Novel and Secure Outsourcing Algorithms for Multiple Bilinear Pairings with Single Untrusted Server

Jiaxiang Yang[1], Yanping Li[1], and Yanli Ren[2]
(Corresponding author: Yanping Li)
School of Mathematics and Information Science, Shaanxi Normal University, Xi'an 710119, China[1]
School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China[2]
(Email: lyp@snnu.edu.cn)
(Received Mar. 9, 2018; Revised and Accepted June 7, 2018; First Online Mar. 2, 2019)

Abstract

Bilinear pairing is one of the most widely-used and time-consuming operations in public key cryptographic algorithms and schemes. Generally, most of the schemes need two or more pairing operations. However, almost all the existing outsourcing algorithms for bilinear pairings can only outsource one pairing operation at one time, and multiple pairings need to be outsourced one by one in a sequence, which may be more inefficient and time-consuming. Currently, the most efficient outsourcing algorithm for one bilinear pairing has a checkability about 2/5 under the one-untrusted-program (OUP). Two novel outsourcing algorithms for multiple bilinear pairings under the same security assumption are proposed in this paper. One can outsource two asymmetric bilinear pairings simultaneously with checkability about 1/4 and the other can outsource two symmetric bilinear pairings simultaneously with checkability about 2/7, both of which have higher checkability than the current most efficient outsourcing algorithm on the condition it also outsource two bilinear pairings. Finally, we proved the security of the two algorithms and analyzed the efficiency by comparing them with prior works. The performance analysis showed that our algorithms are more efficient and practical.

Keywords: Bilinear Pairing; Cloud Computing; Secure Outsourcing; Single Untrusted Server

1 Introduction

With the development of cloud computing technology, outsourcing computation has attracted extensive attention of academia and industry. More and more mobile intelligent terminals, such as smart-phone, tablets and iPad, become the integral part of our life [22, 23, 27, 30]. These resource-constrained devices will face the shortcoming of limited computation when they come across complex computational problems. Outsourcing computation is an important way to solve this type of problem [12, 17]. Therefore, more and more mobile smart devices become a strong demand and driving force for outsourcing computation. Cloud outsourcing computation enables the cloud service providers to provide unlimited computing resources to users, which not only save the users' computational cost, but also improve the users' computation efficiency. Hence, outsourcing computation became a new and popular computing paradigm [2, 3, 11, 16, 20].

Generally, outsourcing tasks are some computations with higher complexity. Especially, as the computing parameters become larger, the computation gets more time-consuming and computationally expensive [10, 18], such as bilinear pairings which are considered the most common and expensive operations in cryptographic algorithms and schemes. Since bilinear pairings play a very important and significant role [9, 19, 21, 31], a large quantity of pairing-based algorithms and protocols are proposeed. Due to its widespread application and higher complexity, outsourcing computation of bilinear pairing is a realistic problem in practice.

A growing number of concrete outsourcing schemes for bilinear pairing have been put forward [6, 13, 25, 26, 29] in the last few years. These algorithms allow that computation-limited users delegate the computing tasks to the cloud, successfully outsourcing computation of bilinear pairing. However, it also inevitably faces some new challenges, which can be summed up as the following three aspects.

Assumption: The number and trustability of cloud servers are the crucial factors influencing the practicality of the scheme. At present, outsourcing algorithms are based on three assumptions. The one-untrusted program (OUP) supposes that one server implements an algorithm and the server could be malicious. The one-malicious version of two-untrusted program (OMTUP) that assumes two servers per-

form an algorithm and only one of them is malicious. The two-untrusted program (TUP) demands that two servers carry out an algorithm and they could be malicious. Since it is difficult to find fully trusted server and two servers require more hardware resources. Obviously OUP assumption is the most practical.

Secrecy: The cloud server of outsourcing computation may be untrusted, and outsourcing data often contains users' sensitive information that cannot be leaked to the cloud server. That is, the cloud server cannot get the contents of the outsourcing data. Hence, it is required that the cloud server should learn nothing useful about what it is actually computing after outsourcing computation.

Checkability: Driven by the cloud server's own economic interests, or because of the failure of software and hardware, the cloud server may return some incorrect or incomplete results to the user. Therefore, the outsourcers should have the ability to check the correctness of the results with some certain checkability, that is to say, the construction not only needs to have higher efficiency, but also higher checkability.

In order to protect data privacy and solve checkability problems, Gennaro *et al.* [8] proposed a checkable outsourcing computation algorithm, in which the inputs and the outputs are confidential to the server, *i.e.*, the server cannot obtain the exact value of the outsourced computation task. Additionally, the user is able to check the correctness of the server's return value. Since then, almost all of outsourcing algorithms and schemes focus on protecting the privacy of outsourced data and pursuing higher checkability of the return value.

1.1 Previous Work

In 2010, Chevallier-Mames *et al.* first proposed the outsourcing algorithm for secure delegation of elliptic-curve bilinear pairing based on an untrusted server, which suggests that a computation-limited terminal outsources the computation of bilinear pairing to a more resourceful server [5]. If the server returns a random value instead of the true computational result (*i.e.*, the server does not do the computation), the outsourcer can check the correctness of the return value with a probability about 1. based on Chevallier-Mames *et al.*'s algorithm, Chen *et al.* make an improvement to reduce the user's point multiplication and exponentiation by pre-computation [19]. Unfortunately, the checkability of server's outputs has dropped from 1 to 1/2. Later, Tian *et al.* proposed two outsourcing algorithms A and B for bilinear pairings [20], which reduce the user's computation amount by changing the complexity of the pre-computation, that is, improve the outsourcing efficiency. However, its assumption, finding two servers, of which at least one server is honest, is very hard to realize in the real cloud computing environment. Therefore, more practical outsourcing computing should be based on a single server without the honest assumption of servers [32]. Then Jiang and Ren proposed an algorithm under the OUP model [15], but the checkability is only 2/5.

Generally, many signature schemes and cryptographic protocols require two or more bilinear pairings [1,7,14,24]. However, almost all the existing outsourcing algorithms for bilinear pairings can only outsource a single bilinear pairing at one time. And if there are multiple bilinear pairings to outsource, it has to outsource one by one in a certain order, which is very time-consuming and inefficient, and maybe results in lower checkability. If we could outsource multiple pairings of computation at one time, and the resourceful server could do the computation task in parallel and return the results quickly, the time cost could be saved greatly. Based on such simple idea, we try to design two algorithms in this paper, which outsource two bilinear pairings at one time to improve the outsourcing efficiency under OUP model with the improved checkability.

1.2 Our Contributions

Based on Jiang and Ren algorithm [15], this paper proposes two novel outsourcing algorithms (**Pai** and **SPai**). Unlike most of existing algorithms, the algorithm **Pai** can outsource two asymmetric bilinear pairings to an untrusted server at the same time with checkability about 1/4. While the algorithm **SPai** can more efficiently outsource two symmetric bilinear pairings simultaneously, which not only decreases the users' computation overhead and protects the users' data privacy, but also improves the checkability. Compared to the existing related algorithms, **Pai** and **SPai** have the following advantages.

First, since the OMTUP model with only one server being malicious is too strong and the TUP model with two untrusted servers is impractical, the OUP model with a single untrusted server is more reasonable and more practical. Both algorithms **Pai** and **SPai** in this paper are designed under the OUP model and can be provably secure.

Second, **Pai** and **SPai** can provide the privacy protection of user data by obfuscating inputs. What's more important, **Pai** and **SPai** can outsource two bilinear pairings at the same time, which reduces the users' computation overhead and saves the outsourcing time-cost greatly.

Third, currently, the most efficient outsourcing algorithm of bilinear pairings under the OUP model has a checkability about 2/5 [15]. If two bilinear pairings are outsourced one by one, the checkability is reduced to 4/25. While our algorithm **Pai** outsources two asymmetric bilinear pairings to an untrusted server at the same time with checkability about 1/4. And the algorithm **SPai** can outsource two symmetric bilinear pairings simultaneously with checkability about 2/7.

In conclusion, algorithms **Pai** and **SPai** can outsource two bilinear pairings simultaneously with improved check-

ability under the most practical OUP model, so our proposed algorithms are more efficient and practical.

1.3 Structure of the Paper

The rest of the paper is structured as follows. In Section 2, some basic knowledge for bilinear pairing are reviewed, and formal security definitions and the system model are given. Novel outsourcing algorithms of **Pai** and **SPai** are presented in Section 3 and their security analyses are demonstrated in Section 4. Performance comparisons with other related algorithms are analyzed in Section 5, and the Section 6 concludes our work.

2 Preliminaries

2.1 Bilinear Pairing

Let G_1 and G_2 be two cyclic additive groups with a large prime order q, and $G_1 =< P_1 >$,$G_2 =< P_2 >$. Let G_T be a cyclic multiplicative group with the same order q. A bilinear pairing is a map $e(\cdot,\cdot) : G_1 \times G_2 \to G_T$ with the following properties:

1) Bilinear: $e(aR, bQ) = e(R, Q)^{ab}$ for all $R \in G_1, Q \in G_2$,
$a, b \in Z_q^*$.

2) Non-degenerate: There exist $R \in G_1$ and $Q \in G_2$ such that $e(R, Q) \neq 1_{G_T}$.

3) Computable: There is an efficient algorithm to compute $e(R, Q)$ for all $R \in G_1, Q \in G_2$.

2.2 Formal Security Definitions

Now we review the formal security definitions of outsourcing algorithm introduced by Hohenberger and Lysyanskaya [10]. Following these definitions, Chen *et al.* and Tian *et al.* proposed their algorithms, respectively. Our algorithms are also based on these security definitions. The detailed definitions of outsourcing computation are introduced below.

The algorithm ***Alg*** includes a trusted party T and an untrusted program U. E represents an untrusted environment. T is a limited computation party who tries to outsource its computation task to the party U. T^U denotes T carries out the computation by invoking U. An adversary A is simulated by a pair of algorithms (E, U'), where E denotes the adversarial environment that submits malicious inputs to ***Alg*** and represents malicious software written by E. As described in [10], we assume that the two adversaries (E, U') can make direct communication only before the execution of T^U, and in other cases, they can only communicate with each other by passing messages through the outsourcer T.
The formal definitions of outsource-inputs/outputs are given as follows:

Definition 1. *(Algorithm with outsource-I/O) The algorithm* **Alg** *includes five inputs and three outputs. The first three inputs are generated by the trusted of T, and are classified as according to how much information the adversary $A = (E, U')$ learns about them, they secret, protected and unprotected. The first input is honest, secret, which is unknown to E and U'. The second input is honest and protected, which is public for E, but is kept secret from U'. The third input is honest and unprotected, which is known by both E and U'. The last two inputs are chosen by the malicious environment E. One is the adversarial protected input that E know it and is secret for U'. The other is the adversarial unprotected input that are open to both E and U'.*

Definition 2. *(Outsource-security) Let* **Alg** *be an algorithm with outsource-I/O. The implementation of* **Alg** *is secure if:*

1) *Correctness: $T^{U'}$ is a correct implementation of* **Alg**.

2) *Security: For all probabilistic polynomial time (PPT) adversaries $A = (E, U')$, there exist expected probabilistic polynomial time simulations (S_1, S_2) such that the following pairs of random variables are computationally indistinguishable.*

Pair one: $EVIEW_{real} \sim EVIEW_{ideal}$:

The adversarial environment E can obtain nothing about inputs or outputs during the execution of T^U. The real process and ideal process proceed in turn.

$$
\begin{aligned}
&EVIEW^i_{real} = \\
&\quad\{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1}); \\
&\quad\ (estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \\
&\qquad \leftarrow E(1^k, EVIEW^{i-1}_{real}, x^i_{hp}, x^i_{hu}); \\
&\quad\ (tstate^i, ustate^i, y^i_s, y^i_p, y^i_u) \\
&\qquad \leftarrow T^{U'(ustate^{i-1})}(tstate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, \\
&\qquad\qquad x^{j^i}_{hu}, x^{j^i}_{ap}, x^{j^i}_{au}) : (estate^i, y^i_p, y^i_u)\} \\
&EVIEW_{real} = EVIEW^i_{real} \text{ if } stop^i = TRUE.
\end{aligned}
$$

An honest process I inputs a security parameter k and its $i-1$ round internal state $istate^{i-1}$ to produce its i round honest state and honest inputs $x^i_{hs}, x^i_{hp}, x^i_{hu}$ for $T^{U'}$. In the same way, the adversarial environment E takes its $i-1$ round view $EVIEW^{i-1}_{real}$, k and x^i_{hp}, x^i_{hu} as inputs to produce its i round internal state $estate^i$, the order of honest inputs j^i, the i round malicious inputs x^i_{ap}, x^i_{au}, and a signal sign $stop^i$. The adversary U takes its $i-1$ round internal state $ustate^{i-1}$ to react with T in the ith round. The implementation of T^U takes five inputs and the $i-1$ round internal state $tstate^{i-1}$ to produce i round internal states of T and U, and the i

round outputs y_s^i, y_p^i, y_u^i. The view of the real process in round i consists of $estate^i$ and the values of y_p^i, y_u^i.

$$
\begin{aligned}
&EVIEW_{ideal}^i = \\
&\{(istate^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, istate^{i-1}); \\
&(estate^i, j^i, x_{ap}^i, x_{au}^i, stop^i) \\
&\quad \leftarrow E(1^k, EVIEW_{ideal}^{i-1}, x_{hp}^i, x_{hu}^i); \\
&(astate^i, y_s^i, y_p^i, y_u^i) \\
&\quad \leftarrow Alg(astate^{i-1}, x_{hs}^{j^i}, x_{hp}^i, x_{hu}^i); \\
&(sstate^i, ustate^i) \\
&\quad \leftarrow S_1^{U'(ustate^{i-1})}(sstate^{i-1}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^{j^i}, \\
&\qquad x_{au}^{j^i}, y_p^i, y_u^i); \\
&(z_p^i, z_u^i) = replace^i(Y_p^i, Y_u^i) \\
&\quad +(1 - replace^i)(y_p^i, y_u^i) : (estate^i, z_p^i, z_u^i)\} \\
&EVIEW_{ideal} = EVIEW_{ideal}^i \text{ if } stop^i = TRUE.
\end{aligned}
$$

In the ideal process, we have a stateful simulator S_1 to participate the algorithm. The algorithm ***Alg*** takes its $i-1$ round internal state $astate^{i-1}$ and five inputs to get i round internal state $astate^i$ and three outputs. The simulated implementation $S_1^{U'}$ inputs its $i-1$ round internal state $sstate^{i-1}$, all the protected and unprotected inputs and outputs to produce the i round internal state of S_1 and U', the simulated protected and unprotected, and a signal $replace^i \in \{0,1\}$. The response signal is used to determine i round (z_p^i, z_u^i) for $EVIEW_{ideal}^i$.

Pair two: $UVIEW_{real} \sim UVIEW_{ideal}$: The view that the untrusted software obtains by participating in the process is described in **Pair One**. So $UVIEW_{real} = ustate^i$ if $stop^i = TRUE$. The ideal process is as follows:

$$
\begin{aligned}
&UVIEW_{real}^i = \\
&\{(istate^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, istate^{i-1}); \\
&(estate^i, j^i, x_{ap}^i, x_{au}^i, stop^i) \\
&\quad \leftarrow E(1^k, estate^{i-1}, x_{hp}^i, x_{hu}^i, y_p^{i-1}, y_u^{i-1}); \\
&(astate^i, y_s^i, y_p^i, y_u^i) \\
&\quad \leftarrow Alg(astate^{i-1}, x_{hs}^{j^i}, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i); \\
&(sstate^i, ustate^i) \\
&\quad \leftarrow S_2^{U'(ustate^{i-1})}(sstate^i, x_{hu}^{j^i}, x_{au}^i)\} \\
&UVIEW_{ideal} = UVIEW_{ideal}^i \text{ if } stop^i = TRUE.
\end{aligned}
$$

The algorithms I, E are the same as those in the $EVIEW_{real}^i$ of the above Pair One definition. While the algorithm ***Alg*** is also defined in the same way as that in the $EVIEW_{ideal}^i$ of **Pair One** definition. The simulated implementation $S_2^{U'}$ takes the ith round internal state $sstate^{i-1}$ and two unprotected inputs to produce the state of $sstate^i, ustate^i$.

Assume that T^U is a correct execution of ***Alg***, some definitions could be reached in the following.

Definition 3. *(α-efficient, secure outsourcing): If for any input x, the running time of T is no more than an α-multiplicative factor of the running time of* **Alg**, *then the algorithm (T, U) is α-efficient secure outsourcing.*

Definition 4. *(β-checkable, secure outsourcing): If for any input x, T could detect any error with a probability no less than β if the U' works maliciously during the execution of $T^{U'}$, then the algorithm (T, U) is β-checkable secure outsourcing.*

Definition 5. *((α, β)-outsource-security): If an algorithm (T, U) is α-efficient and β-checkable, then it will be said to be an (α, β)-outsource-secure implementation of* **Alg**.

2.3 System Model

There are two parties involved in our schemes, that is, the user T and the cloud server U who may be malicious, as shown in Figure 1. Our model can be described in the following.

1) Given two bilinear pairings which will be computed, the user T invokes ***Rand***.
2) ***Rand*** returns a random five-tuple to the user T.
3) T blinds the inputs with the random five-tuple and sends the blind values to cloud sever U.
4) On receiving the obfuscated values, U computes and returns the results to T.
5) After receiving the results from U, T verifies the correctness of the results. If the results are not correct, T will output "error". Otherwise, T will compute the values of the given two bilinear pairings by using the returned results from U.

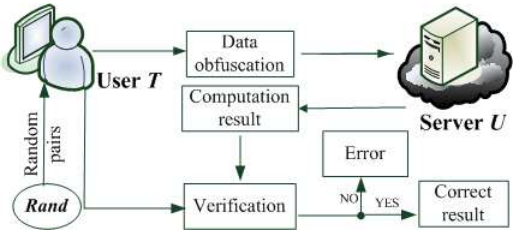

Figure 1: The system architecture of our algorithms

In [10], a subroutine ***Rand***, which can generate a random five-tuple, is used to speed up the computations. The user T invokes this subroutine many times to get a table of random five-tuple. T retrieves some new pairs in the table when needed. We call this table-lookup method. Similarly, we also adopt such a subroutine, whose specific workflow is given as follows:

Input: A large prime q, two cyclic additive groups G_1 and G_2 with order q and a bilinear pairing e.

Output: $(V_1, V_2, v_1V_1, v_2V_2, e(v_1V_1, v_2V_2))$, where v_1, v_2 $\in_R Z_p^*$, $V_1 \in G_1$ and $V_2 \in G_2$.

3 Novel Outsourcing Algorithms for Multiple Bilinear Pairings

In this section, two novels outsourcing algorithms of bilinear pairings **Pai** and **SPai** are proposed. Both **Pai** and **SPai** outsource two bilinear pairings simultaneously to a single untrusted server. Furthermore, the algorithms we proposed also keep the privacy of outsourcing data without reducing checkability.

3.1 Pai: Outsourcing e(A,B), e(C,D) Simultaneously

Pai algorithm can simultaneously outsource $e(A, B)$ and $e(C, D)$, where $A, C \in G_1$ and $B, D \in G_2$, $e(\cdot, \cdot) : G_1 \times G_2 \to G_T$ is an asymmetric bilinear pairing. To ensure the privacy of the outsourcing data, A, B, C, D should be kept secret from the server U. The concrete steps are described as follows:

1) T runs ***Rand*** three times to obtain three random five-tuple: $(V_1, V_2, v_1V_1, v_2V_2, e(v_1V_1, v_2V_2))$, $(X_1, X_2, x_1X_1, x_2X_2, e(x_1X_1, x_2X_2))$, and $(Y_1, Y_2, y_1Y_1, y_2Y_2, e(y_1Y_1, y_2Y_2))$.

   Let $\mu = e(v_1V_1, v_2V_2)$, $\mu_1 = e(x_1X_1, x_2X_2)$, and $\mu_2 = e(y_1Y_1, y_2Y_2)$.

2) T randomly selects $t, t \in \{1, 2, \cdots, s\}$,where $s \in_R Z_p^*$. Considering the efficiency and security, s should be a smaller number. T queries U in random order as follows. U returned the α_i, θ_j, $1 \leqslant i \leqslant 6$, $1 \leqslant j \leqslant 2$.

$$
\begin{aligned}
&U(A + tv_1V_1, B + tv_2V_2)\\
&\quad \to \alpha_1 = e(A + tv_1V_1, B + tv_2V_2),\\
&U(-tA - v_1V_1, v_2V_2)\\
&\quad \to \alpha_2 = e(-tA - v_1V_1, v_2V_2),\\
&U(-v_1V_1, tB + t^2v_2V_2)\\
&\quad \to \alpha_3 = e(-v_1V_1, tB + t^2v_2V_2),\\
&U(C + tv_1V_1, D + tv_2V_2)\\
&\quad \to \alpha_4 = e(C + tv_1V_1, D + tv_2V_2),\\
&U(-tC - v_1V_1, v_2V_2)\\
&\quad \to \alpha_5 = e(-tC - v_1V_1, v_2V_2),\\
&U(-v_1V_1, tD + t^2v_2V_2)\\
&\quad \to \alpha_6 = e(-v_1V_1, tD + t^2v_2V_2),\\
&U(x_1X_1, x_2X_2) \to \theta_1 = e(x_1X_1, x_2X_2),\\
&U(y_1Y_1, y_2Y_2) \to \theta_2 = e(y_1Y_1, y_2Y_2).
\end{aligned}
$$

3) T checks the outputs from U, if $\theta_1 = \mu_1$ and $\theta_2 = \mu_2$, it shows that the of U outputs are correct, otherwise the outputs of U are wrong.

4) T calculates the final results $e(A, B) = \alpha_1\alpha_2\alpha_3\mu$ and $e(C, D) = \alpha_4\alpha_5\alpha_6\mu$.

3.2 SPai: Outsourcing e(A, B), e(A, C) Simultaneously

A large quantity of cryptographic schemes employ symmetric bilinear pairings, namely, $G_1 = G_2 =< P >$. And they often require to calculate $e(A, B)$ and $e(A, C)$. Under such situation, a special outsourcing algorithm **SPai** with much higher efficiency and checkability is put forward in this subsection. The concrete steps are given as follows:

1) T runs ***Rand*** three times to get three random five-tuple: $(V_1, V_2, v_1V_1, v_2V_2, e(v_1V_1, v_2V_2))$, $(X_1, X_2, x_1X_1, x_2X_2, e(x_1X_1, x_2X_2))$, and $(Y_1, Y_2, y_1Y_1, y_2Y_2, e(y_1Y_1, y_2Y_2))$.

   Let $\mu = e(v_1V_1, v_2V_2)$, $\mu_1 = e(x_1X_1, x_2X_2)$, and $\mu_2 = e(y_1Y_1, y_2Y_2)$.

2) T randomly selects t as same as **Pai** algorithm. T queries U in random order as follows. U returned the β_i, χ_j, $1 \leqslant i \leqslant 5$, $1 \leqslant j \leqslant 2$.

$$
\begin{aligned}
&U(A + tv_1V_1, B + tv_2V_2)\\
&\quad \to \beta_1 = e(A + tv_1V_1, B + tv_2V_2),\\
&U(-tA - v_1V_1, v_2V_2)\\
&\quad \to \beta_2 = e(-tA - v_1V_1, v_2V_2),\\
&U(-v_1V_1, tB + t^2v_2V_2)\\
&\quad \to \beta_3 = e(-v_1V_1, tB + t^2v_2V_2),\\
&U(A + tv_1V_1, C + tv_2V_2)\\
&\quad \to \beta_4 = e(A + tv_1V_1, C + tv_2V_2),\\
&U(-v_1V_1, tC + t^2v_2V_2)\\
&\quad \to \beta_5 = e(-v_1V_1, tC + t^2v_2V_2),\\
&U(x_1X_1, x_2X_2) \to \chi_1 = e(x_1X_1, x_2X_2),\\
&U(y_1Y_1, y_2Y_2) \to \chi_2 = e(y_1Y_1, y_2Y_2).
\end{aligned}
$$

3) T checks the outputs from U, if $\chi_1 = \mu_1$ and $\chi_2 = \mu_2$, it shows that the outputs of U are correct, otherwise the outputs of U are wrong.

4) T calculates the final results $e(A, B) = \beta_1\beta_2\beta_3\mu$ and $e(A, C) = \beta_2\beta_4\beta_5\mu$.

4 Security Analysis

4.1 Correctness

If the server honestly performs the algorithm **Pai**, the user T should be able to compute the correct value of the given bilinear pairings $e(A, B)$ and $e(C, D)$ successfully.

Proof.

$$\begin{aligned}
\alpha_1 &= e(A+tv_1V_1, B+tv_2V_2)\\
&= e(A,B)e(A,tv_2V_2)e(tv_1V_1,B)e(tv_1V_1,tv_2V_2)\\
\alpha_2 &= e(-tA-v_1V_1, v_2V_2)\\
&= e(-tA, v_2V_2)e(-v_1V_1, v_2V_2)\\
\alpha_3 &= e(-v_1V_1, tB+t^2v_2V_2)\\
&= e(-v_1V_1, tB)e(-v_1V_1, t^2v_2V_2)\\
\alpha_4 &= e(C+tv_1V_1, D+tv_2V_2)\\
&= e(C,D)e(C,tv_2V_2)e(tv_1V_1,D)e(tv_1V_1,tv_2V_2)\\
\alpha_5 &= e(-tC-v_1V_1, v_2V_2)\\
&= e(-tC, v_2V_2)e(-v_1V_1, v_2V_2)\\
\alpha_6 &= e(-v_1V_1, tD+t^2v_2V_2)\\
&= e(-v_1V_1, tD)e(-v_1V_1, t^2v_2V_2).
\end{aligned}$$

Because $e(aR,bQ) = e(R,Q)^{ab}$ for all $R \in G_1, Q \in G_2$, $a, b \in Z_q^*$. So

$$\begin{aligned}
&\alpha_1\alpha_2\alpha_3\mu\\
=\ & e(A,B)e(A,tv_2V_2)e(tv_1V_1,B)\\
&\cdot e(tv_1V_1,tv_2V_2)e(-tA,v_2V_2)e(-v_1V_1,v_2V_2)\\
&\cdot e(-v_1V_1,tB)e(-v_1V_1,t^2v_2V_2)e(v_1V_1,v_2V_2)\\
=\ & e(A,B)e(A,v_2V_2)^t e(v_1V_1,B)^t e(v_1V_1,v_2V_2)^{t^2}\\
&\cdot e(A,v_2V_2)^{-t} e(v_1V_1,v_2V_2)^{-1} e(v_1V_1,B)^{-t}\\
&\cdot e(v_1V_1,v_2V_2)^{-t^2} e(v_1V_1,v_2V_2)\\
=\ & e(A,B).
\end{aligned}$$

$$\begin{aligned}
&\alpha_4\alpha_5\alpha_6\mu\\
=\ & e(C,D)e(C,tv_2V_2)e(tv_1V_1,D)\\
&\cdot e(tv_1V_1,tv_2V_2)e(-tC,v_2V_2)e(-v_1V_1,v_2V_2)\\
&\cdot e(-v_1V_1,tD)e(-v_1V_1,t^2v_2V_2)e(v_1V_1,v_2V_2)\\
=\ & e(C,D)e(C,v_2V_2)^t e(v_1V_1,D)^t e(v_1V_1,v_2V_2)^{t^2}\\
&\cdot e(C,v_2V_2)^{-t} e(v_1V_1,v_2V_2)^{-1} e(v_1V_1,D)^{-t}\\
&\cdot e(v_1V_1,v_2V_2)^{-t^2} e(v_1V_1,v_2V_2)\\
=\ & e(C,D).
\end{aligned}$$

□

The above equations indicate that the algorithm **Pai** is correct.

Since algorithm **SPai** is a special case of algorithm **Pai**, the correctness proof of algorithm **Pai** is enough to illustrate the correctness of algorithm **SPai**. Therefore, the correctness of **SPai** will not be discussed again because of the limited space.

4.2 Security Proof

Here we will take algorithm **Pai** as example to demonstrate the security of algorithms **Pai** and **SPai**.

Theorem 1. *In the OUP model, the algorithm is an outsource-secure implementation of algorithm **Pai**, where the inputs A, B, C, D may be honest, secret; or honest, protected; or adversarial, protected.*

Proof. Firstly, we prove that **Pair one** $EVIEW_{real} \sim EVIEW_{ideal}$. □

Note that we only consider three types of input (A,B)(as well as (C,D)): honest, secret; honest, protected; or adversarial, protected. If the input (A,B) is anything or other than honest, secret (this means that the input (A,B) is honest, protected or malicious, protected. Obviously, neither types of input (A,B) is secret), then the simulation S_1 is trivial. That is, the simulator S_1 behaves in the same way as in the real execution. Trivially, S_1 never requires to access the secret input (A,B) since neither types of input is secret.

If (A,B) is an honest and secret input, then the simulator S_1 behaves as follows: upon receiving the input on round i, S_1 ignores it, randomly chooses a random five-tuple numbers and submits it to the untrusted server U'. When U' returns the results, S_1 randomly verifies two outputs from U'. If an error is detected, S_1 saves all states and outputs $Y_p^i = "error", Y_p^i = \varphi, rep^i = 1$. If no error is detected, S_1 checks the remaining three outputs. If all checks go through, S_1 outputs $Y_p^i = \varphi, Y_p^i = \varphi, rep^i = 0$; otherwise, S_1 selects a random element r and outputs $Y_p^i = r, Y_p^i = \varphi, rep^i = 0$. In either case, S_1 saves the appropriate states.

The inputs distributed to U' in the real and ideal experiments are computationally indistinguishable. In the ideal experiment, the inputs are uniformly chosen at random. In the real experiment, each part of all queries that T makes is independently re-randomized, where the re-randomization factors are also randomly generated with the naive table-lookup method.

If U' behaves honestly in the ith round, then $EVIEW_{real}^i \sim EVIEW_{ideal}^i$ because T^U perfectly executes **Pai** in the real experiment and S_1 simulates with the same outputs in the ideal experiment.

If U' is dishonest in the ith round, and it has been detected by both T and S_1 (with probability 1/4), then it will produce an error output. In the real experiment, the output of **Pai** looks random to the environment E. In the ideal experiment, S_1 also simulates with a random value $r \in G_T$ as the output. Thus $EVIEW_{real}^i \sim EVIEW_{ideal}^i$, even when U' is dishonest. By the hybrid argument, we conclude that $EVIEW_{real} \sim EVIEW_{ideal}$.

Secondly, we prove **Pair two** $UVIEW_{real} \sim UVIEW_{ideal}$.

The simulator S_2 always behaves as follows: upon receiving the input on the ith round, S_2 ignores it and randomly selects a random five-tuple submits it to the untrusted server U'. Then S_2 saves its states and the states of U'. The environment E can easily distinguish between these real and ideal experiments (note that the output in the ideal experiment is never corrupted). However, E cannot communicate this information with U'. This is because T always re-randomize its inputs to U' in the

ith round of the real experiment. In the ideal experiment, S_2 always generates random, independent queries for U'. Thus, for each *ith* round, we have $UVIEW^i_{real} \sim UVIEW^i_{ideal}$. By the hybrid argument, we conclude that $UVIEW_{real} \sim UVIEW_{ideal}$.

Theorem 2. *In the one-untrusted program (OUP) model, the algorithm (T, U) is an $(O(1/n), 1/4)$ outsource-secure implementation of **Pai**, where n is the bit length of the order q of bilinear groups.*

Proof. The proposed algorithm **Pai** makes three calls to ***Rand*** plus $t^2 + t + 8$ point addition in G_1 or G_2, and 6 multiplication in G_T in order to compute $e(A, B)$ and $e(C, D)$. On one hand , the computation for ***Rand*** is negligible when using the table-lookup method, and a smaller t value can be seen as a point addition. On the other hand, it takes roughly $O(n)$ multiplications finite filed to compute the bilinear pairings. Thus, the algorithms (T, U) are an $O(1/n)$-efficient implementation of **Pai**. If U' fails during any execution of **Pai**, it will be detected with probability 1/4 . □

Theorem 3. *In the one-untrusted program (OUP) model, the algorithm (T, U) is an $(O(1/n), 2/7)$ outsource-secure implementation of **SPai**, where n is same as above.*

Similarly, the security proof of algorithm **SPai** is same as the above proof in essence. Due to the limited space, the proof is omitted here. It is worth mentioning that the high checkability of algorithm **SPai** is attributed to the particularity of the outsourced values.

5 Performance Comparisons

In this section, we compare our algorithms **Pai** and **SPai** with the algorithms in [4,15,28]. As shown in Table 1, let ME denote a modular exponentiation in G_1 or G_2, MI be a modular inverse in G_1 or G_2, MM be a modular multiplication in G_T, PM be a point multiplication in G_1 or G_2 and PA be a point addition in G_1 or G_2. SQT indicates **the number of servers and users' query times**. We omit other operations such as modular additions in Z^*_q which are more lightweight. Note that our algorithms outsource two bilinear pairings at one time, while other algorithms only outsource a bilinear pairing. Therefore, we should comprehensively take into account the above situation and guarantee the fairness of the comparison.

Table 1: Notations

ME	Modular exponentiation
MI	Modular inverse
MM	Modular multiplication
PM	Point multiplication
PA	Point addition
SQT	The number of server and query times

Table 2, Table 3 display the comparison of the efficiency and security properties between our algorithms and the algorithms in [4, 15, 28],respectively. All the algorithms invoke the ***Rand*** subroutine to accelerate the computations, so ***Rand*** can be ignored during the comparison process. For the efficiency comparison, we need to take into account that two bilinear pairings are outsourced by using our algorithms **Pai** and **SPai** and algorithms in [4, 15, 28], respectively. Our algorithms **Pai** and **SPai** are simultaneously outsourcing two bilinear pairings, the algorithms in [4,15,28] can outsource one bilinear pairing at one time. When they outsource two bilinear pairings, they need to be outsourced one by one, that is, their computational overhead need to be multiplied by two. It is obvious that our algorithms have better efficiency than algorithms in [4, 28], and the same efficiency as the algorithm in [15].

From Table 3, we can see the comprehensively performance of our algorithms is better than the other algorithms. Firstly, the efficiency of our algorithms is relatively high since our algorithm requires less computation cost under different security models. Secondly, our algorithms require the minimum query times of user which also can save computational resources. Thirdly, our OUP model hypothesis is the most closest to reality and practical applications. Finally, our algorithms can outsource multiple bilinear pairings simultaneously, and solve the privacy problem with higher checkability in the OUP model. At the same time, **Pai** and **SPai** also reduce the computation and communication cost of users and cloud servers to a certain extent.

6 Conclusions

In this paper, two novel and efficient outsourcing algorithms for multiple bilinear pairings under the OUP security model are put forward. Currently, almost all of the existing outsourcing algorithms for bilinear pairings are based on two servers which occupy large computation resources. Besides, existing outsourcing algorithms can only outsource a bilinear pairing once. When there are multiple bilinear pairings to be outsourced, it has to outsource one by one that is easy to result in inefficiency. To avoid this, we use an untrusted server that is a more practical assumption. To improve the outsourcing efficiency, our scheme allows two bilinear pairings to be outsourced simultaneously with improved checkability and data privacy-preserving. Performance analyses shows that the algorithms **Pai** and **SPai** use fewer resources and query times (economic costs) without decreasing checkability. Hence, our algorithms are comprehensively excellent. The ongoing works focus on how to improve the checkability and realize full verification.

Table 2: Efficiency comparison of the related algorithms

	ME	MI	MM	PM	PA
Algorithm [4] $\times 2$	20	4	12	12	8
Algorithm [28] $\mathbf{A} \times \mathbf{2}$	0	0	6	0	8
Algorithm [28] $\mathbf{B} \times \mathbf{2}$	0	0	$O(logs)$	0	$O(logs)$
Algorithm [15] $\times 2$	0	0	4	0	$O(logs)$
Algorithm **Pai**	0	0	4	0	$O(logs)$
Algorithm **SPai**	0	0	4	0	$O(logs)$

Table 3: Properties comparison of the related algorithms

	SQT	Security model		Checkability
Algorithm [4] $\times 2$	$8U$	OMTUP	(Algorithm [4])^2	1
Algorithm [28] $\mathbf{A} \times \mathbf{2}$	$4U_1$+$8U_2$	TUP	(Algorithm [28]$\mathbf{A}$)^2	1/4
Algorithm [28] $\mathbf{B} \times \mathbf{2}$	$6U_1$+$6U_2$	TUP	(Algorithm [28]$\mathbf{B}$)^2	$(1-\frac{1}{3s})^4$
Algorithm [15] $\times 2$	$10U$	OUP	(Algorithm [15])^2	4/25
Algorithm **Pai**	$8U$	OUP	Algorithm **Pai**	1/4
Algorithm **SPai**	$7U$	OUP	Algorithm **SPai**	2/7

Acknowledgments

This work are partly supported by the National Natural Science Foundation of China under grant 61802243, 61602232, 61572246, the Key R&D Program in industry field of Shaanxi Province under grant 2019GY-013, the Fundamental Research Funds for the Central Universities (GK201803005, GK201903011).

References

[1] A. Ara, M. Al-Rodhaan, T. Yuan, and A. Al-Dhelaan, "A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems," *IEEE Access*, no. 99, pp. 1–1, 2017.

[2] X. F. Chen, J. Li, J. F. Ma, Q. Tang, and W. J. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2386–2396, 2014.

[3] X. F. Chen, J. Li, J. F. Ma, Q. Tang, and W. J. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *European Symposium on Research in Computer Security*, pp. 541–556, Sep. 2012.

[4] X. F. Chen, W. Susilo, J. Li, D. S. Wong, J. F. Ma, S. H. Tang, and Q. Tang, "Efficient algorithms for secure outsourcing of bilinear pairings," *Theoretical Computer Science*, vol. 562, no. C, pp. 112–121, 2015.

[5] B. Chevalliermames, J. S. Coron, N. Mccullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Ifip Wg 8.8/11.2 International Conference on Smart Card Research and Advanced Application*, pp. 24–35, Apr. 2010.

[6] B. Dan, L. Ben, and S. Hovav, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[7] M. Dong, Y. L. Ren, and X. P. Zhang, "Fully verifiable algorithm for secure outsourcing of bilinear pairing in cloud computing," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 7, pp. 3648–3663, 2017.

[8] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers," *Annual Cryptology Conference*, pp. 465-482, 2010.

[9] S. Guo and H. X. Xu, "A secure delegation scheme of large polynomial computation in multi-party cloud," *International Journal of Grid and Utility Computing*, vol. 6, no. 1, pp. 1–7, 2015.

[10] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *International Conference on Theory of Cryptography*, pp. 264–282, June 2005.

[11] W. Hsien, C. Yang and M. S. Hwang, "A survey of public auditing for secure data storage in cloud computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133-142, 2016.

[12] M. S. Hwang, C. C. Lee, T. H. Sun, "Data error locations reported by public auditing in cloud storage service," *Automated Software Engineering*, vol. 21, no. 3, pp. 373–390, Sep. 2014.

[13] M. S. Hwang, S. F. Tzeng, C. S. Tsai, "Generalization of proxy signature based on elliptic curves", *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 73–84, 2004.

[14] S. H. Islam and G. P. Biswas, *A provably secure identity-based strong designated verifier proxy signature scheme from bilinear pairings*. Amsterdam: Elsevier Science Inc., vol. 26, no. 1, pp. 55-67, 2014.

[15] T. J. Jiang and Y. L. Ren, "Secure outsourcing algorithm of bilinear pairings with single server (in chinese)," *Journal of Computer Applications*, vol. 36, no. 07, pp. 1866–1869, 2016.

[16] X. Lin, H. Qu, and X. Zhang, "New efficient and flexible algorithms for secure outsourcing of bilinear pairings," *International Association for Cryptologic Research*, vol. 76, pp. 1–16, 2016.

[17] C. W. Liu, W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing", *International Journal of Network Security*, vol. 18, no. 4, pp. 650–666, 2016.

[18] L. H. Liu and Z. J. Cao, "A note on efficient algorithms for secure outsourcing of bilinear pairings," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 30–36, 2016.

[19] L. H. Liu, Z. J. Cao, C. Mao, and J. B. Wang, "Computational error analysis of two schemes for outsourcing matrix computations," *International Journal of Electronics and Information Engineering*, vol. 7, no. 1, pp. 23–31, 2017.

[20] L. Liu, Z. Cao, C. Mao, "A note on one outsourcing scheme for big data access control in cloud," *International Journal of Electronics and Information Engineering*, vol. 9, no. 1, pp. 29–35, 2018.

[21] M. Manoharan and S. Selvarajan, "An efficient methodology to improve service negotiation in cloud environment," *International Journal of Grid and Utility Computing*, vol. 6, no. 3, pp. 150–158, 2015.

[22] P. Morreale, A. Goncalves, and C.Silva, "Mobile ad hoc network communication for disaster recovery," *International Journal of Space-Based and Situated Computing*, vol. 5, no. 3, pp. 178–186, 2015.

[23] A. Mosa, H. M. El-Bakry, S. M. Abd El-Razek, S. Q. Hasan, "A proposed E-government framework based on cloud service architecture," *International Journal of Electronics and Information Engineering*, vol. 5, no. 2, pp. 93–104, 2016.

[24] L. Oliveira, V. Sucasas, G. Mantas, and J. Rodriguez, "Implementation of a pseudonym-based signature scheme with bilinear pairings on android," in *International Conference on Cognitive Radio Oriented Wireless Networks*, pp. 75–87, Sep. 2017.

[25] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *International Conference on Theory of Cryptography*, pp. 422-439, 2012.

[26] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69-73, 2012.

[27] S. Rezaei, M. Ali Doostari, and M. Bayat, "A lightweight and efficient data sharing scheme for cloud computing," *International Journal of Electronics and Information Engineering*, vol. 9, no. 2, pp. 115–131, 2018.

[28] H. B. Tian, F. G. Zhang, and K. Ren, "Secure bilinear pairing outsourcing made more efficient and flexible," in *ACM Symposium on Information, Computer and Communications Security*, pp. 417-426, 2015. ISBN: 978-1-4503-3245-3

[29] S. F. Tzeng, M. S. Hwang, "Digital signature with message recovery and its variants based on elliptic curve discrete logarithm problem", *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 61–71, Mar. 2004.

[30] G. Varaprasad, S. Murthy, J. Jose, R. J. D'Souza, "Design and development of efficient algorithm for mobile ad hoc networks using cache," *International Journal of Space-Based and Situated Computing*, vol. 1, no. 2/3, pp. 183–188, 2011.

[31] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proceedings of IEEE*, 2011. (`https://ieeexplore.ieee.org/document/5935305`)

[32] Y. J. Wang, Q. H. Wu, D. S.Wong, B. Qin, S. S. M.Chow, Z. Liu, and X. Tan, "Securely outsourcing exponentiations with single untrusted program for cloud storage," in *European Symposium on Research in Computer Security*, pp. 326–343, Sep. 2014.

Biography

Jiaxiang Yang received her B.S. degree from Zhengzhou University, Zhengzhou, China, in 2016. She now is a M.S. degree candidate in Applied Mathematics with the School of Mathematics and Information Science, Shaanxi Normal University, Xi'an, China. Her research interests include secure outsourcing computating protocols and its analysis.

Yanping Li received her M. S. degree from Shaanxi Normal University in 2004 and Ph. D degree from Xidian University in 2009, Xi'an, China. She now is an associate professor with the School of Mathematics and Information Science, Shaanxi Normal University. Her research interests include applied cryptography and its applications.

Yanli Ren is a professor in School of Communication and Information Engineering at Shanghai University, China. She was awarded a M.S. degree in applied mathematics in 2005 from Shaanxi Normal University, China, and a PhD degree in computer science and technology in 2009 from Shanghai Jiaotong University, China. Her research interests include secure outsourcing computing and network security.