Methods for Solving Abstract Equations with Secret Parameters Using External Computer

Yerzhan Seitkulov, Seilkhan Boranbayev, Dina Satybaldina and Askar Boranbayev

September 27, 2021

Methods for solving abstract equations with secret parameters using external computer

Yerzhan Seitkulov[1], Seilkhan Boranbayev[1], Dina Satybaldina[1]

and Askar Boranbayev[2]

[1] Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan
[2] Nazarbayev University, Nur-Sultan, Kazakhstan
yerzhan.seitkulov@gmail.com

Abstract. In this paper we investigate methods for solving abstract equations with secret parameters using external computer. 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 methods and technologies for protecting information. 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. 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.

Keywords: Information security, Cloud computing, Big Data, Secure outsourcing, Internet of Things.

1 Introduction

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 [1-8]. 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

1. 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.

2. The client solves A1, and sends the A2 to the server.

3. The server solves the computationally-complex problem A2, and returns the result of the calculation to the client.

4. 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 $y = x^d \mathrm{mod}\, n$, 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 primes 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 $\mathrm{Comp}_C\,(\beta) = \infty$.

2 Methods for solving abstract equations with secret parameters using external computer

2.1 Methods for solving abstract equations with secret parameters

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

$$B:\ M \rightarrow\ M.$$

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.

Consider the problem

$$Bx = b, \tag{1}$$

where $x \in\ M,\ b \in M$.

Suppose that problem (1) is uniquely solvable.

Let the client needs to approximately solve a computationally-complex equation (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 equation (1) only at the ideological level, since the considered equation (1), generally speaking, is abstract.

Task Z_2: The client needs to approximately solve equation (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 equation (1) remain a secret.

Protocol Z_2

1. 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.

2. The server solves the equation Gy = b and returns to the client an approximate solution y.

3. The client finds an approximate solution to equation (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 $\mathrm{Comp}_C\ (x : Bx = b)$ we denote the time (which can be equal to ∞) required for the client to solve equation (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;
- 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 equation (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 equation (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 equation (1); therefore, the assumption $T(Z_2) < \mathrm{Comp}_C(x{:}\, Bx = b)$, which determines the correctness of the protocol, is justified.

Task Z_3: The client needs to approximately solve equation (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 equation (1) remain a secret.

Protocol Z_3

1. The client randomly finds bijective operators D, K: $M \to M$. Next, calculates $KBD \equiv G$, $Kb \equiv g$ 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.

2. The server solves the equation $Gy = g$ and returns an approximate solution y to the client.

3. The client finds an approximate solution to equation (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 $\mathrm{Comp}_C\ (x : Bx = b)$ we denote the time (which can be equal to ∞) required for the client to solve equation (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;
- 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 .

1) 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: $\rho\ (Gy, g) < \varepsilon$.

2) 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 equation (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 equation (1); therefore, the assumption $T(Z_3) < \mathrm{Comp}_C(x: Bx = b)$, which determines the correctness of the protocol, is justified.

In protocols Z_2 and Z_3, it was assumed that in equation (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 equation (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;
- 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 Eq. (1) remains a secret.

2.2 Methods for solving linear equations with secret parameters

Consider the system of algebraic linear equations

$$\mathrm{Bx} = \mathrm{b} \qquad (2)$$

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.

Protocol Z_4

1. The client takes an n-dimensional vector at random

$\mathrm{w} = (\mathrm{w}[0],\ldots \mathrm{w}[\mathrm{n}-1])$ and calculates $\mathrm{b} - \mathrm{Bw} = \mathrm{g}$ 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 $\mathrm{By} = \mathrm{g}$, and keeps the vector w as secret.

2. The server solves the equation

$$\mathrm{By} = \mathrm{g}$$

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

3. The client finds a solution to equation (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 equations (2) without the help of the server.

Acknowledgements

The authors would like to thank to the Ministry of Digital Development, Innovation and Aerospace, Republic of Kazakhstan, grant No AP06850817.

References

1. Seitkulov Ye. New methods of secure outsourcing of scientific computations // The Journal of Supercomputing, Springer US, Print ISSN 0920-8542. -2013. -Vol. 65, Issue 1. -P.469-482.
2. Jianhua Yu, Xueli Wang, Wei Gao Improvement and applications of secure outsourcing of scientific computations // Journal of Ambient Intelligence and Humanized Computing. -2015. -Vol. 6, Issue 6. -P.763–772.
3. Xing Hu, Chunming Tang Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix / Journal of Cloud Computing, Springer Berlin Heidelberg, 4:7 DOI 10.1186/s13677-015-0033-9, 2015, ISSN 2192-113X. - URL: https://eprint.iacr.org/2014/442.pdf.
4. Cong Wang, Kui Ren, Jia Wang Secure Optimization Computation Outsourcing in Cloud Computing: A Case Study of Linear Programming // IEEE Transactions on Computers. -2016. –Vol. 65, Issue 1. -P.216-229.
5. Vyas R., Singh A., Singh J., Soni G., Purushothama B.R.: Design of an efficient verification scheme for correctness of outsourced computations in cloud computing // Security in Computing and Communications, Springer, 2015. -Vol. 536. -P.66–77.
6. Atallah M., Frikken K. Securely outsourcing linear algebra computations // In: Proceedings of ASIACCS. New York. -2010. -P.48-59.
7. Benjamin D., Atallah M. Private and cheating-free outsourcing of algebraic computations // Proceedings of 6th conference on privacy, security, and trust (PST). -2008. -P.240-245.
8. Tsutomu Matsumoto, Koki Kato, Hideki Imai Speeding Up Secret Computations with Insecure Auxiliary Devices // CRYPTO 1988: Advances in Cryptology — CRYPTO' 88. -1998. -P.497-506.