More Efficient Secure Outsourcing Methods for Bilinear Maps *

Öznur Arabacı, Mehmet Sabir Kiraz, İsa Sertkaya, and Osmanbey Uzunkol

Mathematical and Computational Sciences
TÜBİTAK BİLGEM, Turkey
{oznur.arabaci, mehmet.kiraz, isa.sertkaya, osmanbey.uzunkol}@tubitak.gov.tr

Abstract. Bilinear maps are popular cryptographic primitives which have been commonly used in various modern cryptographic protocols. However, the cost of computation for bilinear maps is expensive because of their realization using variants of Weil and Tate pairings of elliptic curves. Due to increasing availability of cloud computing services, devices with limited computational resources can outsource this heavy computation to more powerful external servers. Currently, the checkability probability of the most efficient outsourcing algorithm is 1/2 and the overall computation requires 4 point addition in the preimage and 3 multiplications in the image of the bilinear map under the one-malicious version of a two-untrusted-program model. In this paper, we propose two efficient new algorithms which decrease not only the memory requirement but also the overall communication overhead.

Keywords: Outsourcing computation, Bilinear maps, Secure delegation, Secure Cloud Computing.

1 Introduction

The improvements in the cloud computing services result in variety of new security and privacy challenges. Many cryptographic mechanisms involving complex computations such as bilinear maps are proposed to overcome these challenges [5, 6, 14]. Since speeding up the computation of bilinear maps is crucial in real-life applications, many schemes are suggested to reduce the computational cost of pairing computation [2, 3, 6, 11, 16, 19]. Especially, Hess introduced a general framework encompassing different types of pairing functions giving optimum numbers of computation steps [12]. However, these computations are still infeasible or unaffordable for resource constrained devices including mobile phones, tablets, smart or RFID cards.

Since Hohenberger and Lysyanskaya stated the question of how a computationally limited device may outsource its computation to another, potentially malicious, but much more computationally powerful device [13], it has been

* This paper is going to appear in the 8th International Conference Information Security and Cryptology (ISCTURKEY 2015).

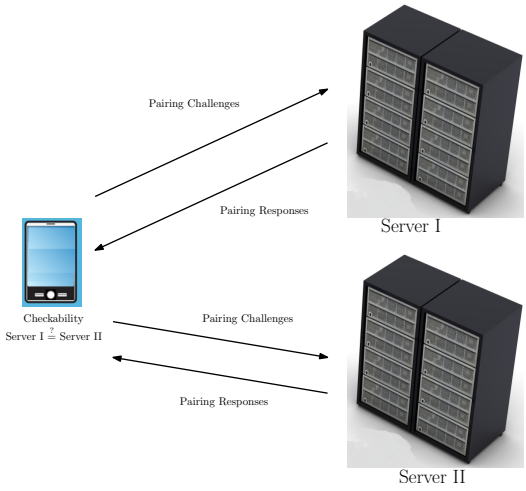

Fig. 1. Outsourcing Bilinear Maps with Two Untrusted Cloud Servers

studied extensively. Outsourcing the complex computations to external powerful devices dates back to Matsumoto, in which the RSA signature generation problem is considered [17]. More specifically, Chevallier-Mames *et al.* proposed a protocol enabling a computationally limited device to outsource the computation of bilinear maps into a more resourceful device [8]. However, this delegation process brought new concerns. Firstly, the external device should learn nothing about the secrets. Also, the computationally-limited device should be able to check whether the external device computed correctly, at least with certain probability. These two concerns can be eliminated by masking the secret values with the cost of some extra computations before sending to the external server, and then removing the masking values together with a way of validating the outsourced computation.

Besides the efficiency constraints, secrecy is the main objective of the security model, since the input and output pair of a client is used for cryptographic purposes. Henceforth, outsource mechanisms surely follow a security model in which the client (the energy limited trusted device that needs to delegate the computation) does not trust the servers (which perform the needed computations). Thus, in the security model, it is assumed that the client is honest but the servers are untrusted. Furthermore, checkability, validation of the computation processes, should be also addressed.

As simulated in Figure 1, outsource computation protocols may utilize one or more servers. Based on the number of servers utilized Tian *et al.* classified them as follows [20]:

- One-Untrusted Program (OUP): One malicious server performs the computation.
- One-Malicious version of a Two-Untrusted Program (OMTUP): Two untrusted servers perform the computation but only one of them may behave maliciously.
- Two-Untrusted Program (TUP): Two untrusted servers perform the computation and both of them may behave maliciously, but they do not maliciously collude.

Following the work of Chen *et al.*, [1], Tian *et al.* proposed two algorithms [20] in the OMTUP setting. First algorithm achieves less computational complexity and the second one improves the checkability at the cost of some additional computations. These outsource protocols are composed of both offline and online computation steps. In the offline phase, the client prepares the necessary values. During the online phase, the client creates masked values based on the precomputed offline values and requests the bilinear map computation.

In this paper, we propose two new algorithms following the steps of Chen *et al.* and Tian *et al.*'s work. We further analyze the protocols under the OMTUP assumption and reduce not only the computational complexity of the offline computations, but also the memory needed to store the values resulting from the offline computations together with the communication overhead. While doing so, we do not increase the computation costs that need to be handled by the client.

1.1 Related work

Weil and Tate pairings are firstly used as cryptanalytic tools for reducing the discrete logarithm problem (DLP) on some elliptic curves to DLP on finite fields [4]. Later, Boneh *et al.* and Joux constructed new cryptographic protocols based on bilinear maps [5,6,14]. Reducing the computational cost of bilinear maps are suggested in [2,3,6,10–12,16,19].

First protocol for secure outsourcing of elliptic curve pairings were proposed by Chevallier-Mames *et al.* [8]. The algorithm assumes the OUP setting and it is 1-checkable. However, the algorithm requires expensive computations, namely multiple membership test operation which is equivalent to an exponentiation over the finite field and inversion on the exponents. Later on, under the same OUP assumption, Kang *et al.* [15] and Canard *et al.* [7] improved the computational complexity results. However, the solutions were not feasible since exponentiation, membership test, and inversion were still required. Tsang *et al.* made a taxonomy for pairing based computations and constructed a batch pairing delegation mechanism [21]. Chow *et al.* studied server aided signature verification [9].

Chen *et al.* broke the paradigm by utilizing two servers under the OMTUP assumption and by performing some computations during an idle time of the resource-limited device [1]. As a result, this outsourcing mechanism of bilinear maps was the first one which does not depend on the membership test operations and exponentiations over the finite field. Additionally, this scheme decreased the online computations on the client side. The user had to perform only 5 point

additions in $\mathbb{G}_1$ and $\mathbb{G}_2$, and 4 multiplications in $\mathbb{G}_3$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is the underlying bilinear map. Later, Tian *et al.* proposed a more efficient algorithm [20] and reduced the online computation phase to 4 point additions in $\mathbb{G}_1$ and $\mathbb{G}_2$, and 3 multiplications in $\mathbb{G}_3$.

1.2 Our contributions

In this paper, we propose two efficient new algorithms for secure outsourcing of bilinear maps. Compared to the state of the art algorithms (especially Tian *et al.*'s [20]), our algorithms need less offline computations, less memory, and less queries to the servers. In order to manage that, different from the previous studies, we use negation of an input value (it is almost for free since it is located over the elliptic curve), and we also send the same checking computation to both servers. Since it is assumed that the two servers do not collude, we reduce the computation costs without affecting the checkability of the system. The second algorithm may for instance be utilized in signature verification applications, in which we evade from at least one multiplication. For both propositions, we also provide the security model following exactly the lines of the security model of Hohenberger and Lysyanskaya [13]. We conclude the paper by comparing the efficiency of the system with the very recent work of Tian *et al.* [20].

1.3 Roadmap

In Section 2, we give the security definitions for the outsourcing algorithm. Then, we present some background and preliminaries that will be needed throughout the manuscript, and we propose our two main algorithms together with their security analysis in Section 3. Next, in Section 4, we analyze complexity of our new algorithms and compare it to the complexity of the best known algorithm [20]. Finally, we conclude the paper in Section 5.

2 Security Model

Chen *et al.*. [1] and Tian *et al.* [20] follow the security model proposed by Hohenberger and Lysyanskaya [13]. We remark especially that we also follow exactly their security model [13].

Definition 1. *An algorithm is said to obey the outsource 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. The first input is called the honest, secret input, which is unknown to both E and U; the second is called the honest, protected input, which may be known by E, but is protected from U; and the third is called 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: the adversarial, protected input, which is known to E, but protected from U; and the the adversarial.*

Definition 2. *Let* $\mathsf{Alg}(.,.,.,.,.)$ *be an algorithm with outsource-IO. A pair of algorithms* (T,U) *is said to be an outsource-secure implementation of an algorithm* Alg *if:* ***Correctness.*** T^U *is a correct implementation of Alg.*
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:*** $EVIEW^i_{real}$~$EVIEW_{ideal}$ *(The external adversary,* E*, learns nothing.):*
  - *The view that the adversarial environment* E *obtains by participating in the following REAL process:*

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

    $EVIEW_{real}=EVIEW^i_{real}$ *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, based on its view from the last round, chooses* (0) *the value of its* $estate_i$ *variable as a way of remembering what it did next time it is invoked;* (1) *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);* (2) *the adversarial, protected input* x^i_{ap}*;* (3) *the adversarial, unprotected input* x^i_{au}*;* (4) *the Boolean variable* $stop^i$ *that determines whether roundi is the last round in this process. Next, the algorithm* $T^{U'}$ *is run on the inputs* $(tstate^{i-1},x^{j^i}_{hs},x^{j^i}_{hp},x^{j^i}_{hu},x^i_{ap},x^i_{au})$*, where* $tstate^{i-1}$ *is* T*s previously saved state, and produces a new state* $tstate^i$ *for* T*, as well as the secret* y^i_s*, protected* y^i_p *and unprotected* y^i_u *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 roundi consists of* $estate^i$*, and the values* y^i_p *and* y^i_u*. 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:*

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

$$(z_p^i, z_u^i) = replace^i(Y_p^i, Y_u^i) + (1 - replace^i)(y_p^i, y_u^i) : (estate^i, z_p^i, z_u^i)\}$$

$EVIEW_{ideal} = EVIEW_{ideal}^i$ *if* $stop^i = TRUE$.
The ideal process also proceeds in rounds. In the ideal process, we have a stateful simulator S_1 *who, shielded from the secret input* x_{hs}*, but given the non-secret outputs that* 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* $replace^i$ *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 state as in the real experiment.*

- ***Pair Two****:* $UVIEW_{real}$~$UVIEW_{ideal}$ *(The untrusted software,* (U_1, U_2)*, learns nothing.):*
  - *The view that the untrusted software* U' *obtains by participating in the REAL process described in Pair One.* $UVIEW_{real} = ustate^i if stop^i = TRUE$.
  - *The* $IDEAL$ *process:*

$$UVIEW_{ideal}^i = \{(istate^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x_{ap}^i, x_{au}^i, stop^i) \leftarrow E(1^k, estate^{i-1}, x^i{}_{,hp}, x_{hu}^i, y_p^{i-1}, y^{i-1}{}_u);$$
$$(astate^i, y_s^i, y_p^i, y_u^i) \leftarrow \mathsf{Alg}(astate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i);$$
$$(sstate^i, ustate^i) \leftarrow S_2^{U'(ustate^{i-1})}(sstate^{i-1}, x_{hu}^{j^i}, x_{au}^i) : (ustate^i)\}$$

  $UVIEW_{ideal} = UVIEW_{ideal}^i$ *if* $stop^i = TRUE$.
  In the ideal process, 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.*

In our security model we assume one-malicious version of a two-untrusted program (OMTUP) model. More concretely, there are two untrusted cloud servers in this model performing the outsourced computation, where only one of them is assumed to be malicious.

Definition 3. *A pair of algorithms* (T, U_1, U_2) *are an* α*-efficient implementation of an algorithm* Alg *if* (1) *they are an outsource-secure implementation of* Alg*, and* (2) $\forall$ *inputs* x*, the running time of* T *is* $\leq$ *an* α*-multiplicative factor of the running time of* $\mathsf{Alg}(x)$.

Definition 4. *A pair of algorithms* (T, U_1, U_2) *are an* β*-checkable implementation of an algorithm* Alg *if* (1) *they are an outsource-secure implementation of* Alg*, and* (2) $\forall$ *inputs* x*, if* $U_i', i = 1, 2$ *deviates from its advertised functionality during the execution of* $T^{(U_1', U_2')}(x)$*,* T *will detect the error with probability* $\geq \beta$.

Definition 5. *A pair of algorithms* (T, U_1, U_2) *are an* (α, β)*-outsource secure implementation of an algorithm* Alg *if they are both* α*-efficient and* β*-checkable.*

3 Algorithms for outsourcing of bilinear maps

3.1 Preliminaries: Bilinear Maps

Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, +)$ be two additive cyclic groups of order q with $\mathbb{G}_1 =< Q >$ and $\mathbb{G}_2 =< P >$, $(\mathbb{G}_3, \cdot)$ be a multiplicative cyclic group of order q, where q is a prime number and $0_{\mathbb{G}_1}, 0_{\mathbb{G}_2}$ and $1_{\mathbb{G}_3}$ are the identity elements of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$, respectively. Assume that Discrete Logarithm Problem (DLP) is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$ (i.e., given a random $y \in \mathbb{G}_1$ (or $\in \mathbb{G}_2$), it is computationally infeasible to find an integer $x \in \mathbb{Z}$ such that $y = g^x$). If it is clear from the context we write 0 for the identity elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and 1 for $\mathbb{G}_3$. A *bilinear map* is a map $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ satisfying the following properties [4]:

- **Bilinearity:** For all $P_1, Q_1 \in \mathbb{G}_1, P_1', Q_1' \in \mathbb{G}_2$, e is a group homomorphism in each component, i.e.
  1. $e(P_1 + Q_1, P_1') = e(P_1, P_1') \cdot e(Q_1, P_1')$,
  2. $e(P_1, P_1' + Q_1') = e(P_1, P_1') \cdot e(P_1, Q_1')$.
- **Non-degeneracy:** e is non-degenerate in each component, i.e.,
  1. For all $P \in \mathbb{G}_1$, $P \neq 0$, there is an element $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$,
  2. For all $Q \in \mathbb{G}_2$, $Q \neq 0$, there is an element $P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.
- **Computability:** There exists an algorithm which computes the bilinear map e efficiently.

3.2 Algorithm 1

Precomputations Like all existing outsourcing algorithms, some precomputations are performed to speed up the proposed algorithms following the method of [20]. It includes a static table ST and a dynamic table DT. The values stored in the dynamic table are replaced while they are used, and then the table is reconstructed in an idle time of the device. We next describe the steps of the Rand1 algorithm to generate random group elements.

Rand1

- **Preprocessing Step:** Let P_1 and P_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Generate n random elements $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}/q\mathbb{Z}$. For $j = 1, \cdots, n$ compute $\beta_{j_1} = \alpha_j \cdot P_1$ and $\beta_{j_2} = \alpha_j \cdot P_2$, and store the values of α_j, β_{j_1} and β_{j_2} in ST. Compute $e(P_1, P_2) \in \mathbb{G}_3$ and store it in ST.
- **Generation of Precomputed Values:** A new entry in DT is computed as follows: Generate randomly $S \in \{1, \cdots, n\}$ such that $|S| = k$. For each $j \in S$, select randomly $K_j \in \{1, \cdots, h-1\}$, where $h > 1$ is a small integer. Compute

$$x_1 \equiv \sum_{j \in S} \alpha_j K_j \bmod q.$$

If $x_1 \equiv 0 \bmod q$, start again. Otherwise, compute

$$x_1 \cdot P_1 \equiv \sum_{j \in S} K_j \cdot \beta_{j_1} \bmod q.$$

Following the above procedure, compute similarly the elements $(x_2, x_2 \cdot P_2), (x_3, x_3 \cdot P_1)$, and $(x_4, x_4 \cdot P_2)$. Then compute

1. $2x_1 \cdot P_1$,
2. $-2x_2 \cdot P_2$,
3. $e(P_1, P_2)^{2x_1x_2}$,
4. $e(P_1, P_2)^{x_3x_4}$.

The entry

$$(x_1 \cdot P_1, 2x_1 \cdot P_1, x_3 \cdot P_1, x_2 \cdot P_2, -2x_2 \cdot P_2,$$

$$x_4 \cdot P_2, e(P_1, P_2)^{2x_1x_2}, e(P_1, P_2)^{x_3x_4})$$

is stored in DT. On each invocation of Rand1, an entry is returned and removed from DT. Further, a new set of values is used as fresh random values.

Proposed Algorithm 1 Our algorithm takes $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ as inputs and produces $e(A, B)$ as output. In what follows, T denotes a trusted device with limited computation resources, and $U_i(A, B) \rightarrow e(A, B), i \in \{1, 2\}$ denotes party U_i taking (A, B) as inputs and returning $e(A, B)$ as output.

– **Initialization:** T calls Rand1 to get random values

$$(x_1 \cdot P_1, 2x_1 \cdot P_1, x_3 \cdot P_1, x_2 \cdot P_2, -2x_2 \cdot P_2,$$

$$x_4 \cdot P_2, \lambda = e(P_1, P_2)^{2x_1x_2}, e(P_1, P_2)^{x_3x_4}).$$

– **Computation:** In random orders, T sends the following values to U_1
  1. $U_1(A + 2x_1 \cdot P_1, -B - 2x_2 \cdot P_2) \rightarrow \alpha_1$,
  2. $U_1(x_3 \cdot P_1, x_4 \cdot P_2) \rightarrow \alpha_1'$.

  Similarly, in random orders, T sends the following values to U_2
  1. $U_2(A + x_1 \cdot P_1, B + x_2 \cdot P_2) \rightarrow \alpha_2$,
  2. $U_2(x_3 \cdot P_1, x_4 \cdot P_2) \rightarrow \alpha_2'$.

– **Recover:** T checks whether $\alpha_1' \stackrel{?}{=} \alpha_2' \stackrel{?}{=} e(P_1, P_2)^{x_3x_4}$. If the verifications are successful then it computes

$$\beta = \alpha_1 \alpha_2^2 \lambda$$

and produces β as output. Otherwise, it rejects and gives an "Error".

Security Analysis

Theorem 1. *Under the OMTUP assumption, the algorithms $(T, (U_1, U_2))$ of "Algorithm 1" are an outsource-secure implementation of a pairing evaluation, where the input (A, B) may be honest, secret, honest, protected, or adversarial, protected.*

Proof. The correctness follows easily by bilinear property of e:

$$\begin{aligned}\beta &= \alpha_1 \cdot \alpha_2^2 \cdot \lambda \\ &= e(A + 2x_1 \cdot P_1, -B - 2x_2 \cdot P_2) \cdot \\ &e(A + x_1 \cdot P_1, B + x_2 \cdot P_2)^2 \cdot e(P_1, P_2)^{2x_1x_2} \\ &= e(A, B)^{-1} \cdot e(P_1, P_2)^{-4x_1x_2} \cdot e(A, B)^2 \cdot \\ &e(P_1, P_2)^{2x_1x_2} \cdot e(P_1, P_2)^{2x_1x_2} \\ &= e(A, B).\end{aligned}$$

We next prove the security of the algorithm. Let (E, U_1', U_2') be a PPT adversary that interacts with a PPT algorithm T in the OMTUP model.
Pair One: $EVIEW^i_{real} \sim EVIEW_{ideal}$ (The external adversary, E, learns nothing.):
For a round i, if the input (A, B) is other than secret, honest, (i.e., honest, protected or adversarial, protected) the simulator S_1 behaves as in the real round. S_1 never requires to access the secret input, since there is none. So suppose that the input (A, B) is honest, secret. In that case, the simulator S_1 behaves as follows: On receiving input in the ith round, S_1 ignores it and instead make random queries to the servers U_1', U_2':

- $U_1'(x_1 \cdot P_1, x_2 \cdot P_2) \to \alpha_1$,
- $U_1'(x_3 \cdot P_1, x_4 \cdot P_2) \to \alpha_1'$,
- $U_2'(x_5 \cdot P_1, x_6 \cdot P_2) \to \alpha_2$,
- $U_2'(x_3 \cdot P_1, x_4 \cdot P_2) \to \alpha_2'$.

After getting responses from U_1' and U_2', S_1 checks:

- If $\alpha_1' \neq \alpha_2'$ or $e(x_3 \cdot P_1, X_4 \cdot P_2) \neq \alpha_1'$, S_1 produces $Y_p^i =$"Error", $Y_u^i = \emptyset$ and $rep^i = 1$.
- If all responses are correct, S_1 sets $Y_p^i = \emptyset, Y_u^i = \emptyset$, and $rep^i = 0$.
- Otherwise, S_1 selects a random value $s_r \in \mathbb{G}_3$ and sets $Y_p^i = s_r, Y_u^i = \emptyset$ and $rep^i = 1$.

For all cases, S_1 saves the appropriate states.
The input distributions in the real and ideal experiments are computationally indistinguishable for U_1' and U_2'. The inputs to U_1' and U_2' are chosen uniformly at random in the ideal experiment. In a real experiment, each part of all queries that T makes to any one program in the computation step is independently re-randomized, and the re-randomization factors(i.e., outputs of Rand1) are either truly randomly generated by naive table-lookup approach or computationally indistinguishable from random by the assumption of the EBPV generator [18]. Now, there are three possibilities to consider.

- If U_1' and U_2' behave honestly in the round i, S_1 gives the correct output, using Alg, which is the same as the output of T^{U_1, U_2}.

- If one of (U_1', U_2') give an incorrect output in the ith round and it has been detected by both T and S_1 with probability $1/2$, then it will result in an "Error".
- Finally, we consider the case, where one of (U_1', U_2') give an incorrect output in the ith round and it is not caught with probability $1/2$. In the real experiment,the two outputs generated by (U_1', U_2') are multiplied together along with a random value λ.

Thus, a corrupted output looks random to the environment E, in the real experiment, S_1 also simulates with a random value in $\mathbb{G}_3$ as the output. So, $EVIEW^i_{real} \sim EVIEW^i_{ideal}$ even when one of (U_1', U_2') is dishonest. Now, by the hybrid argument, we conclude that $EVIEW_{real} \sim EVIEW_{ideal}$.

Pair Two: $UVIEW_{real} \sim UVIEW_{ideal}$ (The untrusted software, (U_1, U_2), learns nothing.):
Here, regardless of the input type, the simulator S_2 always behaves the same way. Upon receiving an input on round i, S_2 ignores it and instead makes four random queries to U_1' and U_2'. Then S_2 saves its own state and the states of (U_1', U_2'). E can easily distinguish between these real and ideal experiments (output of the ideal experiment is never corrupted), but we want to show that E cannot share this information with (U_1', U_2'). This happens because in the ith round of the real experiment, T always re-randomizes the inputs to (U_1', U_2'), and in the ideal experiment S_2 creates random, independent queries for (U_1', U_2'). So, for each round i, we have $UVIEW^i_{real} \sim UVIEW^i_{ideal}$. Then, by the hybrid argument, we get the desired result $UVIEW_{real} \sim UVIEW_{ideal}$.

Theorem 2. *The algorithms $(T, (U_1, U_2))$ of "Algorithm 1" are a $(\mathcal{O}(\frac{1}{\log q}), 1/2)$-outsource secure implementation of a pairing evaluation under the OMTUP assumption.*

Proof. By the above theorem, U_1 and U_2 cannot distinguish a test query from a real query. Without loss of generality, assume that U_1 is honest while U_2 is dishonest (since we are under the OMTUP assumption). Thus, U_2 fails with a probability $1/2$.

3.3 Algorithm 2

Precomputations Rand2

- **Preprocessing Step:** Generate n random elements $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}_q$. For $j = 1, \cdots, n$ compute $\beta_{j_1} = \alpha_j \cdot P_1$ and $_{j_2} = \alpha_j \cdot P_2$, and store the values of α_j, β_{j_1} and β_{j_2} in a static table ST. Compute $e(P_1, P_2)$ and store the value in ST.
- **Generation of Precomputed Values:** When a table DT needs a new entry, it is produced as follows. Randomly generate $S \in \{1, \cdots, n\}$ such

that $|S| = k$. For each $j \in S$, randomly select $K_j \in \{1, \cdots, h-1\}$, where $h > 1$ is a small integer. Compute

$$x_1 \equiv \sum_{j \in S} \alpha_j K_j \bmod q.$$

If $x_1 \equiv 0 \bmod q$, start again. Otherwise, compute

$$x_1 P_1 \equiv \sum_{j \in S} K_j \cdot \beta_{j_1} \bmod q.$$

Following the above procedure, compute similarly the elements $(x_2, x_2 \cdot P_2), (x_3, x_3 \cdot P_1), (x_4, x_4 \cdot P_1)$, and $(x_5, x_5 \cdot P_2)$. Then compute
1. $(x_2 - x_1^{-1} x_2 x_3) \cdot P_2$,
2. $e(P_1, P_2)^{x_4 x_5}$.

The entry

$$(x_1 \cdot P_1, x_3 \cdot P_1, x_4 \cdot P_1, x_2 \cdot P_2, x_5 \cdot P_2,$$
$$(x_2 - x_1^{-1} x_2 x_3) \cdot P_2, e(P_1, P_2)^{x_4 x_5})$$

is stored in DT. On each invocation of Rand2, an entry is returned and removed from DT. Further, a new set of values is used as fresh random values.

Proposed Scheme 2 Algorithm 2 takes $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ as inputs and produces $e(A, B)$ as output. In what follows, T denotes a trusted device with limited computation resources, and $U_i(A, B) \to e(A, B), i \in \{1, 2\}$ denotes party U_i taking (A, B) as inputs and giving $e(A, B)$ as output.

- **Init:** T calls Rand2 to get random values

$$(x_1 \cdot P_1, x_3 \cdot P_1, x_4 \cdot P_1, x_2 \cdot P_2, x_5 \cdot P_2,$$
$$(x_2 - x_1^{-1} x_2 x_3) \cdot P_2, e(P_1, P_2)^{x_4 x_5}).$$

- **Computation:** In random orders, T sends the following to U_1
  1. $U_1(A + x_1 \cdot P_1, B + x_2 \cdot P_2) \to \alpha_1$,
  2. $U_1(x_4 \cdot P_1, x_5 \cdot P_2) \to \alpha_1'$.

  Similarly, in random orders, T sends the following to U_2
  1. $U_2(A + x_3 \cdot P_1, -x_2 \cdot P_2) \to \alpha_2$,
  2. $U_2(-x_1 \cdot P_1, B + (x_2 - x_1^{-1} x_2 x_3) \cdot P_2) \to \alpha_3$
  3. $U_2(x_4 \cdot P_1, x_5 \cdot P_2) \to \alpha_2'$.
- **Recover:** T checks whether $\alpha_1' \stackrel{?}{=} \alpha_2' \stackrel{?}{=} e(P_1, P_2)^{x_4 x_5}$. If the verifications are successful then it computes

$$\beta = \alpha_1 \alpha_2 \alpha_3$$

and produces β as output. Otherwise, it rejects and gives an "Error".

Security Analysis

Theorem 3. *Under the OMTUP assumption, the algorithms* $(T, (U_1, U_2))$ *of "Algorithm 1" are an outsource-secure implementation of a pairing evaluation, where the input* (A, B) *may be honest, secret, honest, protected, or adversarial, protected.*

Proof. The correctness is straight forward:

$$\begin{aligned}\beta &= \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \\ &= e(A + x_1 \cdot P_1, B + x_2 \cdot P_2) \cdot e(A + x_3 \cdot P_1, -x_2 \cdot P_2) \cdot \\ &e(-x_1 \cdot P_1, B + (x_2 - x_1^{-1} x_2 x_3) \cdot P_2) \\ &= e(A, B) e(P_1, P_2)^{x_1 x_2} \cdot e(P_1, P_2)^{-x_2 x_3} \cdot \\ &e(P_1, P_2)^{-x_1 (x_2 - x_1^{-1} x_2 x_3)} \\ &= e(A, B).\end{aligned}$$

The proof of the security part follows analogously to the proof of Theorem 1.

4 Complexity Analysis

4.1 Comparisons

We compare the precomputation algorithms of our proposed schemes with Tian *et al.*'s algorithm [20]. In the following tables, SM denotes scalar multiplication in $\mathbb{G}_1, \mathbb{G}_2$, ME modular exponentiation, PC pairing computation on the server side, FM field multiplication, and PA point addition in $\mathbb{G}_1, \mathbb{G}_2$. Furthermore, k denotes the size of the set S in the algorithms Rand1 and Rand2. Table I compares the precomputation, Table II compares the client's workload, and Table III compares the server's workload. Table IV compares the communication overhead between the client and the servers, and finally Table V gives the memory requirements for ST and DT by means of counting the number of group elements.

	Algoritm [20]	**Algorithm 1**	**Algorithm 2**
SM	3	2	1
ME	2	2	1
PA	5(k+h-3)	4(k+h-3)	5(k+h-3)

Table 1. Comparison of Precomputation

	Algoritm [20]	**Algorithm 1**	**Algorithm 2**
PA	4	4	4
FM	3	3	2

Table 2. Comparison of Client's Workload

	Algoritm A [20]	**Algorithm 1**	**Algorithm 2**
PC	6	4	5

Table 3. Comparison of Server's Workload

	Algoritm A [20]	**Algorithm 1**	**Algorithm 2**
PC	$\approx$ 0,117 KB	$\approx$ 0,078 KB	0,098 KB

Table 4. Comparison of Communication Overhead for 80-Bit Security

	ST	DT
Algorithm 1	3	8
Algorithm 2	3	7

Table 5. Memory Requirements for Rand algorithms.

5 Conclusion

In this paper, we studied outsourcing the computation of bilinear maps and proposed two new efficient algorithms decreasing both the memory requirement and the overall communication overhead. We defined the necessary security model, and proved the correctness and the security of the proposed secure outsourcing algorithms. We further gave the comparisons of our algorithms with a very recent outsourcing mechanism of Tian *et al.* [20] with respect to the offline and online computations, and the memory to be used. In this way, we show that our algorithms are more efficient than all previously proposed solutions.

Acknowledgment

Kiraz's and Arabaci's works are supported by a grant from Ministry of Development of Turkey provided to the Cloud Computing and Big Data Research Lab Project. Uzunkol's work is supported by the project (114C027) funded by EU FP7-The Marie Curie Action and TÜBİTAK (2236-CO-FUNDED Brain Circulation Scheme).

References

1. 0001, X.C., Susilo, W., 0002, J.L., Wong, D.S., Ma, J., Tang, S., Tang, Q.: Efficient algorithms for secure outsourcing of bilinear pairings. Theor. Comput. Sci. 562, 112–121 (2015), http://dblp.uni-trier.de/db/journals/tcs/tcs562.html#ChenSLWMTT15

2. Barreto, P., Galbraith, S., higeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007), http://dx.doi.org/10.1007/s10623-006-9033-6
3. Beuchat, J.L., Gonzlez-Daz, J., Mitsunari, S., Okamoto, E., Rodrguez-Henrquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barretonaehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography - Pairing 2010, Lecture Notes in Computer Science, vol. 6487, pp. 21–39. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-17455-1_2
4. Blake, I., Seroussi, G., Smart, N., Cassels, J.W.S.: Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series). Cambridge University Press, New York, NY, USA (2005)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer Berlin Heidelberg (2001), http://dx.doi.org/10.1007/3-540-44647-8_13
6. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptology 17(4), 297–319 (2004), http://dx.doi.org/10.1007/s00145-004-0314-9
7. Canard, S., Devigne, J., Sanders, O.: Delegating a pairing can be both secure and efficient. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol. 8479, pp. 549–565. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-07536-5_32
8. Chevallier-Mames, B., Coron, J.S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. Cryptology ePrint Archive, Report 2005/150 (2005)
9. Chow, S.S., Au, M.H., Susilo, W.: Server-aided signatures verification secure against collusion attack. Information Security Technical Report 17(3), 46 – 57 (2013), http://www.sciencedirect.com/science/article/pii/S1363412712000489, security and Privacy for Digital Ecosystems
10. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113 – 3121 (2008), http://www.sciencedirect.com/science/article/pii/S0166218X08000449, applications of Algebra to Cryptography
11. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. Information Theory, IEEE Transactions on 52(10), 4595–4602 (Oct 2006)
12. Hess, F.: Pairing lattices. In: Galbraith, S., Paterson, K. (eds.) Pairing-Based Cryptography Pairing 2008, Lecture Notes in Computer Science, vol. 5209, pp. 18–38. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85538-5_2
13. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3378, pp. 264–282. Springer (2005), http://www.iacr.org/cryptodb/archive/2005/TCC/3678/3678.pdf
14. Joux, A.: A one round protocol for tripartite diffiehellman. Journal of Cryptology 17(4), 263–276 (2004), http://dx.doi.org/10.1007/s00145-004-0312-y
15. Kang, B.G., Lee, M.S., Park, J.H.: Efficient delegation of pairing computation (2005)

16. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N. (ed.) Cryptography and Coding, Lecture Notes in Computer Science, vol. 3796, pp. 13–36. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11586821_2
17. Matsumoto, T., Kato, K., Imai, H.: Speeding up secret computations with insecure auxiliary devices. In: Goldwasser, S. (ed.) Advances in Cryptology CRYPTO 88, Lecture Notes in Computer Science, vol. 403, pp. 497–506. Springer New York (1990), http://dx.doi.org/10.1007/0-387-34799-2_35
18. Nguyen, P., Shparlinski, I., Stern, J.: Distribution of modular sums and the security of the server aided exponentiation. In: Cryptography and Computational Number Theory, Progress in Computer Science and Applied Logic, vol. 20, pp. 331–342. Birkhuser Basel (2001)
19. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, Lecture Notes in Computer Science, vol. 4249, pp. 134–147. Springer Berlin Heidelberg (2006), http://dx.doi.org/10.1007/11894063_11
20. Tian, H., Zhang, F., Ren, K.: Secure bilinear pairing outsourcing made more efficient and flexible. In: Bao, F., Miller, S., Zhou, J., Ahn, G. (eds.) Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015. pp. 417–426. ACM (2015), http://doi.acm.org/10.1145/2714576.2714615
21. Tsang, P., Chow, S., Smith, S.: Batch pairing delegation. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) Advances in Information and Computer Security, Lecture Notes in Computer Science, vol. 4752, pp. 74–90. Springer Berlin Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-75651-4_6