Secure Computation Outsourcing for Inversion in Finite Field

Qianqian Su[1,2], Rong Hao[1*], Shaoxia Duan[1], Fanyu Kong[3], Xiaodong Liu[3], and Jia Yu[1]
[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
hr@qdu.edu.cn, shaoxiaduan@163.com, qduyujia@gmail.com
[2]State Key Laboratory of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, Beijing 100093, China
suqianqian@iie.ac.cn
[3]School of Software, Shandong University, Jinan 250101, China
{fanyukong, liuxiaodong}@sdu.edu.cn

Abstract

With the widespread of cloud service and the popularity of mobile devices, more and more researchers are working on the technologies that can securely outsource expensive computation tasks to a single semi-trusted cloud server. Inversion has always been considered one of the most basic and prohibitively expensive arithmetic operations in cryptographic system based on finite field or elliptic curve. In this paper, we construct two algorithms for secure outsourcing inversion operation with the help of single and untrusted cloud server. The first algorithm, named **Inv**, is designed for secure outsourcing single element's inversion. The second algorithm, named **MInv** is designed for secure outsourcing multiple elements' inversion. Compared with using the extended Euclidean algorithm, the client can achieve higher efficacy by using the first proposed algorithm Inv. The efficiency of the second proposed algorithm on the cloud side is superior to invoking the first algorithm multiple times. At the same time, there is no increase in computational burden on the client side. The formal security analysis shows that our algorithms satisfy the security of verifiability, input-privacy, and output-privacy. Furthermore, we simulate our proposed algorithms to evaluate its performance. The simulation results demonstrate that our proposed algorithms are valid and practical.

Keywords: Secure computation outsourcing, Inversion, Verifiability, Privacy

1 Introduction

With the emergence and the widely used of cloud computing, it becomes feasible to access almost limitless computing resources according to people's demand. Computation outsourcing, as one of the fundamental advantages of cloud computing, makes cloud customers with expensive computation tasks get rid of the limitation of their resource-constraint devices. At the same time, computation outsourcing gets more widespread because of the popularity of mobile devices. Using this model, users constrained by mobile or device with limited resources can complete some resource consumption computations, such as modular exponentiation [1] and linear programming [20]. By outsourcing the computation tasks, companies and individuals can avoid the cost of deploying and maintaining hardware or software for they can pay to the cloud service provider according to the amount of resources used.

Although computation outsourcing has tremendous advantages, outsourcing computation tasks to the commercial public cloud servers inevitably bring new security issues and challenges [8, 13, 25, 26]. Users no longer have direct control of their data during the computation. This makes the cloud servers not fully trustworthy [19, 24]. A secure computation outsourcing should meet the following three requirements. First, because the semi-trusted cloud servers may return incorrect results to the client, the clients

Journal of Internet Services and Information Security (JISIS), volume: 10, number: 2 (May 2020), pp. 35-48
DOI: *10.22667/JISIS.2020.05.31.035*
*Corresponding author: College of Computer Science and Technology, Qingdao University, Qingdao 266071, China

should be able to detect the dishonest behavior of the cloud servers. Second, the verification process on the client side should not contain any complicated computations due to the resource-limited devices may be incapable of accomplishing the complicated verification. At least, the amount of computation required by the client to perform the validation should be much less than that required to complete the original task. Otherwise, it would be meaningless to outsource the computation tasks. Third, the tasks may include some sensitive informations, such as the financial records, the secret key of the client and personally identifiable health information. These information should not be exposed to cloud servers. Computation outsourcing must guarantee that cloud servers should not learn any useful information.

Currently, researches on the computation outsourcing can be divided into two directions: the general outsourcing model [4, 10, 28] and the specific outsourcing model [3, 2, 12, 23, 29]. For generic outsourcing model, the first formal definition of the concept of verifiable computation outsourcing is given by Gennaro et al, and a verifiable computation outsourcing scheme for arbitrary functions is constructed in [10]; An outsourcing algorithm for inverting homomorphic functions based on computation disequilibrium was proposed by Zhang et al. in [28]; In [4], an improved scheme without any garbled circuit was proposed for generic computation outsourcing by Chung et al. However, to verify the result, the users are asked to make some pre-computation in [4]. For specific outsourcing model, The first secure outsourcing algorithm for modular exponentiation was proposed by Hohenberger and Lysyanskaya in [12] based on two untrusted cloud servers, and further study was made by Chen et al in [1], and a more efficient algorithm proposed in [3, 2, 23, 29]; An efficient scheme for secure outsourcing the bilinear pairings was proposed by Chen et al. in [18], and a more efficient algorithm was proposed in [21]; Based on the interactive methods, a secure outsourcing mechanism was constructed in [14] by Wang et al. for secure outsourcing large-scale systems of linear equations. For secure outsource the inversion modulo a large composite number, Su et al. proposed an efficient scheme in [16].

Inversion as one of the most basic scientific computation is widely used in many public-key cryptosystems [5, 6, 9] such as the Elliptic Curves Cryptosystems (ECC). ECC [24] uses inversion in its fundamental operations, such as point addition and scalar multiplication. With the increasing applications of cryptosystems based on ECC [15, 27, 17] and bilinear pairings [7, 11, 22], inversion in finite filed plays more and more predominant role in both theory and applications. However, inversion as a time-consuming operation may not be performed on some resource-constraint devices. Therefore, it is vital to outsource the inversion operation to the cloud server. We focus on secure outsourcing the inversion in finite field in this paper.

Our Contribution. We construct an algorithm for secure outsource single element's inversion with the aid of a semi-trusted cloud server. The proposed algorithm meets all requirements stated before. A client can effectively outsource the single element's inversion with privacy and verifiability. Also, we construct an algorithm called for secure outsource multiple elements' inversion. This algorithm can obtain multiple elements' inverses through one-time outsourcing. Compared with invoking the first proposed algorithm multiple times, the computational overhead on the cloud side can be reduced to by using the multiple inversion algorithm (if the client wants to calculate inverses) while the additional computational overhead on the client side does not need. It means the client only needs to pay less money to the cloud service provider if it adopts the second outsourcing scheme. By giving the security analysis, we prove the security of our algorithms. Finally, we conduct several simulation experiments, and the experimental results demonstrate the effectiveness of our proposed algorithm.

Organization. The rest of this paper is organized as follows. The formal definition and security model of secure computation outsourcing are presented in Section 2. In Section 3, we give a computation outsourcing algorithm for single element's inversion and we also construct a secure outsourcing algorithm for multiple elements' inversion. The security proof and efficiency comparisons are presented in Section 4. Finally, we give a conclusion in Section 5.

2 Formal Definition and Security Model

2.1 Formal Definition of Computation Outsourcing

A computation outsourcing model contains two participants: cloud and client. The client is resource-constrained and tries to outsource a computation task $f(x)$ to an untrusted cloud, where the computation task f takes x as the input. In order to guarantee the privacy of x, the client encodes x into σ_x. Then the client sends σ_x and the description of f to the cloud. The cloud computes σ_y after received f and σ_x. Next, the cloud sends σ_y back to the client as the result, and the client verifies the correctness of σ_y. If it is correct, the client accepts it; otherwise, the client refuses it. Finally, the client obtains $f(x)$ from σ_y. The concrete procedure is described in Fig. 1.

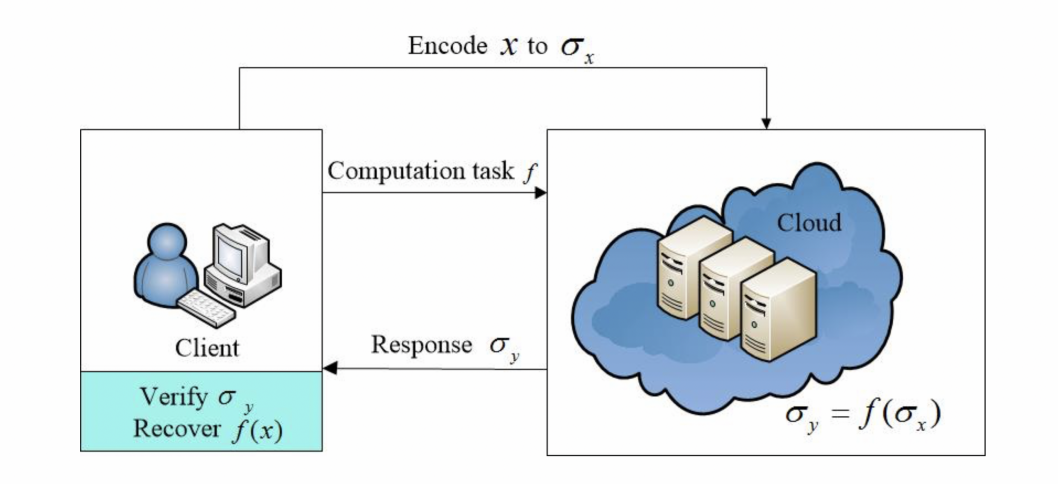


Figure 1: The Computation Outsourcing Model

A secure computation outsourcing scheme, which denoted by (f,x) , contains the following probabilistic polynomial-time (PPT) algorithms.

- $KeyGen(1^k) \rightarrow \tau$. Takes the security parameter k as input, outputs the random value τ.
- $Encrypt(x,\tau) \rightarrow \sigma_x$. Takes the computation task x and the random value τ as inputs, outputs σ_x.
- $Compute(f,\sigma_x) \rightarrow \sigma_y$. Takes the computation task f and σ_x as inputs, outputs σ_y.
- $Verify(\sigma_x,\sigma_y,\tau) \rightarrow \{y,\perp\}$. Takes σ_x, σ_y, and the random number τ as inputs, outputs y indicting the result is correct, or $\perp$ indicting the result is incorrect.

2.2 Security Model

There are two requirements for an (f,x)-computation outsourcing algorithm: verifiability and privacy. Verifiability implies that the client should have the ability to detect whether the result from the cloud is correct or not. It means that the cloud could not fool the client by returning a corrupted $\hat{\sigma}_y$. From the point of mathematics, if $\hat{\sigma}_y = \sigma_y$, we say that the probability of $Verify(\sigma_x,\sigma_y,\tau) = y$ is negligible. Privacy includes the input privacy and the output privacy. The privacy requirement can guarantee that the cloud could not get any useful information about the input x and the output $f(x)$. Firstly, we formalize the concept of verifiability with the following experiments.

Experiment $Exp_{\mathscr{A}}^{verify}[f,k]$

Query and response:

$x_0 = \sigma_{x_0} = \beta_0 = \perp$

For $i = 1$ to l
- $x_i \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$;
- $\tau_i \leftarrow KeyGen(1^k); \sigma_{x_i} \leftarrow Encrypt(\tau_i, x_i)$;
- $\sigma_{y_i} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1}, \sigma_{x_i})$;
- $\beta_i = Vefiry(\sigma_{x_i}, \sigma_{y_i}, \tau_i)$.

Challenge:
- $x \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l)$;
- $\tau \leftarrow KeyGen(1^k); \sigma_x \leftarrow Encrypt(\tau, x)$;
- $\sigma_y \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l, \sigma_x)$;
- $\hat{y} = Vefiry(\sigma_x, \sigma_y, \tau)$.

If $\hat{y} \neq f(x)$ and $\hat{y} \neq \perp$, output 1; else output 0.

In the query and the response phase, the untrusted cloud is given access to the three oracles *KeyGen*, *Encrypt*, and $Verify$. The adversary succeeds if the result $\hat{f(x)}$ accepted by the client. However, it fails. The (f,x)-computation outsourcing algorithm is verifiable only if the probability that any adversary succeeds in this experiment is negligible. We give a more precise definition as follows.

Definition 1. (Verifiability) The (f,x)-computation outsourcing algorithm is verifiable if the success probability of any polynomial-time adversary $\mathscr{A}$ in the above experiment $Exp_{\mathscr{A}}^{verify}[f,k]$ is negligible. In other words, $\Pr[Exp_{\mathscr{A}}^{verify}[f,k] = 1]$ is negligible.

According to the typical indistinguishability argument, the privacy of input and output can be defined. Input privacy and output privacy requires that no useful knowledge about the input and the output are disclosed. Briefly speaking, when the encryption algorithm *Encrypt* inputs two different values, the output of the encryption algorithm Encrypt are indistinguishable for the cloud, we believe that the computation outsourcing algorithm can guarantee the privacy of the input. Next, we formalize the concept of input privacy with the following experiments.

Experiment $Exp_{\mathscr{A}}^{Inprivacy}[f,k]$

Query and response:

$x_0 = \sigma_{x_0} = \beta_0 = \perp$

For $i = 1$ to l
- $x_i \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}})$;
- $\tau_i \leftarrow KeyGen(1^k); \sigma_{x_i} \leftarrow Encrypt(\tau_i, x_i)$.

Challenge:
- $(x^{(0)}, x^{(1)}) \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l})$;
- $b \leftarrow_R \{0,1\}, \tau^b \leftarrow KeyGen(1^k), \sigma_{x^{(b)}} \leftarrow Encrypt(\tau^{(b)}, x^{(b)})$;
- $\hat{b} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l, \sigma_{x^{(b)}})$.

If $\hat{b} = b$, output 1; else output 0;

In the experiment $Exp_{\mathscr{A}}^{Inprivacy}[f,k]$, the *KeyGen* and *Encrypt* oracles are available to the adversary $\mathscr{A}$. In the challenge phase, if the adversary $\mathscr{A}$ can distinguish the outputs of the encryption algorithm, i.e., $\hat{b} = b$. we say that the adversary succeeds. A more precise definition is given as follows.

Definition 2. (Input privacy) The privacy of input is achieved in (f,x)-computation outsourcing algorithm if the advantage of any PPT adversary $\mathscr{A}$ success in the experiment $Exp_A^{Inprivacy}[f,k]$ is negligible. That is, $\Pr[Exp_{\mathscr{A}}^{Inprivacy}[f,k] = 1] - 1/2$ is negligible for any PPT adversary $\mathscr{A}$.

The privacy of output can be defined similarly to input privacy. The following experiment formalizes the definition of output privacy.

Experiment $Exp_{\mathscr{A}}^{Outprivacy}[f,k]$

Query and response:

$x_0 = \sigma_{x_0} = \beta_0 = \perp$

For $i = 1$ to l

- $x_i \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$;
- $\tau_i \leftarrow KeyGen(1^k); \sigma_{x_i} \leftarrow Encrypt(\tau_i, x_i)$;
- $\sigma_{y_i} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1}, \sigma_{x_i})$;
- $\beta_i = Verify(\sigma_{x_i}, \sigma_{y_i}, \tau_i)$.

Challenge:

- Choose $x^{(0)}, x^{(1)} \in \chi$
- $b \leftarrow_R \{0,1\}$;
- $\sigma_{x^{(b)}} \leftarrow Encrypt(\tau^{(b)}, x^{(b)})$, $\sigma_{y^{(b)}} \leftarrow Compute(f, \sigma_{x^{(b)}})$;
- $\hat{b} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l)$.

If $\hat{b} = b$, output 1; else output 0;

In query phase, the adversary is given the oracle access to $Encrypt$ and $Verify$. The adversary succeeds if he/she can distinguish the outputs of $Verify$ in the challenge phase.

Definition 3. (Output privacy) The privacy of input is achieved in (f,x)-computation outsourcing algorithm if the advantage of any PPT adversary $\mathscr{A}$ success in the experiment $Exp_{\mathscr{A}}^{Out\,privacy}[f,k]$ is negligible. More intuitive, $\Pr[Exp_{\mathscr{A}}^{Out\,privacy}[f,k] = 1] - 1/2$ is negligible for any PPT adversary.

3 Secure Computation Outsourcing for Inversion in Finite Field

In this section, we describe the proposed algorithms. The first is a secure outsourcing algorithm for single element's inversion which sends one request to the cloud to obtain an inverse of the specified element. The second algorithm is for secure outsourcing multiple elements' inversion which can obtain the batch of inverses through one request.

3.1 Secure Outsourcing Algorithm for Single Element's Inversion

3.1.1 Outsourcing Algorithm

The secure outsourcing algorithm called **Inv** is proposed for a single element's inverse with the aid of a single and untrusted cloud server. The privacy requirement for algorithm **Inv** is that the cloud server could not obtain any useful information about the element and its inversion. Let G is a multiplicative cyclic group with order p and x is an element in G, where p is a prime number. The algorithm **Inv** takes p and x as inputs. The algorithm **Inv** outputs y, which meets the equation $xy \equiv 1 \bmod p$. The algorithm **Inv** is constructed based on the extended Euclidean algorithm [11], which can be shown in Algorithm 1.

3.1.2 Example

Let $(Z_p^*, \cdot)$ is a multiplicative group of modulus p, where p is a prime number. In this example, we set $p = 79$ and $x = 29$. We want to calculate $y = 29^{-1} \bmod 79$. We invoke the extended Euclidean algorithm and the proposed algorithm **Inv** to calculate y respectively. The concrete procedures of using the extended Euclidean algorithm and using the proposed algorithm **Inv** are described in Table 1 and Table 2.

Using the extended Euclidean algorithm, the client needs to perform 12 times multiplications and 6 times divisions locally. By using the proposed algorithm **Inv**, the computational cost on the client side can be reduced for the client only needs to execute are 3 times multiplications in step 2, 7 and 8. It is easily observed that the proposed algorithm **Inv** reduces the computation cost of the client from 12 times multiplications and 6 times divisions to 3 times multiplications.

Algorithm 1 Secure Outsourcing Algorithm Inv for Single Element Inversion

Input: Prime p; Element $x \in G$
Output: $y = x^{-1} \bmod p$

1: Client randomly chooses an element $\tau \leftarrow_R G$.
2: Client uses the random element τ to encrypt input x: $\sigma_x \leftarrow x \times \tau \bmod p$.
3: Client sends the prime p and the encoded σ_x to cloud.
4: Cloud initializes the parameters: $v \leftarrow p$, $u \leftarrow \sigma_x$, $x_1 \leftarrow 1$, $x_2 \leftarrow 0$.
5: Cloud does as follows.
6: **while** $u \neq q$ **do**
7: $\quad q \leftarrow \lfloor v/u \rfloor$;
8: $\quad r \leftarrow v - qu$, $x_3 \leftarrow x_2 - qx_1$;
9: $\quad v \leftarrow u$, $u \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x_3$;
10: Cloud returns $\sigma_y = x_1 \bmod p$ as the calculation result back to the client.
11: Client verifies the validity: it computes $\beta \leftarrow \sigma_x \times \sigma_y \bmod p$.
12: If $\beta = 1$, **goto** Step 13; Else return $\perp$.
13: Client recovers result: $y \leftarrow \sigma_y \times \tau \bmod p$.
14: **return** (y)

Table 1: Calculate y using the extended Euclidean algorithm

1	$q = \lfloor 79/29 \rfloor = 2$, $r = 79 - 2 \times 29 = 21$, $x_3 = 0 - 2 \times 1 = -2$;
2	$q = \lfloor 29/21 \rfloor = 1$, $r = 29 - 1 \times 21 = 8$, $x_3 = 1 - 1 \times (-2) = 1 + 2 = 3$
3	$q = \lfloor 21/8 \rfloor = 2$, $r = 21 - 2 \times 8 = 5$, $x_3 = -2 - 2 \times 3 = -8$
4	$q = \lfloor 8/5 \rfloor = 1$, $r = 8 - 1 \times 5 = 3$, $x_3 = 3 - 1 \times (-8) = 11$
5	$q = \lfloor 5/3 \rfloor = 1$, $r = 5 - 1 \times 3 = 2$, $x_3 = -8 - 1 \times 11 = -19$
6	$q = \lfloor 3/2 \rfloor = 1$, $r = 3 - 1 \times 2 = 1$, $x_3 = 11 - 1 \times (-19) = 30$
7	$29^{-1} \bmod 79 = 30$

3.2 Secure Outsourcing Algorithm for Multiple Elements' Inversion

3.2.1 Outsourcing Algorithm

The secure outsourcing algorithm **MInv** is designed for multiple elements' inversion with the aid of a single and untrusted cloud server. Based on $x^{-1} = y \cdot (xy)^{-1}$ and $y^{-1} = x \cdot (xy)^{-1}$ hold, we construct the proposed algorithm **MInv**.

Let G is a multiplicative cyclic group with order p and $x_1, x_2, ..., x_m$ are nonzero elements in G, where p is a prime number. The inputs of algorithm **MInv** are $x_1, x_2, ..., x_m$ and p. The outputs of algorithm **MInv** are $x_1^{-1}, x_2^{-1}, ..., x_m^{-1}$, such that $x_i x_i^{-1} \equiv 1 \bmod p$, where $i = 1, 2, ..., m$. The proposed algorithm **MInv** can described as shown in Algorithm 2.

3.2.2 Example

Let $(Z_p^*, \cdot)$ is a multiplicative group of modulus p, where p is a prime number. In this example, we set $x_1 = 4$, $x_2 = 9$, $x_3 = 14$, $x_4 = 15$, $x_5 = 18$ and $p = 79$. We want to calculate the inverse of 4, 9, 14, 15 and 18. i.e., $x_1^{-1} = 4 \bmod 79$, $x_2^{-1} = 9 \bmod 79$, $x_3^{-1} = 14 \bmod 79$, $x_4^{-1} = 15 \bmod 79$ and $x_5^{-1} = 18 \bmod 79$. The concrete procedures of using the proposed algorithm **MInv** are described in Table 3.

Using the extended Euclidean algorithm, 30 times multiplications and 15 times divisions are needed on the cloud side, and 15 times multiplications are needed on the client side. In contrast, by using the

Table 2: Calculate y using the proposed algorithm **Inv**

1	$\tau = KeyGen(Z_{79}^*) = 3$; $\sigma_x = 3 \times 29 \pmod{79} = 8$;
2	$\sigma_y = 8^{-1} \pmod{79} = 10$
3	$\sigma_x \times \sigma_y \pmod{79} = 8 \times 10 \pmod{79} = 1$
4	$y = 3 \times 10 \pmod{79} = 30$

Algorithm 2 Secure Outsourcing Algorithm for Multiple Elements' Inversion

Input: Prime p; Nonzero element $x_1, x_2, ..., x_m \in G$
Output: $x_1^{-1}, x_2^{-1}, ..., x_m^{-1}$, where $x_i x_i^{-1} \equiv 1 \bmod p$

1: Client randomly chooses an element $\tau \leftarrow_R G$.
2: Client does: $c_1 \leftarrow x_1 \times \tau \pmod{p}$.
3: **for** $i = 2 \rightarrow m$ **do**
4: $\quad c_i \leftarrow c_{i-1} \times x_i \pmod{p}$.
5: Client does: $\sigma_x \leftarrow c_m$
6: Client sends the prime p and the encoded σ_x to cloud.
7: Cloud initializes the parameters: $u \leftarrow \sigma_x$, $v \leftarrow p$, $z_1 \leftarrow 1$, $z_2 \leftarrow 0$.
8: Cloud does as follows.
9: **while** $u \neq 1$ **do**
10: $\quad q \leftarrow \lfloor v/u \rfloor$;
11: $\quad r \leftarrow v - q \times u$, $z_3 \leftarrow z_2 - q \times z_1$;
12: $\quad v \leftarrow u$, $u \leftarrow r$, $z_2 \leftarrow z_1$, $z_1 \leftarrow z_3$;
13: Cloud returns $\sigma_y = z_1 \mod p$ as the result back to the client.
14: Client does: $\beta \leftarrow \sigma_x \times \sigma_y \pmod{p}$;
15: Client verifies: If $\beta = 1$, **goto** Step 16; Else return $\perp$.
16: Set $w \leftarrow \sigma_y$.
17: Client recovers result as follows.
18: **for** $i = m \rightarrow 2$ **do**
19: $\quad x_i^{-1} \leftarrow c_{i-1} \times w \pmod{p}$.
20: $\quad w \leftarrow w \times x_i \pmod{p}$.
21: $x_1^{-1} \leftarrow w \times \tau \pmod{p}$.
22: **return** $(x_1^{-1}, x_2^{-1}, ..., x_m^{-1})$

algorithm **MInv**, on the one hand, the cloud needs to perform 8 times multiplications and 4 times divisions. On the other hand, the client needs to perform 15 times multiplications. Clearly, the computation overhead on the cloud side is significantly reduced. Meanwhile, the overhead on the client side does not increase. The more the inverses are calculated, the more the calculation can be reduced on the cloud side.

4 Security Proofs and Efficiency Comparisons

4.1 Security Proofs

Although there are some differences between the algorithm **Inv** and the algorithm **MInv**, in terms of security, no distinct difference can be aware of. we security. Here, we mainly focus on proving the first proposed algorithm **Inv** meets the security requirements. Similarly, the security of the second proposed algorithm **MInv** can be proved. In **Inv**, *KeyGen* includes step 1, *Encrypt* includes step 2, and $Verify$ includes steps 7,8.

Table 3: Calculate $x_1^{-1}, x_2^{-1}, x_3^{-1}, x_4^{-1}, x_5^{-1}$, using the proposed algorithm **MInv**

1	$\tau = KeyGen(Z_{79}^*) = 3$; $c_1 = \tau \times x_1 = 3 \times 4 \pmod{79} = 12$; $c_2 = c_1 \times x_2 = 12 \pmod{79} = 29$; $c_3 = c_2 \times x_3 = 29 \times 14 \pmod{79} = 11$; $c_4 = c_3 \times x_4 = 11 \times 15 \pmod{79} = 7$; $c_5 = c_4 \times x_5 = 7 \times 18 \pmod{79} = 47$; $\sigma_x = 47$;
2	$q = \lfloor 79/47 \rfloor = 1$, $r = 79 - 1 \times 47 = 32$, $x_3 = 0 - 1 \times 1 = -1$; $q = \lfloor 47/32 \rfloor = 1$, $r = 47 - 1 \times 32 = 15$, $x_3 = 1 - 1 \times (-1) = 2$; $q = \lfloor 32/15 \rfloor = 2$, $r = 32 - 2 \times 15 = 2$, $x_3 = -1 - 2 \times 2 = -5$; $q = \lfloor 15/2 \rfloor = 7$, $r = 15 - 7 \times 2 = 1$, $x_3 = 2 - 7 \times (-5) = 37$; $\sigma_y = 37 \pmod{79} = 37$;
3	$\sigma_x \times \sigma_y \pmod{79} = 37 \times 47 \pmod{79} = 1$
4	$y = 3 \times 10 \pmod{79} = 30$

Theorem 1. *According to Definition 1, our proposed algorithm **Inv** is verifiable.*

Proof. To prove the proposed algorithm **Inv** is verifiable, we will prove that the single untrusted cloud cannot cheat the client by any incorrect results. According to the experiment $Exp_{\mathscr{A}}^{verify}[f,k]$ which is defined in Section 2, our proof can be described as following.

Query and response:

For $i = 1$ to l

-$x_i \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$.

-$\tau_i \leftarrow KeyGen(1^k)$; $\sigma_{x_i} \leftarrow Encrypt(\tau_i, x_i)$, where $\sigma_{x_i} = \tau_i \times x_i \mod p$ and $\tau_i \leftarrow_R G$.

- $\sigma_{y_i} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1}, \sigma_{x_i})$.

- $\beta_i = Vefiry(\sigma_{x_i}, \sigma_{y_i}, \tau_i)$.

Challenge:

-$x \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l)$.

-$\tau \leftarrow KeyGen(1^k)$; $\sigma_x \leftarrow Encrypt(\tau, x)$, where $\sigma_x = \tau \times x \mod p$ and $\tau \leftarrow_R G$.

- $\sigma_y \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l, \sigma_x)$.

- $\hat{y} = Vefiry(\sigma_x, \sigma_y, \tau)$.

Set $\sigma_y = f(\sigma_x)$, and σ_y is an element in G for $\sigma_x \in G$. According to the challenge phase, we can get:

- If $\hat{\sigma}_y \notin G$, then $Verify(\sigma_x, \hat{\sigma}_y, \tau_i)$ will output $\perp$.

- If $\hat{\sigma}_y \times \sigma_x \neq 1 \bmod p$, then $Verify(\sigma_x, \hat{\sigma}_y, \tau_i)$ will output $\perp$.

- If $\hat{\sigma}_y \times \sigma_x \equiv 1 \bmod p$, then $Verify(\sigma_x, \hat{\sigma}_y, \tau_i)$ will output 1. That is to say, $\hat{\sigma}_y$ is the expected computation result. That is because of the following hold. Let $\hat{\sigma}_y \equiv \sigma_y \times \alpha \bmod p$, α be an element in G. We can get:

$$\begin{aligned} 1 \equiv & \hat{\sigma}_y \times \sigma_x \mod p \\ \equiv & \sigma_y \times \alpha \times \sigma_x = \sigma_y \times \sigma_x \times \alpha \mod p \\ \equiv & 1 \times \alpha \equiv \alpha \mod p. \end{aligned}$$

Consequently, $\alpha \equiv 1 \mod p$, α is an element in G, $\alpha = 1$, thus $\hat{\alpha}_y = \alpha_y$. Therefore, for every query from the adversary, $\Pr[Exp_{\mathscr{A}}^{Outprivacy}[f,k] = 1] = 0$ always holds. □

Theorem 2. *According to Definition 2 and Definition 3 in Section 2, our proposed algorithm **Inv** can achieve the privacy of input and output.*

Proof. To prove the proposed algorithm **Inv** is input-privacy, we will prove $\Pr[Exp_{\mathscr{A}}^{Inprivacy}[f,k]] - 1/2$, which is defined in $Definition2$, is negligible. Our proof can be described as following.

Query and response:
For $i = 1$ to l
-$x_i \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_{i-1}, \sigma_{x_{i-1}})$.
-$\tau_i \leftarrow KeyGen(1^k); \sigma_{x_i} \leftarrow Encrypt(\tau_i, x_i)$, where $\sigma_{x_i} = \tau_i \times x_i \bmod p$ and $\tau_i \leftarrow_R G$.

In the query and response phase, σ_{x_i} can be considered as a ciphertext encrypted with τ_i as the random encrypt key. Since τ_i is randomly chosen and uniformly distributed in G, $\sigma_{x_i} = \tau_i \times x_i \pmod p$ is one-time pad encryption with advantage in perfect privacy. Therefore, no useful knowledge of the input and the output will be leaked to the adversary during the experiment $Exp_{\mathscr{A}}^{Inprivacy}[f,k]$, except the information that the adversary knows beforehand.

Challenge:
-$(x^{(0)}, x^{(1)}) \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l})$.
-$b \leftarrow_R \{0,1\}; \tau^b \leftarrow KeyGen(1^k); \sigma_{x^{(b)}} \leftarrow Encrypt(\tau^{(b)}, x^{(b)})$.
- $\hat{b} \leftarrow \mathscr{A}(x_0, \sigma_{x_0}, \beta_0, ..., x_l, \sigma_{x_l}, \beta_l, \sigma_{x^{(b)}})$.

Similarly, $\sigma_{x^{(b)}} \equiv \tau \times x^{(b)} \pmod p$ can also be considered as one-time pad encryption where the random encrypt key is τ_i, which is randomly chosen and uniformly distributed in G. In the challenge phase, what the only thing that the adversary can do is randomly guess a bit $\hat{b}$, and the probability of success for the adversary is 1/2. Therefore, the advantage of the adversary $\mathscr{A}$ success in the experiment $Exp_{\mathscr{A}}^{Inprivacy}[f,k]$ is negligible. i.e., $\Pr[Exp_{\mathscr{A}}^{Inprivacy}[f,k]] = 1/2$.

Similar to proving the input-privacy, the output-privacy can be proven. Given the perfect privacy advantage of one-time pad encryption, the queries in experiment $Exp_{\mathscr{A}}^{Outprivacy}[f,k]$ does not leak any useful information to the adversary. In the challenge phase, send $(f(x^{(0)}), f(x^{(1)}), \sigma_{x^{(b)}}, \sigma_{y^{(b)}})$ as a challenge to the adversary $\mathscr{A}$, and ask the adversary A to output $\hat{b}$. On the one hand, because $\sigma_x^{(b)} \bmod p$ is a one-time pad encryption and τ is randomly choose and uniformly distributed in G, $x^{(b)}$ is completely invisible to the adversary. On the other hand, because of $\sigma_{y^{(b)}} = f(\tau^{(b)} \times x^{(b)} (\bmod p)) = (\tau^{(b)})^{-1} \times (x^{(b)})^{-1} (\bmod p)$ holds, we can get $(x^{(b)})^{-1} = \sigma_{y^b} \times (\tau^{(b)}) \pmod p$, and $\Pr[f(x^{(0)} = \sigma_{y^{(b)}} \times \tau^{(b)}] = 1/2$.

Therefore, the probability to guess the correct bit $\hat{b}$ for the adversary $\mathscr{A}$ is 1/2.

□

Similar to proving Theorem 1 and Theorem 2, we can prove Theorem 3 and Theorem 4.

Theorem 3. *Our proposed algorithm **MInv** is verifiable according to Definition 1.*

Theorem 4. *According to Definition 2 and Definition 3 in Section 2, our proposed algorithm **MInv** can achieve the privacy of input and output.*

4.2 Efficiency analysis

We simulate the proposed algorithm **Inv** and **MInv**. For convenience, we use *Algorithm 0* to represent the extended Euclidean algorithm, use M to represent the multiplication operation, and use D to represent the division operation. We now give the analysis of the proposed algorithms. To prove the efficiency of the first proposed algorithm **Inv**, we compare the computation cost on the client side between algorithm **Inv** and *Algorithm 0*. In each cycle of the *Algorithm 0*, 1 division and 2 times multiplications are needed to conduct on the client side. The loop of divisions in *Algorithm 0* is $log(p)$ in the worst case, where p is the modulus. Therefore, using *Algorithm 0*, the client is required to execute $2log(p)$ times multiplications and $log(p)$ times divisions in the worst case. By contrast, using the proposed algorithm **Inv**, in any case, the client only needs to perform 3 multiplications. In Table 4, we compare the number of multiplications and divisions involved in *Algorithm 0* with those in algorithm **Inv**.

The second proposed algorithm **MInv** is used to calculate multiple elements' inversion. Trivially, we can invoke the first proposed algorithm **Inv** multiple times to calculate x_1^{-1}, x_2^{-1}, ... , x_m^{-1}. On the

Table 4: Comparison of *Algorithm 0* and algorithm **Inv**

Algorithm 0	$2log(p)M + log(p)D$;
Inv	$3M$

client side, the computational cost is $3m \cdot M$. And on the cloud side, the computation cost of m elements is $m \cdot 2log(p)M + m \cdot log(p)D$. With the proposed algorithm *MInv*, the computational overhead in the cloud is reduced to $2log(p)M + log(p)D$, while the client does not increase the additional computing burden. We show this comparison in Table 5.

Table 5: . The cost for calculating m elements' inversion in **Inv** and **MInv**

	On the Client side	On the cloud side
Inv	$3mM$	$m \cdot 2log(p)M + m \cdot log(p)D$;
MInv	$3mM$	$2log(p)M + m \cdot log(p)D$

Besides, we carry out simulation experiments for the three algorithms. The experiments are executed on Lenovo G450 PC with an Intel Pentium Dual-Core T4300 @ 2.10GHz processor and 4GB of RAM running Ubuntu version 14.04.

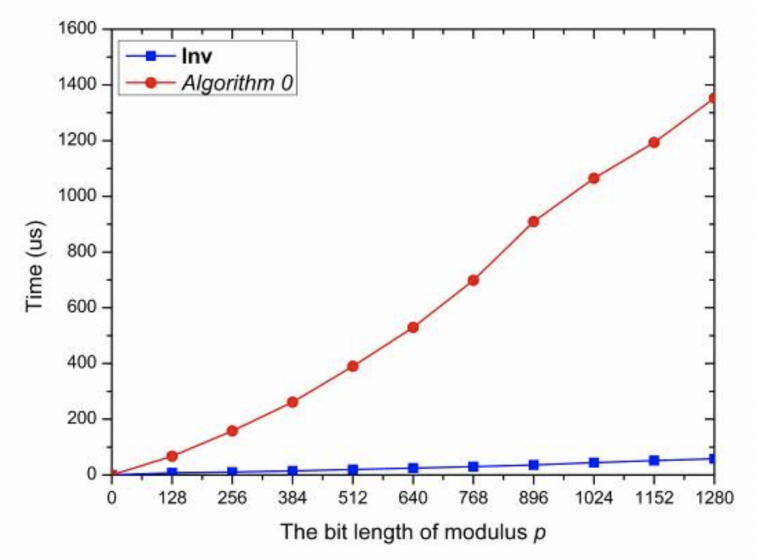


Figure 2: Client's running time of *Algorithm 0* and algorithm **Inv**

Firstly, we compare the running time of *Algorithm 0* and the proposed algorithm **Inv** on the client side. The bit length of p is set from 0 to 1280. From Fig. 2, we can see that the running time in *Algorithm 0* increases much more rapidly than that in the proposed algorithm **Inv**. We can observe this more clearly with the increasing of bit length.

In Fig. 3, we show the running time of the algorithm **Inv** and **MInv** on the client side and on the cloud side, respectively. The bit length of p is set from 0 to 1280 (here we set $m = 8$). The running time of cloud in algorithm **MInv** is 1/8 of that in algorithm **Inv**, while the running time of client in algorithm **MInv** doesn't increase compared with that in algorithm **Inv**.

If client wants to calculate m inverses, the client needs to send m outsourcing requests and the cloud needs to execute m times inverse operations using algorithm **Inv**. However, using algorithm **MInv**,

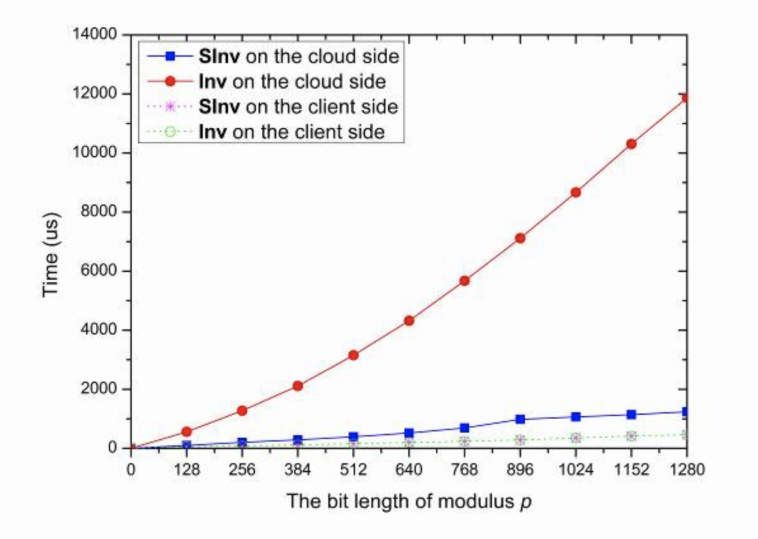


Figure 3: Cloud's running time of algorithm **Inv** and algorithm **MInv** for 8 elements' inversion

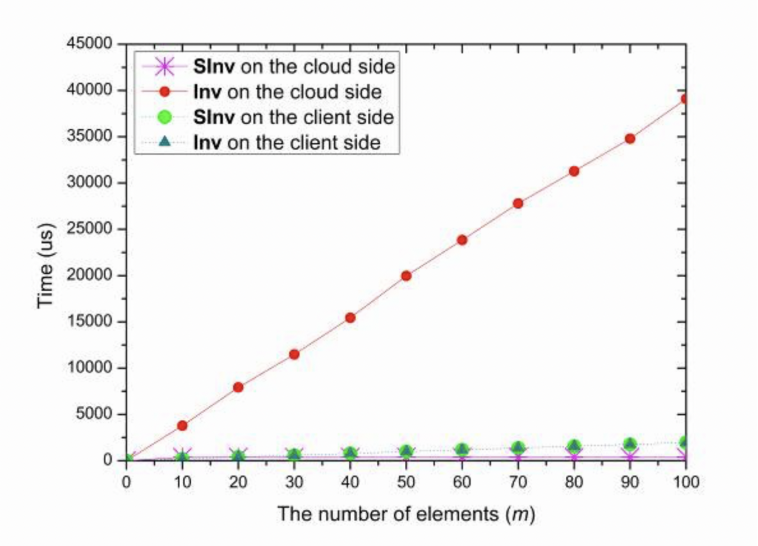


Figure 4: Cloud's running time of **Inv** and **MInv** for multiple elements

the client only needs to send one outsourcing request and the cloud needs to execute one time inverse computation. As the result, the cloud needs to handle one inversion request, in which case the client can get m inverses. That is, the running time of cloud in algorithm **MInv** is 1/8 of that in algorithm **Inv**. Also, the computation cost of client does not increase. We set the number of elements from 0 to 100 and show the experimental results in Fig. 4.

From above analysis, we can conclude that it would be much better to run algorithm **MInv** than to run algorithm **Inv** multiple times when calculate multiple elements' inversion.

5 Conclusion

In order to make public-key cryptography more accessible to resource-constrained devices, we mainly explore how to securely outsource the group inversion to a semi-trusted cloud server in this paper. We firstly propose an algorithm **Inv** for securely outsourcing a single element's inversion. Then, we propose an algorithm **MInv** for securely outsourcing multiple elements' inversion for the purpose of reducing traffic and economic cost. Based on the definitions of security, the formal security proofs of verifiability, input privacy, and output privacy are given. In the algorithm **Inv** and **MInv**, the client and the cloud only need to exchange one message with each other. The simulation results of the proposed algorithm **Inv** and **MInv** shows that our algorithms are efficient.

Acknowledgments

This research is supported by National Natural Science Foundation of China (61572267, 61602275), National Cryptography Development Fund of China (MMJJ20170118).

References

[1] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou. New algorithms for secure outsourcing of modular exponentiations. *IEEE Transactions on Parallel & Distributed Systems*, 25(9):2386–2396, September 2014.

[2] X. Chen, W. Susilo, J. Li, D. S. Wong, J. Ma, S. Tang, and Q. Tang. Efficient algorithms for secure outsourcing of bilinear pairings. *Theoretical Computer Science*, 562:112–121, January 2015.

[3] C. Chevalier, F. Laguillaumie, and D. Vergnaud. Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In *Proc. of the 21st European Symposium on Research in Computer Security (ESORICS'16), Heraklion, Greece*, volume 9878 of *Lecture Notes in Computer Science*, pages 261–278. Springer, Cham, September 2016.

[4] K.-M. Chung, Y. Kalai, and S. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proc. of the 30th Annual Cryptology Conference (CRYPTO'10), Santa Barbara, California, USA*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501. Springer, Berlin, Heidelberg, August 2010.

[5] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proc. of the 18th Annual International Cryptology Conference (CRYPTO'98), Santa Barbara, California, USA*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, Berlin, Heidelberg, August 1998.

[6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, December 1976.

[7] M. S. Farash, M. A. Attari, R. E. Atani, and M. Jami. A new efficient authenticated multiple-key exchange protocol from bilinear pairings. *Computers & Electrical Engineering*, 39(2):530–541, February 2013.

[8] A. Fu, Y. Li, S. Yu, Y. Yu, and G. Zhang. Dipor: An ida-based dynamic proof of retrievability scheme for cloud storage systems. *Journal of Network & Computer Applications*, 104(15):97–106, February 2017.

[9] T. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, January 1985.

[10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Proc. of the 30th Annual Cryptology Conference(CRYPTO'10), Santa Barbara, California, USA*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, Berlin, Heidelberg, Aug 2010.

[11] D. Hankerson, A. Menezes, and S. Springer. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, January 2004.

[12] S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In *Proc. of the 2nd Theory of Cryptography Conference (TCC'05), Cambridge, Maryland, USA*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282. Springer, Berlin, Heidelberg, February 2005.

[13] L. Huang, G. Zhang, and A. Fu. Privacy-preserving public auditing for non-manager group shared data. In *Proc. of the 2017 IEEE International Conference on Communications (ICC'17), Paris, France*, pages 1–6. IEEE, March 2017.

[14] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[15] Z. LAI, Z. ZHANG, and D. TAO. Algorithm for directly computing 7p elliptic curves and its application. *Journal of Computer Applications*, 33(7):1870–1874, October 2013.

[16] Q. Su, J. Yu, C. Tiana, H. Zhang, and R. Haoa. How to securely outsource the inversion modulo a large composite number. *Journal of Systems and Software*, 129:26–34, July 2017.

[17] H. Sun, X. Zheng, and Y. Yu. A proactive secret sharing scheme based on elliptic curve cryptography. In *Proc. of the First International Workshop on Education Technology and Computer Science (ETCS'09), Wuhan, Hubei, China*, pages 666–669. IEEE, March 2009.

[18] H. Tian, F. Zhang, and K. Ren. Secure bilinear pairing outsourcing made more efficient and flexible. In *Proc. of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS'15), Singapore, Republic of Singapore*, pages 417–426. ACM, April 2015.

[19] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel & Distributed Systems*, 23(8):1467–1479, August 2012.

[20] C. Wang, K. Ren, and J. Wang. Secure and practical outsourcing of linear programming in cloud computing. In *Proc. of the 30th IEEE International Conference on Computer Communications (INFOCOM'11), Shanghai, China*, pages 820–828. IEEE, April 2011.

[21] C. Wang, K. Ren, J. Wang, and K. Urs. Harnessing the cloud for securely solving large-scale systems of linear equations. *IEEE Transactions on Parallel & Distributed Systems*, 24(6):549–558, July 2011.

[22] H. Wang. Signer-admissible strong designated verifier signature from bilinear pairings. *Security and Communication Networks*, 7(2):422–428, Feb 2014.

[23] Y. Wang, Q. Wu, D. Wong, B. Qin, S. Chow, Z. Liu, and X. Tan. Securely outsourcing exponentiations with single untrusted program for cloud storage. In *Proc. of the 19th European Symposium on Research in Computer Security (ESORICS'14), Wroclaw, Poland*, volume 8712 of *Lecture Notes in Computer Science*, pages 326–343. Springer, Cham, September 2014.

[24] J. Yu, K. Ren, and C. Wang. Enabling cloud storage auditing with verifiable outsourcing of key updates. *IEEE Transactions on Information Forensics and Security*, 11(6):1362–1375, June 2016.

[25] S. Yu. Big privacy: Challenges and opportunities of privacy study in the age of big data. *IEEE Access*, 4:2751–2763, January 2016.

[26] Y. Yu, Y. Luo, D. Wang, S. Fu, and M. Xu. Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations. In *Proc. of the 2016 IEEE International Conference on Communications (ICC'16), Kuala Lumpur, Malaysia*, pages 1–6. IEEE, May 2016.

[27] F. Zhang and K. Kim. Efficient id-based blind signature and proxy signature from bilinear pairings. In *Proc. of the 8th Australasian Conference-Information Security and Privacy (ACISP'03), Wollongong, Australia*, volume 2727 of *Lecture Notes in Computer Science*, pages 312–323. Springer, Berlin, Heidelberg, July 2003.

[28] F. Zhang, X. Ma, and S. Liu. Efficient computation outsourcing for inverting a class of homomorphic functions. *Information Sciences*, 286:19–28, December 2014.

[29] L. Zhao, M. Zhang, H. Shen, Y. Zhang, and J. Shen. Privacy-preserving outsourcing schemes of modular exponentiations using single untrusted cloud server. *KSII Transactions on Internet and Information Systems*, 11(2):826–845, February 2017.

Author Biography

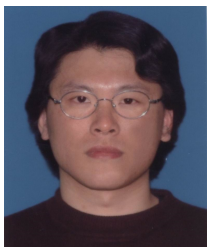
Qianqian Su received the B.S. and M.S. degrees from the College of Computer Science and Technology, Qingdao University, China, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include security, privacy and blockchain.

Rong Hao received M.S. degree in Institute of Network Security from Shandong University, China, in 2006. She is currently a lecture in the College of Information Engineering at Qingdao University, China. Her research interests include digital signature and secret sharing.

Shaoxia Duan received the B.S. and M.S. degrees from the College of Computer Science and Technology, Qingdao University, China, in 2013 and 2016, respectively. She is currently an engineer in China Unicom Network Communications Co., Ltd. JiNan Software Labs.

Fanyu Kong received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. He is currently an associate professor at Shandong University, China. His research interests include cryptanalysis, digital signature, and network security.

Xiaodong Liu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2001 and 1998, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2008. He is currently a lecture at Shandong University, China. His research interests include cryptanalysis, digital signature, and network security.



Jia Yu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. He is currently an professor at Qingdao University, China. His research interests include key evolving cryptography, digital signature, cryptographic protocol, and network security.