Asian Journal of Mathematics and Computer Research

Volume 31, Issue 2, Page 87-95, 2024; Article no.AJOMCOR.12064
ISSN: 2395-4205 (P), ISSN: 2395-4213 (O)

Bilinear Pairings: A Secured Outsourcing with a Single Server

Qing Ying [a*]

[a] *College of Science, North China University of Technology, Beijing-100144, P. R. China.*

Author's contribution

The sole author designed, analysed, interpreted and prepared the manuscript.

Article Information

DOI: 10.56557/AJOMCOR/2024/v31i28681

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://prh.ikprress.org/review-history/12064

Original Research Article

Received: 26/02/2024
Accepted: 30/04/2024
Published: 04/05/2024

Abstract

The computation of bilinear pairs is the most computationally expensive computation for bilinear pair-based cryptographic protocols in practice. Currently, most bilinear pairing outsourcing algorithms have little checkability, or the outsourcer requires two servers for expensive calculations. In this paper, we propose an efficient and secure single-server bilinear pairing outsourcing algorithm, which allows the outsourcing party to detect dishonest errors in cloud servers with probability 1, and the outsourcing party does not need to perform some complex computations.

Keywords: Elliptic curve; outsourcing; bilinear pairing.

1 Introduction

With the growth of cloud computing and the popularity of mobile devices, computing outsourcing has attracted a lot of attention. Outsourcing computing tasks to cloud computing servers can significantly reduce users' computing costs. However, the server is not completely trusted and it faces many security challenges. To prevent untrusted cloud servers leaking and misusing users' data, users need to deliver to cloud server operations. The data is encrypted or blinded to ensure data privacy. In addition, users can check the computation results returned

---

**Corresponding author: Email: bestadver@gmail.com, gaussandjacobi@gmail.com;*

from the cloud server to get the correct computation results [1].

In the field of cryptography, outsourcing expensive calculations to semi-trusted devices has been extensively studied. Chaum and Pedersen [2] introduced the definition of wallets with observers, installing hardware devices on users' hosts to perform complex operations. Chaum and Pedersen [2] formally defined this model and proposed an outsource algorithm to compute modular exponentiation used two servers that cannot collaborate with each other. Hohenberger and Lysyanskaya [3] proposed an improved algorithm. In this improved algorithm, the computational efficiency and verifiable probability of the outsourced algorithm are improved. Green et al. [1] proposed an outsourced decryption algorithm with attribute-based encryption (ABE) that reduces the user's decryption cost. However, the algorithm cannot verify the correctness of the outsourced results. Green et al. [4] proposed a verifiable outsourced decryption algorithm in the ABE scheme that users can check the outsourcing results.

In the Cryptography, Bilinear pairs are widely used in many aspects. Therefore, a lot of work has been done on improving computational efficiency [5,6]. However, users need to perform some highly complex operations such as modular exponentiations. The complexity of these operations was equivalent to the calculation of bilinear pairings. Subsequently, this problem has been found in the outsourcing of bilinear pairs [7,8,9,10]. Chen et al. [11] presents practical bilinear-pairing outsourcing algorithm used two untrustworthy servers. The user only needs to perform five times of point addition and four times of modulus. The multiplication operation did not require any high-complexity operation and was suitable for the calculation of limited devices. The probability of verification of the algorithm was only 50%, that is, the server can still deceive users with a probability of 50%. Recently, [12] and [13] independently propose two bilinear algorithm based on two servers. In these model, the probability that the outsourcer can detect an error is 1, but the cost of outsourcing to two servers is higher than the cost of outsourcing to a single server.

In this paper, we propose a fast and secure single-server bilinear pairing algorithm for outsourcing. In this model, the probability of an error being detected by the outsourcer is 100% and the outsourcer does not have to perform any complex calculations at the same time. And we can reduce the cost of outsourcing to the server.

2 Preliminaries

Let $G_1 = \langle P_1 \rangle$ and $G_2 = \langle P_2 \rangle$ be two additive cyclic groups, and $|G_1| = |G_2| = q$. Let G_T be a multiplicative cyclic group with order q . The bilinear pairing is a map $e: G_1 \times G_2 \to G_T$ with the following properties:

1. Bilinear: $e(aS, bT) = e(S,T)^{ab}$, $\forall S \in G_1, T \in G_2$, $a,b \in Z_q^*$.
2. Non-degenerate: $\exists S \in G_1, T \in G_2$ such that $e(S,T) \neq 1$.
3. Computable: There exist an efficient algorithm to compute $e(S,T)$.

Suppose that $A\lg$ be a cryptographic algorithm. Informally, we say that T securely outsources some work to U and that (T, U) is an implementation of $A\lg$ if T and U implement $A\lg$, that is $A\lg = T^U$, and suppose that an adversary V (and not U), gains oracle access to T , which logs all computations made by T over time, and tries to take malicious action. But V cannot learn any interesting information about T^V 's inputs and outputs. In the following, we present the formal definition of cryptographic algorithm security outsourcing [14].

Definition 1 (Algorithm with outsource-I/O) We say that Algorithm Alg adheres to the outsourcing input/output specification if it produces three outputs from the following five inputs. Let E is a hostile environment，V is the adversarial software that runs in place of Oracle U. The first three inputs are generated by the honest party and categorised according to how well the adversary $A=(E,V)$ knows them. The first input is called the honest secret input, which is unknown to both E and V . The second input is known as the protected honesty input, which may be known to E, but not to V. The third input is called the honest, unprotected input, and both E and V may know it. In addition, there are two inverse selection inputs generated by the environment E: antagonistic, protected inputs, where E knows it but V does not; and antagonistic, unprotected inputs, where both E and V may know it. Similarly, the first output is called secret and neither E nor V knows it; the second output is protected and E may know it but V does not; and the third output is unprotected and both E and V may know it.

Definition 2 (Outsource-Security) Let $\left(T,U\right)$ are an outsource-secure implementation of algorithm $A\lg$, if the following requirements are met:

1. Correctness: T^U correctly implements algorithm $A\lg$.
2. Security: For all probabilistic polynomial time opponents $\mathrm{A}=\left(\mathrm{E},\mathrm{U}^{'}\right)$, there exists a probabilistic expectation polynomial time simulator S_1 and S_2 such that the following pairs of random variables are computationally indistinguishable.

Pair One: $EVIEW_{real} \square EVIEW_{ideal}$, This means that E knows nothing about the inputs and outputs during T^U . The adversarial environment E obtain a view by engaging in the following real-world process:

$$EVIEW_{real}^{i}=\left(istate^{i},x_{hs}^{i},x_{hp}^{i},x_{hu}^{i}\right)\leftarrow I\left(1^{k},istate^{i-1}\right);$$

$$\left(estate^{i},j^{i},x_{ap}^{i},x_{au}^{i},stop^{i}\right)\leftarrow E\left(1^{k},EVIEW_{real}^{i-1},x_{hp}^{i},x_{hu}^{i}\right);$$

$$\left(tstate^{i},ustate^{i},y_{s}^{i},y_{p}^{i},y_{u}^{i}\right)\leftarrow T^{U^{'}\left(ustate^{i-1}\right)}\left(tstate^{i-1},x_{hs}^{j^{i}},x_{hp}^{j^{i}},x_{hu}^{j^{i}},x_{ap}^{i},x_{au}^{i}\right):\left(estate^{i},y_{p}^{i},y_{u}^{i}\right)$$

$$EVIEW_{real}=EVIEW_{ideal}\quad if\quad stop^{i}=TRUE$$

The real process runs in rounds. In round i, the honest (secret, protected, and unprotected inputs $\left(x_{hs}^{i},x_{hp}^{i},x_{hu}^{i}\right)$ A is selected using an honest, stateful process i, which is not accessible to environment E. Then E generates the following 5 outputs

a. the value of its $estate^i$ variable so that it remembers what it did the next time it is called;
b. Hand over the previously generated honest inputs $\left(x_{hs}^{i},x_{hp}^{i},x_{hu}^{i}\right)$ to T^V (note that E can only specify the index j^i of these inputs, but not their values);
c. x_{ap}^{i} , Protected adversarial inputs;
d. x_{au}^{i} , Adversarial, unprotected inputs;
e. Boolean variable $stop^i$, which determines whether round i is the last round of the process.

Then algorithm T^V runs on inputs $\left(tstate^{i-1},x_{hs}^{j^{i}},x_{hp}^{j^{i}},x_{hu}^{j^{i}},x_{ap}^{i},x_{au}^{i}\right)$, where $tstate^{i-1}$ is T's previously saved state, and produces a new state $tstate^i$ for T : secret y_s^i , protected y_p^i ,unprotected y_u^i . The oracle V is given by state $ustate^{i-1}$. The current state of V is saved in the variable $ustate^i$. The view of the real process in round i includes $estate^i$, y_p^i and y_u^i . The overall point of view E in the actual process is simply its point of view in the final round (that is i for $stop^i=TRUE$).

The ideal process:

$$EVIEW^i_{ideal} = \left(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}\right) \leftarrow I\left(1^k, istate^{i-1}\right);$$
$$\left(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i\right) \leftarrow E\left(1^k, EVIEW^{i-1}_{ideal}, x^i_{hp}, x^i_{hu}\right);$$
$$\left(astate^i, y^i_s, y^i_p, y^i_u\right) \leftarrow Alg\left(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}\right);$$
$$\left(sstate^i, ustate^i, Y^i_p, Y^i_u, replace^i\right) \leftarrow S_1^{U'\left(ustate^{i-1}\right)} \times \left(sstate^{i-1}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}, y^i_p, y^i_u\right);$$
$$\left(z^i_p, z^i_u\right) = replace^i\left(Y^i_p, Y^i_u\right) + \left(1 - replace^i\right)\left(y^i_p, y^i_u\right) : \left(estate^i, z^i_p, z^i_u\right)$$

Pair Two: $UVIEW_{real} \square UVIEW_{ideal}$, That is, the untrusted software V written by E will not know anything about the inputs and outputs. Untrusted software V obtains views by participating in a real process, which is described in **Pair One**. $UVIEW_{real} = ustate^i$ when $stop^i = TRUE$.

The ideal process:

$$UVIEW^i_{ideal} = \left(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}\right) \leftarrow I\left(1^k, istate^{i-1}\right);$$
$$\left(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i\right) \leftarrow E\left(1^k, estate^{i-1}, x^i_{hp}, x^i_{hu}, y^i_p, y^i_u\right);$$
$$\left(astate^i, y^i_s, y^i_p, y^i_u\right) \leftarrow Alg\left(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}\right);$$
$$\left(sstate^i, ustate^i\right) \leftarrow S_2^{U'\left(ustate^{i-1}\right)}\left(sstate^{i-1}, x^{j^i}_{hu}, x^i_{au}\right) : \left(ustate^i\right)$$
$$UVIEW_{real} \square UVIEW_{ideal} \quad if \quad stop^i = TRUE$$

In the ideal process, we have a stateful simulator S_2 that is equipped with only unprotected inputs $\left(x^i_{hu}, x^i_{au}\right)$ and queries V.

3 New Outsourcing Algorithm of Bilinear Pairings

3.1 Security model

Similar to Hohenberger and Lysyanskaya [14], we use $Rand$ in order to speed up calculations. The inputs for $Rand$ are G_1, G_2, e and possibly some other (random) values, the outputs for each call are a random independent six-tuple $\left(V_1, V_2, \delta_1 V_1, \delta_2 V_1, \delta_2 V_2, e\left(\delta_1 V_1, \delta_2 V_2\right)\right)$, where $\delta_1, \delta_2 \in \square^*_q$, $V_1 \in G_1$, and $V_2 \in G_2$. A simple way to implement this functionality is to have a trusted server pre-compute a random, independent table of six-tuple and then load it into T's memory. Each time $Rand$ is called, it simply retrieves a new six-tuple in the table (the lookup table method).

3.2 Outsourcing algorithm

In our paper, T invokes the subroutine $Rand$ to outsourcing its pairing computations to U. One requirement is that adversary A cannot know any useful information about inputs and outputs.

Note that $A \in G_1$, $B \in G_2$ may be secret two random points and the output $e(A, B)$ is always secret or protected. Furthermore, both A and B know nothing about U computationally. The process of input $\left(\Lambda_1, \Lambda_2\right)$

and output $e\left(\Lambda_1,\Lambda_2\right)$ is denoted as $U\left(\Lambda_1,\Lambda_2\right)\to e\left(\Lambda_1,\Lambda_2\right)$. The newly constructed outsourcing algorithm in this paper includes the following steps:

1. By running Rand using U and T, the following two blinding six-tuples $\left(V_1,V_2,\delta_1V_1,\delta_2V_1,\delta_2V_2,e\left(\delta_1V_1,\delta_2V_2\right)\right)$ and $\left(V_3,V_4,\delta_3V_3,\delta_4V_3,\delta_4V_4,e\left(\delta_3V_3,\delta_4V_4\right)\right)$ are obtained. We denote $\lambda_1=e\left(\delta_1V_1,\delta_2V_2\right)$ and $\lambda_2=e\left(\delta_3V_3,\delta_4V_4\right)$.

2. The main trick of our paper is to logically split A and B into random fragments that can be computed by U .Without loss of generality, let $\alpha_1=e\left(A+\delta_1V_1,B+\delta_2V_2\right)$, $\alpha_2=e\left(\delta_2A+\delta_2V_1,V_2\right)$, $\alpha_3=e\left(V_1,\delta_1\left(B+V_2\right)\right)$, $\alpha_4=e\left(\alpha_1V_1+\alpha_2V_1,V_2\right)$, $\beta_1=e\left(A+\delta_3V_3,B+\delta_4V_4\right)$, $\beta_2=e\left(\delta_4A+\delta_4V_3,V_4\right)$, $\beta_3=e\left(V_3,\delta_3\left(B+V_4\right)\right)$, and $\beta_4=e\left(\alpha_3V_3+\alpha_4V_3,V_4\right)$.

Note that

$$
\begin{aligned}
\alpha_1&=e\left(A,B\right)e\left(A,\delta_2V_2\right)e\left(\delta_1V_1,B\right)e\left(\delta_1V_1,\delta_2V_2\right)\\
\alpha_2&=e\left(A,\delta_2V_2\right)e\left(V_1,\delta_2V_2\right)\\
\alpha_3&=e\left(\delta_1V_1,B\right)e\left(\delta_1V_1,V_2\right)\\
\alpha_4&=e\left(\delta_1V_1,V_2\right)e\left(\delta_2V_1,V_2\right)
\end{aligned}
$$

And

$$
\begin{aligned}
\beta_1&=e\left(A,B\right)e\left(A,\delta_4V_4\right)e\left(\delta_3V_3,B\right)e\left(\delta_3V_3,\delta_4V_4\right)\\
\beta_2&=e\left(A,\delta_4V_4\right)e\left(V_3,\delta_4V_4\right)\\
\beta_3&=e\left(\delta_3V_3,B\right)e\left(\delta_3V_3,V_4\right)\\
\beta_4&=e\left(\delta_3V_3,V_4\right)e\left(\delta_4V_3,V_4\right)
\end{aligned}
$$

Therefore, $e\left(A,B\right)=\alpha_1\alpha_2^{-1}\alpha_3^{-1}\lambda_1^{-1}\alpha_4=\beta_1\beta_2^{-1}\beta_3^{-1}\lambda_2^{-1}\beta_4$.

(1) T uses a random order query U as follows

$$
\begin{aligned}
&U\left(A+\delta_1V_1,B+\delta_2V_2\right)\to\alpha_1\\
&U\left(\delta_2A+\delta_2V_1,V_2\right)\to\alpha_2\\
&U\left(V_1,\delta_1\left(B+V_2\right)\right)\to\alpha_3\\
&U\left(\alpha_1V_1+\alpha_2V_1,V_2\right)\to\alpha_4\\
&U\left(A+\delta_3V_3,B+\delta_4V_4\right)\to\beta_1\\
&U\left(\delta_4A+\delta_4V_3,V_4\right)\to\beta_2\\
&U\left(V_3,\delta_3\left(B+V_4\right)\right)\to\beta_3\\
&U\left(\alpha_3V_3+\alpha_4V_3,V_4\right)\to\beta_4
\end{aligned}
$$

(2) Finally, T queries U and outputs the correct result, i.e., $\alpha_1\alpha_2^{-1}\alpha_3^{-1}\lambda_1^{-1}\alpha_4$ and $\beta_1\beta_2^{-1}\beta_3^{-1}\lambda_2^{-1}\beta_4$ for test checks. If $\alpha_1\alpha_2^{-1}\alpha_3^{-1}\lambda_1^{-1}\alpha_4=\beta_1\beta_2^{-1}\beta_3^{-1}\lambda_2^{-1}\beta_4$, T can compute $e(A,B)=\alpha_1\alpha_2^{-1}\alpha_3^{-1}\lambda_1^{-1}\alpha_4$, otherwise, T outputs error.

Remark 1 T can be easily computed $-P$ for any random point $P\in G_1$ or $P\in G_2$.

Hence, T can query

$$\begin{aligned}
&U\left(\delta_2A+\delta_2V_1,-V_2\right)\to e\left(\delta_2A+\delta_2V_1,-V_2\right)\to\alpha_2^{-1}\\
&U\left(-V_1,\delta_1\left(B+V_2\right)\right)\to e\left(-V_1,\delta_1\left(B+V_2\right)\right)\to\alpha_3^{-1}\\
&U\left(\delta_4A+\delta_4V_3,-V_4\right)\to e\left(\delta_4A+\delta_4V_3,-V_4\right)\to\beta_2^{-1}\\
&U\left(-V_3,\delta_3\left(B+V_4\right)\right)\to e\left(-V_3,\delta_3\left(B+V_4\right)\right)\to\beta_3^{-1}
\end{aligned}.$$

Considering $\left(V_1,V_2,\delta_1V_1,\delta_2V_1,\delta_2V_2,e\left(\delta_1V_1,\delta_2V_2\right)^{-1}\right)$ and $\left(V_3,V_4,\delta_3V_3,\delta_4V_3,\delta_4V_4,e\left(\delta_3V_3,\delta_4V_4\right)^{-1}\right)$ as the result of a run of Rand, T does not require an inverse operation in G_T.

4 Security Proof

4.1 Security analysis

Theorem 1 Algorithm (T,U) is an outsourced secure implementation in which input (A,B) can be honest and secret; or honest and protected; or adversarial and protected.

Proof: Correctness is obvious, we only prove safety. Let $\mathrm{A}=\left(\mathrm{E},\mathrm{U}'\right)$ be a probabilistic polynomial-time adversary that interacts with a probabilistic polynomial-time algorithm T. In the first instance, we prove **Pair One** $EVIEW_{real}$ □ $EVIEW_{ideal}$:

If (A,B) is an honest, secret input, then the simulator S_1 behaves as follows: On receiving the input on round i, S_1 ignores it, instead of making four random queries of the form $\left(P_j,Q_j\right)$ to U'. Randomly, S_1 discovers two outputs (i.e., $e\left(P_j,Q_j\right)$) from each programme. If an error is discovered, S_1 saves all states and outputs $Y_p^i=\text{"}error\text{"}$, $Y_u^i=\varnothing$, $rep^i=1$ (i.e., the output for ideal process is $\left(estate^i,\text{"}error\text{"},\varnothing\right)$). If no error is discovered, S_1 tests the remaining two outputs. If all checks pass, S_1 outputs $Y_P^i=Y_u^i=\varnothing$, $rep^i=0$ (that is, the output for ideal process is $\left(estate^i,y_p^i,y_u^i\right)$); otherwise, S_1 chooses a random element r and outputs $Y_u^i=\varnothing,Y_P^i=r$, $rep^i=1$ (i.e., the output for ideal process is $\left(estate^i,r,\varnothing\right)$). In either case, S_1 preserves the appropriate states.

Distributing the input to V in the real and ideal experiments are computationally indistinguishable. In the ideal setting, the inputs are randomly selected and assigned. In real experiments, all the numbers in the query that T presents to the program in step (3) of this paper are independently re-randomised, and the re-randomisation factors are randomly generated by the mundane look-up table method. We will consider the following scenarios:

If V is honest in the i-th round, then $EVIEW^{i}_{real} = EVIEW^{i}_{ideal}$. In the actual environment, the output looks randomly to the environment E. In the actual environment, S_1 also simulate random values $r \in G_T$ as the output. Thus, $EVIEW^{i}_{real} = EVIEW^{i}_{ideal}$. Hence we get $EVIEW_{real} = EVIEW_{ideal}$.
Secondly, we prove that **Pair Two** $UVIEW_{real}$ □ $UVIEW_{ideal}$:

The behavior of the simulator S_2 is following: When the input is received on i-th round, S_2 ignores it and asks V four random queries of the form $\left(P_j, Q_j\right)$. Then S_2 keeps its own state and V's state. It is easy to distinguish between these real and ideal environments (note that the output of the ideal environment is never corrupted). However, E is unable to communicate this information to V. This is because in round i of the real environment, T always re-randomises its input to V . In the ideal environment, E always generates random, independent queries for V . Hence we have $UVIEW^{i}_{real}$ □ $UVIEW^{i}_{ideal}$. Through the above discussion we get $UVIEW_{real}$ □ $UVIEW_{ideal}$.

Theorem 2 Suppose the order of the bilinear group is q and the bit length of q is n, then the algorithm $\left(T, U\right)$ in this paper are an $(O(1/n),1)$ -outsource-secure implementation.

Proof : In order to calculate $e\left(A, B\right)$, Rand needs to be called twicein the algorithm in this paper, In addition cost 10 addition operation in G_1 (or G_2), 8 multiplication operation in G_T, and 4 point multiplication operation in G_1 (or G_2) in order to compute $e\left(A, B\right)$. When using the look-up table method, the calculation of Rand is negligible. On the one hand, and it takes approximately $O\left(n\right)$ multiplication operations to compute the bilinear pairings in resulting finite filed. In summary, the theorem is proved.

On the other hand, T can check the returned queries from the real environment. If U fails during any execution of this paper, it is detected with probability 1.

4.2 Comparison

We compare the algorithm in this paper with the algorithm in [8]. Addition and multiplication in G_1 or G_2 are denoted as A and M respectively. The multiplication, inverse and exponential operation in G_T are denoted as TM, TI and TE respectively. Denote by PG the computation of the bilinear pairing. We omit the modular addition operation in □ $_q$.

Table 1. Comparison between the various algorithms

Algorithm	[6]	[9]	Alg-1in [10]	Alg-2in [10]	[13]	[11]	Ours
A	4	5	4	$O\left(\log s\right)$	8	6	10
TM	6	4	3	$O\left(\log s\right)$	14	19	8
TE	10	0	0	0	0	0	0
M	6	0	0	0	0	0	4
PG(U)	4	8	6	6	6	10	8
Number of servers	1	2	2	2	2	2	1
Checkability	1	$1/2$	$1/2$	$\left(1-1/3s\right)^2$	1	1	1

Table 1 shows the comparison among different outsourcing algorithms of bilinear pairings. Compared to the algorithm [8], the proposed algorithm is superior in terms of efficiency and checkability. Be more specific, compared with the algorithm [11,12], this algorithm improves the checkability to 1. Moreover, compared with algorithm [13,15], this proposed algorithm achieve outsourcing computing of bilinear pairings with single server. In this algorithm, little computational cost is appended.

On the other hand, in our algorithm, server U needs to perform 8 bilinear pairing operations. In addition, the computation for $Rand$ is about 2A + 2TE + 6M, this value is negligible due to the use of the look-up table method. Hence, the algorithm in this paper requires more computation on the server side compared to Chevallier-Mames et al. [8]. Note that, the computational power of the server is much more powerful, so in this sense the efficiency of our algorithm will not be affected.

5 Conclusion

This paper presents an efficient and secure single server bilinear pairing outsourcing algorithm. A significant feature of our proposed algorithm is that the outsourcer does not need to perform some expensive operations, such as exponential operations, and if the cloud server is dishonest, the outsourcer can detect the error with probability 1.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Chen X F, Li J, Ma J F, et al. New algorithms for secure outsourcing of modular exponentiations. IEEE Trans Parall Distrib Syst. 2014;25:2386–2396.

[2] Chaum D, Pedersen T. Wallet databases with observers. Proceedings of 12th Annual Conference on Advances in Cryptology. Berlin: Springer. 1992;89–105.

[3] Hohenberger S, Lysyanskaya A. How to securely outsource cryptographic computations. Proceedings of the 2nd International Conference on Theory of Cryptography. Berlin: Springer. 2005;264–282.

[4] Green M, Hohenberger S, Waters B. Outsourcing the decryption of ABE ciphertexts. Proceedings of the 20th USENIX Conference on Security. New York: ACM. 2011;34.

[5] Qianqian Su;Rui Zhang;Rui Xue. Secure Outsourcing Algorithms for Composite Modular Exponentiation Based on Single Untrusted Cloud. The Computer Journal. 2020;63(8):1271-1271.

[6] Yunhai Zheng and Chengliang Tian. Lattice-Based Cryptanalysis on Outsourcing Scheme of Modular Exponentiations ExpSOS.Journal of Frontiers of Computer Science and Technology. 2022;16(05):1087-1095.

[7] Lai JZ, Deng RH, Guan CW, et al. Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Foren Secur. 2013;8:1343–1354.

[8] Chevallier-Mames B, Coron J, McCullagh N, et al. Secure delegation of elliptic-curve pairing. Proceedings of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application. Berlin: Springer. 2010;24–35.

[9] Tsang P, Chow S, Smith S. Batch pairing delegation. Proceedings of the 2nd International Workshop on Security. Berlin: Springer. 2007;74–90.

[10] Chow S, Au M, Susilo W. Server-aided signatures verification secure against collusion attack.

Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. New York: ACM, 2011. 401–405.

[11] Chen XF, Susilo W, Li J, et al. Efficient algorithms for secure outsourcing of bilinear pairings. Theor Comput Sci. 2015;562:112–121.

[12] Tian HB, Zhang FG, Ren K. Secure bilinear pairing outsourcing made more efficient and flexible. Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. New York: ACM. 2015;417–426.

[13] Min dong, Yanli Ren, Xinpeng Zhang. Fully Verifiable Algorithm for Secure Outsourcing of Blinear Pairing in Cloud Computing. KSII Transactions on Internet and Information Systems. 2017;7(11):3648-3663.

[14] Hohenberger S, Lysyanskaya A. How to Securely Outsource Cryptographic Computations.in Proc. TCC, LNCS Springer-Verlag: New York, NY, USA. 2005;3378:264-282.

[15] Ren Y, Ding N, Wang T. et al., New algorithms for verifiable outsourcing of bilinear pairing, Science China Information Sciences. 2016;59:099103.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
https://prh.ikprress.org/review-history/12064