Methods for secure cloud processing of big data

Yerzhan N. Seitkulov, Seilkhan N. Boranbayev, Gulden B. Ulyukova, Banu B. Yergaliyeva, Dina Satybaldina
Gumilyov Eurasian National University, Kazakhstan

Article Info

Article history:

Received Apr 23, 2021
Revised Apr 23, 2021
Accepted May 5, 2021

Keywords:

Big data
Cloud computing
Information security
Internet of things
Secure outsourcing

ABSTRACT

We study new methods of secure cloud processing of big data when solving applied computationally-complex problems with secret parameters. This is one of the topical issues of secure client-server communication. As part of our research work, we model the client-server interactions: we give specific definitions of such concepts as “solvable by the protocol”, “secure protocol”, “correct protocol”, as well as actualize the well-known concepts-“active attacks” and “passive attacks”. First, we will outline the theory and methods of secure outsourcing for various abstract equations with secret parameters, and then present the results of using these methods in solving applied problems with secret parameters, arising from the modeling of economic processes. Many economic tasks involve processing a large set of economic indicators. Therefore, we are considering a typical economic problem that can only be solved on very powerful computers.

Corresponding Author:

Yerzhan N. Seitkulov
Gumilyov Eurasian National University
2 Satpayev str., Nur-Sultan city, Kazakhstan
Email: yerzhan.seitkulov@gmail.com

1. INTRODUCTION

In this paper we investigate new methods for secure cloud processing of big data when solving applied computationally-complex problems with secret parameters. As a rule, standard cryptographic protocols are used to ensure the security of client-server communications. These cryptographic methods are effective for big data storage tasks, but are not always acceptable for secure information processing tasks. For example, the well-known mathematical methods of homomorphic encryption still have no practical application due to the huge computational costs on the client side. Therefore, along with classical cryptographic methods, it is necessary to use alternative (non-cryptographic) methods and technologies for protecting information. Such methods, as a rule, are used for the safe processing of big data arising in the mathematical modeling of economic problems, in linear programming problems, which, for one reason or another, may contain secret parameters [1]-[24].

Our problem can be described as follows. We will assume that a "client" is an entity who wishes to secure use an insecure server to solve some computationally-complex problem, that is, the client wishes to secure process big data on the server. As a server a supercomputer can be used, which is feasible for the implementation of this computationally-complex task. It is important to note that in this setting, the server is not trusted by the client; therefore, sensitive or confidential information must be protected from the server, and the results of the server's computations, generally speaking, must be easily verified by the client. Further, own computing devices can act as a server, but controlled unscrupulous or careless employees. Therefore, an adversary who wants to intercept classified information can also act as a server. So, formally, the server is simultaneously an adversary, and the data sent to it represents a computationally-complex problem that it must solve in encrypted form.

Further, the concept of "client" will be relative, and it will depend on the considered class of problems with secret parameters, which must be solved using the server in encrypted form. Therefore, a "regular" computer, but with limited computing resources, can serve as a client. In this case, the server acts as an ideal supercomputer that the client can use on a contractual basis to solve it. This client-server interaction can be given the following protocol form (see [8]).

Let the client need to solve some computationally-complex problem Z, depending on the secret parameter α: $Z(\alpha)$. Suppose that there is a certain algorithm (scheme) A for solving problem $Z(\alpha)$, which can be efficiently implemented using the computing resources of the server, but not on the client side. Protocol Z: i) The client decomposes Algorithm A into two Algorithms A1 and A2, so that three conditions are met: firstly, solving algorithms A1 and A2 allows solving problem Z; secondly, the A1 algorithm can depend on the secret parameter, and the A2 algorithm either does not depend on the secret parameter α at all, or the time required for the server to reveal the secret from the A2 algorithm is unacceptable for it. Thirdly, the client can calculate A1 quickly enough, ii) the client solves A1, and sends the A2 to the server, iii) the server solves the computationally-complex problem A2, and returns the result of the calculation to the client, iv) the client, having received the result of computing the computationally-complex problem A2 from the server, solves the original problem Z.

It should be noted that, generally speaking, the obtained solution to Problem Z may not be a secret. For example, suppose the client needs to compute $y = x^{d} \bmod n$, where d is the client's secret parameter, while integers n and e such that $ed \equiv 1(\varphi(n))$, are public. The integer n is the product of the secret prime's p and q. If the interceptor knows the numbers y, x and n, then it is almost impossible to determine the secret parameter d anyway, and this problem is known as the discrete logarithm problem.

We need the following generally accepted definitions [8]:

- Definition 1. We say that a computationally-complex problem is solvable by some protocol if the client receives a solution to the original problem as a result of executing each step of this protocol. In all cases, by a solution we mean an approximate solution. The task that the client sends to the server is first reduced to a certain scheme, according to which it will be solved on a supercomputer. That is, the client orders the server to solve the problem according to some scheme (algorithm) with a given accuracy.
- Definition 2. We will say that a protocol is secure if the client's secret parameters cannot be declassified during interaction with the server. Moreover, if the server determines a certain set, the elements of which are probable secret parameters, then the cardinality of the set must be at least countable (this excludes the probabilistic approach and the possibility of enumeration).
  The following concepts are also important.
- Definition 3. An active attack is a case when the server can send false decisions to the client. A protocol is called resistant to active attack if the client can verify the solution received from the server within a reasonable time for the client. For example, if the server sends the client an approximate solution x of some matrix equation Ax=f, then the client can verify the server's computation result by simply multiplying the matrix A by the vector x, which should be approximately equal to the vector f. That is, the client solves a direct problem.
- Definition 4. We say that a protocol is correct if the total time required to implement the protocol is less than the time the client solves the problem on its own, without the help of the server. In this case, $\mathrm{Comm}(\alpha)$, $\mathrm{Comp}_C(\beta)$, $\mathrm{Comp}_S(\gamma)$ Comm (α)-denote the time required to transmit a message α between the server and the client, the time the client executes algorithm β and the time it takes to execute the algorithm γ by the server, respectively. And by T (Z) we denote the time required to implement the protocol Z. If some algorithm β is not calculated at all on the client's side, then we will write $\mathrm{Comp}_C(\beta) = \infty$.

The time T (Z) required to implement protocol Z will not include the time required by the client to test the protocol for resistance to an active attack. That is, T (Z) is the time required to implement the Z protocol, if the server does not deviate from the protocol, that is, it sends only the correct decisions to the client (in this case, we speak of a passive attack, that is, the real attack occurs by an information interceptor). There is an explanation for this: if the server sends a false solution to some computationally complex problem, and the client detects this within an acceptable time for it, then the goal is not achieved, that is, the protocol is not implemented. Therefore, it is advisable to denote by T (Z) - the time required only with a clear implementation of the Z protocol.

All considered protocols in the article will be correct, because we will assume that the task is not calculated at all on the client's side, or it is calculated in an unacceptable time. Therefore, the concept of correctness will be relative and depends on a specific class of problems. Each time the correctness of the protocol will be substantiated that the computationally-complex part A2 of the problem Z (α) is transferred to the server, and the client needs to solve the computationally easy task A1.

2. RESEARCH METHOD

Let M be a complete metric space and B a continuous operator taking an element from M to itself, that is;

$$B : M \to M.$$

The completeness of the space M is necessary for the possibility of finding an approximate solution. Generally speaking, if M is assumed to be an arbitrary metric space, then some problems may turn out to be algebraic. For example, if M consists of two numbers 0 and 1, then the problem of finding an approximate solution as such is not worth it. As shown where $x \in M$, $b \in M$.

Consider the problem;

$$Bx = b, \qquad (1)$$

Suppose that problem (1) is uniquely solvable. Let the client needs to approximately solve a computationally-complex in (1) with respect to the unknown x. To find an approximate solution on a computer, in many cases it is required to reduce the equation to a discrete analogue. However, we will present several protocols for solving in (1) only at the ideological level, since the considered (1), generally speaking, is abstract. Task Z_2: The client needs to approximately solve (1) for an unknown $x \in M$. Suppose that transformation B is a secret element of the client, and the right-hand side $b \in M$ is not a secret. We also require that the solution to (1) remain a secret.

2.1. Protocol Z_2

- The client finds a bijective operator at random $D: M \to M$. Next, it calculates the composition $BD \equiv G$ and sends the server to solve the equation with accuracy ε:

  $$Gy = b,$$

  while the client keeps operator D secret.
- The server solves the equation Gy = b and returns to the client an approximate solution y.
- The client finds an approximate solution to (1) by the formula;

  $$x = Dy$$

Let $T_1 = \mathrm{Comp}_C(D, BD)$ be the time required for the client to construct a bijective operator D and calculate the composition BD, T_2 $\mathrm{Comm}(G)$ is the time required to transmit a message G to the server, $T_3 = \mathrm{Comp}_S(y: Gy = b)$ is the time it takes for the server to solve the equation Gy=b, $T_4 = \mathrm{Comm}(y)$ is the time it takes for the server to send the message y, and $T_5 = \mathrm{Comp}_C(Dy)$ is the time required by the client to calculate Dy. By Comp_C $(x : Bx = b)$ we denote the time (which can be equal to ∞) required for the client to solve (1) without the help of the server. Let $T(Z_2) = T_1 + T_2 + T_3 + T_4 + T_5 < \mathrm{Comp}_C(x: Bx = b)$.

Statement 1. Task Z_2 is solvable by protocol Z_2 if BD and Dy are calculated on the client side, and the equation Gy = b is solved on the server. Further, the Z_2 protocol is Resistant to active attack if Gy is calculated on the client side; and secure.

Indeed, we have;

$$B(Dy) = B\big(D(G^{-1}b)\big) = BDD^{-1}B^{-1}b = b,$$

therefore, if the server does not deviate from the protocol, then the client finds an approximate solution to (1) by the formula x=Dy, that is, the problem Z_2 is solvable by this protocol.

- Resistance to active attack. Since the server sends the solution y to the client, the client verifies the server's computation result by simply calculating the direct problem Gy, which should be approximately equal to b: $\rho(Gy, b) < \varepsilon$.

- Security. The server knows the composition of the two operators BD = G, but separately the operators B and D are not known to the server. Therefore, the secret parameter B, as well as the solution x of (1), remain secret from the server.

Correctness of the protocol Z_2. Constructing an arbitrary bijective operator D is often less difficult than finding a solution to an arbitrary (1); therefore, the assumption $T(Z_2) < \mathrm{Comp}_C(x: Bx = b)$, which determines the correctness of the protocol, is justified. Example. Suppose the client needs to solve the equation 5x=150, where the number 5 is the client's secret parameter.

2.2. Protocol Z_2'

- Let 3 be a random secret number taken by the client. Now the product 5*3=15 and the right side 150 the client sends to the server.
- The server solves the equation 15y=150 and receives the number y = 10, which it sends to the client.
- The client finds a solution to the original equation by the formula x=3*10=30.

Task Z_3: The client needs to approximately solve (1). Suppose that the client's secret parameter is the right-hand side b of the equation, and operator B is not a secret. We also require that the solution to (1) remain a secret.

2.3. Protocol Z_3

- The client randomly finds bijective operators D, K: M → M. Next, calculates KBD ≡ G, Kb ≡ g and sends them to the server so that it solves the following equation with accuracy ε

$$Gy = g,$$

the client keeps the D and K operators as secrets.
- The server solves the equation Gy = g and returns an approximate solution y to the client.
- The client finds an approximate solution to (1) by the formula;

$$x = Dy.$$

Let $T_1 = \mathrm{Comp}_C(D, K, KBD)$ be the time required for the client to build reversible operators D and K and calculate the composition KBD, $T_2 = \mathrm{Comm}(G, g)$ is the time required to transmit the messages G, g to the server, $T_3 = \mathrm{Comp}_S(y: Gy = g)$ is the time it takes for the server to solve the equation Gy=g, $T_4 = \mathrm{Comm}(y)$ is the time it takes for the client to send the message y to the server, and $T_5 = \mathrm{Comp}_C(Dy)$ is the time it takes for the client to compute Dy. By Comp_C (x : Bx = b) we denote the time (which can be equal to ∞) required for the client to solve (1) without the help of the server. Let $T(Z_3) = T_1 + T_2 + T_3 + T_4 + T_5 < \mathrm{Comp}_C(x: Bx = b)$.

Statement 2. Task Z_3 is solvable by protocol Z_3 if KBD, Kb and Dy are computable on the client side, and Gy=g is solvable on the server. Further, the protocol Z_3 is resistant to active attack if Gy is computable on the client side; and secure.
Indeed, we have;

$$B(Dy) = B\big(D(G^{-1}g)\big) = B(D(D^{-1}B^{-1}K^{-1}g))=K^{-1}g=b.$$

Hence x = Dy. That is, the task Z_3 is resolvable by the protocol Z_3 .

- Resistance to active attack. Since the server sends the client an approximate solution y, the client verifies it by calculating it simply by solving the direct problem Gy, which should be approximately equal to g: $\rho(Gy, g) < \varepsilon$.
- Security. The server knows the composition of the operators KBD = G, but separately the operators K and D are not known to the server; the server also knows the result of calculating the two secret elements K and b: Kb = g, so the operator K also remains a secret. This means that element b remains secret. It also follows from this that the solution x of (1) remains a secret.

Correctness of the protocol Z_3. In the general case, the construction of arbitrary bijective operators D and K is often less difficult than finding a solution to an arbitrary (1); therefore, the assumption $T(Z_3) < \mathrm{Comp}_C(x: Bx = b)$, which determines the correctness of the protocol, is justified. Example. Suppose the client needs to solve the equation 5x=150, and the number 150 is the client's secret parameter.

2.4. Protocol Z_3'

- Let K = 3 and D = 10 be random secret numbers taken by the client. Now the client sends the calculation results of the products 3*5*10 = 150 and 3*150 = 450 to the server.
- The server solves the equation 150y = 450 and receives the number y = 3, which it sends to the client.

– The client finds a solution to the original equation by the formula x = D*y = 10*3 = 30.

In protocols Z_2 and Z_3, it was assumed that in (1) the secret parameter is either operator B or the right-hand side of b. Sometimes it may turn out that the secret parameters of the client are both operator B and the right side of b. In this case, the task for the client is simplified, since the less the server knows about the task in question, the more difficult it is for him to recognize it.

Problem Z_3^*: Suppose the client needs to approximately solve in (1), keeping the operator B, the right-hand side b and the desired solution x secret. The Z_3 protocol is used for this task. Statement 2': Task Z_3^* is solvable by protocol Z_3 if KBD, Kb and Dy are computable on the client side, and protocol Z_3 is resistant to active attack if Gy is computable on the client side; and secure, i.e., the secrecy of B and b is maintained.

It is enough to show the security of the protocol. The server knows the composition of the operators $KBD = G$, but separately the operators K, B and D are not known to the server, which means that B remains secret. Also, the server knows the result of calculating two secret elements K and b*Kb = g, so the element b also remains secret. It also follows from this that the solution x of (1) remains a secret.

Consider the system of algebraic linear equations

$$Bx = b \tag{2}$$

where B is a rectangular m × n matrix with elements B [i] [j], (i = 0, ..., m - 1; j = 0, ..., n - 1), and b is a vector of length m with elements b [k], (k = 0, ..., m - 1). Suppose that system (2) is consistent, that is, it has at least one solution. Vector b is the client's secret parameter. Then problem (2) according to the LE protocol from [1] is solved as follows.

2.5. Protocol Z_4

– The client takes an n-dimensional vector at random

```
w = (w[0],...w[n − 1]) and calculates b − Bw = g by the algorithm
for(i = 0; i < m; i + +)
{
c = 0;
for(j = 0; j < n; j + +)
c = c + B[i][j] * w[j];
g[i] = b[i] - c;
}
```

Now the client is sending to the server the equation $By = g$, and keeps the vector w as secret.

– The server solves the equation

$$By = g$$

and returns an approximate solution to the client $y = (y[0],...y[n-1])$.

– The client finds a solution to (2) using the algorithm

```
for(j = 0; j < n; j + +)
x[j] = y[j] + w[j];
```

This shows that the client needs to do only a few arithmetic operations to solve a system of linear equations. That is, for large n and m, the computational costs of the client are much less than if the client solved the system of linear algebraic (2) without the help of the server.

3. RESULTS AND DISCUSSION

Linear problems are especially common when modeling economic problems, in the problem of finding the extremum of a function, and linear programming. In general, if a certain process is described by a linear mathematical model, then it can be easily solved by the methods described above. In this section, we will look at two specific problems. First, we consider the problem of finding the extremum of a function with secret parameters, and then an applied problem with secret parameters that arises when modeling economic processes.

3.1. Problems of finding the extremum of a function with secret parameters

Consider the problem of determining the extremum of the function:

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j - \sum_{k=1}^{n} b_kx_k \tag{3}$$

subject to communication.

$$\sum_{i=1}^{n} 2c_i x_i = P \tag{4}$$

Here $a_{ij} = a_{ji}$. The client needs to find the extremum of the function (3), subject to the connection (4), using the computing facilities of the server. In this case, the server does not need to know the following secret parameters of the client: $b_1, b_2, \dots, b_n$ и P. Numbers a_{ij} and c_k are not secrets. Protocol Q:

– The client composes a system of linear algebraic equations

$$Aq = b \tag{5}$$

where $b = (b_1, b_2, \dots, b_n, P)$ is free column, $q = (x_1, x_2, \dots, x_n, \lambda)$ − unknown vector, and matrix A has the form.

$$A = \begin{pmatrix} 2a_{11} & 2a_{12} & \dots 2a_{1n} & 2c_1 \\ 2a_{12} & 2a_{22} & \dots 2a_{2n} & 2c_2 \\ \dots & \dots & \dots \dots & \dots \\ 2a_{1n} & 2a_{2n} & \dots 2a_{nn} & 2c_n \\ 2c_1 & 2c_2 & \dots\ 2c_n & 0 \end{pmatrix} \tag{6}$$

Next, the client takes a random vector w, computes $b - w = f$, and sends the equation to the server

$$Ay = f \tag{7}$$

– The server solves (7) and returns solution y to the client.
– The client finds a solution to (6) by the formula

$$x = y + z$$

and finds an extremal point for function (3) subject to constraint (4).

Statement 3. Point $x^* = (x_1, x_2, \dots, x_n)$ is an extreme point, and protocol Q is resistant to active attack; and secure. Resistance to active attack and security follows from the LE protocol from [1]. To find the extremum, the Lagrange method is used.

The Lagrange function has the form:

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j - \sum_{k=1}^{n} b_k x_k + \lambda\left(\sum_{i=1}^{n} 2c_i x_i - P\right)$$

Let q denote the column $(x_1, x_2, \dots, x_n, \lambda)^t$. Then the extreme points are found from the system of linear algebraic (5).

3.2. A linear programming task with secret parameters

Many economic tasks deal with processing a large set of economic indicators. In this subsection we consider a typical economic problem that can be solved only on very powerful computers. The statements of some of the problems considered are taken from [25], and all concepts of a purely economic nature are also defined in this work.

Consider a macroeconomic model [25], in which n products $G_1, \dots, G_n$, m consumers $A_1, \dots, A_m$ and one manufacturer participate. Let some period (for example, a year) produce X_i units of the product G_i. Let, further, the values $P_1, \dots, P_m$ mean, respectively, the incomes of consumers $A_1, \dots, A_m$ for the same period. Suppose that each consumer spends all his income on purchasing products from the manufacturer, and the relative utility of a unit of product G_i for consumer A_s is estimated by the non-negative number $c_i^{(s)}$. Non-negative n-dimensional vector $x^{(s)} = (x_1^{(s)}, \dots, x_n^{(s)})$, whose component $x_i^{(s)}$ is the number of units of product G_i purchased by consumer A_s, assortment set of this consumer. Let $C = (c_{ij})$ be a non-negative matrix, each column and each row of which has at least one positive element. Let $P_1, \dots, P_m$ be positive numbers.

Problem statement: it is required to find a vector $x^{(s)}$ maximizing form:

$$\sum_{s=1}^{m} P_s \ln\left(\sum_{i=1}^{n} c_i^{(s)} x_i^{(s)}\right) \rightarrow \max \tag{8}$$

$$\sum_{s=1}^{m} x_i^{(s)} = X_i, i = 1, \dots, n, \tag{9}$$

$$x_i^{(s)} \geq 0, i = 1, \dots, n, s = 1, \dots, m \tag{10}$$

Here the maximizing function;

$$\sum_{s=1}^{m} P_s \ln(\sum_{i=1}^{n} c_i^{(s)} x_i^{(s)})$$

is the entropy of the macroeconomic system.

Let the client need to solve problem (8)-(10). The client's secret elements are matrix C and vector $X = (X_1, \dots, X_n)$. Vector P is not a secret. Protocol E:

- The client finds diagonal matrices at random $D^j = \mathrm{diag}(d_1{}^j, \dots, d_n{}^j), d_i{}^j > 0, i = 1, 2, \dots, n; j = 1, \dots, m$ and calculate $r_i^{(s)} = c_i^{(s)} d_i^{(s)}$. Further, for s=2,...,m computes n $k_i^{(s)} = \frac{d_i^s}{d_i^1}$, $b_i = \frac{X_i}{d_i^1}$ and sends to the server task (11)-(13) :

$$\sum_{s=1}^{m} P_s \ln(\sum_{i=1}^{n} r_i^{(s)} z_i^{(s)}) \to \max \quad (11)$$

$$z_i^1 + \sum_{s=2}^{m} k_i^{(s)} z_i^{(s)} = b_i, i = 1, \dots, n \quad (12)$$

$$z_i^{(s)} \geq 0, i = 1, \dots, n; s = 1, \dots, m \quad (13)$$

- The server solves the problem (11)-(13) and solution $z_i^{(s)}$ returns to the client.
- The client finds a solution to problem (8)-(10) by the formula

$$x_i^{(s)} = d_i^s z_i^{(s)} \quad (14)$$

Statement 4. Task (8)-(10) is solvable by protocol E, and protocol E is secure. Let's make a replacement;

$$x_i^{(s)} = d_i^s z_i^{(s)}.$$

Then, considering that $r_i^{(s)} = c_i^{(s)} d_i^s, k_i^{(s)} = \frac{d_i^s}{d_i^1}$, $b_i = \frac{X_i}{d_i^1}$, conditions (8) and (9) take the form (11) and (12), respectively. Further, since d_i^s are positive numbers, then condition (10) can be written in the form of condition (13). Protocol security. The server receives a system of equations;

$$r_i^{(s)} = c_i^{(s)} d_i^s \quad (15)$$

$$k_i^{(s)} = \frac{d_i^s}{d_i^1}, \text{ s=2, \dots, m} \quad (16)$$

$$b_i = \frac{X_i}{d_i^1} \quad (17)$$

system (15)-(17) contains $2nm + n$ unknowns, and equations 2nm. Therefore, system (15)-(17) is not uniquely solvable, that is, the server will not be able to determine the secret elements of the client.

4. CONCLUSION

Generally speaking, for the numerical solution of the equation Gy = b, the bijectivity of the operator D is insufficient (for example, continuity is also required). Therefore, this and other protocols that solve the abstract problem (1) should be perceived at the conceptual level. Further, since B and D are one-to-one maps, the equation Gy=b can have only a unique solution.

In the task Z_3^* it is assumed that the secret parameters of the client are all elements of (1), that is, in fact, the server will only know the form of (1). Note that in this setting, the client's task becomes less difficult, since the less the adversary knows about the task under consideration, the more difficult it is to determine the secret parameters. But this does not correspond to practice, since usually the interested enemy knows some parameters of the task. Therefore, in order for the tasks under consideration to have a practical meaning, it is desirable to assume that the enemy knows as much as possible about the computationally complex task under consideration. The joint system of linear algebraic (2) can have infinitely many solutions. Therefore, in the Z_4 protocol, at step 2, the server returns to the client an arbitrary approximate solution of the equation by=g.

ACKNOWLEDGEMENTS

This research is funded by the Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan, grant No. AP09559249, and partially supported by the the Ministry of Digital Development, Innovation and Aerospace of the Republic of Kazakhstan, grant No AP06850817. Also, many thank to Mrs. Ainur Zheptispayeva for technical support.

REFERENCES

[1] Y. N. Seitkulov, "New methods of secure outsourcing of scientific computations," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 469-482, 2013, doi: 10.1007/s11227-012-0809-3.

[2] J. Yu, X. Wang, and Wei Gao, "Improvement and applications of secure outsourcing of scientific computations," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, pp.763–772, 2015, doi: 10.1007/s12652-015-0280-0.

[3] X. Hu and C. Tang, "Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix," *Journal of Cloud Computing*, vol. 4, no. 1, 2015, doi: 10.1186/s13677-015-0033-9.

[4] C. Wang, K. Ren, and J. Wang, "Secure Optimization Computation Outsourcing in Cloud Computing: A Case Study of Linear Programming," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 216-229, 1 Jan. 2016, doi: 10.1109/TC.2015.2417542.

[5] R. Vyas, A. Singh, J. Singh, G. Soni, and B. R. Purushothama, "Design of an efficient verification scheme for correctness of outsourced computations in cloud computing," *Security in Computing and Communications*, vol. 536, pp.66–77, 2015, doi: 10.1007/978-3-319-22915-7_7.

[6] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," *ASIACCS '10: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 2010, pp.48-59, doi: 10.1145/1755688.1755695.

[7] D. Benjamin and M. J. Atallah, "Private and Cheating-Free Outsourcing of Algebraic Computations," *2008 Sixth Annual Conference on Privacy, Security and Trust*, 2008, pp. 240-245, doi: 10.1109/PST.2008.12.

[8] T. Matsumoto, K. Kato, and H. Imai, "Speeding up Secret Computations with Insecure Auxiliary Devices," *Goldwasser S. (eds) Advances in Cryptology-CRYPTO' 88. CRYPTO 1988. Lecture Notes in Computer Science,* vol. 403, pp.497-506, 1988, doi: 10.1007/0-387-34799-2_35.

[9] T. Mefenza and D. Vergnaud, "Cryptanalysis of Server-Aided RSA Protocols with Private-Key Splitting", *The Computer Journal*, vol. 62, no. 8, pp. 1194-1213, 2019, doi: 10.1093/comjnl/bxz040.

[10] K. Zhou, M. H. Afifi, and J. Ren, "ExpSOS: Secure and Verifiable Outsourcing of Exponentiation Operations for Mobile Cloud Computing," in *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2518-2531, Nov. 2017, doi: 10.1109/TIFS.2017.2710941.

[11] S. Hohenberger and A. Lysyanskaya, "How to Securely Outsource Cryptographic Computations," *Theory of Cryptography Conference*, vol. 3378, 2005, pp.264–282, doi: 10.1007/978-3-540-30576-7_15.

[12] P. Béguin and J. Quisquater, "Fast Server-Aided RSA Signatures Secure Against Active Attacks," *Annual International Cryptology Conference*, vol. 963, 1995, pp. 57-69, doi: 10.1007/3-540-44750-4_5.

[13] C. H. Lim and P. J. Lee, "Security and Performance of Server-Aided RSA Computation Protocols," *Annual International Cryptology Conference*, vol. 963, 1995, pp.70–83, doi: 10.1007/3-540-44750-4_6.

[14] C. Castelluccia, E. Mykletun, and G. Tsudik, "Improving Secure Server Performance by Re-balancing SSL//TLS Handshakes," *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006, pp.26–34, doi: 10.1145/1128817.1128826.

[15] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New Algorithms for Secure Outsourcing of Modular Exponentiations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2386-2396, Sept. 2014, doi: 10.1109/TPDS.2013.180.

[16] Y. Wang, *et al.*, "Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage," *European Symposium on Research in Computer Security*, vol. 8712, pp.326–343, 2014, doi: 10.1007/978-3-319-11203-9_19.

[17] P. Q. Nguyen and I. Shparlinski, "On the Insecurity of a Server-Aided RSA Protocol," *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, vol. 2248, 2001, pp. 21–35, doi: 10.1007/3-540-45682-1_2.

[18] J. Merkle, "Multi-round Passive Attacks on Server-Aided RSA Protocols", *CCS '00: Proceedings of the 7th ACM conference on Computer and Communications Security*, 2000, pp.102–107, doi: 10.1145/352600.352616.

[19] B. Pfitzmann and M. Waidner, "Attacks on Protocols for Server-Aided RSA Computation," *Workshop on the Theory and Application of of Cryptographic Techniques*, vol. 658, pp. 153–162, 1993, doi: 10.1007/3-540-47555-9_13.

[20] M. Jakobsson and S. Wetzel, "Secure Server-Aided Signature Generation," *International Workshop on Public Key Cryptography*, vol. 1992, pp. 383–401, 2001, doi: 10.1007/3-540-44586-2_28.

[21] J. Merkle and R. Werchner, "On the Security of Server-Aided RSA Protocols," *International Workshop on Public Key Cryptography*, vol. 1431, pp. 99–116, 1998, doi: 10.1007/BFb0054018.

[22] Y. Aono, "A New Lattice Construction for Partial Key Exposure Attack for RSA," *International Workshop on Public Key Cryptography*, vol.5443, pp. 34–53, 2009, doi: 10.1007/978-3-642-00468-1_3.

[23] J. Blömer and A. May, "New Partial Key Exposure Attacks on RSA," *Annual International Cryptology Conference*, vol. 2729, 2003, pp. 27–43, doi: 10.1007/978-3-540-45146-4_2.

[24] M. Ernst, E. Jochemsz, A. May, and B. de Weger, "Partial Key Exposure Attacks on RSA up to Full Size Exponents," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 3494, pp. 371–386, 2005, doi: 10.1007/11426639_22.

[25] S. E. Kerimkulov, "Mathematical methods for studying macroeconomic processes in Kazakhstan," Disertation, Department Economi Sciences, Almaty, Kazakhstan, pp. 230, 2003.

BIOGRAPHIES OF AUTHORS

Yerzhan N. Seitkulov, was born 1979, Kazakhstan. Current position is PhD, Associate Professor, Director of Information Security and Cryptology Institute, Gumilyov Eurasian National University, Kazakhstan. His interest is information security. Scopus author ID: 24280172200 and researcher ID web of science: P-4905-2014

Seikhan N. Boranbayev, was born 1953, Kazakhstan. Current position is PhD, Professor, departnaent of information sistems, Gumilyov Eurasian National University, Kazakhstan. His interest is information security. scopus author ID: 36238718000 and web of science ID: P-7585-2014

Gulden B. Ulyukova, was born 1990, Kazakhstan. Current position is scientific researcher, information security and cryptology Institute, Gumilyov Eurasian National University, Kazakhstan. Her interest is information security. Scopus author ID: 57192183921.

Banu B. Yergaliyeva, was born 1985, Kazakhstan. Current position is scientific researcher, information security and cryptology Institute, Gumilyov Eurasian National University, Kazakhstan. Her interest is information security. Scopus Author ID: 57191241749.

Dina Satybaldina, was born 1971, Kazakhstan. Current position is Dr, Accociate Professor, Head of Information Security Departent, Gumilyov Eurasian National University, Kazakhstan. Her scientific interest is information security. scopus author ID: 57193740669 and web of science ID: P-1120-2014