CExp: secure and verifiable outsourcing of composite modular exponentiation with single untrusted server

Digital Communications and Networks 3 (2017) 236–241

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.elsevier.com/locate/dcan

CExp: secure and verifiable outsourcing of composite modular exponentiation with single untrusted server

Shuai Li [a], Longxia Huang [a,b], Anmin Fu [a,*], John Yearwood [b]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[b] School of Information Technology, Deakin University, Geelong, VIC 3220, Australia

ARTICLE INFO

Keywords:
Cloud computing
Outsourcing computation
Verifiable computation
Modular exponentiation

ABSTRACT

Outsourcing computing allows users with resource-constrained devices to outsource their complex computation workloads to cloud servers that may not be honest. In this paper, we propose a new algorithm for securing the outsourcing of composite modular exponentiation, which is one of the most complex computing tasks in discrete-log based cryptographic protocols. Unlike algorithms based on two untrusted servers, we outsource modular exponentiation operation to only a single server, which eliminates the potential for a collusion attack when using two servers. Moreover, our proposed algorithm can hide the base and exponent of the outsourced data, which prevents the exposure of sensitive information to cloud servers. In addition, compared with the state-of-the-art algorithms, our scheme has remarkably better checkability. The user could detect any misbehavior with a probability of one if the server returns a fault result.

1. Introduction

Cloud computing is a service that enables convenient, on-demand network access to a shared pool of configurable computing resources that can be instantly deployed and released using minimal management effort [1]. With the rapid development of cloud computing, outsourcing computing has emerged as a new paradigm. This allows users with resource-constrained devices to outsource their complex computational tasks to cloud servers and enjoy unlimited computing resources in a convenient pay-per-use manner. During the whole process of outsourcing, clients do not need to buy any physical servers, storage, software, etc., and do not need to worry about the source of the service. They can directly use the desired services and applications through the network.

Although outsourcing computation provides great convenience for users, it inevitably creates some new security concerns and challenges [2]. First, outsourced computation tasks often contain sensitive information [3] such as personal medical records, personal income status and family member information, which should not be exposed to cloud servers. Therefore, protecting a user's sensitive input/output information is one of the biggest challenges that has arisen with the progress on outsourcing data [4]. Second, because cloud servers are not fully trusted [5], they may return incorrect results. For example, outsourced calculations often require large quantities of computing resources. If users cannot determine whether the cloud server's outputs is correct, the cloud server may be "lazy" to save resources [6]. In addition, software bugs and malicious attacks from adversaries may affect the correctness of the calculation results [7]. As a result, efficiently verifying the outsourced computation results is another important security challenge that has arisen with the progress on outsourcing data. If the outsourcer can verify the results returned by the server with high probability, it is easy to detect any misbehavior of the server.

To address these two key issues, many studies have been performed to find ways securely outsource expensive computations. Chaum et al. [8] first introduced the concept of "wallets with observers", which allows users to install a piece of hardware on their computers to handle some expensive computing workloads. Atallah et al. [16] studied the secure outsourcing of numerical calculations and scientific computing for the first time, and proposed a few effective hiding techniques for scientific computations such as matrix multiplication, inequalities and linear equations. These hiding techniques can guarantee the security and privacy of users' data. Hohenberger et al. [9] first provided a formal security definition and a security model for outsourcing computations, and proposed a secure outsourcing algorithm for modular exponentiation based on two non-colluding cloud servers.

In this paper, we propose a new algorithm, called CExp for the secure outsourcing of composite modular exponentiation. Our main contributions are summarized as follows:

* Corresponding author.
E-mail address: fuam@njust.edu.cn (A. Fu).

http://dx.doi.org/10.1016/j.dcan.2017.05.001
Received 20 February 2017; Received in revised form 19 April 2017; Accepted 3 May 2017
Available online 13 June 2017

1) In contrast to algorithms based on two untrusted servers, we outsource the modular exponentiation operation to only a single server. As a result, it can eliminate the potential for a collusion attack that exists when using two servers.
2) Our proposed algorithm can hide the base and exponent of the outsourced data utilizing a new mathematical division operation. Therefore, our algorithm can realize complete input and output privacy.
3) In particular, compared with the state-of-the-art algorithms, our algorithm gets remarkably better in checkability. The outsourcer can detect any misbehavior with a probability of one. Thus it is not possible for the outsourcer to be cheated by the cloud server.

The remainder of this paper is organized as follows. Section 2 presents a review of the related work on secure outsourcing computation schemes. Section 3 introduces the security definition and system model. The proposed algorithm for modular exponentiation is presented in section 4. We provide a security analysis of the proposed algorithm in section 5. Section 6 evaluates the performance of CExp. Finally, this paper is concluded in section 7.

2. Related work

Atallah et al. [16] first studied the secure outsourcing of numerical calculations and scientific computing, and proposed some disguise techniques for scientific computations such as matrix multiplication, inequalities and linear equations. These disguise techniques can ensure the security and privacy of the user's data. However, they did not discuss the problem that the results cannot be verified. Benjamin et al. [17] used a semantically secure encryption scheme for expensive linear algebra computations (e.g. the multiplication of huge matrices) to construct a verifiable security outsourcing computing protocol for complicated calculations. Gentry et al. [18] first introduced and formalized the notion of verifiable computation, and many effectual studies can be found in this area [19–22]. Moreover, Gentry et al. [18] proposed a fully-homorphic encryption scheme to achieve exceedingly efficient outsourcing, and this scheme can ensure the privacy of the client's inputs and outputs. Atallah et al. [23] proposed a protocol that outsources matrix multiplication based on the weak secret hiding assumption, but this protocol requires the client to perform homomorphic encryption. Subsequently, Wang et al. [24] utilized hiding techniques, iterative methods and semantically secure additive homomorphic encryption, and proposed efficient mechanisms for solving large-scale systems of linear equations in the cloud.

In the cryptographic community, various outsourcing modular exponentiation algorithms [8]– [15] have been designed. Chaum et al. [8] first introduced the notion of "wallets with observers", which allows users to install a piece of hardware on their computers to handle some expensive computing workloads. Hohenberger et al. [9] provided a formal security definition for outsourcing computations, and proposed a secure outsourcing algorithm for exponentiation modulo a prime based on two non-colluding cloud servers. Based on these protocols, Hohenberger et al. [9] achieved outsource-secure Cramer-Shoup encryptions and Schnorr signatures in the cloud environment. However, the checkability of the results with their scheme is only 1/2. Chen et al. [10] proposed a new secure outsourcing scheme for modular exponentiation based on two untrusted program models. Compared with the algorithm [9], the proposed scheme is superior in both efficiency and checkability. Meanwhile, Chen et al. also proposed the first secure and efficient outsourcing algorithm for simultaneous modular exponentiations in [10]. Ye et al. [11] proposed a novel secure outsourcing algorithm for modular exponentiation based on a new mathematical division operation under two non-colluding cloud servers. The base and exponent of the outsourced data can be kept private and the user's computational burden is greatly reduced. Note that all the algorithms in [9]–[11] are based on two untrusted servers. Therefore, they may suffer from a collusion attack [10]. Dijk et al. [12] proposed the first outsourcing scheme for modular exponentiation with a single untrusted program. Unfortunately, because the base of the outsourced exponentiation is exposed to the untrusted server, their scheme cannot guarantee the privacy of the queried input. Ma et al. [13] proposed a new outsourcing scheme for multiple modular exponentiations based on a single untrusted server. Compared with the schemes in [9,10], it does not require complex pre-computation operations which makes it more efficient when solving multiple modular exponentiations. However, their scheme has obvious defects in relation to privacy and the bases and exponents of the outsourced data are public for the servers. Wang et al. [14] proposed a new outsourcing scheme for modular exponentiation with a single untrusted server. This scheme can guarantee the privacy of the exponent and base, but the checkability of the result is just 1/2. Recently, Ding et al. [15] proposed a new secure outsourcing scheme for modular exponentiations based on a single untrusted program model. The checkability of the algorithm in [15] is much better than that of the algorithm proposed in Ref. [14].

3. Security definition and system model

3.1. Security definition

We say that a trusted entity T securely outsources some tasks to an untrusted program U, and (T,U) turns into an outsource-secure implementation of a cryptographic algorithm Alg if (1) T and U implement Alg together, i.e., $Alg=T^U$ and (2) even if T provides oracle access to a malicious U' that remembers all of its computations over time and tries to behave maliciously, U' cannot learn any useful information about the input or output of $T^{U'}$. In the following, we introduce the security definition for the secure outsourcing algorithm [9].

Definition 1. (Algorithm with outsource-I/O). An algorithm Alg complies with the outsource input/output specification if it takes five inputs and generates three outputs. An honest party generates the first three inputs, which are classified by how much the adversary $A=(E,U')$ knows about them, where E is the adversarial environment that submits adversarial chosen inputs to Alg, and U' is the adversarial software operation that is used to replace oracle U. The first input is called the honest, secret input, which is unknown to both E and U'; the second input is called the honest, protected input, which may be known to E, but is shielded from U'; the third input is called the honest, unprotected input, which may be known to E and U. The other two inputs are generated by the environment E and include the adversarial, protected input , which is known by E, but shielded from U', and the adversarial, unprotected input, which may be known to both E and U. Similarly, the first output is called the honest output, which is unknown to both E and U'; the second is protected, which may be known by E, but not U'; and the third is unprotected, which may be known by both parties of A.

The following definition of outsource-security guarantees that the malicious environment E cannot obtain any useful information of the secret inputs and outputs of $T^{U'}$, even if T uses the malicious software U' written by E.

Definition 2. (Outsource-security). Let Alg be an algorithm with outsource input/output. We call a pair of algorithms (T,U) an outsource-secure actualization of Alg if the fllowings are true:

1) Correctness: T^U is a correct implementation of Alg.
2) Security: For all probabilistic polynomial-time adversaries $A=(E,U')$, the existence of probabilistic expected polynomial-time simulators (S_1,S_2) makes no difference in the computation of the following pairs of random variables.
   a) *Pair One:* $EVIEW_{real} \sim EVIEW_{ideal}$ (The external adversary, E, learns nothing)
   b) *Pair Two:* $UVIEW_{real} \sim UVIEW_{ideal}$ (The untrusted software, U', learns nothing)

Here we only provide a brief summary of definition 2. For a complete introduction can refer to [9].

Definition 3. (α−*efficient, secure outsourcing*). A pair of algorithms (T,U) is an α−*efficient* implementation of Alg if (1) T^U is a correct implementation of Alg and (2) ∀ inputs x, and the running time of T is no more than an α-multiplicative factor of the running time of Alg(x).

Definition 4. (β-*checkable, secure outsourcing*). A pair of algorithms (T,U) is a β-*checkable* implementation of Alg if (1) T^U is a correct implementation of Alg and (2) ∀ inputs x, and T can detect the error with a probability of no less than β when U' deviates from its advertised functionality during the execution of $T^{U'}(x)$.

3.2. System model

In this paper, our model includes two entities, the user T and cloud server U. The user T has some complex computing tasks to calculate but lacks the computing power. The cloud server has significant computing power, but can only be assumed to be semi-trusted. When users want to compute modular exponentiations utilizing the cloud server, the process is as follows:

1) Given the modular exponentiation, user T makes some logical divisions to hide the original values and then sends the blinded values to cloud server U.
2) After receiving the blinded values, U calculates these values and returns the results to T.
3) After receiving all of the results from U, T can verify their correctness.

The system model for this scenario is shown in Fig. 1.

4. Proposed algorithm for modular exponentiation

We propose a new secure outsourcing algorithm CExp for exponentiation modulo a composite in the one-malicious model [9]. In CExp, T uses a subroutine called *RandN* [25], which can speed up computations and reduce the computational overhead. A requirement for CExp is that an adversary cannot know any useful information about the inputs and outputs of CExp. In the following algorithm, $U(x,y) \rightarrow y^x$ expresses that U takes (x,y) as the input and outputs $y^x mod N$.

Let p,q be two large primes and$N = pq$. The input of CExp is $u \in Z_N^*$ and $d \in Z_{\phi(N)}^*$, where $\phi(N) = (p-1)(q-1)$. The output of CExp is$u^d \ mod\ N$.

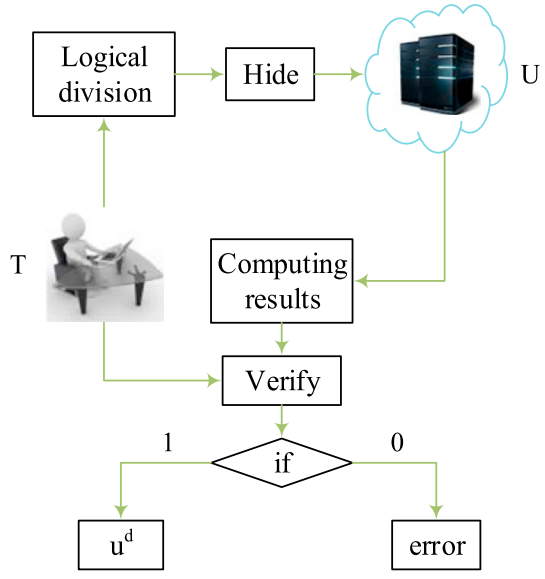

Fig. 1. System model.

Both u and d are computationally blinded to U. (Being computationally blinded means the base u and exponent d are unknown to the server in the data outsourcing calculation process). In order to speed up the computation, we utilize a subroutine called *RandN*. Upon each invocation, *RandN* takes a composite $N = pq$, an exponent $e \in Z_N^*$ and possibly some other values as inputs, where p,q are primes. The output for each invocation is a random, independent pair of the form (x, $x^e \ mod\ N$), where $x \in Z^*$. The proposed algorithm CExp consists of the following sub-algorithms:

4.1. Set up

First, T computes a, where $ad = 1 mod \phi(N)$. Then T randomly chooses $r \in Z_N^*$ and computes $r^a \ mod\ N$. In order to reduce the computational overhead, T chooses a number r that is as small as possible (e.g. $r \in [2,10]$), so that T can easily compute $r^a mod N$. Next T runs*RandN* four times to create four blinding pairs (g_1, g_1^e),(g_2, g_2^e),(g_3, g_3^e) and (g_4, g_4^e). We denote $v_1 = g_1^e \bmod N w_1 = g_2^e \bmod N$,$v_2 = g_3^e \bmod N$ and $w_2 = g_4^e \bmod N$.

4.2. Division

Our main method is to logically split u and d into random looking pieces that can be easily computed by U. The first logical divisions are

$$\begin{aligned} u^d &= (g_1 w)^d \\ &= v_1 g_1^{r_1} w^d \end{aligned} \tag{1}$$

where $w = u/g_1$ and $r_1 = d - e$.

Through the first logical divisions, the base u has been hidden, which is expressed based on random values for v_1,g_1 and w. Next, we also need to hide the exponent d. In order to achieve this goal, the second logical divisions are

$$\begin{aligned} u^d &= v_1 g_1^{r_1} w^d \\ &= v_1 (c_1 g_2)^{r_1} w^d \\ &= v_1 c_1^{r_1} w_1 g_2^{t_1} w^d \\ &= v_1 c_1^{r_1} w_1 g_2^{t_1} w^{l_1 + k_1 t_1} \\ &= v_1 c_1^{r_1} w_1 w^{l_1} \left(w^{k_1} g_2\right)^{t_1} \end{aligned} \tag{2}$$

where $c_1 = g_1/g_2$, $t_1 = r_1 - e$ and $d = l_1 + k_1 t_1$.

The second logical divisions cause d to be blinded by random values l_1,k_1 and t_1. In order to enhance the security of the inputs, the random blinding factor t_1 should be at least 64 bits long.

Meanwhile, T transforms $(u')^d$ using a similar approach,

$$\begin{aligned} (u')^d &= (g_3 w')^d \\ &= v_2 g_3^{r_1} (w')^d \end{aligned} \tag{3}$$

where $u' = u \cdot r^a$, $w' = u'/g_3$ and $r_1 = d-e$.

$$\begin{aligned} (u')^d &= v_2 g_3^{r_1} (w')^d \\ &= v_2 (c_2 g_4)^{r_1} (w')^d \\ &= v_2 c_2^{r_1} w_2 g_4^{t_1} (w')^d \\ &= v_2 c_2^{r_1} w_2 g_4^{t_1} (w')^{l_1 + k_1 t_1} \\ &= v_2 c_2^{r_1} w_2 (w')^{l_1} \left((w')^{k_1} g_4\right)^{t_1} \end{aligned} \tag{4}$$

where $c_2 = g_3/g_4$, $t_1 = r_1 - e$ and $d = l_1 + k_1 t_1$.

4.3. Solve

Using these values (r_1,c_1), (r_1,c_2), (l_1,w), (k_1,w), (l_1,w'), and (k_1,w') transmitted by T, U computes the following modular exponentiations and returns the corresponding computing results to T.

$$\begin{aligned}&(r_1,c_1)\to c_1^{r_1};\\&(r_1,c_2)\to c_2^{r_1};\\&(l_1,w)\to w^{l_1};\\&(k_1,w)\to w^{k_1};\\&(l_1,w')\to (w')^{l_1};\\&(k_1,w')\to (w')^{k_1}.\end{aligned}\tag{5}$$

4.4. Verify

T checks whether U has produced the correct output, i.e.,

$$r\left(v_1c_1^{r_1}w_1w^{l_1}\left(w^{k_1}g_2\right)^{t_1}\right)\stackrel{?}{=}v_2c_2^{r_1}w_2(w')^{l_1}\left((w')^{k_1}g_4\right)^{t_1}\tag{6}$$

If not, it indicates that U has produced wrong responses, and T outputs "error"; otherwise, T can compute the final result

$$u^d=v_1c_1^{r_1}w_1w^{l_1}\left(w^{k_1}g_2\right)^{t_1}\tag{7}$$

5. Security evaluation

In this section, we provide a security analysis for the proposed algorithm CExp in the one-malicious model, along with a comparison of different algorithms.

5.1. Security analysis

Theorem 1. **(*Outsource-security*).** *Based on the single untrusted program model, the above algorithms (T,U) are an outsource-secure implementation of CExp, where the input (d,u) may be honest, secret; honest, protected; or adversarial, protected.*

Proof: *The correctness property is obvious. If U performs honestly, T can compute*

$$\begin{aligned}&u^d=v_1c_1^{r_1}w_1w^{l_1}\left(w^{k_1}g_2\right)^{t_1}\\&=v_1c_1^{r_1}w_1g_2^{t_1}w^{l_1+k_1t_1}\\&=v_1c_1^{r_1}w_1g_2^{t_1}w^{d}\\&=v_1(c_1g_2)^{r_1}w^{d}\\&=v_1g_1^{r_1}w^{d}\\&=(g_1w)^{d}\\&=u^{d}\end{aligned}\tag{8}$$

$$\begin{aligned}&(u')^d=v_2c_2^{r_1}w_2(w')^{l_1}\left((w')^{k_1}g_4\right)^{t_1}\\&=v_2c_2^{r_1}w_2g_4^{t_1}(w')^{l_1+k_1t_1}\\&=v_2c_2^{r_1}w_2g_4^{t_1}(w')^{d}\\&=v_2(c_2g_4)^{r_1}(w')^{d}\\&=v_2g_3^{r_1}(w')^{d}\\&=(g_3w')^{d}\\&=(u')^{d}\end{aligned}\tag{9}$$

Based on equations (8) and (9), equation (6) *is established as long as U generates correct responses.*

$$\begin{aligned}&ru^d=(u')^d\\&rv_1c_1^{r_1}w_1w^{l_1}\left(w^{k_1}g_2\right)^{t_1}=v_2c_2^{r_1}w_2(w')^{l_1}\left((w')^{k_1}g_4\right)^{t_1}\end{aligned}\tag{10}$$

Next we prove the security. First, we prove Pair One $EVIEW_{real}\sim EVIEW_{ideal}$: *if the input (d,u) is not the case of an honest, secret input,E can always know the input information. Obviously, the execution process of the simulator* S_1 *will be the same as that of a real experiment in this case. Consequently, we only need to consider the case of an honest, secret input. In the ideal experiment, the execution process of the simulator* S_1 *is as follows: after receiving the input on round i,* S_1 *ignores it and instead makes six random queries of the form* (α_j,β_j) *to* U'. *After that* S_1 *checks six outputs* $\beta_j^{a_j}$ *from* U'. *If an error has been detected,* S_1 *saves all of the states and outputs* $Y_p^i=$ "error", $Y_u^i=\varphi$, $rep^i=1$, *and at this time the output for ideal process is* $(estate^i,$"error"$,\varphi)$. *If all the checks have been passed,* S_1 *outputs* $Y_p^i=\varphi$, $Y_u^i=\varphi$, $rep^i=0$, *and at this point, the output for the ideal process is* $(estate^i,y_p^i,y_u^i)$.

Additionally, we need to illustrate that the input distributions to U' *are computationally indistinguishable in the real and ideal experiments. In the ideal experiment, the inputs are selected randomly and uniformly. In the real experiment, each part of the query that T makes to U is independently re-randomized and computationally indistinguishable. If* U' *behaves honestly on round i, then* $EVIEW_{real}^i\sim EVIEW_{ideal}^i$. *Because* $T^{U'}$ *correctly executes CExp in the real experiment and* S_1 *simulates with the same outputs in the ideal experiment, i.e.,* $rep^i=0$. *If* U' *is dishonest on round i and returns incorrect calculation results, then the error will be detected by both T and* S_1 *with a probability of one, resulting in an output of "error". Thus,* $EVIEW_{real}^i\sim EVIEW_{ideal}^i$ *even when* U' *misbehaves. By the hybrid argument, we conclude that* $EVIEW_{real}\sim EVIEW_{ideal}$.

Second, we prove Pair $TwoUVIEW_{real}\sim UVIEW_{ideal}$. *If the input (d,u) is not honest, secret; honest, protected; or adversarial, protected,* U' *can always know the input information. Obviously, the execution process of the simulator* S_2 *will be the same as a real experiment in this case. Consequently, we only need to consider the case where the input is honest, secret; honest, protected; or adversarial, protected. In the ideal experimental environment,* S_2 *always behaves as follows: after getting the input on round i,* S_2 *ignores it and instead makes six random queries of the form* (α_j,β_j) *to* U'. *After that* S_2 *saves its states and the state of* U'. *Similar to that previously shown, it can be demonstrated that* $UVIEW_{real}^i\sim UVIEW_{ideal}^i$ *and the inputs to* U' *in the real experiment are computationally indistinguishable from those in the ideal one randomly selected by* S_2. *Consequently, although E can easily distinguish between these real and ideal experiments, E cannot communicate this information to* U'. *Therefore, we conclude that* $UVIEW_{real}\sim UVIEW_{ideal}$.

Theorem 2. $\left(O\left(\frac{2k+\log t_1+\log a}{n}\right)\right)$**-*efficient CExp*).** *In the single untrusted program model, the above algorithms (T,U) are an* $O\left(\frac{2k+\log t_1+\log a}{n}\right)$*-efficient secure implementation of CExp.*

Proof: *On one hand, the proposed algorithm CExp makes* four *calls toR and N plus* $(14+3\log t_1+1.5\log a)$ *Modular Multiplication (MM) and* four *Modular Inverse (MInv)* operations *in order to compute* u^d *modN*. In addition, *CExp takes* $2k$ *MM using the BPV generator in* [25], *where k is a parameter related to the size of the subset sum. On the other hand, it takes roughly* $1.5n$ *MM to calculate* u^d *mod N by the square-and-multiply method, where n is the bit of d. Thus, the algorithms (T,U) are an* $O\left(\frac{2k+\log t_1+\log a}{n}\right)$*-efficient implementation of CExp.*

Theorem 3. **(1-*checkable CExp*).** *In the single untrusted program model, the above algorithms(T,U) are a* 1*-checkable implementation of CExp.*

Proof: *When T receives the computing results from U, T can verify whether* equation (6) *is established.*

$$r\left(v_1c_1^{r_1}w_1w^{l_1}\left(w^{k_1}g_2\right)^{t_1}\right)\stackrel{?}{=}v_2c_2^{r_1}w_2(w')^{l_1}\left((w')^{k_1}g_4\right)^{t_1}$$

If the equation does not hold, this indicates that U has produced wrong responses. T will detect the misbehavior of server U with a probability of one.

5.2. Comparison

We compare the proposed algorithm with the state-of-the-art algorithms [10] [14]. Let MM denote a modular multiplication, MInv denote a modular inverse, and Invoke denote an invocation of the subroutine*RandN*. We omit other operations such as modular additions in these algorithms because the computation of these operations is negligible compared to the others. A comparison of the efficiency and checkability of these algorithms is provided in Table 1.

According to Table 1, we can see that the checkability of CExp is much higher than those of Exp [10], and GExp [14]. In CExp, the outsourcer

Table 1
Comparison of the algorithms.

	Exp [10]	GExp [14]	Our CExp
Security Model	Two UP	Single UP	Single UP
MM	7	$12 + 1.5\log\gamma$	$14+3\log t_1+1.5\log a$
MInv	3	4	4
Invoke (Rand)	5	6	4
Invoke (U)	6	4	6
Checkability	2/3	1/2	1

("Two/Single UP" denotes Two/Single Untrusted Program Model respectively).

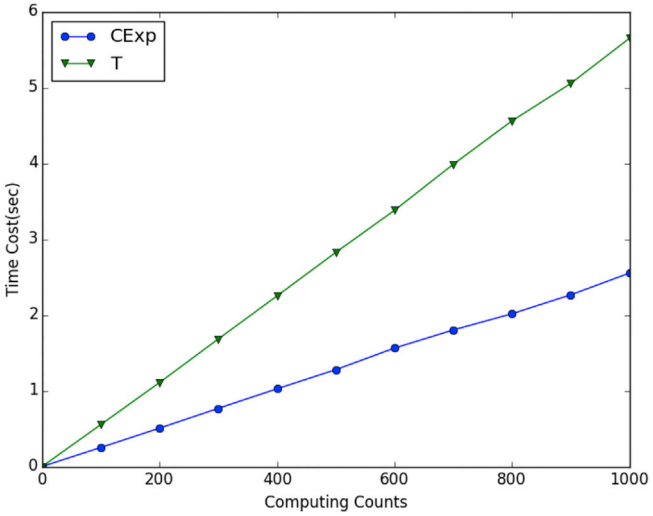

Fig. 2. Simulation for CExp algorithm.

can detect any misbehavior with a probability of one. Thus, it is not possible for the outsourcer to be cheated by the cloud server. Although Exp [10] performs better in terms of the benchmark MM, it is based on the two untrusted program models. Thus, Exp [10] may suffer from collusion attack. Compared with GExp [14], our CExp needs to pay additional $1.5\log a$ MM to calculate the value of r^a. Note that when it comes to variable-based and fixed-exponent exponentiation case, we can reduce the computational overhead by amortization, since we can reuse the r^a.

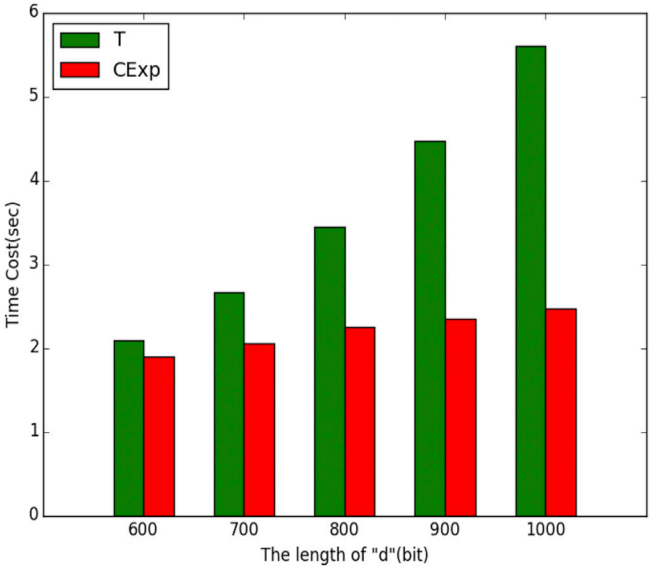

Fig. 3. Time cost of.u^d.

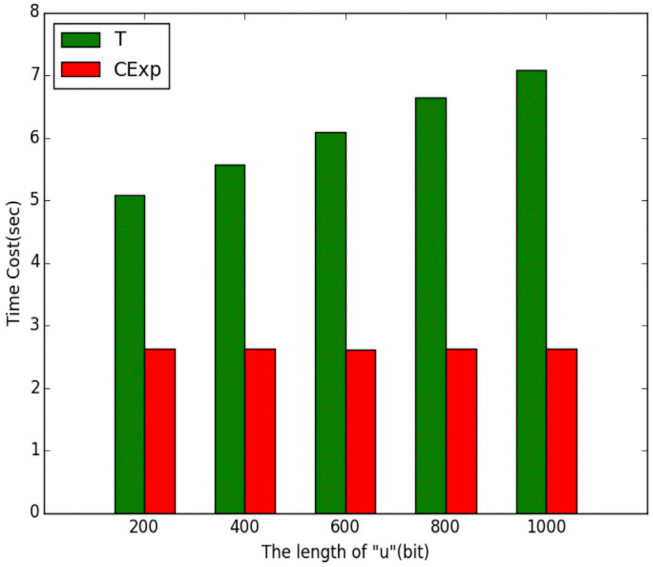

Fig. 4. Time cost of.u^d.

6. Performance evaluation

We implemented our proposed outsourcing algorithm using the Python language. Our experiment was conducted on two machines with Intel Core i5 processors running at 3.20 GHz with 4G of memory (cloud server), and an Intel Core i5 processor running at 2.10 GHz with 2G of memory (local user).

Fig. 2 shows the time cost results of computing the modular exponentiation with the CExp algorithm and without the outsourcing algorithm. The time cost for CExp is smaller than that for computing modular exponentiation without outsourcing. By utilizing our proposed outsourcing algorithm, the time cost can be greatly reduced because CExp delegates some heavy calculations to U. In this way the user can afford the computational overhead, which is much less than before. In Fig. 3, we provide the simulation results of the CExp algorithm, with the time costs of different exponent bit lengths. Computing the modular exponentiation without outsourcing is rather time-consuming. Note that the time cost of computing the modular exponentiation without an outsourcing algorithm increases relatively quickly as the bit length of the exponent grows. By utilizing our proposed outsourcing algorithm CExp, the time cost increases slowly with the bit length of the exponent. The time costs with different bit lengths for the base is shown in Fig. 4. As the bit length of the base increases, the time cost of the modular exponentiation without outsourcing increases, but the time cost for CExp is almost constant.

7. Conclusion

In this paper, we proposed a new algorithm for the secure outsourcing of composite modular exponentiation with only one untrusted server, which avoids the risk of a collusion attack. In addition, the proposed algorithm can guarantee the privacy of the outsourced data. After logically splitting the data into random looking pieces, the cloud server cannot obtain any sensitive information. In particular, in our proposed algorithm, the outsourcer can detect any misbehavior of the server with a probability of one. In addition, the analysis and experimental results show that our proposed algorithm is provably secure and efficient.

Acknowledgments

This work was supported by the National Science Foundation of China (61572255), Natural Science Foundation of Jiangsu Province,

China (BK20141404 and BK20150787), Six talent peaks project of Jiangsu Province, China (XYDXXJS-032) and Innovation of graduate student training project in Jiangsu Province, China (KYLX16_0465).

References

[1] K. Ren, C. Wang, Q. Wang, Security challenges for the public cloud, IEEE Internet Comput. 16 (1) (Jan. 2012) 69–73.
[2] S. Yu, Big privacy: challenges and opportunities of privacy study in the age of big data, IEEE Access 4 (6) (Jun. 2016) 2751–2763.
[3] L. Huang, G. Zhang, A. Fu, Certificateless public verification scheme with privacy-preserving and message recovery for dynamic group, in: Proc. ACSW'17, 2017.
[4] Y. Ren, N. Ding, X. Zhang, et al., Verifiable Outsourcing Algorithms for Modular Exponentiations with Improved Checkability, in: Proc. 2016 ASIA CCS, pp. 293–303.
[5] A. Fu, N. Qin, J. Song, et al., Privacy-Preserving public auditing for multiple managers shared data in the cloud, J. Comput. Res. Dev. 52 (10) (Oct., 2015) 2353–2362.
[6] X. Ma, J. Li, F. Zhang, Outsourcing computation of modular exponentiations in cloud computing, Clust. Comput. 16 (4) (Dec. 2013) 787–796.
[7] L. Huang, G. Zhang, A. Fu, Privacy-preserving public auditing for dynamic group based on hierarchical tree, J. Comput. Res. Dev. 53 (10) (2016) 2334–2342.
[8] D. Chaum, P. Torben, Wallet databases with observers, in: Proc. 1992 CRYPTO. pp. 89–105.
[9] S. Hohenberger, A. Lysyanskaya, How to securely outsource cryptographic computations, in: Proc. 2005 TCC. pp. 264–282.
[10] X. Chen, J. Li, J. Ma, et al., New algorithms for secure outsourcing of modular exponentiation, IEEE Trans. Parallel Distrib. Syst. 25 (9) (Sep. 2014) 2386–2396.
[11] J. Ye, Z. Xu, Y. Ding, Secure outsourcing of modular exponentiations in cloud and cluster computing, Clust. Comput. 19 (2) (Jun. 2016) 811–820.
[12] M. Van Dijk, D. Clarke, B. Gassend, et al., Speeding up exponentiation using an untrusted computational resource, Des. Codes Cryptogr. 39 (2) (May. 2006) 253–273.
[13] X. Ma, J. Li, F. Zhang, Outsourcing computation of modular exponentiations in cloud computing, Clust. Comput. 16 (4) (Dec. 2013) 787–796.
[14] Y. Wang, Q. Wu, D. S. Wong, et al., Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage, in: Proc. 2014 ESORICS. pp. 326–343.
[15] Y. Ding, Z. Xu, J. Ye, et al., Secure outsourcing of modular exponentiations under single untrused programme model, J. Comput. Syst. Sci. (Nov. 2016), http://dx.doi.org/10.1016/j.jcss.2016.11.005.
[16] M.J. Atallah, K.N. Pantazopoulos, J.R. Rice, et al., Secure outsourcing of scientific computations, Adv. Comput. 54 (May, 2008) 215–272.
[17] D. Benjamin D, M.J. Atallah, Private and cheating-free outsourcing of algebraic computations, in: Proc. 2008 PST, pp. 240–245.
[18] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers, in: Proc. 2010 CRYPTO. pp. 465–482.
[19] D. Fiore, R. Gennaro, Publicly verifiable delegation of large polynomials and matrix computations with applications, in: Proc. 2012 ASIA CCS. pp. 501–512.
[20] S. Benabbas, R. Gennaro, and Y. Vahlis, Verifiable delegation of computation over large datasets, in: Proc. 2011 EUROCRYPT. pp. 111–131.
[21] R. Canetti, B. Riva, and G. Rothblum, Two protocols for delegation of computation, in: Proc 2012 ICITS. pp. 37–61.
[22] D. Beaver, J. Feigenbaum, J. Kilian, et al., Locally random reductions: improvements and applications, J. Cryptol. 10 (1) (Dec. 1997) 17–37.
[23] M.J. Atallah, K.B. Frikken, Securely outsourcing linear algebra computations, in: Proc. 2010 ASIA CCS. pp. 48–59.
[24] C. Wang, K. Ren, J. Wang, et al., Harnessing the cloud for securely solving large-scale systems of linear equations, in: Proc. 2011 ICDCS, pp. 549–558.
[25] V. Boyko, M. Peinado, R. Venkatesan, Speeding up discrete log and factoring based schemes via precomputations, in: Proc. 1998 EUROCRYPT. pp. 221–235.