Hindawi
Security and Communication Networks
Volume 2017, Article ID 4892814, 9 pages
https://doi.org/10.1155/2017/4892814

Research Article

Noninteractive Verifiable Outsourcing Algorithm for Bilinear Pairing with Improved Checkability

Yanli Ren,[1] Min Dong,[1] Zhihua Niu,[2] and Xiaoni Du[3]

[1] *School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China*
[2] *School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China*
[3] *College of Mathematics and Information Science, Northwest Normal University, Lanzhou 730070, China*

Correspondence should be addressed to Yanli Ren; renyanli@shu.edu.cn

Received 19 June 2017; Revised 27 August 2017; Accepted 6 September 2017; Published 15 October 2017

Academic Editor: Rémi Cogranne

It is well known that the computation of bilinear pairing is the most expensive operation in pairing-based cryptography. In this paper, we propose a noninteractive verifiable outsourcing algorithm of bilinear pairing based on two servers in the one-malicious model. The outsourcer need not execute any expensive operation, such as scalar multiplication and modular exponentiation. Moreover, the outsourcer could detect any failure with a probability close to 1 if one of the servers misbehaves. Therefore, the proposed algorithm improves checkability and decreases communication cost compared with the previous ones. Finally, we utilize the proposed algorithm as a subroutine to achieve an anonymous identity-based encryption (AIBE) scheme with outsourced decryption and an identity-based signature (IBS) scheme with outsourced verification.

1. Introduction

Outsourcing computation has received widespread attention with the development of cloud computing and the proliferation of mobile devices [1]. Despite of the huge benefits, it also encounters some security concerns and challenges. Firstly, the computation tasks often include some private information that should not be disclosed to the cloud servers, since the servers are not fully trusted. Secondly, the cloud servers may return an invalid result, but the outsourcer fails to detect the error [1]. Therefore, two main security challenges of the outsourcing computation are privacy and checkability: (1) the cloud servers cannot learn anything about the private inputs and the outputs of the computation outsourced to them; (2) the outsourcer can detect any failure if the cloud servers return a wrong computation result.

Verifiable computation (VC) allows a client with limited computation capability to outsource evaluation of a function on some inputs to a powerful but semitrusted server [2, 3]. The client in this model first executes a lot of off-line computation and encrypts the function which will be evaluated and then sends the encrypted function to the server. The server then performs the computation on the encoded function and responds with a result and a proof that the result is correct. Finally, the client verifies whether the computation has been carried out honestly based on the server's proof. During the whole process, the computation cost of the client is less than computing the function directly itself.

Our Contributions. In this paper, we propose a noninteractive verifiable outsourcing algorithm of bilinear pairing in the one-malicious model of two untrusted servers, which improves the checkability of the outsourcer without any interactive operation between the outsourcer and the server. In the proposed algorithm, the outsourcer could detect any failure with a probability close to 1 if one of the servers returns the false result. The proposed algorithm improves the checkability at the expense of only a little efficiency when compared with previous algorithms. Finally, we utilize the proposed algorithm as a subroutine to achieve an AIBE scheme with outsourced decryption and an IBS scheme with outsourced verification.

1.1. Related Works. In the cryptographic community, outsourcing expensive operations to a semitrusted device is widely studied. Chaum and Pedersen [4] introduced the concept of "wallets with observers" that allows installing a piece of hardware on the client's device to execute some operations for each transaction. Hohenberger and Lysyanskaya formalized this model [5] and presented algorithms for the computation of modular exponentiations (MExps) based on two noncolluding servers. Further, Chen et al. [1] proposed a new outsourcing algorithm for MExps with improved efficiency and checkability based on the same model as [5]. However, it is still possible for the outsourcer to be cheated by the server. Ren et al. then constructed a verifiable outsourcing scheme of MExps, where the outsourcer can detect the error with a probability of 1 if the server misbehaves [6]. Lai et al. [7] proposed an attribute-based encryption (ABE) scheme with verifiable outsourcing decryption, which guaranteed that an outsourcer can efficiently detect the wrong results. Qin et al. [8] then proposed new ABE scheme with outsourced decryption, where the outsourcer could verify the outsourcing results with a high efficiency at the expense of minimal overhead. Chen et al. first considered the problem of outsourcing computation in attribute-based signature (ABS) schemes and delegated the verification of signature to an untrusted server [9]. Yu et al. [10] proposed a secure and efficient cloud storage auditing scheme with verifiable outsourcing of key updates. The process of key updates is outsourced to the third party auditor (TPA), and the TPA only knows the encrypted secret key. Meanwhile, the outsourcer could verify the effectiveness of encrypted secret keys when uploading new files to the cloud server. Also, Wang et al. [11] proposed a privacy-preserving public auditing system for data storage security and extended it to handle the problem of multiple auditing, where the TPA could learn nothing about data and the integrity of data could be verified publicly. Other works target specific classes of functions, such as revocable identity-based encryption [12], solution of linear equations [13], and image features extraction [14].

In recent years, bilinear pairings have various applications in constructing new cryptographic primitive, for example, identity-based encryption [15], short signature [16], and key agreement protocol [17]. In pairing-based cryptography, the computation of bilinear pairing is the most expensive operation and it has important effects on efficiency of these schemes or protocols. Thus, a lot of research work has been done to compute bilinear pairing efficiently [18, 19].

Chevallier-Mames et al. [20] presented the first algorithm for secure outsourcing of bilinear pairings based on an untrusted server, where the outsourcer could detect any failure with probability of 1 if the server returns an incorrect result. However, the outsourcer must execute some other expensive operations such as scalar multiplications and modular exponentiations, where these computations are even comparable to those of bilinear pairings in some scenarios [19, 21]. Subsequently, other works on delegation of bilinear pairings [22, 23] also suffer from the same problems. Chen et al. proposed the first efficient outsourcing algorithm of bilinear pairing in the one-malicious version of two untrusted program models [24], where the outsourcer only carried out 5 point additions and 4 multiplications without any expensive operations, which is suitable for the computation-limited client. However, the checkability of the algorithm in [24] is only 1/2, and the outsourcer may accept a false result returned by a malicious server with probability of 1/2. Tian et al. presented two outsourcing algorithms of bilinear pairing based on two servers [25]. One is more efficient than the algorithm of [24], and the outsourcer needs to execute 4 point additions and 3 multiplications with the same checkability. The other algorithm is more flexible based on two untrusted servers with improved checkability. As we know, it is also possible for the outsourcer to be cheated by the server and the error cannot be detected successfully. Recently, Ren et al. presented a new outsourcing algorithm of bilinear pairing, which improves the checkability of the outsourcer to 1, and it is impossible for the server to cheat the outsourcer to accept a false outsourcing result [26]. However, it needs two interactive rounds between the outsourcer and the server and increases the communication cost, though the checkability is improved to 1.

1.2. Organization. The rest of this paper is organized as follows. In Section 2, we introduce the definition of bilinear pairing and security model of the outsourcing scheme. A noninteractive verifiable outsourcing algorithm of bilinear pairing is presented and its security analysis is given in Section 3. In Section 4, we introduce two applications of the proposed outsourcing scheme: an AIBE scheme with outsourced decryption and an IBS scheme with outsourced verification. The performance evaluation of the proposed scheme is presented in Section 5. In Section 6, we conclude the paper.

2. Definitions

In this section, we introduce the properties of bilinear pairings, security definition, and model of the proposed outsourcing algorithms.

2.1. Bilinear Pairings. Let q be a large prime, $G, \widehat{G}$ are two cyclic addition groups of order q, and G_T is a cyclic multiplicative group of order q. P, Q are generators of $G, \widehat{G}$, respectively. $e : G \times \widehat{G} \rightarrow G_T$ is a bilinear map with the following properties [15, 16, 21]:

1. Bilinear: $e(a_0 R, b_0 V) = e(R, V)^{a_0 b_0}$ for any $R \in G$, $V \in \widehat{G}$, and $a_0, b_0 \in {Z_q}^*$
2. Nondegenerate: there exist $R_0 \in G$, $V_0 \in \widehat{G}$ such that $e(R_0, V_0) \neq 1$
3. Computable: there is an efficient algorithm to compute $e(R, V)$ for any $R \in G$, $V \in \widehat{G}$

The bilinear map and the bilinear pairing can be realized by supersingular elliptic curves or hyperelliptic curves over finite groups and Weil or Tate pairings, respectively [15, 16, 21].

2.2. Security Definition and Model. Now we review the formal security definition of an outsourcing algorithm introduced by [5]. An algorithm *Alg* includes a trusted part T and an untrusted program U, and T^U denotes the works carried out by T invoking U. An adversary A is simulated by a pair of algorithms (E, U'), where E denotes the adversarial environment that submits adversarial inputs for *Alg* and U' represents adversarial software written by E. As described in [5], we assume that the two adversaries (E, U') can only make direct communication before the execution of T^U, and, in other cases, they can only communicate with each other by passing messages through the outsourcer T.

Definition 1 (algorithm with outsource *I/O*). An algorithm *Alg* takes five inputs and generates three outputs. The first three inputs are chosen by an honest party, and the last two inputs are generated by the environment E. The first input is honest and secret, which is unknown for both E and U'; the second is honest and protected, which may be public for E but is private for U'; the third is honest and unprotected, which may be public for both E and U'; the fourth is adversarial and protected, which is public for E but is protected from U'; and the last one is adversarial and unprotected, which is public for E and U'. Similarly, the first output is secret, which is protected from E and U'; the second is protected, which may be public for E but not U'; and the third is unprotected, which may be public for both E and U'.

The following security definition ensures that both E and U' cannot obtain any information about the private 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 *I/O*. T^U is called an outsource-secure implementation of *Alg* if the following conditions hold:

(1) *Correctness*: T^U is a correct implementation of *Alg*

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

Pair One. $EVIEW_{\text{real}} \sim EVIEW_{\text{ideal}}$, which means that the malicious environment E cannot gain anything interesting about the private inputs and outputs during the execution of T^U. The detailed definitions of the real process and the ideal process are omitted because of limited space; please see [5] for the details.

Pair Two. $UVIEW_{\text{real}} \sim UVIEW_{\text{ideal}}$, which means that the untrusted software U' written by E learns nothing about the inputs and outputs during the execution of T^U. Please also see [5] for the detailed definitions.

Assume that T^U is a correct implementation of *Alg*; we have the following definitions.

Definition 3 (α-efficient, secure outsourcing). A pair of algorithms (T, U) are α-efficient if the running time of $T$is not more than an α-multiplicative factor of that of *Alg* for any input x.

Definition 4 (β-checkable, secure outsourcing). A pair of algorithms (T, U) are β-checkable if T detects any deviation of U' from its advertised functionality during the implementation of $T^{U'(x)}$ with probability not less than β for any input x.

Definition 5 ((α, β)-outsource-security). A pair of algorithms (T, U) are called an (α, β)-outsource-secure implementation of *Alg* if they are both α-efficient and β-checkable.

The proposed algorithms are executed based on two untrusted program models introduced by [5]. In this model, the adversarial environment E writes two programs $U' = (U'_1, U'_2)$, and T installs these programs in a manner such that all subsequent communication between any two of E, U'_1, and U'_2 must pass through T. The new adversary attacking T is $A = (E, U'_1, U'_2)$. We assume that at most one of the programs misbehaves, but we do not know which one. It is named as the one-malicious version of two untrusted models. In the real world, it is equivalent to buying two copies of the untrusted software from different vendors and achieving the outsource security as long as one of them is honest [1].

3. Verifiable Secure Outsourcing of Bilinear Pairing

As [5], a subroutine named Rand is used to speed up the computations. The inputs for Rand are a prime q, two cyclic addition groups $G, \widehat{G}$ of order q, and a bilinear map $e : G \times \widehat{G} \rightarrow G_T$, where G_T is a cyclic multiplicative group of order q and the output for each invocation is a random, independent vector of the following form:

$$\begin{aligned}
\big(&t_1, t_2, a_1P + a_2P, a_3P, a_4P, b_1P + b_2P, b_3P, \\
&- (a_1P + a_2P + b_3P), - (t_2a_1P + a_2P), \\
&- (a_1P + t_2a_2P), a_1Q + a_2Q, a_3Q, b_1Q + b_2Q, b_3Q, \\
&b_4Q, - (b_1Q + b_2Q + a_3Q), - (t_1b_1Q + b_2Q), \\
&- (b_1Q + t_1b_2Q), e(a_3P, a_3Q), e(b_3P, b_3Q), \\
&e(a_4P, b_1Q + b_2Q)^{t_1+1}, e(a_1P + a_2P, b_4Q)^{t_2+1}, \\
&e(a_1P + a_2P, b_1Q + b_2Q)^{-1}\big),
\end{aligned} \tag{1}$$

where $a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4 \in_R Z_q{}^*$, $t_1, t_2 \in [2, 3, \ldots, s]$, and s is a small number.

We can use the table-lookup method to implement this functionality. First, a trusted server computes a table of random, independent vectors in advance and then stores it into the memory of T. For each invocation of i, T needs to retrieves a new vector in the table.

3.1. Verifiable Outsourcing Algorithm. We propose a noninteractive verifiable outsourcing algorithm *NIVBP* for bilinear pairing in the one-malicious model. In *NIVBP* algorithm, T outsources its bilinear pairing computations to U_1 and U_2 by invoking the subroutine Rand. A requirement for *NIVBP* is that the adversary A cannot know any useful information about the inputs and outputs of *NIVBP*.

Let q be a large prime. The input of *NIVBP* is $A \in G$ and $B \in \widehat{G}$, and the output is $e(A, B)$. A and B are both computationally blinded to U_1 and U_2. The proposed *NIVBP* algorithm is described as follows:

(1) T firstly runs Rand one time to create a blinding vector as (1).

(2) T queries U_1 in random order as follows:

$$\begin{aligned}
&U_1\left(A + a_1P + a_2P, B + b_1Q + b_2Q\right) \longrightarrow \\
&\alpha_{11} = e\left(A + a_1P + a_2P, B + b_1Q + b_2Q\right), \\
&U_1\left(A + b_1P + b_2P, a_3Q\right) \longrightarrow \\
&\alpha_{12} = e\left(A + b_1P + b_2P, a_3Q\right), \\
&U_1\left(-b_3P - a_1P - a_2P, B + b_3Q\right) \longrightarrow \\
&\alpha_{13} = e\left(-b_3P - a_1P - a_2P, B + b_3Q\right), \\
&U_1\left(A + a_4P, -t_1b_1Q - b_2Q\right) \longrightarrow \\
&\alpha_{14} = e\left(A + a_4P, -t_1b_1Q - b_2Q\right), \\
&U_1\left(-t_2a_1P - a_2P, B + b_4Q\right) \longrightarrow \\
&\alpha_{15} = e\left(-t_2a_1P - a_2P, B + b_4Q\right).
\end{aligned} \tag{2}$$

Similarly, T queries U_2 in random order as follows:

$$\begin{aligned}
&U_2\left(A + a_1P + a_2P, B + b_1Q + b_2Q\right) \longrightarrow \\
&\alpha_{21} = e\left(A + a_1P + a_2P, B + b_1Q + b_2Q\right), \\
&U_2\left(A + a_3P, -a_3Q - b_1Q - b_2Q\right) \longrightarrow \\
&\alpha_{22} = e\left(A + a_3P, -a_3Q - b_1Q - b_2Q\right), \\
&U_2\left(b_3P, B + a_1Q + a_2Q\right) \longrightarrow \\
&\alpha_{23} = e\left(b_3P, B + a_1Q + a_2Q\right), \\
&U_2\left(A + a_4P, -b_1Q - t_1b_2Q\right) \longrightarrow \\
&\alpha_{24} = e\left(A + a_4P, -b_1Q - t_1b_2Q\right), \\
&U_2\left(-a_1P - t_2a_2P, B + b_4Q\right) \longrightarrow \\
&\alpha_{25} = e\left(-a_1P - t_2a_2P, B + b_4Q\right).
\end{aligned} \tag{3}$$

(3) T verifies whether U_1 and U_2 generate the correct outputs, which means that (4)–(6) hold.

(a)

$$\alpha_{11} = \alpha_{21} \tag{4}$$

(b)

$$\begin{aligned}
&\left(\alpha_{12} \cdot \alpha_{22} \cdot e\left(a_3P, a_3Q\right)\right)^{t_1+1} \\
&\quad = \alpha_{14} \cdot \alpha_{24} \cdot e\left(a_4P, b_1Q + b_2Q\right)^{t_1+1}
\end{aligned} \tag{5}$$

(c)

$$\begin{aligned}
&\left(\alpha_{13} \cdot \alpha_{23} \cdot e\left(b_3P, b_3Q\right)\right)^{t_2+1} \\
&\quad = \alpha_{15} \cdot \alpha_{25} \cdot e\left(a_1P + a_2P, b_4Q\right)^{t_2+1}.
\end{aligned} \tag{6}$$

If not, T outputs “error”; otherwise, T outputs

$$\begin{aligned}
&\alpha_{12} \cdot \alpha_{22} \cdot e\left(a_3P, a_3Q\right) = e\left(A, b_1Q + b_2Q\right)^{-1}, \\
&\alpha_{13} \cdot \alpha_{23} \cdot e\left(b_3P, b_3Q\right) = e\left(a_1P + a_2P, B\right)^{-1},
\end{aligned} \tag{7}$$

$$\begin{aligned}
&e\left(A, B\right) \\
&\quad = \alpha_{11} \cdot e\left(A, b_1Q + b_2Q\right)^{-1} \cdot e\left(a_1P + a_2P, B\right)^{-1} \\
&\qquad \cdot e\left(a_1P + a_2P, b_1Q + b_2Q\right)^{-1}.
\end{aligned} \tag{8}$$

Correctness. It is obvious that formula (4) holds if two servers are all honest. In addition,

$$\begin{aligned}
&\alpha_{12} \cdot \alpha_{22} \cdot e\left(a_3P, a_3Q\right) \\
&\quad = e\left(b_1P + b_2P, a_3Q\right) e\left(A + a_3P, -b_1Q - b_2Q\right) \\
&\quad = e\left(A, b_1Q + b_2Q\right)^{-1},
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\alpha_{13} \cdot \alpha_{23} \cdot e\left(b_3P, b_3Q\right) \\
&\quad = e\left(-a_1P - a_2P, B + b_3Q\right) e\left(b_3P, a_1Q + a_2Q\right) \\
&\quad = e\left(a_1P + a_2P, B\right)^{-1},
\end{aligned} \tag{10}$$

$$\begin{aligned}
&\left(\alpha_{12}\alpha_{22}e\left(a_3P, a_3Q\right)\right)^{t_1+1} \\
&\quad = \alpha_{14}\alpha_{24}e\left(a_4P, b_1Q + b_2Q\right)^{t_1+1} \\
&\quad = e\left(A + a_4P, b_1Q + b_2Q\right)^{-(t_1+1)} e\left(a_4P, b_1Q + b_2Q\right)^{t_1+1} \\
&\quad = e\left(A, b_1Q + b_2Q\right)^{-(t_1+1)},
\end{aligned} \tag{11}$$

$$\begin{aligned}
&\left(\alpha_{13}\alpha_{23}e\left(b_3P, b_3Q\right)\right)^{t_2+1} \\
&\quad = \alpha_{15}\alpha_{25}e\left(a_1P + a_2P, b_4Q\right)^{t_2+1} \\
&\quad = e\left(a_1P + a_2P, B + b_4Q\right)^{-(t_2+1)} e\left(a_1P + a_2P, b_4Q\right)^{t_2+1} \\
&\quad = e\left(a_1P + a_2P, B\right)^{-(t_2+1)}.
\end{aligned} \tag{12}$$

Therefore, formulas (4)–(6) hold according to the above analysis. Finally, T obtains $e(A, B)$ as (8).

Remark 6. If one of the servers is dishonest, the results could be verified successfully with a probability close to 1 except that the dishonest server knows the values of $\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}$ (or $\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \alpha_{25}$) and t_1, t_2. As we know, five queries sent to U_1 and U_2 are submitted in random order and $t_1, t_2 \in [2, 3, \dots, s]$. So, the dishonest server could guess the values of $\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}$ (or $\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \alpha_{25}$) and t_1, t_2 with the probabilities of $1/5!$ and $1/(s-1)^2$, respectively. Therefore, the checkability of the *NIVBP* algorithm is

$$1 - \frac{1}{5!\,(s-1)^2} = 1 - \frac{1}{120\,(s-1)^2} \approx 1. \tag{13}$$

Remark 7. The proposed algorithm *NIVBP* is also applicative in the condition where $G, \widehat{G}$ are two cyclic multiplication groups. Let $g, \widehat{g}$ be generators of $G, \widehat{G}$, respectively. $e : G \times \widehat{G} \to G_T$ is a bilinear map. In this case, the inputs of *NIVBP* are also $A \in G$ and $B \in \widehat{G}$, and the output is also $e(A, B)$. The details are also omitted because of limited space.

3.2. Security Analysis

Theorem 8. *In the one-malicious model, the proposed algorithm $(T, (U_1, U_2))$ is an outsource-secure implementation of NIVBP, where the input (A, B) may be honest and secret or honest and protected or adversarial and protected.*

Proof. Let $A = (E, U_1', U_2')$ be a PPT adversary that interacts with a PPT algorithm T in the one-malicious model.

First, we prove that $\mathrm{EVIEW}_{\mathrm{real}} \sim \mathrm{EVIEW}_{\mathrm{ideal}}$, which means that the environment E learns nothing during the execution of $(T, (U_1, U_2))$. If the input (A, B) is honest and protected or adversarial and protected, it is obvious that the simulator S_1 behaves the same as in the real execution. Therefore, we only need to prove the case where (A, B) is an honest, secret input.

So, suppose that (A, B) is an honest, secret input. The simulator S_1 in the ideal experiment behaves as follows. On receiving the input on round i, S_1 ignores it and instead makes five random queries of the form (α_j, β_j) to both U_1' and U_2' Finally, S_1 randomly checks one output $e(\alpha_j, \beta_j)$ from each program. If an error is detected, S_1 saves all states and outputs $Y_p^i =$ "error," $Y_u^i = \varnothing$, $\mathrm{rep}^i = 1$, and thus the final output for ideal process is (estate^i, "error," $\varnothing$). If no error is detected, S_1 checks the remaining four outputs. If all checks pass, S_1 outputs $Y_p^i = \varnothing$, $Y_u^i = \varnothing$, $\mathrm{rep}^i = 0$; that is, the final output for ideal process is $(\mathrm{estate}^i, y_p^i, y_u^i)$; otherwise, S_1 selects a random element r and outputs $Y_p^i = r, Y_u^i = \varnothing, \mathrm{rep}^i = 1$, and the output for ideal process is $(\mathrm{estate}^i, r, \varnothing)$. □

In addition, we need to show that the inputs to (U_1', U_2') in the real experiment are computationally indistinguishable from those in the ideal one. In the ideal experiment, the inputs are selected uniformly at random. In the real one, each part of all five queries that T makes to any program is generated by invoking the subroutine Rand and thus is computationally indistinguishable from random numbers. Therefore, we consider three possible conditions. If (U_1', U_2') both are honest in round i, $\mathrm{EVIEW}^i_{\mathrm{real}} \sim \mathrm{EVIEW}^i_{\mathrm{ideal}}$, since the outputs of *NIVBP* are not replaced and $\mathrm{rep}^i = 0$; if one of (U_1', U_2') is dishonest in round i, the fault must be detected by both T and S_1 with a probability close to 1, resulting in an output of "error"; otherwise, the output of *NIVBP* is corrupted with a probability of $1/120(s-1)^2$. In the real experiment, the five outputs generated by (U_1', U_2') are multiplied together along with a random value. Thus, $\mathrm{EVIEW}^i_{\mathrm{real}} \sim \mathrm{EVIEW}^i_{\mathrm{ideal}}$ even when one of (U_1', U_2') misbehaves, so we conclude that $\mathrm{EVIEW}_{\mathrm{real}} \sim \mathrm{EVIEW}_{\mathrm{ideal}}$.

Second, we prove that $\mathrm{UVIEW}_{\mathrm{real}} \sim \mathrm{UVIEW}_{\mathrm{ideal}}$, which means that the untrusted software (U_1', U_2') learns nothing during the execution of $(T, (U_1', U_2'))$. In the ideal experiment, the simulator S_2 always behaves as follows: when receiving the input on round i, S_2 ignores it but submits five random queries of the form (α_j, β_j) to U_1' and U_2'. Then S_2 saves its states and those of (U_1', U_2'). Since the honest, secret or honest, protected or adversarial, protected inputs are all private for (U_1', U_2'), the simulator S_2 is applicable to all those conditions. As shown in Pair One, the inputs to (U_1', U_2') in the real experiment are computationally indistinguishable from those in the ideal one randomly chosen by S_2. Thus, $\mathrm{UVIEW}^i_{\mathrm{real}} \sim \mathrm{UVIEW}^i_{\mathrm{ideal}}$ for each round i, and so $\mathrm{UVIEW}_{\mathrm{real}} \sim \mathrm{UVIEW}_{\mathrm{ideal}}$.

Theorem 9. *In the one-malicious model, the proposed algorithm $(T, (U_1, U_2))$ in Section 3.1 is verifiable; that is, the outsourcer can test the error with a probability close to 1 if one of the servers outputs the false result.*

Proof. Assume that U_1 is an honest server and U_2 is a malicious server. At the end of the algorithm, the outsourcer verifies the results by formulas (4)–(6). It is obvious that U_2 must generate the correct value of α_{21}; otherwise, formula (4) cannot pass the verification with a probability of 1. Thus, the only possibility of U_2 cheating T is returning the false value of $\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \alpha_{25}$, which is denoted by $\overline{\alpha_{21}}, \overline{\alpha_{22}}, \overline{\alpha_{23}}, \overline{\alpha_{24}}, \overline{\alpha_{25}}$, respectively.

Assume that $\overline{\alpha_{21}}, \overline{\alpha_{22}}, \overline{\alpha_{23}}, \overline{\alpha_{24}}, \overline{\alpha_{25}}$ could pass the verification of formulas (5) and (6); that is,

$$\begin{aligned}
&\left(\alpha_{12} \cdot \overline{\alpha_{22}} \cdot e\left(a_3P, a_3Q\right)\right)^{t_1+1} \\
&\quad = \alpha_{14} \cdot \overline{\alpha_{24}} \cdot e\left(a_4P, b_1Q + b_2Q\right)^{t_1+1}, \\
&\left(\alpha_{13} \cdot \overline{\alpha_{23}} \cdot e\left(b_3P, b_3Q\right)\right)^{t_2+1} \\
&\quad = \alpha_{15} \cdot \overline{\alpha_{25}} \cdot e\left(a_1P + a_2P, b_4Q\right)^{t_2+1},
\end{aligned} \tag{14}$$

which means that

$$\begin{aligned}
\frac{\overline{\left(\alpha_{22}\right)^{t_1+1}}}{\overline{\alpha_{24}}} &= \frac{\alpha_{14} \cdot e\left(a_4P, b_1Q + b_2Q\right)^{t_1+1}}{{\alpha_{12}}^{t_1+1} \cdot e\left(a_3P, a_3Q\right)^{t_1+1}}, \\
\frac{\overline{\left(\alpha_{23}\right)^{t_1+1}}}{\overline{\alpha_{25}}} &= \frac{\alpha_{15} \cdot e\left(a_1P + a_2P, b_4Q\right)^{t_2+1}}{{\alpha_{13}}^{t_2+1} \cdot e\left(b_3P, b_3Q\right)^{t_2+1}}.
\end{aligned} \tag{15}$$

Since U_1 is an honest server, $\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}$ must be correct. In addition, $e(a_3P, a_3Q)$, $e(b_3P, b_3Q)$, $e(a_4P, b_1Q + b_2Q)^{t_1+1}$, $e(a_1P + a_2P, b_4Q)^{t_2+1}$ are generated randomly by Rand subroutine, and so these values must be true. Thus, the

Table 1: Comparison of the outsourcing algorithms for bilinear pairing.

Algorithm	Pair [24]	TZR1 [25]	TZR2 [25] ($s = 4$)	VBP [26]	NIVBP ($s = 4$)
PA (T)	5	4	11	8	8
M (T)	4	3	9	11	19
Invoke (Rand)	3	1	2	2	1
Pair (U)	8	6	6	6	10
MExp (U)	0	0	0	4	0
Interactive	No	No	No	Yes	No
Servers	Two	Two	Two	Two	Two
Checkability	0.5	0.5	0.84	1	0.999

values of $\overline{(\alpha_{22})}^{t_1+1}/\overline{\alpha_{24}}$ and $\overline{(\alpha_{23})}^{t_2+1}/\overline{\alpha_{25}}$ should be true even if $\overline{\alpha_{21}}, \overline{\alpha_{22}}, \overline{\alpha_{23}}, \overline{\alpha_{24}}, \overline{\alpha_{25}}$ are incorrect; otherwise, they could not pass the verification of formulas (5) and (6).

In order to obtain the true values of $\overline{(\alpha_{22})}^{t_1+1}/\overline{\alpha_{24}}$ and $\overline{(\alpha_{23})}^{t_2+1}/\overline{\alpha_{25}}, U_2$ must guess the values of $\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \alpha_{25}$ and t_1, t_2. As shown in Section 3.1, the probabilities of guessing the true values of $\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \alpha_{25}$ and t_1, t_2 are $1/5!$ and $1/(s-1)^2$, respectively. Therefore, the outsourcer can test the error with a probability of $1 - 1/5!(s-1)^2 = 1 - 1/120(s-1)^2 \approx 1$. □

Theorem 10. *In the one-malicious model, the proposed algorithm $(T, (U_1, U_2))$ is an $(O(s/m), \approx 1)$-outsource-secure implementation of NIVBP, where s is a small positive integer and m is the bit length of q and q is the order of $G, \widehat{G}$.*

Proof. The proposed algorithm *NIVBP* makes one call to Rand and 8 point additions (PA) in G or $\widehat{G}$ and $O(s)$ multiplication in G_T in order to compute $e(A, B)$. As shown in [24], it takes roughly $O(m)$ multiplications in resulting finite field to compute the bilinear pairing, where m is the bit length of q. Thus, the proposed algorithm is an $O(s/m)$-efficient implementation of *NIVBP*. On the other hand, it must be detected with a probability close to 1 if U_1 or U_2 fails during any execution of *NIVBP* from Theorem 9. □

3.3. Comparison. We compare the outsourcing algorithms for bilinear pairing with input privacy in Table 1, where s is a small positive integer and "PA" and "M" denote the operation of point addition in G or $\widehat{G}$ and multiplication in G_T, respectively.

From Table 1, we conclude that the *NIVBP* algorithm increases checkability of the outsourcer, though a little computation cost is appended compared with Pair and TZR1 algorithms. In addition, the *NIVBP* algorithm improves computation efficiency and checkability of the outsourcer simultaneously compared with TZR2 algorithm for the same parameter: $s = 4$. The efficiency and checkability of the *NIVBP* algorithm are nearly the same as those of VBP algorithm, but it decreases the communication cost, since it is noninteractive while the VBP algorithm is interactive. Therefore, the *NIVBP* algorithm increases checkability and decreases communication cost of the outsourcer, although a little computation cost is appended.

4. Applications

In this section, we introduce two applications of the proposed *NIVBP* algorithm: anonymous identity-based encryption (AIBE) scheme [27] and identity-based signature (IBS) scheme [28].

Let $G, \widehat{G}, G_T$ be three cyclic multiplication groups of order q, and let $g, \widehat{g}$ be generators of $G, \widehat{G}$, respectively. $e : G \times \widehat{G} \rightarrow G_T$ is a bilinear map. In the following schemes, $G = \widehat{G}$.

4.1. Boyen-Waters AIBE Scheme with Outsourcing Decryption. The proposed outsource-secure AIBE scheme consists of the following algorithms.

Setup. It chooses a random generator $g \in G$, random group elements $g_0, g_1 \in G$, and random exponents $\omega, t_1, t_2, t_3, t_4 \in Z_q$. The master key MSK $= \{\omega, t_1, t_2, t_3, t_4\}$, and the public parameters PK are as follows:

$$\left\{e(g,g)^{t_1 t_2 \omega}, g, g_0, g_1, v_1 = g^{t_1}, v_2 = g^{t_2}, v_3 = g^{t_3}, v_4 = g^{t_4}\right\}. \tag{16}$$

Extract (MSK, ID). To issue a private key for identity ID, it chooses two random exponents $r_1, r_2 \in Z_q$ and computes the private key $\mathrm{SK}_{\mathrm{ID}} = \{d_0, d_1, d_2, d_3, d_4\}$ as follows:

$$\begin{aligned} d_0 &= g^{r_1 t_1 t_2 + r_2 t_3 t_4}, \\ d_1 &= g^{-\omega t_2} \left(g_0 g_1^{\mathrm{ID}}\right)^{-r_1 t_2}, \\ d_2 &= g^{-\omega t_1} \left(g_0 g_1^{\mathrm{ID}}\right)^{-r_1 t_1}, \\ d_3 &= \left(g_0 g_1^{\mathrm{ID}}\right)^{-r_2 t_4}, \\ d_4 &= \left(g_0 g_1^{\mathrm{ID}}\right)^{-r_2 t_3}. \end{aligned} \tag{17}$$

Encrypt (PK, ID, M). To encrypt a message $M \in G_T$ for an identity ID, it chooses random $s, s_1, s_2 \in Z_q$ and creates the ciphertext $\mathrm{CT} = \{C', C_0, C_1, C_2, C_3, C_4\}$ as follows:

$$\left\{Me(g,g)^{t_1 t_2 \omega s}, \left(g_0 g_1^{\mathrm{ID}}\right)^{s}, v_1^{s-s_1}, v_2^{s_1}, v_3^{s-s_2}, v_4^{s_2}\right\}. \tag{18}$$

Decrypt (PK, ID, CT). The outsourcer T executes the *NIVBP* algorithm for five times and obtains

$$e(C_i, d_i) = NIVBP(C_i, d_i), \quad i = 0, 1, 2, 3, 4 \tag{19}$$

and then computes $C' \prod_{i=0}^{4} e(C_i, d_i) = M$.

4.2. Paterson-Schuldt IBS Scheme with Outsourcing Verification.

The detailed scheme is shown as follows.

Setup. It picks $\alpha \in Z_q$, $g_2 \in G$, and computes $g_1 = g^{\alpha}$. Further, choose $u', m' \in G$ and vectors $U = (u_i)$, $M = (m_i)$ of length n_u and n_m, respectively, where u_i, m_i are random elements from G. The public parameters are $\mathrm{PK} = \{g, g_1, g_2, u', U, m', M, e(g_2, g_1)\}$ and the master secret key is g_2^{α}.

Extract. Let u be a bit string of length n_u representing an identity and let $u[i]$ be the i-th bit of u. Set $U \subset \{1, 2, \ldots, n_u\}$ as the set of index i such that $u[i] = 1$. To construct the private key d_u of the identity u, pick $r_u \in Z_q$ and compute

$$d_u = \left(g_2^{\alpha}\left(u'\prod_{i\in U}u_i\right)^{r_u}, g^{r_u}\right). \tag{20}$$

Sign. Let $M \subset \{1, \ldots, n_m\}$ be the set of index j such that $m[j] = 1$, where m is a message and $m[j]$ is the j-th bit of m. To generate a signature σ for the message m, randomly choose $r_m \in Z_q$ and compute

$$\sigma = \left(g_2^{\alpha}\left(u'\prod_{i\in U}u_i\right)^{r_u}\left(m'\prod_{j\in M}m_j\right)^{r_m}, g^{r_u}, g^{r_m}\right). \tag{21}$$

Verify. Given a signature $\sigma = (V, R_u, R_m)$ of an identity u for a message m, the outsourcer T executes the *NIVBP* algorithm and obtains

$$\begin{aligned} e(V, g) &= NIVBP(V, g), \\ e\left(u'\prod_{i\in U}u_i, R_u\right) &= NIVBP\left(u'\prod_{i\in U}u_i, R_u\right), \\ e\left(u'\prod_{i\in U}u_i, R_u\right) &= NIVBP\left(m'\prod_{j\in M}m_j, R_m\right). \end{aligned} \tag{22}$$

And verify

$$\begin{aligned} &e(V, g) \\ &\quad = e(g_2, g_1)\, e\left(u'\prod_{i\in U}u_i, R_u\right) e\left(u'\prod_{i\in U}u_i, R_u\right). \end{aligned} \tag{23}$$

It is obvious that the two outsourcing schemes are verifiable and secure, since the *NIVBP* algorithm is verifiable with input privacy as described in Section 3.

5. Performance Evaluation

In this section, we provide an experimental evaluation of the proposed outsourcing algorithms. Our experiment is simulated on two machines with Intel Xeon Processor running at 3.4 GHz with 32 G memory (cloud server) and Intel Celeron Processor running at 1.2 GHz with 2 G memory (the outsourcer), respectively. The programming language is Java, using Java Pairing-Based Cryptography (JPBC) Library. The parameter q is a 160-bit prime that is randomly generated.

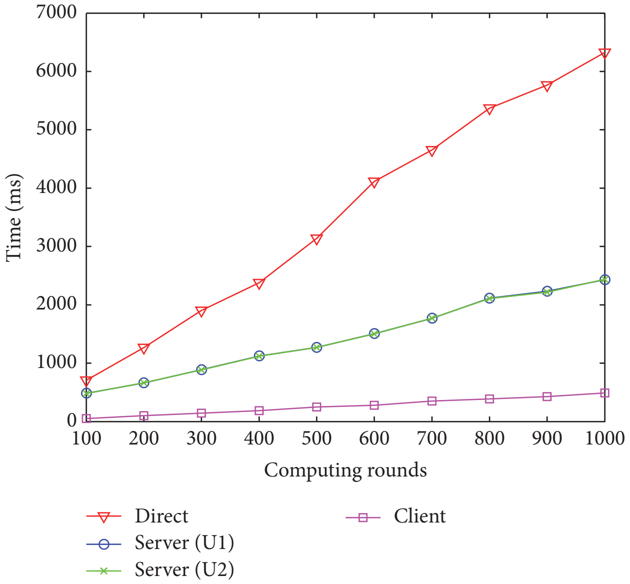

Figure 1: Simulation for the NIVBP algorithm.

In Figure 1, we provide the simulation of *NIVBP* algorithm, which means that the fault can be found with a probability close to 1 if one of the servers misbehaves. It is obvious that the time cost for the outsourcer T is much smaller than that for directly computing bilinear pairing, since a number of computations have been delegated to two servers. Therefore, the proposed *NIVBP* algorithm is the implementation of secure and verifiable outsourcing for bilinear pairing.

In Figure 2, we compare the evaluation times of the outsourcing algorithms for bilinear pairing proposed in [24–26] and this paper, respectively. From Figure 2, we conclude that, for the outsourcer T, the *NIVBP* algorithm is superior to TZR2 algorithm in efficiency, and it appends small computation cost to improve the checkability compared with Pair and TZR1 algorithms. In addition, the *NIVBP* algorithm is nearly the same as VBP algorithm in efficiency, but it is noninteractive and decreases the communication cost of the outsourcer. Thus, the proposed *NIVBP* algorithm improves the checkability and decreases communication cost for the outsourcer simultaneously based on two servers in the one-malicious model.

6. Conclusions

In this paper, we propose a noninteractive verifiable outsource-secure algorithm for bilinear pairing. The security model of our proposed algorithm is based on two noncolluding servers, and the outsourcer can detect any failure with a probability close to 1 if one of the servers misbehaves. Compared with the previous ones, the proposed algorithm improves the checkability and communication efficiency simultaneously for the outsourcer.

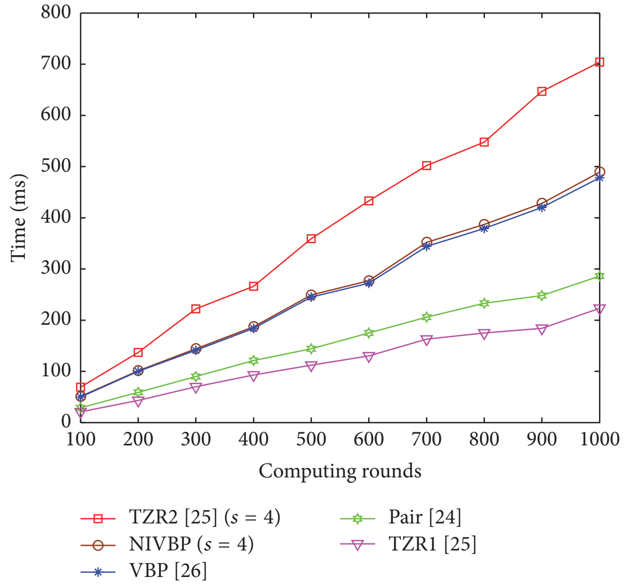

Figure 2: Efficiency comparison of the outsourcing algorithms for bilinear pairing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work described in this paper was supported by the National Natural Science Foundation of China (Grant no. 61572309), Natural Science Foundation of Shanghai (no. 16ZR1411200), and Program for New Century Excellent Talents in University (NCET-12-0620).

References

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

[2] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: outsourcing computation to untrusted workers," in *Advances in cryptology—CRYPTO 2010*, vol. 6223 of *Lecture Notes in Comput. Sci.*, pp. 465–482, Springer, Berlin, Germany, 2010.

[3] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Advances in cryptology—CRYPTO 2010*, vol. 6223 of *Lecture Notes in Comput. Sci.*, pp. 483–501, Springer, Berlin, Germany, 2010.

[4] D. Chaum and T. Pedersen, "Wallet databases with observers," in *Advances in Cryptology—CRYPTO' 92*, vol. 740 of *Lecture Notes in Computer Science*, pp. 89–105, Springer, Berlin, Germany, 1993.

[5] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proceedings of the TCC 2005*, vol. 3378 of *Lecture Notes in Computer Science*, pp. 264–282, Springer.

[6] Y. Ren, N. Dingy, X. Zhang, H. Lu, and D. Gu, "Verifiable outsourcing algorithms for modular exponentiations with improved checkability," in *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2016*, pp. 293–303, ACM, June 2016.

[7] J.-Z. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[8] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.

[9] X. Chen, J. Li, X. Huang, J. Li, Y. Xiang, and D. S. Wong, "Secure outsourced attribute-based signatures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3284–3294, 2014.

[10] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362–1375, 2016.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE INFO-COM*, pp. 525–533, San Diego, Calif, USA, March 2010.

[12] Y. Ren, N. Ding, X. Zhang, H. Lu, and D. Gu, "Identity-based encryption with verifiable outsourced revocation," *Computer Journal*, vol. 59, no. 11, pp. 1659–1668, 2016.

[13] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Transactions on Information and Forensics Security*, vol. 10, no. 1, pp. 69–78, 2015.

[14] Y. Ren, X. Zhang, G. Feng, Z. Qian, and F. Li, "How to Extract Image Features based on Co-occurrence Matrix Securely and Efficiently in Cloud Computing," *IEEE Transactions on Cloud Computing*, 2017.

[15] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer, Berlin, Germany, 2001.

[16] J. C. Cha and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Proceedings of the PKC*, vol. 2567 of *Lecture Notes in Computer Science*, pp. 18–30, Springer.

[17] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in *Proceedings of the ANTS*, vol. 1838 of *Lecture Notes in Computer Science*, pp. 385–393, Springer, 2000.

[18] M. Scott, N. Costigan, and W. Abdulwahab, "Implementing cryptographic pairings on smartcards," in *Proceedings of the CHES*, vol. 4249 of *LNCS*, pp. 134–147, 2006.

[19] P. S. Barreto, S. D. Galbraith, C. Heigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Designs, Codes and Cryptography*, vol. 42, no. 3, pp. 239–271, 2007.

[20] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Proceedings of the CARDIS 2010*, vol. 6035 of *LNCS*, pp. 24–35, 2010.

[21] S. D. Galbraith, K. G. Paterson, and N. . Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[22] P. Tsang, S. Chow, and S. Smith, "Batch pairing delegation," in *Proceedings of the IWSEC 2007*, 90, 74 pages, 2007.

[23] S. S. M. Chow, M. H. Au, and W. Susilo, "Server-aided signatures verification secure against collusion attack," in *Proceedings of the 6th International Symposium on Information, Computer and Communications Security, ASIACCS 2011*, pp. 401–405, March 2011.

[24] X. Chen, W. Susilo, J. Li et al., "Efficient algorithms for secure outsourcing of bilinear pairings," *Theoretical Computer Science*, vol. 562, pp. 112–121, 2015.

[25] H. Tian, F. Zhang, and K. Ren, "Secure bilinear pairing outsourcing made more efficient and flexible," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2015*, pp. 417–426, April 2015.

[26] Y. Ren, N. Ding, T. Wang, H. Lu, and D. Gu, "New algorithms for verifiable outsourcing of bilinear pairings," *Science China Information Sciences*, vol. 59, no. 9, Article ID 99103, 2016.

[27] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Advances in cryptology—CRYPTO 2006*, vol. 4117 of *Lecture Notes in Computer Science*, pp. 290–307, Springer, Berlin, Germany, 2006.

[28] K. G. Paterson and J. C. N. Schuldt, "Efficient identity-based signatures secure in the standard model," in *Proceedings of the ACISP 2006*, vol. 4058 of *Lecture Notes in Computer Science*, pp. 207–222, Springer.