy and the industry.

Alternatively we designed and developed a solution for secure Database [15]. We use proprietary developed OPE [16], proposed in this article FHE and strong deterministic encryptions. Main idea of our approach to secure database is to intercept user SQL queries on a flexibly configurable proxy server, encrypt vulnerable user's data and change the syntax of queries according to encryption's output ciphertext. Responses from DBMS are decrypted in a proper way and displayed to the user. The feature of granular security allows different encryptions to be applied to different columns in SQL table and perfectly accommodates user's requirements. Combination of implemented encryptions with carefully designed secure database architecture allowed us to achieve significantly low overhead of data flow and SQL queries' execution time. Estimated average overhead is around 20%.

This project allowed us to validate developed homomorphic encryption and to show its practical acceptance. Thus, we can perform secure computations over ecrypted data directly in protected database due to the properties of FHE. That is why such an application is primary for the proposed fully homomorphic encryption.

B. Cloud Computation

Cloud technologies are very popular and wide spread nowadays. Although customers of cloud services are very excited by cloud features and benefits that cloud has brought to enterprises, they are very concerned about security, particularity confidentiality, of data stored and processed in a cloud [7]. Those concerns are caused by several security issues of cloud technology in common, such as insider threat [8], possible security breach [9], intervention of special services into citizens privacy [10] and any other case of unauthorized access to vulnerable user data. There are multiple solutions [11][12] to described problem and one of them is usage of encryptions. Using homomorphic encryption or order preserving encryptions will allow business users to perform variety of operations over data stored in cloud data centers without necessity of massive computations on customers' side. Such a scenario will possibly lower expenses, while ensuring confidentiality of customer's data.

C. Constructing Public-Key Cryptosystem

Firstly we consider the application of fully homomorphic encryption for constructing linear and polynomial public-key cryptosystems. It is worth to note that we use the simplified method of the proposed encryption with fixed parameters: $k = 1, n = 4$. It means that we have the only modulus m and the only secret vector x.

The linear one is based on the Hill cipher [20]. In common way Hill cipher matches an original vector p to a ciphertext c according to the rule: $c = A \cdot p \ mod \ m$, where a square matrix A and a modulus m are secret. Besides, the matrix A should be invertible by the modulus m in order to provide the correctness of the decryption process. It is obvious that such a method is vulnerable to the plaintext attack. That is why the main idea of our approach is to hide the secret matrix A using the proposed FHE for its encryption. Also we encrypt the initial message with fully homomorphic algorithm E. Then we get a ciphertext, a result of public-key encryption, according to the rule: $c = E(A) \cdot E(p) \ mod \ m$.

The second, polynomial, cryptosystem is based on the analogue of the well-known RSA algorithm [19] where the modulus m is secret. Unfortunately this construction is unstable, but we can modify it using our fully homomorphic encryption. Thus, we propose to encrypt original number with the FHE algorithm E and after that raise the result of encryption to the power: $\big(E(p)\big)^e \ mod \ m$.

Let us consider the details of the polinomial cryptosystem via some examples.

1) Keys generation: Secret key consists of the components of the proposed homomorphic encryption's key:

$$m = 659$$

$$x = (176\ 657\ 361\ 197)$$

Public key includes an integer number $e = 3$ that is invertible by modulus $\phi(e)$ (where $\phi(a)$ is the Euler function for a) and the multiplication table γ_{ijk} that contains 16 vectors (or $4^3 = 64$ elements):

$$\gamma_{11} = (319\ 77\ 626\ 452)$$

$$\gamma_{12} = (80\ 182\ 161\ 229)$$

$$\gamma_{13} = (542\ 527\ 513\ 623)$$

$$\gamma_{14} = (2\ 148\ 241\ 557)$$

$$\gamma_{21} = (281\ 131\ 618\ 399)$$

$$\gamma_{22} = (568\ 414\ 276\ 590)$$

$$\gamma_{23} = (404\ 220\ 384\ 640)$$

$$\gamma_{24} = (238\ 252\ 389\ 179)$$

$$\gamma_{31} = (253\ 620\ 610\ 313)$$

$$\gamma_{32} = (304\ 88\ 55\ 421)$$

$$\gamma_{33} = (5\ 565\ 352\ 650)$$

$$\gamma_{34} = (63\ 390\ 604\ 279)$$

$$\gamma_{41} = (478\ 460\ 120\ 176)$$

$$\gamma_{42} = (78\ 568\ 258\ 224)$$

$$\gamma_{43} = (59\ 332\ 90\ 33)$$

$$\gamma_{44} = (432\ 103\ 198\ 222)$$

The size of secret and public keys is 2.5 Kb for the chosen parameters k and n.

2) Encryption: The initial number is an integer $p = 123$. The first step of the algorithm is to encrypt p using our fully homomorphic encryption. In other words, we should match p with a vector c that satisfies the following condition: $(c, x)\ mod\ m = p$.

$$123 \xrightarrow{Hom} \begin{pmatrix} 27458280 \\ 16546176 \\ 35555955 \\ 21767475 \end{pmatrix}$$

The second step is to raise the result of the FH encryption to the appropriate power e:

$$z = \begin{pmatrix} 27458280 \\ 16546176 \\ 35555955 \\ 21767475 \end{pmatrix}^3 =$$

$$= \begin{pmatrix} 3608973865261560248050671547 56 \\ 4770191334233879129224388094 75 \\ 4887824141231792260989933721 32 \\ 5229006672595046416078439201 58 \end{pmatrix}$$

Vector z is a ciphertext for the initial number p.

3) Decryption: First, let us multiply the ciplertext z and the secret vector x. It is obvious that as a result we get the initial number p raised to the power e:

$$(z, x)\ mod\ m = (c^e, x)\ mod\ m = p^e\ mod\ m$$

According to the example:

$$\begin{pmatrix} 360897386526156024805067154756 \\ 477019133423387912922438809475 \\ 488782414123179226098993372132 \\ 522900667259504641607843920158 \end{pmatrix} \cdot \begin{pmatrix} 176 \\ 657 \\ 361 \\ 197 \end{pmatrix}^T mod\,659 =$$

$$= 510$$

Then, let us raise the result of the previous operation to the power d, where $d = e^{-1}\ mod\ \phi(m)$:

$$\left(p^e\ mod\ m\right)^d = p^{ed}\ mod\ m = p$$

Next, substitute the real values:

$$d = 3^{-1}\ mod\ 658 = 439$$

$$510^{439}\ mod\ 659 = 123$$

Finally we get the initial number $p = 123$.

Implementation of these cryptosystems demonstrates that all of the arithmetical calculations over encrypted data are correct. Also it proves that the multiplication of ciphertexts doesn't lead to the increase in dimension of multiplication results. This is the illustration of first practical use of the proposed FHE scheme.

D. Government Defensive Purpose

It is obvious that modern warfare needs a lot of computations. A part of these computations is done on machines using a software that are produced in foreign countries (for one fixed country), thus can not be fully trusted, because of possible hardware and software Trojans [17][18]. This problem of lack of trust can be solved by producing in a secure way the FH hardware encryptors. In the same time all untrusted computers will perform computations only over encrypted data.

All the mentioned applications are only examples of secure computation and described in this section as the illustrations of a wide area of the proposed homomorphic encryption usage.

REFERENCES

[1] C. Gentry, "A fully homomorphic encryption scheme," [Online]. Available: http://crypto.stanford.edu/craig/craig-thesis.pdf.

[2] C. Gentry and S. Halevi, "Implementing Gentry's Fully-Homomorphic Encryption Scheme," *in Advances in Cryptology - EUROCRYPT 2011,* pp. 129–148. DOI: 10.1007/978-3-642-20465-4_9. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20465-4_9

[3] D. Knuth, *The Art of Computer Programming Seminumerical Algorithms,* vol. 2, Addison-Wesley Pub. Co., 1981.

[4] "Programming Computation on Encrypted Data," Broad Agency Announcement DARPA-BAA-10-81, Defense Advanced Research Projects Agency, 2010.

[5] R. Rivest, L. Adleman and M. Dertouzos, "On data banks and privacy homomorphisms," *in Foundations of Secure Computation,* 1978, pp. 169–180.

[6] D. Stehle and R. Steinfeld, "Faster Fully Homomorphic Encryption," *on Asiacrypt conference,* http://eprint.iacr.org/2010/299.pdf, 2010.

[7] "Cloud Computing Top Threats in 2013," *The Notorious Nine,* Cloud Security Alliance, [Online]. Available: https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf.

[8] W. R. Claycomb and A. Nicoll, "Insider Threats to Cloud Computing: Directions for New Research Challenges," *in Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference,* 2012, pp. 387–394. DOI: 10.1109/COMPSAC.2012.113. [Online]. Available: http://dx.doi.org/10.1109/COMPSAC.2012.113

[9] "Chronology of data breaches," Privacy Rights Clearinghouse, [Online]. Available: http://www.privacyrights.org/data-breach.

[10] "Interview with Whistleblower Edward Snowden on Global Spying," *Der Spiegel,* 2013.

[11] J. Zhou, "On the security of cloud data storage and sharing," *in Proceedings of the 2nd international workshop on Security in cloud computing,* 2014, pp. 1–2. DOI: 10.1145/2600075.2600087. [Online]. Available: http://doi.acm.org/10.1145/2600075.2600087

[12] A. J. Feldman, W. P. Zeller, M. J. Freedman and E. W. Felten "SPORC: Group collaboration using untrusted cloud resources," *in Proceedings of the 9th Symposium on Operating Systems Design and Implementation,* Vancouver, Canada, 2010.

[13] "OpenSSL Heartbleed Vulnerability," *Cyber Security Bulletins,* Canada, 2014.

[14] S. Tu, M. F. Kaashoek, S. Madden and N. Zeldovich, "Processing Analytical Queries over Encrypted Data," *in Proceedings of the 39th International Conference on Very Large Data Bases (VLDB),* Trento, Italy, 2013, pp. 289–300. DOI: 10.14778/2535573.2488336. [Online]. Available: http://dx.doi.org/10.14778/2535573.2488336

[15] K. Shatilov, V. Boiko, S. Krendelev, D. Anisutina and A. Sumaneev, "Solution for Secure Private Data Storage in a Cloud," *in Proceedings of the Federated Conference on Computer Science and Information Systems,* 2014, pp. 885–889. DOI: 10.15439/2014F43. [Online]. Available: http://dx.doi.org/10.15439/2014F43

[16] M. Usoltseva, S. Krendelev and M. Yakovlev, "Order-preserving encryption schemes based on arithmetic coding and matrices," *in Proceedings of the Federated Conference on Computer Science and Information Systems,* 2014, pp. 891–899. DOI: 10.15439/2014F186. [Online]. Available: http://dx.doi.org/10.15439/2014F186

[17] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *in IEEE Des. Test,* 2010, pp. 10–25. DOI: 10.1109/MDT.2010.7. [Online]. Available: http://dx.doi.org/10.1109/MDT.2010.7

[18] R. Lehtinen, D. Russell and G. T. Gantemi, "Computer Security Basics," *O'Reilly,* 2006.

[19] A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," *CRYPTO,* 1982, pp. 279–288.

[20] L. S. Hill, "Cryptography in an Algebraic Alphabet," *The American Mathematical Monthly,* vol. 36, 1929, pp. 306–312.