Journal of Computer Security and Data Forensics ©2021 ISSN xxxx-xxxx
Volume **1**, Number **2**, June **2021**

Security analysis and improvement of verifiable outsourcing of bilinear pairing scheme

Jen-Chien Hsu[1], Jhe-Wei Lin[1], Sk Md Mizanur Rahman[2], JRaylin Tso[1,*]

[1]Department of Computer Science
National Chengchi University
64, Sec. 2, Zhi-nan Rd., Taipei 11605, China

[2]Information and Communication Engineering Technology (ICET)
School of Engineering Technology and Applied Science (SETAS)
Centennial College, Toronto, ON M1G 3T8, Canada

Received February 2021; revised May 2021
(Communicated by: raylin@cs.nccu.edu.tw)

Abstract. *In the research of cryptography, a very important technique is "pairing-based cryptography ". Its method is based on an element in the additive group G1 and an element in the additive group G2 to perform mapping to generate an element in the multiplicative group G. Pairing is widely used in the field of cryptography and pairing operations are resource-consuming. So, the method of outsourcing pairing calculation has attracted much attention, and Ren et al. proposed a scheme in 2016. It proposes a secure verifiable outsourcing algorithm of single bilinear pairing based on two untrusted servers, and improved verifiability of outsourcing. But the disadvantage of this scheme is existing attack method for verifiability. The attack method proposed by Osmanbey Uzunkol et al. in 2017. So, this paper proposes an improved solution to prevent attacks. In this paper, we first introduce the algorithm for verifiable outsourcing of bilinear pairings proposed by Ren et al., then presents attack methods of this algorithm proposed by Osmanbey Uzunkol et al. Final, we present our method to improve the algorithm and show it is security for protecting user's information.*
Keywords: Bilinear pairings, Verifiable outsourcing, Pairing-based cryptography.

1. Introduction.

1.1. Research Background.
With the proliferation of handheld devices such as mobile phones and smart watches, the rapid spread of mobile technology has prompted the rise of the Internet of Things (IoT) and the development of cloud computing. Even terminal devices with less computing resources can outsource complex operations to cloud servers. By outsourcing complex calculations, devices with fewer resources can outsource complex calculations to cloud servers with more computing resources. Cloud servers usually have powerful computing capabilities. Its task is to perform complex calculations. Although users can outsource computing, they must bear security risks, such as malicious servers returning wrong values or leaking confidential information. Therefore, there are two challenges. The first one is sensitive and confidential information, which should not enable the cloud server to obtain any useful information from the computing process; the second one is if the cloud server is not trusted, malicious servers may return incorrect results. Therefore, the user must have the ability to check, and the algorithm must have checkability. One of the outsourcing technologies that have received much attention is

"verifiable outsourcing of bilinear pairings". Users can outsource part of the bilinear pairing calculation, or check whether the cloud server calculates the result correctly for us, instead of returning invalid or incorrect results. This technology is widely used in attribute-based signature (ABS) systems. The original ABS scheme was developed from identity-based signatures, and more precisely says that it is from fuzzy identity based signature proposed by Yang et al.[1]. The algorithm is more flexible, and it is an important technology in the authentication system to protect user privacy. However, one of the main shortcomings of ABS is that the time required for signature increases with the complexity of the predicate formula. Therefore, OABS is an improvement to ABS. In the Outsourced attribute-based signature (OABS), this method achieves the purpose of reducing the cost of calculation and improving the signature by outsourcing complicated calculation of the signature step to the cloud service provider (CSP) effectively. The part is outsourced in the signing step of OABS usually outsources the bilinear pairing calculation to the cloud server. The user only needs to receive the calculation result returned by the cloud server. However, outsourcing processes and methods often involve security issues and challenges. Chen et al.[4] proposed the first outsourcing bilinear pairing algorithm based on two untrusted servers. The outsourcer does not participate in any complex calculations, but the verifiability of the algorithm is only 1/2. Subsequently, Ren et al.[2] propose a secure verifiable outsourcing algorithm of single bilinear pairing based on two untrusted servers. Unfortunately, in 2017 Uzunkol et al. address the problem of secure and verifiable delegation of general pairing computation. This paper shows the importance of verifiable outsourcing of bilinear pairings (VBP) algorithm research.

1.2. **Organization.**
This paper is divided into to several sections, the contents of which are as follow:

1. This section introduces the research background and motivation.
2. This section introduces the some background knowledge about this paper.
3. This section introduces the verifiable outsourcing of bilinear pairing scheme(VBP).
4. This section introduces the attack protocol of VBP.
5. This section introduces our modified verifiable outsourcing of the bilinear pairing scheme.
6. This section performs security and correctness of our MVBP scheme.
7. This section introduces the conclusion.

2. **Preliminaries.**

2.1. **Bilinear Pairing.**
In cryptography, bilinear pairing is the basis of many techniques. In this section, we introduce the properties of bilinear pairing.

Definition 2.1. *The bilinear pairing defines on three cyclic group of order p, G_1, G_2 and G_T, where G_1, G_2 are additive group and G_T is multiplicative group. The bilinear paring $e : G_1 \times G_2 \leftarrow G_T$ must satisfy three properties:*

- *Bilinearity: $\forall P \in G_1, Q \in G_2$, then $e(aP, bQ) = e(P,Q)^{ab}$.*
- *Non-degeneracy: $\exists P \in G_1, Q \in G_2$ such that $e(P,Q) \neq 1$, where 1 is the identity of G_T.*
- *Computability: There exists an efficient algorithm to compute bilinear pairing.*

2.2. **Outsourcing Computation.**
The formal security definition of the outsourcing algorithm proposed by Hohenberger and Lysyanskaya et al.[5]. The algorithm is named **Alg**, and **Alg** includes three parties, a trusted and resource-limited user (delegator) T and an untrusted but powerful server U. Finally, E stands for untrusted environment. T is limited computing party, and the process of algorithm **Alg** is that T tries to outsource

its computing tasks to the server U. The symbol T^U indicates that T performs calculation by calling U to complete the task. A adversary A will try to simulate the execution of the actual algorithm. Here we use the symbol (E, U') to represent the simulated algorithm, where E represents an untrusted environment, U' Represents a malicious and untrusted server.

Definition 2.2. *(Algorithm with outsource-I/O)*[5] *An algorithm Alg obeys theoutsource input/output specification if it takes five inputs, and produces three outputs. The first three inputs are generated by an honest party, and are classified by how much the adversary A = (E,U) knows about them. There are five inputs:*

1. *The* ***honest, secret input****, which is unknown to both E and U;*
2. *The* ***honest, protected input****, which may be known by E, but is protected from U;*
3. *The* ***honest, unprotected input****, which may be known by both E and U.*

In addition, there are two adversarially-chosen inputs generated by the environment E:

4. *The* ***adversarial, protected input****, which is known to E, but protected from U;*
5. *The adversarial,* ***unprotected input****, which may be known by E and U.*

Similarly, the first output called ***secret*** *is unknown to both E and U; the second is* ***protected****, which may be known to E, but not U; and the third is* ***unprotected****, which may be known by both parts of A.*

Definition 2.3. *(Outsource-security)*[5] *If the following conditions are met, let* ***Alg*** *be an algorithm with outsource-I/O. we say algorithms (T,U) is implementation of an algorithm* ***Alg*** *securely.*

1. *Correctness: T^U executes all steps of Alg correctly.*
2. *Security: For all probabilistic polynomial-time adversaries A = (E,U), there exist probabilistic expected polynomial-time simulators (S_1, S_2) such that the following pairs of random variables are computationally indistinguishable. Let us say that the honestly-generated inputs are chosen by a process I.*

- *Pair One: $EVIEW_{real} \sim EVIEW_{ideal}$(The external adversary, E, learns nothing.):*

The view that the adversarial environment E obtains by participating in the following REAL process:

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

$EVIEW_{real} = EVEIW^i_{real}$ if $stop^i$ =TRUE.
The real process proceeds in rounds. In round i, the honest (secret, protected, and unprotected) inputs $(x^i_{hs}, x^i_{hp}, x^i_{hu})$ are picked using an honest, stateful process I to which the environment does not have access. Then the environment is based on its view from the last round. And, it chooses

1. *the value of its $estate_i$ variable as a way of remembering what it did next time it is invoked;*
2. *which previously generated honest inputs $(x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu},)$ to give to $T^{U'}$ (note that the environment can specify the index j^i of these inputs, but not their values);*
3. *the adversarial, protected input x^i_{ap};*
4. *the adversarial, unprotected input x^i_{au};*

5. *the Boolean variable $stop^i$ that determines whether round i is the last round in this process.*

Next, the algorithm $T^{U'}$ is run on the inputs $(tstate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^{i}, x_{au}^{i})$, where $tstate^{i-1}$ is T's previously saved state, and produces a new state $tstate^i$ for T, as well as the secret j_s^i, protected y_p^i and unprotected y_u^i outputs. The oracle U' is given its previously saved state, $ustate^{i-1}$, as input, and the current state of U' is saved in the variable$ustate^i$. The view of the real process in round i consists of $estate^i$, and the values y_p^i and y_u^i. The overall view of the environment in the real process is just its view in the last round (i.e., i for which $stop^i$ = TRUE).

– *The IDEAL process:*

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

$$EVIEW_{ideal} = EVIEW_{ideal}^i \text{ if } stop^i = TRUE.$$

The ideal process also proceeds in rounds. In the ideal process, we have a stateful simulator S_1 who, shielded from the secret input x_{hs}^i, but given the non-secret outputs Alg produces when run all the inputs for round i, decides to either output the values (y_p^i, y_u^i) generated by Alg, or replace them with some other values (Y_p^i, Y_u^i). (Notationally, this is captured by having the indicator variable $repalce^i$ be a bit that determines whether y_p^i will be replaced with Y_p^i.) In doing so, it is allowed to query the oracle U'; moreover, U' saves its sate as in the real experiment.

- *Part Two: $UVIEW_{real} \sim UVIEW_{ideal}$ (The untrusted software U', learns nothing):*
  – *The view that the untrusted software U' obtains by participating in the REAL process described in Pair One. $UVIEW_{real} = ustate^i$ if $stop^i$=TURE.*
  – *The Ideal process:*

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

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

In the ideal process, we have a stateful simulator S_2 who, equipped with only the unprotected inputs (x_{hu}^i, x_{au}^i), queries U'. As before, U' may maintain state.

Definition 2.4. *(α-Efficient, secure outsourcing) A pair of algorithms (T,U) are an α-efficient implementation of an algorithm Alg if they are outsource-secure implementation of Alg and for all inputs x, the running time of T is $\leq$ an α-multiplicative factor of the running time of Alg(x).*

For example, we assume that when the algorithm does not outsource any calculation steps, the calculation time is one unit time. If we outsource half of the calculation at this time, we only need to execute the remaining half, then we call (T,U) has 1/2 efficiency.

Definition 2.5. *(β-Checkable, secure outsourcing) A pair of algorithms (T,U) are a $\beta-$checkable implementation of an algorithm Alg if they are an outsource-secure implementation of Alg and for all inputs x, if U' deviates form its advertised functionality during the execution of $T^{U'}(x)$, T will detect the error with probability $\geq \beta$.*

For example, if U' returns an error value, the probability that T checks out that the return value is wrong is 100 %, then we say (T,U) has a 1-checkable.

Definition 2.6. *($(\alpha,\beta)-$outsource-security) A pair of algorithm (T,U) are an $(\alpha,\beta)-$outsourcing-secure implementation of an algorithm Alg if they are both α-efficient and β-checkable.*

2.3. **System Model.** The system model we proposed includes three roles, namely the client T and the cloud servers U_1 and U_2, and we assume that one of them may be malicious. We briefly describe the overall execution process in the following steps:

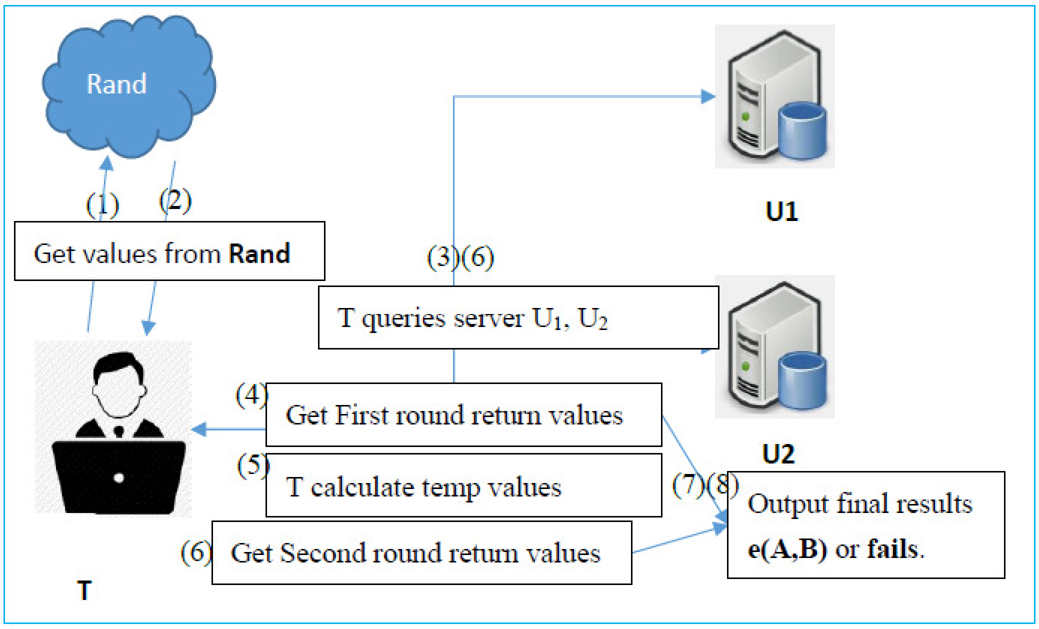

FIGURE 1. Our System Process

1. Given a bilinear pair e(A, B) to be calculated, the user T calls the Rand routine to perform the related pre-computation.
2. Rand routine returns two vector values to T. Among them, Rand is a trusted routine, which is used to help speed up calculations. The two vector values returned are as follows:
   Vector 1:
   $$a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P}), e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}$$
   Vector 2:
   $$a_1^{-1}, b_1^{-1}, a_1P, b_1\hat{P}, a_3P, b_3\hat{P}, e(a_1P, b_1\hat{P}), e(a_3P, b_1\hat{P})^{-1}, e(a_1P, b_3\hat{P})^{-1}$$
   where $e : G_1 \times G_2 \to G_T$ is a bilinear paring, P is the generator of G_1, $\hat{P}$ is the generator of G_2, and $a_i, b_i \in \mathbb{Z}_q^*$ for all i.

3. T sends request to the server U_1, U_2.
4. U_1 and U_2 respectively calculate the values of three sets of bilinear pairing and sends them to user T.
5. T calculate the relevant temporary storage value.
6. Using the temporarily stored value from above step, T sends the second round of calculation request to the servers U_1 and U_2, and at this time, two sets of bilinear paring values are calculated respectively.
7. T checks the return value from these server.
8. If the result of the step 6 is correct, then T calculates and outputs the value of e(A,B); otherwise, T outputs an error message.

3. **Verifiable outsourcing of bilinear pairings(VBP).** In this section, we introduce the verifiable outsourcing of bilinear pairings(VBP) algorithm proposed by Ren et al.[2]. This algorithm are 1-checkable. The detail process of the VBP is as follows these steps:

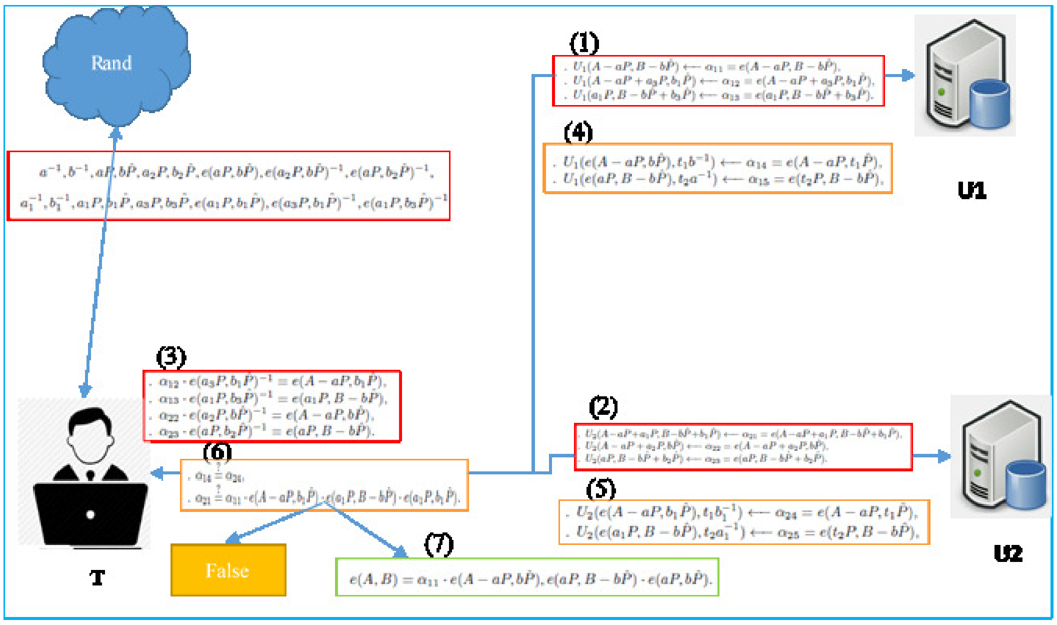

FIGURE 2. VBP algorithm execution Process

First, the user T calls the routine Rand, and Rand will return the following two sets of vector values. The two vector values are as following:

$$a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P}), e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}$$

$$a_1^{-1}, b_1^{-1}, a_1P, b_1\hat{P}, a_3P, b_3\hat{P}, e(a_1P, b_1\hat{P}), e(a_3P, b_1\hat{P})^{-1}, e(a_1P, b_3\hat{P})^{-1}$$

where $e : G_1 \times G_2 \to G_T$ is a bilinear paring, P is the generator of G_1, $\hat{P}$ is the generator of G_2, and $a_i, b_i \in \mathbb{Z}_q^*$ for all i.

The VBP algorithm executing step is as following:

1. T queries U_1 as follows:

$$U_1(A - aP, B - b\hat{P}) \to \alpha_{11} = e(A - aP, B - b\hat{P})$$

$$U_1(A - aP + a_3P, b_1\hat{P}) \to \alpha_{12} = e(A - aP + a_3P, b_1\hat{P})$$

$$U_1(a_1P, B - b\hat{P} + b_3\hat{P}) \to \alpha_{13} = e(a_1P, B - b\hat{P} + b_3\hat{P})$$

, where order is random. Our goal is to calculate the value of e(A,B), but the user T is a terminal with limited computing resources, so we call the Rand routine to calculate the relevant parameter values, combine the values of A and B with the return value of Rand, and hide the information A, B in the query parameters, and U_1 will not know that the actual value to be calculated is e(A, B).

2. Similarly, T queries U_2 as follows:

$$U_2(A - aP + a_1P, B - b\hat{P} + b_1\hat{P}) \to \alpha_{21} = e(A - aP + a_1P, B - b\hat{P} + b_1\hat{P})$$
$$U_2(A - aP + a_2P, b\hat{P} + b_1\hat{P}) \to \alpha_{22} = e(A - aP + a_2P, b\hat{P} + b_1\hat{P})$$
$$U_2(aP, B - b\hat{P} + b_2\hat{P}) \to \alpha_{23} = e(aP, B - b\hat{P} + b_2\hat{P})$$

, where order is random.

3. T obtains the return values $\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}$ and α_{23}. Then calculates the following temporary values.

$$\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1} = e(A - aP, b_1\hat{P})$$
$$\alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1} = e(a_1P, B - b\hat{P})$$
$$\alpha_{22} \cdot e(a_2P, b\hat{P})^{-1} = e(A - aP, b\hat{P})$$
$$\alpha_{23} \cdot e(aP, b_2\hat{P})^{-1} = e(aP, B - b\hat{P})$$

4. T sends the second round of calculation request to U_1.

$$U_1(e(A - aP, b\hat{P}), t_1b^{-1}) \to \alpha_{14} = e(A - aP, t_1\hat{P})$$
$$U_1(e(aP, B - b\hat{P}), t_2a^{-1}) \to \alpha_{15} = e(t_2P, B - b\hat{P})$$

,where $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_p^*$.

5. Similarly, T sends a request to U_2.

$$U_2(e(A - aP, b_1\hat{P}), t_1b_1^{-1}) \to \alpha_{24} = e(A - aP, t_1\hat{P})$$
$$U_2(e(a_1P, B - b\hat{P}), t_2a_1^{-1}) \to \alpha_{15} = e(t_2P, B - b\hat{P})$$

,where $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_p^*$.

6. T checks the following two sets of values to check whether they are equal, if they are equal, the verification is passed. If not, the verification fails.

$$\alpha_{14} \overset{?}{=} \alpha_{24}$$
$$\alpha_{21} \overset{?}{=} \alpha_{11} \cdot e(A - aP, b_1\hat{P}) \cdot e(a_1P, B - b\hat{P}) \cdot e(a_1P, b_1\hat{P}).$$

7. If is passes the inspection of the above step, T will calculate the value of e(A,B) below and output it, otherwise output the error message.

$$e(A, B) = \alpha_{11} \cdot e(A - aP, b_1\hat{P}) \cdot e(aP, B - b\hat{P}) \cdot e(aP, b\hat{P}).$$

4. **Attack of VBP.** In this section, we introduce the attack method proposed by Osmanbey Uzunkol et al.[3] and our attack method. First, we first introduce the method of Osmanbey Uzunkol et al..

The difference from the algorithm proposed by Ren et al. is that in the part of step 2, some changes are made to the return value, and the rest of the steps are the same. The main attack method is to use the addition and subtraction of the parameter value sent by T to get the value that should not be known to the outsourcing server, then attack T. The attack mainly focuses on verifiability. Suppose U_1 is a malicious and U_2 is an honest server. Firstly, it should guess the all correct positions of $\alpha_{1i}, i = 1, 2, 3$ with probability 1/6. Then, U_1 could compute two value $e(a_3P, b_1\hat{P}), 3(a_1P, b_3\hat{P})$ directly. Furthermore, with these two values, U_1 can calculate the following two values:

$$(\alpha_{12} \cdot e(a_3P, b_1\hat{P}^{-1}))^{-1} = e(A - aP, b_1\hat{P})^{-1}$$

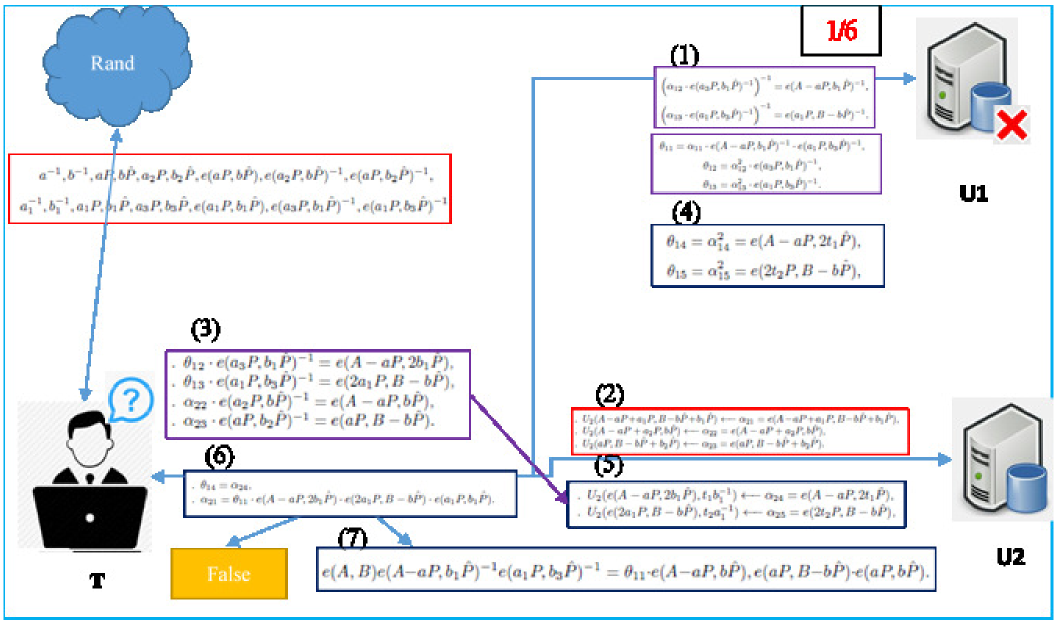
Rand
1/6
(1)
$(\alpha_{12}\cdot e(a_3P,b_1\hat{P})^{-1})^{-1}=e(A-aP,b_1\hat{P})^{-1}$,
$(\alpha_{13}\cdot e(a_1P,b_3\hat{P})^{-1})^{-1}=e(a_1P,B-b\hat{P})^{-1}$.
$\theta_{11}=\alpha_{11}\cdot e(A-aP,b_1\hat{P})^{-1}\cdot e(a_1P,b_3\hat{P})^{-1}$,
$\theta_{12}=\alpha_{12}^2\cdot e(a_3P,b_1\hat{P})^{-1}$,
$\theta_{13}=\alpha_{13}^2\cdot e(a_1P,b_3\hat{P})^{-1}$.
U1
$a^{-1},b^{-1},aP,b\hat{P},a_2P,b_2\hat{P},e(aP,b\hat{P}),e(a_2P,b\hat{P})^{-1},e(aP,b_2\hat{P})^{-1}$,
$a_1^{-1},b_1^{-1},a_1P,b_1\hat{P},a_3P,b_3\hat{P},e(a_1P,b_1\hat{P}),e(a_3P,b_1\hat{P})^{-1},e(a_1P,b_3\hat{P})^{-1}$
(4)
$\theta_{14}=\alpha_{14}^2=e(A-aP,2t_1\hat{P})$,
$\theta_{15}=\alpha_{15}^2=e(2t_2P,B-b\hat{P})$,
(3)
$\theta_{12}\cdot e(a_3P,b_1\hat{P})^{-1}=e(A-aP,2b_1\hat{P})$,
$\theta_{13}\cdot e(a_1P,b_3\hat{P})^{-1}=e(2a_1P,B-b\hat{P})$,
$\alpha_{22}\cdot e(a_2P,b\hat{P})^{-1}=e(A-aP,b\hat{P})$,
$\alpha_{23}\cdot e(aP,b_2\hat{P})^{-1}=e(aP,B-b\hat{P})$.
(2)
$U_2(A-aP+a_1P,B-b\hat{P}+b_1\hat{P})\longleftarrow\alpha_{21}=e(A-aP+a_1P,B-b\hat{P}+b_1\hat{P})$,
$U_2(A-aP+a_2P,b\hat{P})\longleftarrow\alpha_{22}=e(A-aP+a_2P,b\hat{P})$,
$U_2(aP,B-b\hat{P}+b_2\hat{P})\longleftarrow\alpha_{23}=e(aP,B-b\hat{P}+b_2\hat{P})$.
(5)
$U_2(e(A-aP,2b_1\hat{P}),t_1b_1^{-1})\longleftarrow\alpha_{24}=e(A-aP,2t_1\hat{P})$,
$U_2(e(2a_1P,B-b\hat{P}),t_2a_1^{-1})\longleftarrow\alpha_{25}=e(2t_2P,B-b\hat{P})$.
(6)
$\theta_{14}=\alpha_{24}$,
$\alpha_{21}=\theta_{11}\cdot e(A-aP,2b_1\hat{P})\cdot e(2a_1P,B-b\hat{P})\cdot e(a_1P,b_1\hat{P})$.
(7)
$e(A,B)e(A-aP,b_1\hat{P})^{-1}e(a_1P,b_3\hat{P})^{-1}=\theta_{11}\cdot e(A-aP,b\hat{P}),e(aP,B-b\hat{P})\cdot e(aP,b\hat{P})$.
T
False
U2

Figure 3. The Attack Process of Osmanbey Uzunkol et al.

$$(\alpha_{13}\cdot e(a_1P,b_3\hat{P}^{-1}))^{-1}=e(a_1P,B-b\hat{P})^{-1}$$

Then U_1 could send to the delegator T the following wrong values easily instead of $\alpha_{1i}, i = 1,2,3$:

$$\theta_{11}=\alpha_{11}\cdot e(A-aP,b_1\hat{P})\cdot e(a_1P,b_3\hat{P})^{-1}$$
$$\theta_{12}=\alpha_{11}^2\cdot e(a_3P,b_1\hat{P})^{-1}$$
$$\theta_{13}=\alpha_{13}^2\cdot e(a_1P,b_3\hat{P})^{-1}$$

When receiving the values $\theta_{1i},\alpha_{2i}i=1,2,3$, the delegator T would compute the following wrong values without knowing it:

$$\theta_{12}\cdot e(a_3P,b_1\hat{P})^{-1}=e(A-ap,2b_1\hat{P})$$
$$\theta_{13}\cdot e(a_1P,b_3\hat{P})^{-1}=e(2a_1p,B-b\hat{P})$$
$$\theta_{22}\cdot e(a_2P,b\hat{P})^{-1}=e(A-ap,b\hat{P})$$
$$\theta_{23}\cdot e(aP,b_2\hat{P})^{-1}=e(ap,B-b\hat{P})$$

We can see that there is a gap between the value of this step and the value that VBP should have received originally, and some of the parameter values were unknowingly modified to double. In step 4, after T sends the second round of calculation request, U_1 still replaces the original $\alpha_{1i}i=4,5$ with $\theta_{1i},i=45$.

$$\theta_{14}=\alpha_{14}^2=e(A-aP,2t_1\hat{P})$$
$$\theta_{15}=\alpha_{15}^2=e(2t_2P,B-b\hat{P})$$

Then the second server U_2 would compute

$$U_2(e(A-aP,2b_1\hat{P}),t_1b_1^{-1})\leftarrow\alpha_{24}=e(A-aP,2t_1\hat{P})$$
$$U_2(e(2a_1P,B-2b\hat{P}),t_2a_1^{-1})\leftarrow\alpha_{15}=e(2t_2P,B-b\hat{P})$$

Unfortunately, after such modification of the returned values, these values will also pass the verification step of step 6, but these values are wrong. This attack shows that the scheme in [2] does not satisfy the full verifiability, that is, it is not 1-Checkable, it should be changed to 5/6-Checkable. and it is a scheme in which a malicious U_1 could pass the verification step with wrong values with probability at least 1/6. Hence, U_1 could manipulate the output with probability at least 1/6.

Now, we introduce our attack method. We first assume that the data transmission process is insecure, and a third party can obtain relevant parameter values during the transmission process through eavesdropping. Therefore, the third party can obtain the following values. In the VBP algorithm, the final e(A,B) is calculated by multiplying the following values:

$$e(A,B) = \alpha_{11} \cdot e(A - aP, b_1\hat{P}) \cdot e(aP, B - b\hat{P}) \cdot e(aP, b\hat{P})$$

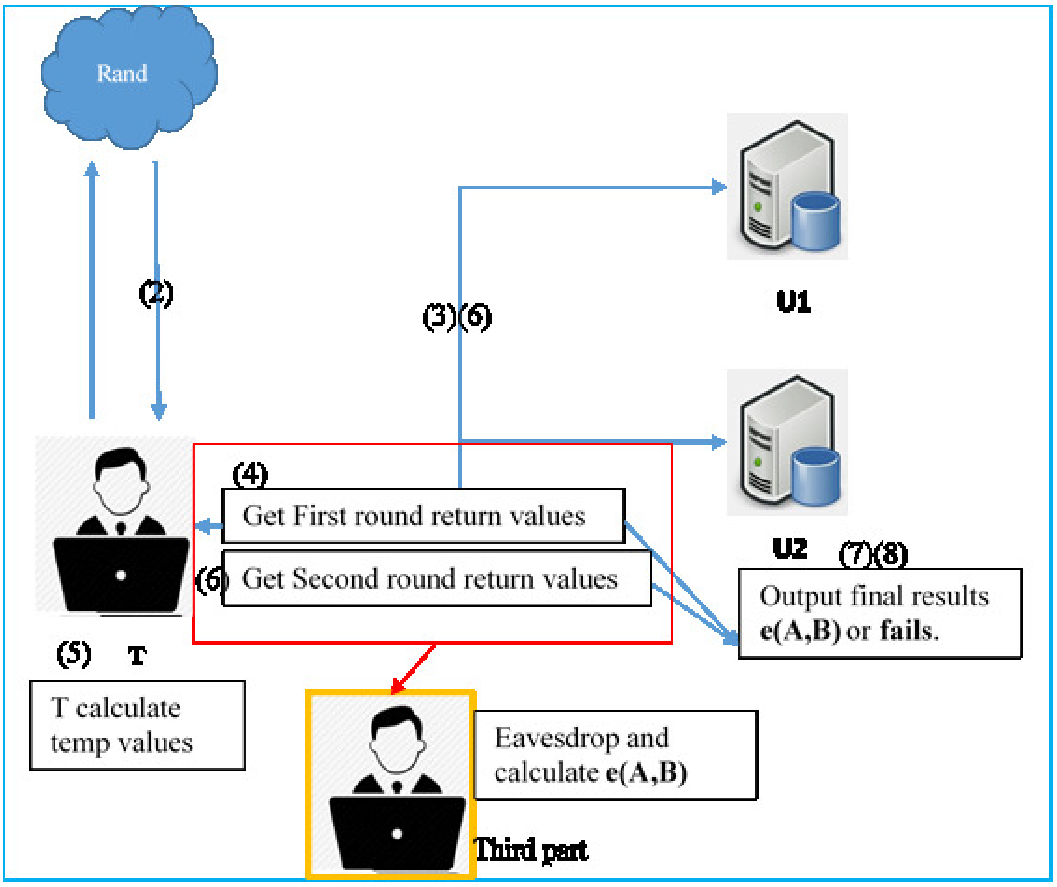
Rand
(2)
(3)(6)
U1
U2
(4)
Get First round return values
(6) Get Second round return values
(7)(8)
Output final results e(A,B) or fails.
(5) T
T calculate temp values
Eavesdrop and calculate e(A,B)
Third part

FIGURE 4. Transmission process of third-party eavesdropping

In step 1:
After T sends the first round of calculation request to U_1, if the third party guesses the correct order of T's request, the parameters(A-aP) and($b_1\hat{P}$) can be obtained, and it can calculate value of e(A-aP,$b_1\hat{P}$).
In step 2:
Similarly, a third party can obtain the parameters (aP) and (b$\hat{P}$), and then calculate the value of e(aP,b$\hat{P}$). The value of α_{11} can also be eavesdropped from the transmission process. Finally, the third party can still calculate the value of e(A,B) after eavesdropping. Therefore, the VBP algorithm proposed by Ren et al. does not satisfy the privacy in calculation mentioned in section 2. In the next section, we will introduce our improvement method and show that this method can avoid attacks.

5. **Modified VBP.** After the introduction of the first two parts, in this part we will introduce our improved part of the VBP algorithm proposed by Ren et al. We named our method Modified VBP (MVBP) and explained that our method can prevent Attacks are introduced in the chapter 4. Next, we will introduce our method in detail.

The difference between our method and VBP is that in the parts of step 1 and step 2, T makes some changes to the parameter values when U_1 and U_2 send the first round of calculation requests. We add $r_i, r_i^{(} - 1) \in \mathbb{Z}_P^*, i = 1\ 6$ to each request parameter. The detailed process of our method is introduced as follows.

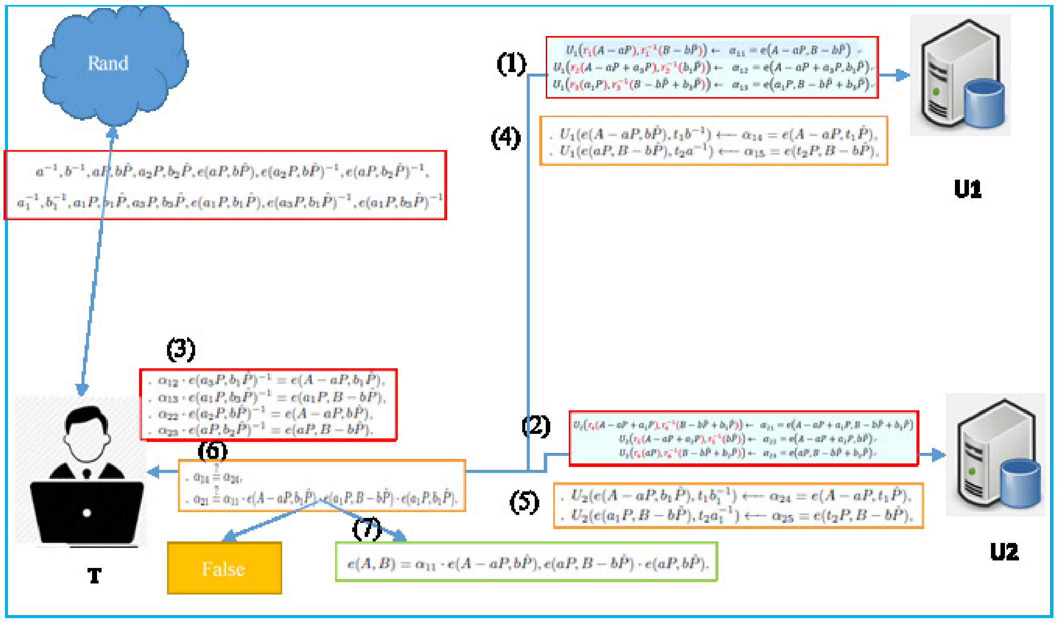

FIGURE 5. Process of Our MVBP

1. T queries U_1 follows:

$$U_1(r_1(A - aP), r_1^{-1}(B - b\hat{P})) \to \alpha_{11} = e(A - aP, B - b\hat{P})$$
$$U_1(r_2(A - aP + a_3P), r_2^{1}(b_1\hat{P})) \to \alpha_{12} = e(A - aP + a_3P, b_1\hat{P})$$
$$U_1(r_3(a_1P), r_3^{-1}(B - b\hat{P} + b_3\hat{P})) \to \alpha_{13} = e(a_1P, B - b\hat{P} + b_3\hat{P})$$

2. Similarly, T queries U_2 as follows:

$$U_2(r_4(A - aP + a_1P), r_4^{-1}(B - b\hat{P} + b_1\hat{P})) \to \alpha_{21} = e(A - aP + a_1P, B - b\hat{P} + b_1\hat{P})$$
$$U_2(r_5(A - aP + a_2P), r_5^{-1}(b\hat{P} + b_1\hat{P})) \to \alpha_{22} = e(A - aP + a_2P, b\hat{P} + b_1\hat{P})$$
$$U_2(r_6(aP), r_6^{-1}(B - b\hat{P} + b_2\hat{P})) \to \alpha_{23} = e(aP, B - b\hat{P} + b_2\hat{P})$$

The following step is same with the original VBP. We can find that just adding 6 multiplication calculations will not affect the overall performance. Because the computational cost of bilinear pairing is much greater than that of multiplication.

Next, we introduce our method flow and show that it can prevent the attacks in section 4.The algorithm flow is shown in Figure 6.
When the attack method tries to calculate the value $e(a_3P, b_1\hat{P})^{(} - 1)$, $e(a_1P, b_3\hat{P})^{(} - 1)$. it will calculate the following wrong value k_1, k_2.

$$k_1 = e(r_2(A - aP + a_3P) - r_1(A - aP), r_2^{-1}(b_1\hat{P}))^{-1}$$
$$k_2 = e(r_3(a_1P), r_3^{-1}(B - b\hat{P} + b_3\hat{P}) - r_1^{-1}(B - b\hat{P}))^{-1}$$

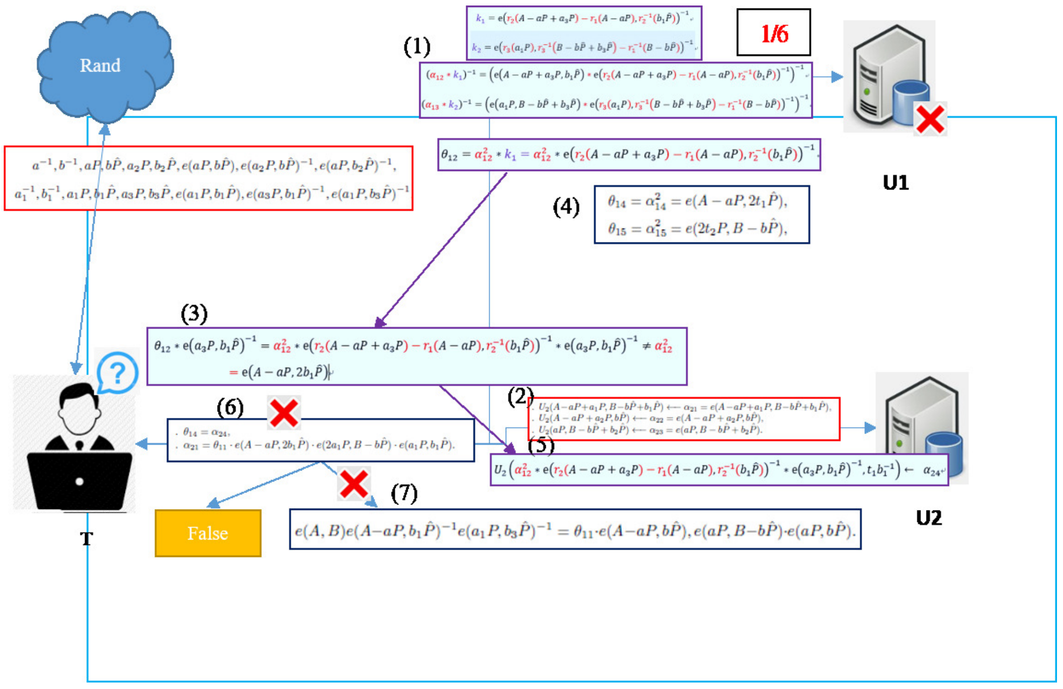

Figure 6. Our MVBP's prevent attack flow

Then $e(A-aP, b_1\hat{P})^{-1}$ and $e(a_1P, B-b\hat{P})^{-1}$ are miscalculated into the following two values:

$$(\alpha_{12}\cdot k_1)^{-1} = (e(A-aP+a_3P, b_1\hat{P})\cdot e(r_2(A-aP+a_3P)-r_1(A-aP), r_2^{-1}(b_1\hat{P}))^{-1})^{-1}$$

$$(\alpha_{13}\cdot k_2)^{-1} = (e(a_1P, B-b\hat{P}+b_3\hat{P})\cdot e(r_3(a_1P), r_3^{-1}(B-b\hat{P}+b_3\hat{P})-r_1^{-1}(B-b\hat{P}))^{-1})^{-1}$$

Then, U_1 easily returns the following error value to T instead of $\alpha_{1i}, i=1,2,3$, and it becomes the following value. We only need to take θ_{12} as an example, and the rest of θ_{11}, θ_{13} are so on and so forth.

$$\theta_{12} = \alpha_{12}^2\cdot k_1 = \alpha_{12}^2\cdot e(r_2(A-aP+a_3P)-r_1(A-aP), r_2^{-1}(b_1\hat{P}))^{-1}$$

When T receives this value, it will calculate the following value, which will be used as a parameter to send to U_2 in step 5:

$$\begin{aligned}\theta_{12}\cdot e(a_3P, b_1\hat{P})^{-1} =& \alpha_{12}^2\cdot e(r_2(A-aP+a_3P)-r_1(A-aP), r_2^{-1}(b_1\hat{P}))^{-1}\cdot e(a_3P, b_1\hat{P})^{-1}\\ \neq& \alpha_{12}^2\end{aligned}$$

Then U_2 calculates the following error value:

$$U_2(e(A-aP+a_3P, b_1\hat{P})\cdot e(r_2(A-aP+a_3P)-r_1(A-aP), r_2^{-1}(b_1\hat{P}))^{-1}, t_1b_1^{-1}) \to \alpha_{24}$$

Then in the step 6 verification step, it will be detected as a malicious return value. Finally, in the case of third-party eavesdropping, our method can also prevent to calculate correct value after eavesdropping, as explained below:
In step 1:
Due to the addition of $r_i, r_i^{-1}\in\mathbb{Z}_p^*, i=1,2,3,4,5,6$, the values of $(A-ap)$ and $(b_1\hat{P})$ cannot be calculated correctly, so $e(A-aP, b_1\hat{P})$ is .
In step 2:
Similarly, due to the addition of $r, r_i^{-1}\in\mathbb{Z}_p^*, i=1,2,3,4,5,6$, the values of (aP) and $(b\hat{P})$ cannot be calculated correctly, so $e(aP, b\hat{P})$ is. Therefore, our method is safe in this case.

6. **Analysis and Proof.**

6.1. **One-malicious Model.** In one-malicious model, we assume that U_1 is malicious, and the attack method in [3] should have a 1/6 probability of guessing the position of $\alpha_{1i}, i = 1, 2, 3$. Then, U_1 can correctly calculate the values $e(a_3P, b_1\hat{P}), e(a_1P, b_3\hat{P})$. Then the method can successfully attack. But using our scheme, the attack method will calculate the wrong values of $\theta_{11}, \theta_{12}, \theta_{13}$.
In this way, it will know that U_1 is malicious, and then, in the final verification step, T finally outputs the error message "fails".

6.2. **Comparison.** In this section, we compare the amount of computation between different bilinear paring outsourcing algorithm. As shown in below table.

Table 1. Comparison of Calculation amount of Different Algorithm

Operation / algorithm	VBP[2]	DBP[6]	MVBP
PA(Time)	8	6	8
M(Time)	14	19	17
Mexp(Time)	0	0	0
SM(Time)	0	0	0
Pair(U)	6	10	6
Mexp(U)	4	0	4
Servers	2	2	2
Checkability	1	1	1
Rounds	4	2	2

The meaning of symbols in the table:

- PA: Point addition in additive group G_1 or G_2.
- M: Multiplication in G_T.
- Eexp: Modular exponentiation in G_T.
- SM: Scale multiplication in additive group G_1 or G_2.
- Pair: Bilinear pairing computation.
- Servers: The number of server required.
- Checkability: Checkability value and range of value in [0,1].
- Rounds: The number of rounds of calculating a pairing.

As shown in Table 1 below, it can be seen that the difference between our MVBP and the original VBP algorithm is only Three M operations are added, but safety is improved. Compared with the DBP algorithm, although the overall performance and safety may not be as good as the DBP algorithm, but our MVBP has four fewer pairing calculations.

6.3. **Security Proof.**

Theorem 6.1. *In a one-malicious model, if the input is honestly, secret; honestly, protected; adversarially, protected, input, the algorithm is an outsourcing-security implementation of algorithm MVBP.*

Proof: The proof is similar to [2, 5], let A=(E,U_1', U_2') is a malicious attacker(adversary) in one-malicious model.

This proof that we cannot be sure which one is malicious. We only assume that one of them is malicious, but we don't know which one is. Therefore, the symbol is first added with ', which means that both are untrusted.

Pair One: $EVIEW_{real} \sim EVIEW_{ideal}$
If the input (A, B) is honestly, secret, the behavior of the simulator S_1 will be the same as the actual execution. The behavior of the simulator is as follows. After receiving the input in the i-th round, S_1 will ignore it and instead perform three or two random calculation requests, such as requesting (P_i, Q_i) from U_1' and U_2'. Then , the simulator tests all the output of the server and sets the basis value based on the result returned by the server. If S_1 detects an error, the simulator S_1will save its state and output $Y_p^i = "error", Y_u^i = \emptyset, replace^i = 1$. If no error is detected, output $Y_p^i = \emptyset, Y_u^i = \emptyset, replace^i = 0$; Otherwise, S_1 selects a random value r and outputs $Y_p^i = r, Y_u^i = \emptyset, replace^i = 1$, in eith case, the appropriate state will be saved.
The input is randomly selected. In ideal and actual experiments, all calculation requests made by T to the server are computationally random. Therefore, we say that the input given to (U_1', U_2') in ideal and real experiments is computationally blind. If (U_1', U_2') works honestly in the i-th round of the real experiment, then $T^{(U_1',U_2')}$ can perform the algorithm perfectly in the real experiment, and S_1 chooses not to replace the output, So we can easily get the result $EVIEW_{real} \sim EVIEW_{ideal}$. If one of (U_1', U_2') in the i-th round is dishonest, T and S_1 will detect all the errors caused by those servers and cause the output "error". So we can also get $EVIEW_{real} \sim EVIEW_{ideal}$.

Pair Two: $UVIEW_{real} \sim UVIEW_{ideal}$
The behavior of the simulator S_2 is as follows. After receiving the input in the i-th round, S_2 will ignore it and make three or two random calculation requests for U_1' and U_2'. After that, S_2 saves its own state and the state of inputs (U_1', U_2'). These differences between the ideal experiment and the real experiment can be easily distinguished by E. However, E cannot communicate, access, or exchange information with $(U_1',\ U_2')$. Because in the i-th real experiment, T always makes the input seem random to $(U_1',\ U_2')$. And in an ideal experiment, S_2 is always to create random and independent calculation requests. Therefore, for each round we can always get $UVIEW_{real}\ UVIEW_{ideal}$.

Theorem 6.2. *In a one-malicious model, $(T, (U_1, U_2))$ is our algorithm's $(\mathcal{O}(\frac{1}{n}), 1)$-outsourcing-secure implementation, where the value of n is the bit length of q.*

Proof:
In the proposed algorithm MVBP, T calls the Rand routine once, and performs 8 addition operations and 17 multiplication operations to obtain the value of e(A, B). In addition, it is well known that it needs to perform $\mathcal{O}(n)$ multiplication in a finite filed to calculate bilinear pairing [4]. Therefore, our algorithm (T,(U_1, U_2)) is an $\mathcal{O}(\frac{1}{n})$-efficient implementation. On the other hand, in the proposed algorithm, all computing requests of T to the untrusted server can be successfully verified. Therefore, if one of (U_1, U_2) is dishonest, the error will be detected with a probability of 1.

6.4. **Proof of Correctness.** In this section, we will prove the correctness of our method. The difference between our MVBP and VBP is that for steps 1, 2, T makes some changes when sending calculation requests to U_1, U_2 in the first round. We add $r_i, r_i^(-1) \in \mathbb{Z}_p^*, i = 1, 2, 3, 4, 5, 6$ to each request, as follows:
In step 1:

$$U_1(r_1(A - aP), r_1^{-1}(B - b\hat{P})) \to \alpha_{11} = e(A - aP, B - b\hat{P})$$

$$U_1(r_2(A - aP + a_3P), r_2^1(b_1\hat{P})) \to \alpha_{12} = e(A - aP + a_3P, b_1\hat{P})$$

$$U_1(r_3(a_1P), r_3^{-1}(B - b\hat{P} + b_3\hat{P})) \to \alpha_{13} = e(a_1P, B - b\hat{P} + b_3\hat{P})$$

In step 2:

$$U_2(r_4(A - aP + a_1P), r_4^{-1}(B - b\hat{P} + b_1\hat{P})) \to \alpha_{21} = e(A - aP + a_1P, B - b\hat{P} + b_1\hat{P})$$
$$U_2(r_5(A - aP + a_2P), r_5^{-1}(b\hat{P} + b_1\hat{P})) \to \alpha_{22} = e(A - aP + a_2P, b\hat{P} + b_1\hat{P})$$
$$U_2(r_6(aP), r_6^{-1}(B - b\hat{P} + b_2\hat{P})) \to \alpha_{23} = e(aP, B - b\hat{P} + b_2\hat{P})$$

According to the related calculation method of bilinear pairing can be extracted forward, take the first request in step 1 as an example :

$$\begin{aligned} U_1(r_1(A - aP), r_1^{-1}(B - b\hat{P})) &= e(r_1(A - aP), r_1^{-1}(B - b\hat{P})) \\ &= (r_1 \cdot r_1^{-1}) \cdot (e(A - aP, B - b\hat{P})) \to \alpha_{11} \\ &= e(A - aP, B - b\hat{P}) \end{aligned}$$

It can be seen that after adding $r_i, r_i^(-1) \in \mathbb{Z}_p^*, i = 1, 2, 3, 4, 5, 6$, the last calculated value of our MVBP in steps 1 and 2 is the same as VBP, so it will not affect the subsequent steps 3 to step 7 calculations. That is, in the end we must be able to correctly calculate the value of e(A, B) we want.

7. **Conclusion.** In this paper, we modified the VBP algorithm proposed by Ren et al. [2] and proposed the Modified VBP (MVBP) algorithm. It strengthens the security, is not successfully attacked by the existing attack method [3], and does not affect the efficiency of the calculation. At the same time, the original Checkability is improved. With the attack method [3], the Checkability is only 5/6-Checkable. But after our improvement, it becomes 1-Checkable. That is, it can be 100% checked whether the outsourced return value is correct. Compared with other existing algorithms, although VBP has been improved to make it more secure, the overall performance and security may not be as good as the DBP algorithm [6]. But our MVBP has also reduced the amount of pairing calculations four times. Therefore, future work is how to improve MVBP to make it better than DBP? Or improve the DBP algorithm to make it faster and more efficient.

REFERENCES

[1] Yang, P., Cao, Z., & Dong, X. (2008). Fuzzy Identity Based Signature. IACR Cryptol. ePrint Arch., 2008, 2.

[2] Ren, Y., Ding, N., Wang, T., Lu, H., & Gu, D. (2016). New algorithms for verifiable outsourcing of bilinear pairings. Science China Information Sciences, 59(9), 99103.

[3] Uzunkol, O., Kalkar, Ö., & Sertkaya, I. (2017). Fully Verifiable Secure Delegation of Pairing Computation: Cryptanalysis and An Efficient Construction. IACR Cryptol. ePrint Arch., 2017, 1173.

[4] Chen, X., Susilo, W., Li, J., Wong, D. S., Ma, J., Tang, S., & Tang, Q. (2015). Efficient algorithms for secure outsourcing of bilinear pairings. Theoretical Computer Science, 562, 112-121.

[5] Hohenberger, S., & Lysyanskaya, A. (2005, February). How to securely outsource cryptographic computations. In Theory of Cryptography Conference (pp. 264-282). Springer, Berlin, Heidelberg.

[6] Dong, M., Ren, Y., & Zhang, X. (2017). Fully Verifiable Algorithm for Secure Outsourcing of Bilinear Pairing in Cloud Computing. KSII Transactions on Internet & Information Systems, 11(7).