Fully Verifiable Algorithm for Secure Outsourcing of Bilinear Pairing in Cloud Computing

Min Dong, Yanli Ren, Xinpeng Zhang
School of Communication and Information Engineering, Shanghai University
Shanghai 200444, China
[e-mail: dongmin3524@i.shu.edu.cn; renyanli@shu.edu.cn, xzhang@shu.edu.cn]
*Corresponding author: Yanli Ren

Received December 1, 2016; revised March 25, 2017; accepted April 17, 2017; published July 31, 2017

Abstract

With the development of cloud computing and widespread availability of mobile devices, outsourcing computation has gotten more and more attention in cloud computing services. The computation of bilinear pairing is the most expensive operation in pair-based cryptographic schemes. Currently, most of the algorithms for outsourcing bilinear pairing have small checkability or the outsourcers need to operate expensive computations. In this paper, we propose an efficient algorithm for outsourcing bilinear pairing with two servers, where the outsourcers can detect the errors with a probability of 1 if the cloud servers are dishonest, and the outsourcers are not involved in any complex computations. Finally, the performance evaluation demonstrates that the proposed algorithm is most efficient in all of fully verifiable outsourcing algorithms for bilinear pairing.

Keywords: Cloud computing, Cryptography, Bilinear pairing, Outsourcing computation, Verifiable computation

The work was supported by the Natural Science Foundation of China (Grant No. 61572309, 61472235, 61525203).

https://doi.org/10.3837/tiis.2017.07.019 **ISSN : 1976-7277**

1. Introduction

With the development of cloud computing and widespread availability of mobile devices, the cloud computing services are gradually coming in our everyday life. Cloud computing is playing a more and more important role in many aspects, and finally it will absolutely change our ways of computing. With the increasing availability of cloud computing services, outsourcing computation has caught much attention from scientific community. By outsourcing computation, resource-limited devices can outsource expensive computation to cloud servers with more powerful computational ability. Though outsourcers can reduce much computational time by outsourcing computation, there are also some security challenges needs we pay attention to. Firstly, the computation tasks always contain sensitive and secret information that should not be exposed to untrusted servers. Therefore, the first challenge is the secrecy of the sensitive inputs and outputs, i.e., untrusted servers should not get anything useful from the process of computing. On the other hand, the cloud servers are not fully trusted and they may return incorrect results, so another problem we should focus on is the checkability of outsourcing computation, i.e., outsourcers should have the ability to detect the error if cloud servers are controlled by adversary and return invalid results. Of course, the process of verifying should not be involved in any expensive computational operations.

Verifiable computation allows outsourcers outsource complex computations to untrusted cloud servers [1, 2]. Firstly, outsourcers operate some simple computations to encrypt the sensitive inputs, and then outsourcers outsource the cryptographic information to untrusted servers. The cloud servers compute the results based on the cryptographic inputs and return them to outsourcers. Finally, outsourcers verify the correctness and get the final outcome by using the results returned from the servers.

The computation of bilinear pairing has been considered as the most expensive operation in pair-based cryptographic schemes, and it is almost incapable for resource limited devices such as RFID tags. The advent of intelligent society makes almost all mobile devices have requirement to connect with other components of the cloud environment. Outsourcing computation can guarantee that these devices are integrated into the network and cloud securely. At the same time, a lot of computational time can be saved for devices those are computationally incapable of carrying out cryptographic algorithms by outsourcing computation to powerful cloud servers.

In this paper, we propose an efficient and fully verifiable algorithm for outsourcing bilinear pairing based on two servers which means the outsourcer can detect the errors those servers may make with probability of 1. By operating the proposed outsourcing algorithm, outsourcer can obtain secure communication channels without exposing of sensitive inputs and outputs when it wants to interact with the cloud environment. Furthermore, the proposed scheme can be applied in cloud to achieve the fully verifiable communication such as proposed algorithm can be used for identity-based signatures (IBS). Meanwhile, one side of the communication can invoke proposed algorithm dynamically which means the proposed one is not only designed for static inputs, i.e., if outsourcer needs to carry out bilinear pairing when receiving the new inputs, the outsourcer can update the inputs and operate proposed scheme to achieve the outsourcing of bilinear pairing.

The rest of this paper is organized as follows. In Section 2, we review the previous work of outsourcing computations. Definitions and security model for secure outsourcing computation are given in Section 3. Then, we propose the algorithm for outsourcing bilinear pairing and

compare proposed algorithm with some previous algorithms in Section 4. In Section 5, we do some experiments to evaluate the efficiency of the proposed algorithm. Finally, we make some conclusions in Section 6.

2. Related Work

In the cryptography, many researchers do the work on outsourcing expensive computations to untrusted servers. Chaum et al. [3] firstly presented the notion of "wallets with observers", i.e., outsourcer's computer installed the secure hardware to operate complex computations. Hohenberger et al. [4] proposed the first formal model and secure-outsourcing algorithm for modular exponentiations based on two untrusted servers. Then Chen et al. [5] proposed an improved algorithm for outsourcing modular exponentiations, and it improved the efficiency and checkability simultaneously. Green et al. [6] proposed the outsourcing algorithm for attribute-based encryption, it reduced the time cost for outsourcers, but outsourcer could not check the correctness of the results in this algorithm.

Bilinear pairing has many applications in many aspects, but it is the most expensive computation in pair-based cryptographic protocols. The computation of bilinear pairing has important influence on the efficiency of the algorithms, so plenty of work has been done to improve the efficiency of computing bilinear pairing [7-10]. Chevalier et al. [11] proposed the first algorithm for outsourcing bilinear pairing with single untrusted cloud server. Outsourcers can detect the error with probability 1 if the server misbehaves in the algorithm, but outsourcers need to do other expensive computations such as scalar multiplications and modular exponentiations. Then Chen et al. [12] presented the first practical algorithm for outsourcing bilinear pairing based on two untrusted servers, the outsourcers were not involved in any complex computations, but the checkability of the algorithm was only $1/2$, i.e., outsourcers also could be cheated by servers with probability $1/2$. Recently, Tian et al. [13] proposed two outsourcing algorithms, and one of them improved the efficiency when keeping the checkability $1/2$, another one improved the checkability to $(1-1/3s)^2$ where s was a small integer, we could see that outsourcers could also be cheated by servers in these two algorithms. Ren et al. [14] proposed an outsourcing algorithm, which had a high checkability, but the outsourcers needed to communicate with the two untrusted servers more than one time to get the results in their algorithm, which appended much more communication cost.

Our contributions. In this paper, we propose an efficient algorithm for outsourcing bilinear pairing based on two untrusted servers. In the algorithm we presented, the outsourcers can detect the errors with probability 1 if the servers misbehave. Compared with the algorithms in [12-14], we improve the checkability to 1 and outsourcers never need to perform any complex computations at the same time.

3. Definitions and Model

3.1 Bilinear Pairing

In this section, we introduce some properties of bilinear pairing [15-18], and give the formal definitions and security model for secure outsourcing computations.

Let q be a large prime, G_1 and G_2 are two cyclic additive groups created by P and Q, and the order of G_1 and G_2 is q. At the same time, we definite G_T to be cyclic multiplicative group of order q. We say $e: G_1 \times G_2 \to G_T$ is a bilinear map with following properties:

- Bilinear: $e(aU, bV) = e(U, V)^{ab}$ for all $a, b \in Z_q^*$, and $U \in G_1, V \in G_2$.
- Non-degenerate: There are $U \in G_1$ and $V \in G_2$ such that $e(U, V) \neq 1$.
- Computable: For all $U \in G_1$ and $V \in G_2$, there is an efficient algorithm to compute $e(U, V)$.

Bilinear pairing can be obtained by Weil and Tate pairing in super- singular elliptic curves over finite fields [15-18].

3.2 Security Definitions

In this section, we introduce security definitions of outsourcing computations [4].

Hohenberger and Lysyanskaya [4] first proposed the formal definitions for secure outsourcing computations, and we use the definitions as [4].

If there is an algorithm *Alg* with two parties, i.e., an honest but resource-limited party T and an untrusted but powerful party U. At the same time, there is a party U' instead of U works maliciously, and records all the computations when it is invoked and T has access to. We uniformly say T outsources some computations to U and (T, U) is an outsource-secure implementation of *Alg*.

Next, we give the definitions of an algorithm with secure outsourcing inputs and outputs.

Definition 1 (Algorithm with Outsource-IO)**.** An *Alg* with outsourcing input/output should take five inputs and generate three outputs. The inputs and outputs are classified by how much the adversary $A = (E, U')$ knows about them, where E is the adversarial environment.

There are five inputs in an algorithm *Alg*. The first one is honest and secret, which is unknown for both E and U; the second one is honest and protected, which may be known by E, but is protected from U; the third one is honest and unprotected, which may be known by both E and U. The other two inputs are generated by the adversarial environment E. The fourth one is adversarial and protected, which is known for E, but protected from U; and the fifth one is adversarial and unprotected, which is known for both E and U. There are three outputs: the first one is secret, which is unknown for both E and U; the second one is protected, which may be known for E, but not U; and the third one is unprotected, which may be known for both E and U.

Definition 2 (Outsource-security). We say a pair of algorithms (T, U) with outsourcing I/O is an outsource-secure implementation of *Alg* if:

Correctness: T^U is a correct implementation of *Alg*.

Security: For all adversaries $A = (E, U')$, there are simulators (S_1, S_2) such that the following pairs of random variables are computationally blind.

Pair One: $EVIEW_{real} \sim EVIEW_{ideal}$, which means that E get nothing useful about the inputs and outputs during the process of T^U.

The adversarial environment E gets the view by participating in the following real process:

$$EVIEW_{real}^i = \{(istate^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x_{ap}^i, x_{au}^i, stop^i) \leftarrow E(1^k, EVIEW_{real}^{i-1}, x_{hp}^i, x_{hu}^i);$$
$$(tstate^i, ustate^i, y_s^i, y_p^i, y_u^i) \leftarrow T^{U'(ustate^{i-1})}$$
$$\times (tstate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i) : (estate^i, y_p^i, y_u^i)\}$$

$EVIEW_{real} = EVIEW_{real}^i$ *if* $stop^i = TRUE$.

The real process proceeds in rounds. In round *i*, the honest (secret, protected, and unprotected) inputs $(x_{hs}^i, x_{hp}^i, x_{hu}^i)$ are selected by honest process *I*, and they cannot be obtained by the adversary environment *E*. Then *E* chooses the values of some variables based on its view from the last round:

- *E* sets the value of $estate_i$ to remember what it will do next time when it is invoked;
- honest inputs $x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}$ based on the inputs $(x_{hs}^i, x_{hp}^i, x_{hu}^i)$ created preciously, and *E* can specify the index j^i of them but not know their values;
- adversarial, protected input x_{ap}^i and unprotected input x_{au}^i ;
- *E* chooses $stop^i$ to determine whether round *i* is the last round in this process.

Next, on receiving the inputs $(tstate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i)$, the algorithm $T^{U'}$ produces a new state $tstate^i$, the secret y_s^i, protected y_p^i and unprotected y_u^i outputs, where $tstate^{i-1}$ is *T's* previous state. U' saves the current state in the variable $ustate^i$ based on the input of $ustate^{i-1}$. Moreover, $(estate^i, y_p^i, y_u^i)$ are the view of the real process in round i. The view in the last round is the overall view of the environment in the real process and $stop^i = TRUE$.

The ideal process:

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

The ideal process also proceeds in rounds. In the ideal process, the stateful simulator S_1 knows nothing about the secret input x_{hs}^i but given the non-secret outputs. There is a *1-bit* variable $replace^i$ that decides whether the real values (y_p^i, y_u^i) outputs will be replaced by some other values (Y_p^i, Y_u^i). In doing so, it is allowed to make queries to oracle U'; moreover, U' saves its state as in the real experiment.

Pair Two: $UVIEW_{real} \sim UVIEW_{ideal}$, i.e., the untrusted software U', which is written by *E* , will not know anything about inputs and outputs.

The untrusted software U' gets the view by participating in the real process which is described in ***Pair One***. $UVIEW_{real} = ustate^i$ if $stop^i = TRUE$.

The ideal process:

$$UVIEW_{ideal}^{i} = \{(istate^{i}, x_{hs}^{i}, x_{hp}^{i}, x_{hu}^{i}) \leftarrow I(1^{k}, istate^{i-1});$$
$$(estate^{i}, j^{i}, x_{ap}^{i}, x_{au}^{i}, stop^{i})$$
$$\leftarrow E(1^{k}, estate^{i-1}, x_{hp}^{i}, x_{hu}^{i}, y_{p}^{i-1}, y_{u}^{i-1});$$
$$(astate^{i}, y_{s}^{i}, y_{p}^{i}, y_{u}^{i})$$
$$\leftarrow Alg(astate^{i-1}, x_{hs}^{j^{i}}, x_{hp}^{j^{i}}, x_{hu}^{j^{i}}, x_{ap}^{i}, x_{au}^{i});$$
$$(sstate^{i}, ustate^{i})$$
$$\leftarrow S_{2}^{U'(ustate^{i-1})} \times (sstate^{i-1}, x_{hu}^{j^{i}}, x_{au}^{i}) : (ustate^{i})\}$$

$UVIEW_{ideal} = UVIEW_{ideal}^{i}$ *if* $stop^{i} = TRUE$.

In the ideal process, another simulator S_2 only knows about the unprotected inputs (x_{hu}^{i}, x_{au}^{i}) and it can make queries to U'. As shown in before, U' may maintain its state.

Definition 3(α-Efficient, Secure Outsourcing). If T^{U} *is* a correct implementation of Alg and the running time of T is no more than an α-multiplicative factor of the running time of *Alg*, we say (T, U) is an α-efficient implementation of Alg.

Definition 4(β -Checkable, Secure Outsourcing). We say that algorithms (T, U) is a β-checkable implementation of *Alg* if it is a correct implementation of *Alg* and if U' work maliciously, *T* will detect the error with probability no less than β.

Definition 5((α, β)-Outsource-Security). Algorithms (T, U) is an (α, β)-outsource-secure implementation of Alg if it is both α-efficient and β-checkable.

3.3 Security Model

In this section, we introduce security model of outsourcing computations [4].

We also introduce the security model of [4] that Hohenberger et al. presented. In their *two untrusted program model*,

- *E* writes the code for two untrusted servers U_1', U_2' and gives it to *T*;
- *T* installs this software in a manner such that all communication between *E* and U_1', U_2' must pass through *T*. We can get the new adversary $A = (E, U_1', U_2')$.

If at most one of the servers works maliciously but we do not know which one, and security means that there is a simulator for both. This is the *one-malicious model* proposed in [4].

4. Outsourcing Bilinear Pairing with Two Untrusted Servers

In this section, we propose an efficient outsourcing algorithm for bilinear pairing based on two untrusted cloud servers. We also analyze security and verifiability of the algorithm, and then compare efficiency of the proposed algorithm with that of previous algorithms.

4.1 The Proposed Outsourcing Algorithm

Similar to [4], we also need a subroutine called *Rand* to accelerate the computations. The inputs for *Rand* is a large prime q , two cyclic additive groups G_1 and G_2 with order q, and bilinear pairing $e: G_1 \times G_2 \rightarrow G_T$. The outputs for each invocation of *Rand* are a random vector shown as follows:

$$a_1P,\ a_2P,\ a_3P,\ a_4P,\ a_5P, a_6P,\ a_7P\ ,$$
$$b_1Q,\ b_2Q,\ b_3Q,\ b_4Q,\ b_5Q,\ b_6Q,\ b_7Q\ ,$$
$$e(a_1P, b_1Q)^{-1}, e(a_2P, (b_4+b_6)Q)^{-1},\ r_i,\ r_i', i \in \{4,5,6,7\},$$
$$e(a_3P, (b_5+b_7)Q)^{-1}, e\big((a_4+a_6)P, b_2Q\big)^{-1}, e((a_5+a_7)P, b_3Q)^{-1}. \quad (1)$$

Where $a_1, \cdots, a_7, b_1, \cdots, b_7 \in Z_q^*$, $r_i, r_i' \in \{-4,-2,-1,1,2,4\}$, P and Q are the generators of G_1 and G_2 respectively; and

$$r_4b_4 = r_6b_6,\ r_5b_5 = r_7b_7;$$
$$r_4'a_4 = r_6'a_6, r_5'a_5 = r_7'a_7;$$
$$\sum_{i=4}^{7} b_i = -b_1;\ \sum_{i=4}^{7} a_i = -a_1. \quad (2)$$

In our secure outsourcing algorithm ***DBP***, T outsources its pairing computation to U_1 and U_2 by invoking *Rand* in one-malicious model. Two untrusted servers cannot know any sensitive information about inputs and outputs of ***DBP***.

Let q be a large prime, the inputs of algorithm ***DBP*** are two random points $A \in G_1$, $B \in G_2$, and the output is $e(A, B)$. Moreover, the inputs A, B and the output $e(A, B)$ are private to U_1 and U_2. The proposed ***DBP*** algorithm is shown as follows.

(1) To implement the outsourcing algorithm, T firstly runs *Rand* once to get a random vector:

$$a_1P,\ a_2P,\ a_3P,\ a_4P,\ a_5P, a_6P,\ a_7P\ ,$$
$$b_1Q,\ b_2Q,\ b_3Q,\ b_4Q,\ b_5Q,\ b_6Q,\ b_7Q\ ,$$
$$e(a_1P, b_1Q)^{-1}, e(a_2P, (b_4+b_6)Q)^{-1},\ r_i,\ r_i', i \in \{4,5,6,7\},$$
$$e(a_3P, (b_5+b_7)Q)^{-1}, e\big((a_4+a_6)P, b_2Q\big)^{-1}, e((a_5+a_7)P, b_3Q)^{-1}.$$

(2) Then T queries U_1 in random order as

$$U_1(A+a_1P, B+b_1Q) \to e(A+a_1P, B+b_1Q) = \theta_{11},$$
$$U_1(A+a_2P, b_4Q) \to e(A+a_2P, b_4Q) = \alpha_{11},$$
$$U_1(A+a_3P, b_5Q) \to e(A+a_3P, b_5Q) = \alpha_{12},$$
$$U_1(a_4P, B+b_2Q) \to e(a_4P, B+b_2Q) = \beta_{11},$$
$$U_1(a_5P, B+b_3Q) \to e(a_5P, B+b_3Q) = \beta_{12}.$$

(3) Similarly, T queries U_2 in random order as

$$U_2(A+a_1P, B+b_1Q) \to e(A+a_1P, B+b_1Q) = \theta_{21},$$
$$U_2(A+a_2P, b_6Q) \to e(A+a_2P, b_6Q) = \alpha_{21},$$
$$U_2(A+a_3P, b_7Q) \to e(A+a_3P, b_7Q) = \alpha_{22},$$
$$U_2(a_6P, B+b_2Q) \to e(a_6P, B+b_2Q) = \beta_{21},$$
$$U_2(a_7P, B+b_3Q) \to e(a_7P, B+b_3Q) = \beta_{22}.$$

(4) Finally, T checks whether U_1 and U_2 create the correct outputs, i.e.,

- $\theta_{11} = \theta_{21}$; (3)

- $\alpha\sim check_j\ (j\in\{1,2\})$
$$\begin{cases} |r_{j+5}|\geq|r_{j+3}|, \begin{cases} \frac{r_{j+5}}{r_{j+3}}>0, \alpha_{1j}=\alpha_{2j}^{\frac{r_{j+5}}{r_{j+3}}} \\ \frac{r_{j+5}}{r_{j+3}}<0, \alpha_{1j}\alpha_{2j}^{-\frac{r_{j+5}}{r_{j+3}}}=1 \end{cases} \\ |r_{j+5}|<|r_{j+3}|, \begin{cases} \frac{r_{j+3}}{r_{j+5}}>0, \alpha_{1j}^{\frac{r_{j+3}}{r_{j+5}}}=\alpha_{2j} \\ \frac{r_{j+3}}{r_{j+5}}<0, \alpha_{1j}^{-\frac{r_{j+3}}{r_{j+5}}}\alpha_{2j}=1 \end{cases} \end{cases}; \quad (4)$$

- $\beta\sim check_j\ (j\in\{1,2\})$
$$\begin{cases} |r'_{j+5}|\geq|r'_{j+3}|, \begin{cases} \frac{r'_{j+5}}{r'_{j+3}}>0, \beta_{1j}=\beta_{2j}^{\frac{r'_{j+5}}{r'_{j+3}}} \\ \frac{r'_{j+5}}{r'_{j+3}}<0, , \beta_{1j}\beta_{2j}^{-\frac{r'_{j+5}}{r'_{j+3}}}=1 \end{cases} \\ |r'_{j+5}|<|r'_{j+3}|, \begin{cases} \frac{r'_{j+3}}{r'_{j+5}}>0, \beta_{1j}^{\frac{r'_{j+3}}{r'_{j+5}}}=\beta_{2j} \\ \frac{r'_{j+3}}{r'_{j+5}}<0, \beta_{1j}^{-\frac{r'_{j+3}}{r'_{j+5}}}\beta_{2j}=1 \end{cases} \end{cases}. \quad (5)$$

If not, T outputs "error"; otherwise, T multiplies the outputs of U_1 and U_2 to obtain that

$$e(A,b_1Q)^{-1}=\alpha_{11}\alpha_{12}\alpha_{21}\alpha_{22}e(a_2P,(b_4+b_6)Q)^{-1}e(a_3P,(b_5+b_7)Q)^{-1}; \quad (6)$$
$$e(a_1P,B)^{-1}=\beta_{11}\beta_{12}\beta_{21}\beta_{22}e((a_4+a_6)P,b_2Q)^{-1}e((a_5+a_7)P,b_3Q)^{-1}. \quad (7)$$

Then we can get:
$$e(A,B)=\theta_{11}\ e(a_1P,b_1Q)^{-1}e(A,b_1Q)^{-1}e(a_1P,B)^{-1}. \quad (8)$$

Correctness. We demonstrate the correctness of equations (3)-(8) here. On receiving the results returned from the servers, we can use the properties of bilinear pairing, as introduced in section 3.1, to prove the correctness of equations (3)-(8).

Equation (3): From step 2 and 3 in the algorithm, it is obvious that equation (3) holds.

Equations (4)-(5): There are some relationship between r_i and r'_i, where $i\in\{4,5,6,7\}$, and $r_i,r'_i\in\{-4,-2,-1,1,2,4\}$. For example, if $j=1$, $|r_6|\geq|r_4|$ and $\frac{r_6}{r_4}>0$, from step 2 and 3, we can get $\alpha_{11}=e(A+a_2P,b_4Q)$ and $\alpha_{21}=e(A+a_2P,b_6Q)$. Meanwhile, from equation (2), $r_4b_4=r_6b_6$ so that we can obtain $\alpha_{11}=\alpha_{21}^{\frac{r_6}{r_4}}$. Similarly, we can get that equations (4)-(5) both hold.

Equation (6):
$$\alpha_{11}\alpha_{12}\alpha_{21}\alpha_{22}e(a_2P,(b_4+b_6)Q)^{-1}e(a_3P,(b_5+b_7)Q)^{-1}$$
$$=e(A+a_2P,b_4Q)e(A+a_3P,b_5Q)e(A+a_2P,b_6Q)e(A+a_3P,b_7Q)$$
$$\cdot e(a_2P,(b_4+b_6)Q)^{-1}e(a_3P,(b_5+b_7)Q)^{-1}$$

$$= e(A + a_2P, b_4Q + b_6Q)e(A + a_3P, b_5Q + b_7Q)e(a_2P, (b_4 + b_6)Q)^{-1}e(a_3P, (b_5 + b_7)Q)^{-1}$$
$$= e(A + a_2P, b_4Q + b_6Q)e(A + a_3P, b_5Q + b_7Q)e(-a_2P, b_4Q + b_6Q)e(-a_3P, b_5Q + b_7Q)$$
$$= e(A + a_2P, b_4Q + b_6Q)e(-a_2P, b_4Q + b_6Q)e(A + a_3P, b_5Q + b_7Q)e(-a_3P, b_5Q + b_7Q)$$
$$= e(A, b_4Q + b_6Q)e(A, b_5Q + b_7Q) = e(A, b_4Q + b_5Q + b_6Q + b_7Q) \quad (9)$$

From equation (2), we know that $\sum_{i=4}^{7} b_i = -b_1$, so

$$\alpha_{11}\alpha_{12}\alpha_{21}\alpha_{22}e(a_2P, (b_4 + b_6)Q)^{-1}e(a_3P, (b_5 + b_7)Q)^{-1}$$
$$= e(A, b_4Q + b_5Q + b_6Q + b_7Q) = e(A, -b_1Q) = e(A, b_1Q)^{-1} \quad (10)$$

i.e., equation (6) is correct.

Equation (7):

$$\beta_{11}\beta_{12}\beta_{21}\beta_{22}e((a_4 + a_6)P, b_2Q)^{-1}e((a_5 + a_7)P, b_3Q)^{-1}$$
$$= e(a_4P, B + b_2Q)e(a_5P, B + b_3Q)e(a_6P, B + b_2Q)e(a_7P, B + b_3Q)$$
$$\cdot e((a_4 + a_6)P, b_2Q)^{-1}e((a_5 + a_7)P, b_3Q)^{-1}$$
$$= e(a_4P + a_6P, B + b_2Q)e(a_5P + a_7P, B + b_3Q)e\big((a_4 + a_6)P, b_2Q\big)^{-1}e\big((a_5 + a_7)P, b_3Q\big)^{-1}$$
$$= e(a_4P + a_6P, B + b_2Q)e(a_5P + a_7P, B + b_3Q)e(a_4P + a_6P, -b_2Q)e(a_5P + a_7P, -b_3Q)$$
$$= e(a_4P + a_6P, B)e(a_5P + a_7P, B) = e(a_4P + a_5P + a_6P + a_7P, B) \quad (11)$$

Similarly, we know that $\sum_{i=4}^{7} a_i = -a_1$ from equation (2), so

$$\beta_{11}\beta_{12}\beta_{21}\beta_{22}e\big((a_4 + a_6)P, b_2Q\big)^{-1}e\big((a_5 + a_7)P, b_3Q\big)^{-1}$$
$$= e(a_4P + a_5P + a_6P + a_7P, B) = e(-a_1P, B) = e(a_1P, B)^{-1} \quad (12)$$

i.e., equation (7) is also correct.

Equation (8):

From equations (6) and (7), we can get the value of $e(A, b_1Q)^{-1}$ and $e(a_1P, B)^{-1}$.

$$\theta_{11}\, e(a_1P, b_1Q)^{-1}e(A, b_1Q)^{-1}e(a_1P, B)^{-1}$$
$$= e(A + a_1P, B + b_1Q)\, e(a_1P, b_1Q)^{-1}e(A, b_1Q)^{-1}e(a_1P, B)^{-1}$$
$$= e(A, B)e(A, b_1Q)e(a_1P, B)e(a_1P, b_1Q)e(a_1P, -b_1Q)e(A, -b_1Q)e(-a_1P, B)$$
$$= e(A, B)e(A, b_1Q)e(A, -b_1Q)e(a_1P, B)e(-a_1P, B)e(a_1P, b_1Q)e(a_1P, -b_1Q) = e(A, B) \quad (13)$$

So we conclude that the proposed algorithm is correct based on the above analysis.

Note: We first give an informal explanation that the inputs and outputs of the ***DBP*** algorithm, i.e. $A, B, e(A, B)$ are private for the servers. The formal proof will be shown later.

In the proposed ***DBP*** algorithm, we can see that there are some relationship between b_4, b_5, a_4, a_5 and b_6, b_7, a_6, a_7, i.e., $r_4b_4 = r_6b_6$, $r_5b_5 = r_7b_7$, $r_4'a_4 = r_6'a_6$, $r_5'a_5 = r_7'a_7$, where $r_i, r_i' \in \{-4, -2, -1, 1, 2, 4\}$.

In the one-malicious model, we suppose that U_1 is dishonest, i.e., U_1 may want to guess $A, B, e(A, B)$. Because U_1 can get b_4Q, b_5Q, a_4P, a_5P with some probability, in order to guess A and B, it needs to use the relationship between b_4, b_5, a_4, a_5 and b_6, b_7, a_6, a_7. Since that r_i, r_i' are chosen randomly in $\{-4, -2, -1, 1, 2, 4\}$, so

b_6Q may be $\{\pm\frac{1}{4}; \pm\frac{1}{2}; \pm1; \pm2; \pm4\}b_4Q$; b_7Q may be $\{\pm\frac{1}{4}; \pm\frac{1}{2}; \pm1; \pm2; \pm4\}b_5Q$.

Similarly, a_6P may be $\{\pm\frac{1}{4}; \pm\frac{1}{2}; \pm1; \pm2; \pm4\}a_4P$; a_7P may be $\{\pm\frac{1}{4}; \pm\frac{1}{2}; \pm1; \pm2; \pm4\}a_5P$.

Thus, U_1 may obtain $b_6Q(\, b_7Q, a_6P, a_7P)$ from $b_4Q(b_5Q, a_4P, a_5P)$ with a probability of 1/10. If U_1 wants to get A and B, the probability of getting correct b_6Q, b_7Q, a_6P, a_7P simultaneously from b_4Q, b_5Q, a_4P, a_5P is 10^{-4}. On the other hand, U_1 must ensure α_{11}, α_{12} , β_{11} , β_{12}, and then get b_4Q, b_5Q, a_4P, a_5P, and the probability is $\frac{1}{A_5^4} = \frac{1}{120}$.

From the theoretical analysis above, we can get that if an untrusted server tries to guess both A and B randomly, and the probability is $10^{-4} \cdot \frac{1}{120} = \frac{1}{12} \cdot 10^{-5}$, which is a small number and it is near to 0. Most important, there is no way for U_1 to check whether the guess is correct. In addition, $e(A, B)$ is private for U_1 because of the randomness of *Rand* subroutine. Therefore, the true inputs and outputs of the ***DBP*** algorithm, i.e. $A, B, e(A, B)$ are all private for the servers.

4.2 An Example of the Proposed Algorithm

Firstly, the outsourcer calls *Rand* once to get a random vector:

$$\begin{gathered} a_1P,\ a_2P,\ a_3P,\ a_4P,\ a_5P,\ a_6P,\ a_7P\ , \\ b_1Q,\ b_2Q,\ b_3Q,\ b_4Q,\ b_5Q,\ b_6Q,\ b_7Q\ , \\ e(a_1P, b_1Q)^{-1}, e(a_2P, (b_4 + b_6)Q)^{-1}, \\ r_4 = 1,\ r_5 = -2,\ r_6 = -1,\ r_7 = 4,\ r_4' = -2,\ r_5' = -2,\ r_6' = -1,\ r_7' = 2, \\ e(a_3P, (b_5 + b_7)Q)^{-1}, e\big((a_4 + a_6)P, b_2Q\big)^{-1}, e((a_5 + a_7)P, b_3Q)^{-1}. \end{gathered} \tag{14}$$

Then T queries U_1 and U_2 in random order to get ten computational results:

$$\begin{gathered} U_1(A + a_1P, B + b_1Q) \to e(A + a_1P, B + b_1Q) = \theta_{11}, \\ U_1(A + a_2P, b_4Q) \to e(A + a_2P, b_4Q) = \alpha_{11}, \\ U_1(A + a_3P, b_5Q) \to e(A + a_3P, b_5Q) = \alpha_{12}, \\ U_1(a_4P, B + b_2Q) \to e(a_4P, B + b_2Q) = \beta_{11}, \\ U_1(a_5P, B + b_3Q) \to e(a_5P, B + b_3Q) = \beta_{12}. \end{gathered}$$

$$\begin{gathered} U_2(A + a_1P, B + b_1Q) \to e(A + a_1P, B + b_1Q) = \theta_{21}, \\ U_2(A + a_2P, b_6Q) \to e(A + a_2P, b_6Q) = \alpha_{21}, \\ U_2(A + a_3P, b_7Q) \to e(A + a_3P, b_7Q) = \alpha_{22}, \\ U_2(a_6P, B + b_2Q) \to e(a_6P, B + b_2Q) = \beta_{21}, \\ U_2(a_7P, B + b_3Q) \to e(a_7P, B + b_3Q) = \beta_{22}. \end{gathered}$$

Next, outsourcer checks the results those returned from the untrusted servers.

- $\theta_{11} = \theta_{21}$ (15)
- $\alpha{\sim}check_1$
  Because $r_4 = 1, r_6 = -1$, i.e., $|r_6| \geq |r_4|$ and $\frac{r_6}{r_4} < 0$. From equation (4), the outsourcer checks whether
  $$\alpha_{11}\alpha_{21} = 1. \tag{16}$$
- $\alpha{\sim}check_2$
  Similarly, $r_5 = -2, r_7 = 4$, i.e., $|r_7| \geq |r_5|$ and $\frac{r_7}{r_5} < 0$. From equation (4), we can also get that the outsourcer checks whether
  $$\alpha_{12}\alpha_{22}^2 = 1. \tag{17}$$
- $\beta{\sim}check_1$
  Because $r_4' = -2, r_6' = -1$, i.e., $|r_6'| < |r_4'|$ and $\frac{r_6'}{r_4'} > 0$. From equation (5), the outsourcer checks whether
  $$\beta_{11}^2 = \beta_{21}. \tag{18}$$

- $\beta \sim check_2$

  We can also get $r_5' = -2, r_7' = 2$, i.e., $|r_7'| \geq |r_5'|$ and $\frac{r_7'}{r_5'} < 0$. From equation (5), the outsourcer checks whether

$$\beta_{12}\beta_{22} = 1. \tag{19}$$

  If equations (15)-(19) all hold, the outsourcer can compute the final result:

$$e(A,B) = \theta_{11}\, e(a_1P, b_1Q)^{-1} e(A, b_1Q)^{-1} e(a_1P, B)^{-1} = e(A,B),$$

  where

$$e(A, b_1Q)^{-1} = \alpha_{11}\alpha_{12}\alpha_{21}\alpha_{22} e(a_2P, (b_4 + b_6)Q)^{-1} e(a_3P, (b_5 + b_7)Q)^{-1};$$
$$e(a_1P, B)^{-1} = \beta_{11}\beta_{12}\beta_{21}\beta_{22} e((a_4 + a_6)P, b_2Q)^{-1} e((a_5 + a_7)P, b_3Q)^{-1}.$$

4.3 An Application of the Proposed Algorithm

In this section, we use the proposed ***DBP*** algorithm to outsource the identity-based signature scheme [19].

Setup. Let G be a group of prime order q. The system chooses a generator P of G, picks a random number s and sets $P_{pub} = sP$. There are two hash functions $H_1: \{0,1\}^* \times G \to Z_q$ and $H_2: \{0,1\}^* \to G$, the master key is s and the system parameter is (P, P_{pub}, H_1, H_2).

Extract. On receiving the identity ID, the private key generator (PKG) computes to get a private key $D_{ID} = sH_2(ID)$, and $Q_{ID} = H_2(ID)$ is a public key.

Sign. On receiving the key D_{ID} and the message m, the signer picks a random number r and computes:

$$U = rQ_{ID}, h = H_1(m, U), V = (r + h)D_{ID},$$

Finally, signer outputs the signature $\sigma = (U, V)$.

Verify. When the verifier receives the message m, identity ID and the signature σ, it operates and computes:

$$h = H_1(m, U), U + hQ_{ID}, \boldsymbol{DBP}(P, V) \to e(P, V),$$
$$\boldsymbol{DBP}(P_{pub}, U + hQ_{ID}) \to e(P_{pub}, U + hQ_{ID}),$$

Finally, it checks whether $\boldsymbol{DBP}(P, V) = \boldsymbol{DBP}(P_{pub}, U + hQ_{ID})$.

Note: The outsourcer only needs to operate 1 point addition by invoking the proposed ***DBP*** algorithm for achieving the identity-based signature in cloud, thus computational cost for outsourcer can be greatly reduced.

4.4 Security Analysis

Theorem 1. In the one-malicious model, if the input (A, B) is honest, secret; or honest, protected; or adversarial, protected, the algorithm $\left(T, (U_1, U_2)\right)$ is an outsourcing-secure implementation of our algorithm ***DBP***.

Proof. The proof is similar to [4, 12]. Let $A = (E, U_1', U_2')$ be an adversary that interacts with algorithm in one-malicious model.

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

If the input (A, B) is honest and secret, the behavior of simulator S_1 will be same as that in real execution. The simulator S_1 will behave as follows. On receiving input in the round i, S_1 will ignore it and instead make five random queries like (P_j, Q_j) to U_1' and U_2'. Then simulator S_1 tests all outputs from servers and sets the values on the basis of the results returned from the servers. If S_1 detects an error, simulator S_1 saves its states and outputs $Y_p^i = "error"$, $Y_u^i = \emptyset$, $replace^i = 1$. If no error is detected, S_1 outputs $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, $replace^i = 0$; otherwise,

S_1 selects a random element r and outputs $Y_p^i = r$, $Y_u^i = \emptyset$ and makes $replace^i = 1$. In either case, S_1 also saves the appropriate states.

The inputs are chosen randomly and all queries those T makes to the servers are computationally random in ideal and real experiment respectively. So we say that the input given to (U_1', U_2') in the ideal and real experiments are computationally blind. If (U_1', U_2') work honestly in round *i* in the real experiment, then $T^{(U_1'U_2')}$ perfectly executes the algorithm in the real experiment and S_1 chooses not to replace the outputs, so we can easily obtain that $EVIEW_{real} \sim EVIEW_{ideal}$. If one of (U_1', U_2') is dishonest in the round *i*, then all errors those servers made will be detected by T and S_1, and resulting in an output of "error". In the real experiment, the nine outputs generated by (U_1', U_2') are multiplied together along with five random values. Thus, even one of (U_1', U_2') behaves dishonestly, we also can get that $EVIEW_{real} \sim EVIEW_{ideal}$.

Pair Two: $UVIEW_{real} \sim UVIEW_{ideal}$

The simulator S_2 behaves as follows. On receiving input in the round *i*, S_2 will ignore it and make five random queries to both U_1' and U_2'. After that S_2 saves its own states and the states of (U_1', U_2'). These differences between ideal and real experiments can be easily distinguished by *E*. However, *E* cannot communicate this information with (U_1', U_2'). Because in the *i*-th round of real experiment, *T* always makes its inputs look random to (U_1', U_2'); and in ideal experiment, S_2 always creates random and independent queries for (U_1', U_2'). Thus, for each round we always have $UVIEW_{real} \sim UVIEW_{ideal}$.

Theorem 2. In the one-malicious model, the algorithm $\left(T, (U_1, U_2)\right)$ is an $(O(\frac{1}{n}),1)$-outsourcing-secure implementation of our algorithm, where *n* is the bit length of *q*.

Proof. In the proposed algorithm ***DBP***, *T* makes 1 call to *Rand* plus 6 point additions and 19 multiplications to get $e(A, B)$. Moreover, it is well known that it takes $O(n)$ multiplications in finite filed to compute bilinear pairing [12]. Thus, our algorithm $\left(T, (U_1, U_2)\right)$ is an $O(\frac{1}{n})$-efficient implementation. On the other hand, all the queries those T makes to the untrusted servers can be verified successfully in proposed algorithm. So errors will be detected with probability 1 if one of (U_1, U_2) is dishonest.

4.5 Comparison

Table 1 presents the comparison among different outsourcing algorithms of bilinear pairing. In the **Table 1**, *s* is a small integer, and we denote by PA a point addition in G_1 or G_2, by M a multiplication in G_T, by MExp a modular exponentiation in G_T, by SM a scalar multiplication in G_1 or G_2, by Pair a bilinear pairing.

Compared with algorithm in [11], the proposed ***DBP*** algorithm is more efficient with the same checkability. On the other hand, the outsourcer needs to operate 10MExp plus 6SM in algorithm BJN [11], and the computational cost of outsourcer is almost equivalent to compute bilinear pairing directly. Compared with algorithms Pair [12] and TZR1 [13], the ***DBP*** algorithm improves the checkability to 1 though a little computational cost is appended. In TZR2 [13] and VBP algorithm [14], the outsourcer can check the error with a probability of 1 or close to 1. The proposed ***DBP*** algorithm is also fully verifiable and improves the efficiency of the outsourcer when comparing with TZR2 [13] and VBP [14]. Thus, the ***DBP*** algorithm is most efficient in all of the fully verifiable outsourcing algorithms for bilinear pairing.

Table 1. Comparison among different algorithms

Algorithm	BJN[11]	Pair[12]	TZR1[13]	TZR2[13]	VBP[14]	DBP
PA(T)	4	5	4	$O(log\ s)$	8	6
M(T)	6	4	3	$O(log\ s)$	14	19
MExp(T)	10	0	0	0	0	0
SM(T)	6	0	0	0	0	0
Pair(U)	4	8	6	6	6	10
MExp(U)	0	0	0	0	4	0
Servers	1	2	2	2	2	2
Checkability	1	$1/2$	$1/2$	$(1-\frac{1}{3s})^2$	1	1
Communication Rounds	4	2	2	2	4	2

5. Performance Evaluation

In this section, we do some experiments to evaluate the efficiency of the proposed algorithm for outsourcing single bilinear pairing to cloud servers. The parameters q is a random 160-bit prime. The programming language is JAVA and we use the database of Java Pairing-Based Cryptography (JPBC).

In the experiments, the cloud servers are simulated by a machine with 32 Intel Xeon CPUs running at 2.6GHz and 16G RAM. At the same time, the outsourcer is simulated by a computer with Intel core i7 CPU running at 2.6GHz and 4G memory.

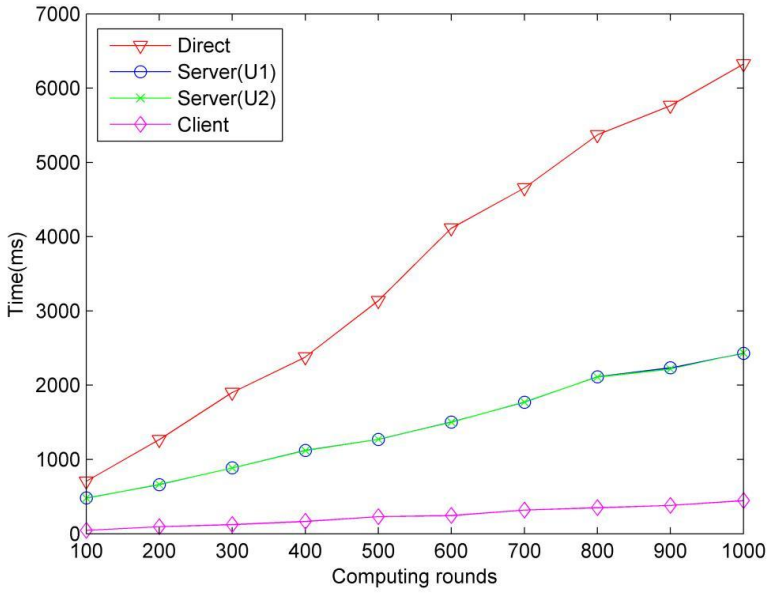


Fig. 1. Simulation for ***DBP*** algorithm

In **Fig. 1**, we give the simulation for the proposed algorithm ***DBP***, the outsourcer can detect the error with a probability of 1 if the servers misbehave. It is obvious that the computation cost for outsourcer is much smaller than that of computing the bilinear pairing directly because a lot of computation has been outsourced to the cloud servers. So we can say that the proposed algorithm ***DBP*** is an efficient outsourcing algorithm with full verifiability.

Then in **Table 2** and **Fig. 2**, we compare the computational time of the outsourcer in ***DBP*** algorithm with that of the algorithms in [11-14].

Table 2. Efficiency comparison among different outsourcing algorithms

Rounds	Computing time of T (ms)					
	BJN[11]	Pair[12]	TZR1[13]	TZR2[13]	VBP[14]	DBP
100	7494	29	21	91	50	45
200	14975	59	43	184	100	95
300	20856	90	70	300	141	122
400	27803	121	93	357	184	166
500	36786	144	112	483	245	229
600	40857	175	130	584	272	245
700	51548	206	163	675	344	319
800	56924	233	175	735	379	350
900	63198	248	184	869	420	381
1000	72041	286	223	953	478	445

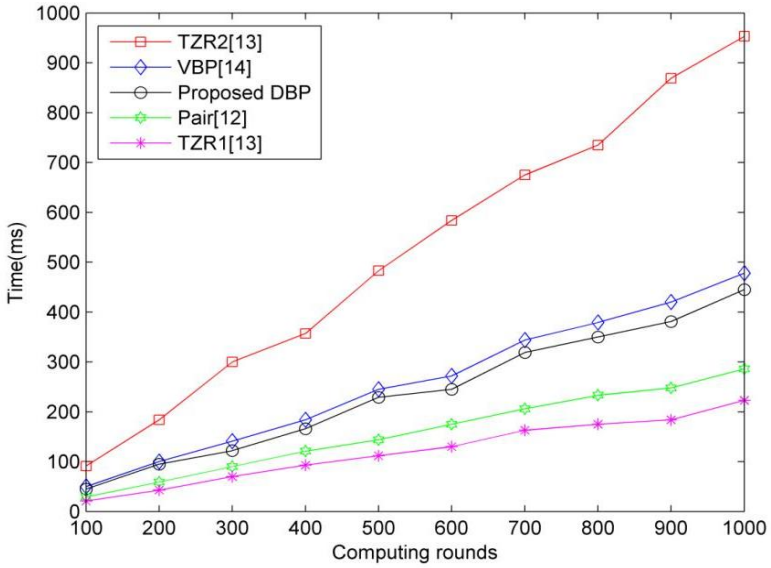


Fig. 2. Efficiency comparison among different algorithms

From **Table 2**, it is obvious that other mentioned algorithms are all more efficient than BJN [11], because the outsourcer needs to carry out modular exponentiation and scalar multiplication for computing bilinear pairing in BJN [11]. From **Fig. 2**, we conclude that the proposed algorithm ***DBP*** is more efficient than TZR2 [13] and VBP [14]. Meanwhile, the outsourcer only needs to interact with the servers once and can obtain fully verifiable results in proposed one. Compared with Pair [12] and TZR1 [13], we improve the checkability to 1 though a little computation cost is appended, so that all potentially invalid results returned from the servers can be detected by the outsourcer, where the checkability is only $1/2$ in Pair [12] and TZR1 [13]. Therefore, the proposed algorithm ***DBP*** is an efficient and fully verifiable outsourcing algorithm for bilinear pairing in the one-malicious model.

6. Conclusion

In this paper, we propose a fully verifiable and efficient algorithm for outsourcing bilinear pairing based on two untrusted cloud servers. Based on the two untrusted servers, the proposed algorithm ***DBP*** keeps both inputs and outputs secret, and all outsourcing errors can be detected by the outsourcer with a probability of 1. Compared with the previous algorithms, the proposed algorithm ***DBP*** improves the checkability and efficiency simultaneously.

The proposed algorithm is constructed by using two non-colluding servers, which may be impractical in current cloud environment. In addition, the outsourcing algorithm for bilinear pairing with single server is not efficient such as [11]. Therefore, the future work is how to outsource bilinear pairing efficiently based on single untrusted cloud server.

References

[1] R. Gennaro, C. Gentry, B. Parno, "Non-interactive verifiable computing: outsourcing computation to untrusted workers," in *Proc. of the 30th Annual Conference on Advances in Cryptology*, pp.465–482, August 15-19, 2010. Article (CrossRef Link)

[2] K. Chung, Y. Kalai, S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. of the 30th Annual Conference on Advances in Cryptology*, pp.483–501, August 15-19, 2010. Article (CrossRef Link)

[3] D. Chaum, T. Pedersen, "Wallet databases with observers," in *Proc. of 12th Annual Conference on Advances in Cryptology*, pp.89–105, August 16–20, 1992. Article (CrossRef Link)

[4] S. Hohenberger, A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. of the 2nd International Conference on Theory of Cryptography*, pp.264–282, February 10-12, 2005. Article (CrossRef Link)

[5] X. Chen , J. Li, J. Ma et al., "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans Parall Distrib Syst*, vol. 25, no.9, pp.2386–2396, 2014. Article (CrossRef Link)

[6] M. Green, S. Hohenberger, B Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. of the 20th USENIX Conference on Security*, pp.34, August 08 - 12, 2011.

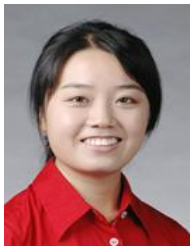
[7] P. Barreto, S. Galbraith, C. Heigeartaigh et al., "Efficient pairing computation on supersingular Abelian varieties," *Design Code Cryptogr*, vol.42, no.3, pp.239–271, 2007. Article (CrossRef Link)

[8] J. Beuchat, J. Gonzalez, S. Mitsunari et al., "High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves," in *Proc. of the 4th International Conference on Pairing-Based Cryptography*, pp.21–39, December, 2010. Article (CrossRef Link)

[9] F. Hess, N. Smart, F. Vercauteren, "The Eta pairing revisited," *IEEE Trans Inf Theory*, vol.52, no.10, pp. 4595–4602, 2006. Article (CrossRef Link)

[10] M. Scott, N. Costigan, W. Abdulwahab, "Implementing cryptographic pairing on smartcards," in *Proc. of the 8th International Conference on Cryptographic Hardware and Embedded Systems*, pp.134–147, October 10-13, 2006. Article (CrossRef Link)
[11] B. Chevallier-Mames, J. Coron, N. McCullagh et al., "Secure delegation of elliptic-curve pairing," in *Proc. of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application*, pp.24–35, April 14-16, 2010. Article (CrossRef Link)
[12] X. Chen, W. Susilo, J. Li et al., "Efficient algorithms for secure outsourcing of bilinear pairing," *Theor Comput Sci*, vol.562, pp.112–121, 2015. Article (CrossRef Link)
[13] H. Tian, F. Zhang, K. Ren, "Secure bilinear pairing outsourcing made more efficient and flexible," in *Proc. of the 10th ACM Symposium on Information, Computer and Communications Security*, pp.417–426, April 14-17, 2015. Article (CrossRef Link)
[14] Y. Ren, N. Ding, T. Wang et al., "New algorithms for verifiable outsourcing of bilinear pairing," *Science China Information Sciences*, vol.59:099103, 2016. Article (CrossRef Link)
[15] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp.213–229, August 19–23, 2001. Article (CrossRef Link)
[16] D. Boneh, H. Shacham, "Short signatures from the Weil pairing," in *Proc. of Advances in Cryptology-Crypto 2001*, pp.514–532, December 9–13, 2001. Article (CrossRef Link)
[17] E. Yoon, W. Lee, K. Yoo, "Secure Remote User Authentication Scheme Using Bilinear Pairings," in *Proc. of 1st International Workshop on Information Security Theory and Practice*, pp.102–114, May 9-11, 2007. Article (CrossRef Link)
[18] S. Galbraith, K. Paterson, N. Smart, "Pairing for cryptographers," *Discrete Appl. Math*, vol.156, no.16, pp. 3113–3121, 2008. Article (CrossRef Link)
[19] J. Cha, J.H. Cheon, "An identity-based signature from gap Diffie–Hellman groups," in *Proc. of Public Key Cryptography-PKC 2003*, pp.18–30, January 6–8, 2003. Article (CrossRef Link)

Min Dong received the BS degree on communication engineering from Shanghai University, Shanghai, China, in 2015. Currently, he is a graduate student at the School of Communication and Information Engineering in Shanghai University, Shanghai, China. His research interests include applied cryptography and secure outsourcing computing



Yanli Ren is an associate 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 Jiao Tong University, China. Her research interests include applied cryptography, secure outsourcing computing, and network security.

Xinpeng Zhang received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.S. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. He was with the State University of New York at Binghamton as a visiting scholar from January 2010 to January 2011, and Konstanz University as an experienced researcher sponsored by the Alexander von Humboldt Foundation from March 2011 to May 2012. His research interests include multimedia security, image processing, and digital forensics. He has published more than 170 papers in these areas.