Fully Verifiable Secure Delegation of Pairing Computation: Cryptanalysis and An Efficient Construction

Osmanbey Uzunkol[1], Öznur Kalkar[2,3], and İsa Sertkaya[2]

[1] FernUniversität in Hagen, Faculty of Mathematics and Computer Science, Germany
osmanbey.uzunkol@gmail.com
[2] Mathematical and Computational Sciences TÜBİTAK BİLGEM, Turkey
{oznur.arabaci,isa.sertkaya}@tubitak.gov.tr
[3] Department of Mathematics, Gebze Technical University

Abstract. We address the problem of secure and verifiable delegation of general pairing computation. We first analyze some recently proposed pairing delegation schemes and present several attacks on their security and/or verifiability properties. In particular, we show that none of these achieve the claimed security and verifiability properties simultaneously. We then provide a fully verifiable secure delegation scheme VerPair under one-malicious version of a two-untrusted-program model (OMTUP). VerPair not only significantly improves the *efficiency* of all the previous schemes, such as fully verifiable schemes of Chevallier-Mames *et al.* and Canard *et al.* by eliminating the impractical *exponentiation-* and *scalar-multiplication-consuming* steps, but also offers for the *first time* the desired *full verifiability* property unlike other practical schemes. Furthermore, we give a more *efficient* and less *memory consuming* invocation of the subroutine Rand for VerPair by eliminating the requirement of offline computations of modular *exponentiations* and *scalar-multiplications.* In particular, Rand includes a fully verifiable partial delegation under the OMTUP assumption. The partial delegation of Rand distinguishes VerPair as a useful lightweight delegation scheme when the delegator is resource-constrained (e.g. RFID tags, smart cards or sensor nodes).

1 Introduction

The rapid proliferation of mobile technologies and steadily increasing ubiquitous networking of various intelligent devices via internet resulted in usable computing paradigms of *mobile computing* and *internet of things* (IoT) with many advantages. At the same time, *cloud computing* as a revolutionary computing paradigm offers innovative, flexible and cost-efficient solutions for both individuals and enterprises, including on-demand self-service capability, pay-per use, ubiquitous network access, location-independent pooling of resources, scalability of services, and rapid elasticity [36]. In other words, recent scientific and technological advances in mobile computing, IoT and cloud computing come with several novel application areas with lots of potential. A very good example is the *delegation* of computational tasks of a client to programs, applications, nearby or remote services and servers. Since the targeted computation is prohibitively high or even impossible in most applications (e.g. *Big Data*), delegation of such rather complex, time-consuming, and usually heavy computational tasks to more powerful entities has become the most expedient option along with its cost-effectiveness. Of particular highly desirable practical interest is to delegate (or outsource) complex computational tasks from resource-constrained and energy-limited units and devices like smart cards, SIM cards, active/passive RFID tags and sensor nodes. Delegating computation to *potentially* untrusted programs comes however also with new, complicated, and sometimes unique security and privacy challenges. Although, it is highly desirable to have a non-interactive computational and bandwidth efficient delegation mechanism working independent of the delegated functionality, general program obfuscation is unfortunately impossible even with fully homomorphic encryption techniques [22]. As a usable and practical solution, delegation of the computation of *costly* cryptographic operations while keeping the *security* and *privacy* properties of the underlying cryptographic mechanism unchanged has been the subject of many recent studies [25,20,17,15,18,29,19]. Beside the usual security and privacy requirements, ensuring the *verifiability* of the delegated computation is of utmost importance. In particular, insufficient verifiability comes with *fatal consequences* especially if it is a part of an authentication protocol or a verification step of a digital signature as also pointed out in [15,29].

Pairings. Pairing-based cryptography is demanded in most cryptographic solutions like one round tripartite key exchange in Joux [26], identity-based encryption scheme in [9] and short signatures of Boneh, Lynn, and Shacham [11]. Furthermore, it is required to obtain new cryptographic tools in the standard model in order to propose innovative cryptographic solutions with a view towards their various applications and deployments in mobile computing, IoT and cloud computing. In particular, most applications benefit from pairing-based cryptography like group and ring signatures [8,21], aggregate and verifiably encrypted signatures [10], signcryption [33], homomorphic linear authenticators [43], cryptographic accumulators [14], functional encryption [1], and zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [5] with the new novel applications to privacy in blockchain technologies, i.e. cryptocurrencies like ethereum and zcash.

Computing pairings however forms one of the most expensive tasks among cryptographic operations. Although reducing the computational cost of pairings has been the subject of numerous studies in the literature [4,6,11,24,32,42], these results are unfortunately still considerably far away of being practical enough to be able to mount pairing computations within resource-constrained and energy-limited units and devices. Even worse, recent advances on solving the discrete logarithm problem in the target group of a paring function enforces to increase the actual key sizes, thence increases further the computational complexity of pairing computations substantially [30,3,31].

Delegation of Pairing Computation. Delegation of pairing computation to more powerful and possibly untrusted servers is therefore a cost-effective way of realizing pairing-based cryptography. Additionally, it is the unique *computation, space* as well as *energy constrained* option for the deployments of pairings in resource-constrained and energy-limited devices since its computation could effectively be realized without necessarily embedding it directly inside the devices. Nevertheless, mobile devices, physically uncontrollable cheap sensor nodes, personal computers and/or remote servers could always be a target for dishonest entities which they could possibly control by means of physical attacks, malwares and/or malicious insiders as also partially pointed out by [15].

Related Work. In the context of delegating modular exponentiations, [25] provided the first formal simulation-based security notions for secure and verifiable delegation of cryptographic computations in the presence of malicious powerful helpers. Following their work, different security models for a delegated pairing computation can be summarized as follows:

- One-Untrusted Program (OUP): There exists a single malicious program performing the delegated computation.
- One-Malicious version of a Two-Untrusted Program (OMTUP): There exist two untrusted programs performing the delegated computation but exactly one of them may also behave maliciously.
- Two-Untrusted Program (TUP): There exist two untrusted programs performing the delegated computation and both of them may simultaneously behave maliciously, but they do not maliciously collude.

[20] introduced the first fully verifiable delegation scheme for pairings under the OUP model which later be improved by [28] and [15]. As pointed out by [18] the main drawback of these schemes is the replacement of the pairing computation with a scheme executing the comparably costly interactive computations of modular exponentiations and elliptic curve scalar multiplications. This contradicts inherently with the major goal of secure delegation stemming originally from the realizability, practicality and usability considerations. With the aim of practically feasible secure delegation of pairing computation, [18] proposed the first efficient delegation scheme under the OMTUP assumption without requiring any interactive computation of modular exponentiations and elliptic curve scalar multiplications of the delegator. [44] and [2] improved the computational overhead of this scheme further. The major drawback of these schemes is however that an untrusted program could cheat the delegator with probability $1/2$ contradicting substantially with the desired verifiability of the delegated computation. Almost at the same time, the first secure delegation scheme with adjustable verifiability is proposed by [29] in the context of group exponentiations under the OUP model. Like [29], [44] proposed for the first time a pairing delegation scheme with the adjustable verifiability probability $(1-1/3s)^2$ under the TUP model, where s is a predetermined parameter. Subsequently, [41] proposed another scheme under the OMTUP assumption. The verifiability of this scheme is claimed to be $(1-1/120(s-1)^2)$ with an adjustable s.

Our Contribution. In this paper, we primarily focus on the analysis of recently proposed pairing delegation schemes, and design the first fully verifiable secure delegation scheme $\mathsf{VerPair}$ for pairings under the OMTUP assumption. We also observe that even though the existing precomputation techniques [13,12,45] do not provide the computation of the form g^{ab} or u^{-1} with $g, u \in \mathbb{G}$ of a (multiplicatively written) group $\mathbb{G}$, most papers use these to produce values of the form (a, b, g^{ab}) and u^{-1} for randomly chosen elements a, b or u as granted [23,39,35,34,27]. Unfortunately, the only way for the delegator to produce such values seems to *compute* them *offline* making the delegation scheme highly unsuitable for resource-constrained devices. To eliminate the requirement of offline computation of modular exponentiations (or elliptic curve scalar multiplications) on the delegator's side, we additionally give a new precomputation subroutine Rand consisting of two parts. While one part of Rand produces values of the form (t, g^t) using the existing precomputation techniques, the other part delegates the computation of the values of the form g^{ab} or u^{-1} to untrusted servers. The delegated part of Rand is constructed also as a fully verifiable secure delegation scheme under the OMTUP assumption. This part delegates the costly offline computation of the precomputation step to untrusted servers in contrast to the previous proposals which implicitly require to compute such values offline by the delegator itself. Besides reducing the offline computation of the delegator, the partial delegation of Rand enables also a more efficient realization of $\mathsf{VerPair}$ by reducing the online workload of the delegator. In particular, we give a complete delegation scheme which requires neither online nor offline computation of modular exponentiations and elliptic curve scalar multiplications.
In particular, this paper has two major goals:

- Recently many new delegation schemes for pairings have been proposed to meet the full verifiability property, (mostly) without interactive computations of modular exponentiations and elliptic curve scalar multiplications [23,39,35,34,27]. Together with [41] we analyze these delegation schemes, and present several attacks on their security and/or verifiability properties:
  1. We show that the scheme in [41] does not satisfy the claimed verifiability. More concretely, a malicious server U_i, $i \in \{1, 2\}$, could cheat the delegator with probability at least $1/10(s-1)$ instead of the authors's claim with probability $1/120(s-1)^2$. Therefore, the scheme offered no significant computational advantage when compared with the scheme in [44]. Additionally, communication overhead is unfortunately much higher (10 calls to the servers in [41] instead of 6 calls in [44]).
  2. We show that the scheme in [23] does not satisfy the claimed security. More concretely, we mount a simple brute-force attack that a malicious server U_i, $i \in \{1, 2\}$, could prepare a look-up table with 3000 entries from $\mathbb{G}_1$. If the private input $A \in \mathbb{G}_1$ is delegated once more to U_i (i.e. to delegate the computation of $e(A, Y)$ for arbitrary $Y \in \mathbb{G}_2$), then U_i could easily find the secret input A with a simple search within the look-up table. Obviously, the secret input B (hence $e(A, B)$) could easily be

found by U_i by a simple search within another look-up table (with 3000 entries from $\mathbb{G}_2$). Hence, the scheme is totally insecure.
3. We show that the scheme in [39] does not satisfy the full verifiability. In particular, a malicious server U_i, $i \in \{1, 2\}$, could cheat the delegator with probability at least $1/6$.
4. We show that the scheme in [35] does not satisfy the full verifiability. In particular, a malicious server U_i, $i \in \{1, 2\}$, could cheat the delegator with probability at least $1/2$.
5. We show that the scheme in [34] does not satisfy full verifiability. In particular, a malicious server U could cheat the delegator with probability at least $1/6$. Even worse, the scheme reveals the private points $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$ resulting in a totally insecure scheme.
6. We show that the scheme in [27] does not satisfy the full verifiability. In particular, a malicious server U_i, $i \in \{1, 2\}$, could always cheat the delegator.

– We introduce the first fully verifiable secure delegation scheme VerPair under the OMTUP assumption which does not require any interactive modular exponentiations and elliptic curve scalar multiplications. Additionally, by reducing the requirement of the invocation of a *Rand* scheme by a partial delegation, VerPair offers a considerably efficient and secure *complete* delegated pairing computation mechanism. In particular, VerPair can effectively be utilized even if the delegator has only highly limited resources.

2 Verifiable & Secure Delegation of General Pairing Computation: Preliminaries and Security Model

In this section, we first revisit the basic notions related to pairings. Then, we give a brief overview for the requirements of a secure and verifiable delegation of the *general* pairing computation. This section ends with a simulation-based security model for the delegation of pairings under the OMTUP assumption.

Remark 1. Our security model basically adapts the ideas of [25] for delegating group exponentiations into the delegation of general pairing computation as in previous works [18,44]. We note that it would also be possible to give a relaxed security model based on indistinguishability. This could be done for example firstly by adapting a security model recently proposed in the extended version of [19] into the delegation of pairing computation which is originally proposed for the delegation of group exponentiations without any verification part. Secondly, the right indistinguishability-based verifiability notion could be adapted into the delegation of pairing computation setting, for instance, by using the verifiability definition of [16] (also originally proposed for the delegation of group exponentiations). However, since the simulation-based security notions form the most strong security models, we prefer also to use these following the lines of the previous results.

2.1 Preliminaries

Pairings. Pairing-based cryptography requires computation of bilinear maps of elliptic curves over finite fields. There are different choices for bilinear maps mainly because of efficiency and security considerations which we mostly see as an abstract generic operation and call them simply *pairings*. More formally, we assume that $(\mathbb{G}_1, +) =< P_1 >$ and $(\mathbb{G}_2, +) =< P_2 >$ are two additive cyclic groups of prime order q and $(\mathbb{G}_3, \cdot)$ be a multiplicative cyclic group of order q, where $P_1 \in \mathbb{G}_1$ and $P_2 \in \mathbb{G}_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, resp. A pairing is a map

$$e : \ \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

satisfying the following properties [7]:

- **Bilinearity:** For all $P, P' \in \mathbb{G}_1, Q, Q' \in \mathbb{G}_2$, e is a group homomorphism in each component, i.e.
  1. $e(P + P', Q) = e(P, Q) \cdot e(P', Q)$,
  2. $e(P, Q + Q') = e(P, Q) \cdot e(P, Q')$.
- **Non-degeneracy:** e is non-degenerate in each component, i.e.,
  1. For all non-zero $P \in \mathbb{G}_1$, there is an element $Q \in \mathbb{G}_2$ such that
  $$e(P, Q) \neq 1,$$
  2. For all non-zero $Q \in \mathbb{G}_2$, 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 in polynomial-time in the length of q.

Throughout the paper, we denote by $\mathbb{Z}_q$ (or $\mathbb{F}_q$) the field $\mathbb{Z}/q\mathbb{Z}$, and by $\mathbb{Z}_q^*$ the multiplicative group of $\mathbb{Z}_q$. The expression $x \leftarrow T$ and $x \leftarrow y$ denote the probabilistic process of random and independent choice of x from a set T, and assigning the value of x to a variable y, respectively.

Secure and Verifiable Delegation of Pairing Computations. Following the lines of [20,15], a secure fully verifiable delegation protocol for pairing computation is expected to satisfy informally the following main properties:

- **Completeness:** After completion of the protocol with an honest program U, the delegator T obtains $e(A, B)$ on the inputs $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, except with negligible probability.
- **Secrecy** & **Privacy:** An untrusted program should not learn any information about the input points $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$. More formally, for any malicious program U, there exists a simulator $\mathcal{S}$ such that for any $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, the output of $\mathcal{S}$, to which the points A are $B$are not given, is computationally indistinguishable from the program's view:
  $$\mathcal{S} \overset{c}{\equiv} \mathsf{View}_U(A, B).$$

– **Verifiability** The delegator should be able to detect a cheating program, except with negligible probability. More formally, for any cheating program U and for any input values $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, the delegator outputs either $\perp$ or $e(A, B) \in \mathbb{G}_3$, except with negligible probability.

We call a program (trusted or untrusted) a server throughout this paper.

Steps of a Delegation Scheme: A delegation scheme for pairing computation under the OMTUP assumption consists of mainly 4 steps:

1. **Precomputation** Rand**:** (Pseudo-)random pairs of the form $(k_1, k_1P_1) \in \mathbb{Z}_q \times \mathbb{G}_1$, $(k_2, k_2P_2) \in \mathbb{Z}_q \times \mathbb{G}_2$, and $(k_3, g^{k_3}) \in \mathbb{Z}_q \times \mathbb{G}_3$ are computed to randomize the input points $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, where $g = e(P_1, P_2) \in \mathbb{G}_3$. Additionally, inverses of the form $t^{-1} \in \mathbb{Z}_q^*$ and some multiplications of (pseudo-)random elements in $\mathbb{Z}_q^*$ are sometimes computed to utilize the delegation efficiently.
2. **Randomizing the input points** $A \in \mathbb{G}_1$**,** $B \in \mathbb{G}_2$**.** The input points are randomized by the client by performing only point additions (PA'S) in $\mathbb{G}_1$ and $\mathbb{G}_2$ with precomputed (pseudo-)random points.
3. **Delegation to servers.** The randomized points (possibly together with some other points from the precomputation step) are queried to the servers U_1 and U_2. This delegation could be performed sequentially (different rounds) or concurrently (more than one delegation of pairing computation in a single round). For $i = 1, 2$, $U_i(X, Y)$, $U_i(\alpha, h)$, and $U_i(\beta, \cdot^{-1})$ denote the delegation of $e(X, Y)$ with $X \in \mathbb{G}_1$, $Y \in \mathbb{G}_2$, h^α with $h \in \mathbb{G}_3$, $\alpha \in \mathbb{Z}_q^*$, and β^{-1} with $\beta \in \mathbb{Z}_q^*$, resp.
4. **Verification of the delegated computation.** Upon receiving the queries from the servers U_1 and U_2 the validity of the delegated computation is verified by performing comparison of the received data, and/or PA's in $\mathbb{G}_1$ and $\mathbb{G}_2$, and modular multiplications (MM's) in $\mathbb{G}_3$ with the received data and some points from the precomputation step. If the verification fails, an error message $\perp$ is returned.
5. **Derandomizing the outputs and computing** $e(A, B) \in \mathbb{G}_3$**.** If the verification is successful, then the output $e(A, B)$ is returned by performing only PA's in $\mathbb{G}_1$ and $\mathbb{G}_2$, and MM's in $\mathbb{G}_3$ of the received data and some points from the precomputation.

Remark 2. 1. Usually, a fixed number of offline elements $(x_i, x_iP_1, x_iP_2, g^{x_i})$ are stored to T at the initialization of the system by a trusted server. Then, Rand computes dynamic pairs of the form $(k_1, k_1P_1), (k_2, k_1P_1), (k_3, g^{k_3})$ by choosing a random subset of the static values and performing PA's and MM's. Rand could be realized in such a way that output distribution is statistically close to the uniform distribution for carefully chosen parameters [13,12,45].
2. In this paper, we initiate a *hybrid approach*, i.e. we partially delegate Rand. Note that the delegation of Rand could be done independent of the main steps of the delegation which provides to reduce the online workload of the

delegator considerably while enabling the delegator to precompute values of the form (a, b, g^{ab}) and u^{-1} for randomly chosen elements $a, b \in \mathbb{Z}_q^*$ of (a multiplicatively written) group $\mathbb{G}$ with $g, u \in \mathbb{G}$. Since, the precomputation techniques [13,12,45] do not include the computation of such g^{ab}, the only realizable way of producing such values is to compute these offline by the delegator although most papers regard them as granted. This makes these schemes totally impractical when the delegator has restricted resources.

2.2 Definitions & Security Model

In this section, we adapt the simulation-based security notions of [25] following the previous delegation schemes [18,44].
Informally, a *trusted* honest but resource-component part T securely delegates some work to a potentially untrusted component U, and (T, U) forms a delegated-secure implementation of a cryptographic scheme Alg if

- T and U jointly implements $\mathsf{Alg} = T^U$,
- if T is given oracle access to a malicious U', $U \neq U'$, then despite the assumption that U' acts maliciously every time it is invoked by recording its own computation over time, it cannot obtain any information about both the input and the output of $T^{U'}$.

Since U' is not the single entity acting maliciously and interacting with Alg, the adversary $\mathcal{A}$ consists of two parts:

- the adversarial component U' operating in place of U,
- an adversarial component *environment* E submitting adversarially chosen inputs to Alg.

The fundamental assumption in [25] is that E and U' may develop a joint strategy before until interacting with T, but they will not have a direct communication channel thereafter. According to this assumption, first logical divisions of inputs to Alg are

- *secret* information solely available to T,
- environmentally *protected* information available to both T and E but nor available to U',
- *protected* information available to both T and U' but nor available to E,
- *unprotected* information available to T, E, and U'.

This division includes in particular the cases where E may have access something about the protected inputs to Alg which is either not available to U' or not available to E. More concretely, T might hide some of these from U' whereas E can clearly see all of its own adversarial inputs to Alg. Likewise, T might hide some information from E whereas U' can see some of protected random inputs generated solely by T which E cannot see as E and U' can only communicate through T. Throughout the paper both environmentally protected and protected

information are called protected if there is no need to distinguish the cases from which adversarial component the information is protected.
Moreover, the above divisions have additional subdivisions depending on whether the inputs were generated *honestly* or *adversarially.* Note that there cannot however exist a adversarial secret input.
Similarly, the outputs of Alg are logically divided into *secret, protected,* and *unprotected* outputs. The simplified formal definition is given as follows (i.e. by neglecting possible relations of the inputs and outputs to each other):

Definition 1. *[25] (**Algorithm with delegated-IO**)*
An algorithm Alg *is said to obey the delegation input/output specification if it takes five inputs, and produces three outputs. The first three inputs are generated by an honest party T, and are classified by how much information about them is available to the adversary $\mathcal{A} = (E, U')$, where E is the adversarial environment submitting adversarially chosen inputs to* Alg*, and U' is the adversarial component operating in place of oracle U. 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 either be known to E, but is protected from U, or known to U, but is protected from E; 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 adversarial, which may be known by both 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 it is protected from U; and the third is unprotected, which may be known by both E and U.*

Definition 2. *[25] (**Delegated Security**)*
Let $\mathsf{Alg}(\cdot,\cdot,\cdot,\cdot,\cdot)$ *be an algorithm with delegated-IO. A pair of algorithms (T, U) is said to be a delegated-secure implementation of an algorithm* Alg *if:*
Completeness. *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. We assume that the honestly-generated inputs are chosen by a process I.*

– ***Pair One:*** *$EVIEW_{real} \sim EVIEW_{ideal}$:*
  • *The view that the adversarial environment E obtains by participating in the following REAL process:*

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

$EVIEW_{real} = EVIEW^i_{real}$ *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 round* i *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. for* i *with* $stop^i = TRUE$*.*

- *The IDEAL process:*

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

$EVIEW_{ideal} = EVIEW^i_{ideal}$ *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^i_p, y^i_u) *generated by* Alg*, or replace them with some other values* (Y^i_p, Y^i_u) *Note that this process is captured by having the indicator variable* $replace^i$ *be a bit determining whether* y^i_p *will be replaced with* Y^i_p*. In doing so, it is allowed to query the oracle* U'*; moreover,* U' *saves its state as in the real experiment.*

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

- *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:*

$$\begin{gathered} UVIEW^i_{ideal} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1}); \\ (estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, estate^{i-1}, x^i_{hp}, x^i_{hu}, y^{i-1}_p, y^{i-1}_u); \end{gathered}$$

$$(astate^i, y_s^i, y_p^i, y_u^i) \leftarrow \mathsf{Alg}(astate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i);$$
$$(sstate^i, ustate^i) \leftarrow S_2^{U'(ustate^{i-1})}(sstate^{i-1}, x_{hu}^{j^i}, x_{au}^i) : (ustate^i)\}$$

$UVIEW_{ideal} = UVIEW^i_{ideal}$ if $stop^i = TRUE$.
In the ideal process, we have a stateful simulator S_2 who, equipped with only the unprotected inputs (x^i_{hu}, x^i_{au}), queries U'. As before, U' may maintain state.

In our security model we assume one-malicious version of a two-untrusted program (OMTUP) model as we discussed in Introduction. More concretely, we have $U = (U_1, U_2)$, where only one of the U_i is assumed to be malicious, $i = 1, 2$.

Definition 3. *(α-**Efficiency**)*
A pair of algorithms (T, U_1, U_2) are an α-efficient delegated-implementation of an algorithm Alg *if (1) T^{U_1,U_2} is a complete implementation of* Alg*, and (2) $\forall$ inputs x, the running time of T is smaller than an α-multiplicative factor of the running time of* $\mathsf{Alg}(x)$*.*

Definition 4. *(β-**Verifiability**)*
A pair of algorithms (T, U_1, U_2) are a β-verifiable delegated implementation of an algorithm Alg *if (1) T^{U_1,U_2} is a complete 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 larger than β. In particular, if T will always detect the error, except with negligible probability, i.e. $1 - \beta$ is negligibly small, then a pair of algorithms (T, U_1, U_2) are a fully verifiable delegated implementation of an algorithm* Alg*.*

Definition 5. *(α, β-**Delegated Secure Implementation**)*
A pair of algorithms (T, U_1, U_2) are an (α, β)-delegated secure implementation of an algorithm Alg *if they are both α-efficient and β-verifiable. In particular, a pair of algorithms (T, U_1, U_2) are a fully verifiable $(\alpha, 1)$-delegated secure implementation of an algorithm* Alg *if they are a fully verifiable delegated implementation of an algorithm* Alg*.*

3 Verifiability And/Or Security Issues of Recent Schemes

In this section, we give security analysis of recently proposed delegation schemes for general pairing computation [41,23,39,35,34,27]. To give a self-contained section, and make the attacks easily understandable for readers, we explain each attack after briefly recalling the original schemes.

3.1 Ren *et al.*'s Scheme from Security and Communication Networks [41]

Ren *et al.* proposed the following scheme: The Rand algorithm outputs the values

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

where P is the generator of $\mathbb{G}_1$ (of prime order q) and Q is the generator of $\mathbb{G}_2$ (of prime order q), $a_i, b_i \in_R \mathbb{Z}_q^*$, $1 \leq i \leq 4$, $t_j \in \{2, 3, \cdots, s\}$, and $s \in \mathbb{N}$ is a small number.
Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The scheme in [41] is given as follows:

1. T queries U_1 in random order as follows:

   . $U_1(X_1 = A + a_1P + a_2P, Y_1 = B + b_1Q + b_2Q) \longleftarrow \alpha_{11} = e(X_1, Y_1)$,
   . $U_1(X_2 = A + b_1P + b_2P, Y_2 = a_3Q) \longleftarrow \alpha_{12} = e(X_2, Y_2)$,
   . $U_1(X_3 = -(a_1P + a_2P + b_3P), Y_3 = B + b_3Q) \longleftarrow \alpha_{13} = e(X_3, Y_3)$,
   . $U_1(X_4 = A + a_4P, Y_4 = -(t_1b_1Q + b_2Q)) \longleftarrow \alpha_{14} = e(X_4, Y_4)$,
   . $U_1(X_5 = -(t_2a_1P + a_2P), Y_5 = B + b_4Q) \longleftarrow \alpha_{15} = e(X_5, Y_5)$.
2. Then similarly, T queries U_2 in random order as follows:
   . $U_2(X_1 = A + a_1P + a_2P, Y_1 = B + b_1Q + b_2Q) \longleftarrow \alpha_{21} = e(X_1, Y_1)$
   . $U_2(X_6 = A - a_3P, Y_6 = -(b_1Q + b_2Q + a_3Q)) \longleftarrow \alpha_{22} = e(X_6, Y_6)$,
   . $U_2(X_7 = b_3P, Y_7 = B + a_1Q + a_2Q) \longleftarrow \alpha_{23} = e(X_7, Y_7)$,
   . $U_2(X_8 = A + a_4P, Y_8 = -(b_1Q + t_1b_2Q)) \longleftarrow \alpha_{24} = e(X_8, Y_8)$,
   . $U_2(X_9 = -(a_1P + t_2a_2P), Y_9 = B + b_4Q) \longleftarrow \alpha_{25} = e(X_9, Y_9)$.
3. Upon receiving computation results from both servers, T checks if
   . $\alpha_{11} \stackrel{?}{=} \alpha_{21}$,
   . $(\alpha_{12}\alpha_{22}e(a_3P, a_3Q))^{t_1+1} \stackrel{?}{=} \alpha_{14}\alpha_{24}e(a_4P, b_1Q + b_2Q)^{t_1+1}$,
   . $(\alpha_{13}\alpha_{23}e(b_3P, b_3Q))^{t_2+1} \stackrel{?}{=} \alpha_{15}\alpha_{25}e(a_1P + a_2P, b_4Q)^{t_2+1}$.

If the check is not successful, then T outputs $\perp$. Otherwise, using some precomputed values and $\alpha_{12}, \alpha_{13}, \alpha_{22}, \alpha_{23}$ the delegator T can compute $e(A, B)$. We refer to [41] for the details.
An Attack on The Verifiability of [41]: Assume that the server U_1 is malicious. The probability that U_1 chooses one of the pairs $(\alpha_{12}, \alpha_{14})$ or $(\alpha_{13}, \alpha_{15})$ out of 10 pairs from 5 random queries is at least 1/5. Then, U_1 can guess the right position of α_{12} or α_{13} with probability at least 1/2. Moreover, U_1 (resp. U_2) could correctly guess the value of the right exponent t_i with probability $1/(s-1)$, $i = 1, 2$. Hence, a malicious server U_1 (resp. U_2) could correctly guess $(\alpha_{12}, \alpha_{14})$ or $(\alpha_{13}, \alpha_{15})$ with the correct exponent t_i with probability at least

$$1/10(s-1).$$

Assume without loss of generality that $(\alpha_{12}, \alpha_{14})$ is the correctly guessed pair with the exponent t_1. Then, the server U_1 could send the bogus values with using an arbitrary element $\theta \in \mathbb{G}_3$

$$\gamma_{12} = \alpha_{12} \cdot \theta, \gamma_{14} = \alpha_{14} \cdot \theta^{t_1+1}$$

to the delegator T which successfully enable U_1 to cheat the delegator T, and pass the verification step with probability at least $1/10(s-1)$. Moreover, since after the verification step, the value α_{12} (resp. α_{13}) is also used to recover $e(A, B)$, the output yields to a bogus value instead of $e(A, B)$ with probability at least $1/10(s-1)$. It is obvious that the same simple attack strategy could be used to manipulate α_{22} or α_{23} and α_{24} or α_{25} for U_2.
For $s = 4$, the verification step in [41] is successful with probability $1 - 1/10 \cdot 3 \approx 0.967$ instead of the author's claim with probability ≈ 0.999. Now, if the verification probabilities of [44] and [41] are chosen to be *the same* (i.e. by choosing appropriate values for the adjustable verification probabilities in both schemes), then it can be seen that the proposed scheme in [41] has almost *no efficiency* benefit when carefully compared with the Tian *et al.*'s scheme [44]. Additionally, it has *worse communication* overhead than [44] (with 10 calls to the servers instead of 6 calls in [44]). Hence, the scheme in [41] is less practical than the scheme in [44] with almost no computational advantages and requirement of additional bandwidth.

3.2 Dong *et al.*'s Scheme from KSII Trans. Int. and Inf, Systems [23]

Dong *et al.* proposed the following scheme: The Rand algorithm outputs the following values

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

where P is the generator of $\mathbb{G}_1$ (of prime order q), Q is the generator of $\mathbb{G}_2$ (of prime order q), $a_j, b_j \in_R \mathbb{Z}_q^*$, $1 \le j \le 7$, $r_i, r_i' \in \{\pm1, \pm2, \pm4\}$, and

$$\begin{gathered}
r_4b_4 = r_6b_6, r_5b_5 = r_7b_7, \\
r_4'a_4 = r_6'a_6, r_5'a_5 = r_7'a_7, \\
\sum_{j=4}^{7} b_j = -b_1, \sum_{j=4}^{7} a_j = -a_1.
\end{gathered}$$

Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The scheme in [23] is as follows:

1. T queries U_1 in random order as follows:

   . $U_1(A + a_1P, B + b_1Q) \longleftarrow \theta_{11} = e(A + a_1P, B + b_1Q)$,
   . $U_1(A + a_2P, b_4Q) \longleftarrow \alpha_{11} = e(A + a_2P, b_4Q)$,
   . $U_1(A + a_3P, b_5Q) \longleftarrow \alpha_{12} = e(A + a_3P, b_5Q)$,
   . $U_1(a_4P, B + b_2Q) \longleftarrow \beta_{11} = e(a_4P, B + b_2Q)$,
   . $U_1(a_5P, B + b_3Q) \longleftarrow \beta_{12} = e(a_5P, B + b_3Q)$.
2. Then similarly, T queries U_2 in random order as follows:
   . $U_2(A + a_1P, B + b_1Q) \longleftarrow \theta_{21} = e(A + a_1P, B + b_1Q)$,
   . $U_2(A + a_2P, b_6Q) \longleftarrow \alpha_{21} = e(A + a_2P, b_6Q)$,
   . $U_2(A + a_3P, b_7Q) \longleftarrow \alpha_{22} = e(A + a_3P, b_7Q)$,
   . $U_2(a_6P, B + b_2Q) \longleftarrow \beta_{21} = e(a_6P, B + b_2Q)$,
   . $U_2(a_7P, B + b_3Q) \longleftarrow \beta_{22} = e(a_7P, B + b_3Q)$.

Then, T performs the verification step using $\theta_{11}, \theta_{12}, \alpha_{ij}, \beta_{ij}$, $1 \leq i, j \leq 2$, and after successful verification T computes $e(A, B)$ by multiplying these values with some precomputed values. We refer to [23] for the details of the verification step and the computation of $e(A, B)$.

An Attack on The Security of [23]: Since by the specification of the scheme, the equations

$$a_4P + a_5P + (r'_4/r'_6)a_4P + (r'_5/r'_7)a_5P = -a_1P$$

$$b_4Q + b_5Q + (r_4/r_6)b_4Q + (r_5/r_7)b_5Q = -b_1Q$$

hold. A malicious server U_1 could prepare a preliminary look-up table with each 1000 entries, since we have 10 possibilities for the correct pair a_4P and a_5P (resp. similarly for b_4Q and b_5Q) and 10 possibilities for each $r_i/r_j \in \{\pm 1/4, \pm 1/2, \pm 1, \pm 2, \pm 4\}$ (resp. $r'_i/r'_j \in \{\pm 1/4, \pm 1/2, \pm 1, \pm 2, \pm 4\}$ with $i \in \{4, 5\}$ and $j \in \{6, 7\}$.

Then for each entry in the preliminary look-up table, we have 3 possibilities for $A + a_1P$ (resp. $B + b_1Q$). Adding all these possibilities to the preliminary look-up table yields to a look up table with 3000 entries for possible candidates of $A \in \mathbb{G}_1$ (resp. another look-up table yields to a look up table with 3000 entries for possible candidates of $B \in \mathbb{G}_2$).

If the delegator T uses $A \in \mathbb{G}_1$ (resp. $B \in \mathbb{G}_2$) to delegate a pairing computation of the form $e(A, Y)$ with $Y \in \mathbb{G}_2$ (resp. $e(X, B)$ with $X \in \mathbb{G}_1$), then a malicious server U_1 could successfully find A (resp. B). Obviously, any delegation of pairing computations of the form $e(A, Y)$ and $e(X, B)$ would leak the output value $e(A, B) \in \mathbb{G}_3$. Hence, the scheme in [23] is *completely insecure* in its current form.

The only possible way of having a *secure version* of the scheme in [23] seems to to choose r_i and r'_i such that bit-lengths of r_i and r'_i are long enough, e.g. at least circa 76-bits for 80-bits security level. On the other side, this would make the scheme *totally inefficient*. In other words, in this case the scheme would have a huge computational overhead when compared with a direct computation of $e(A, B)$ by the delegator T, which could directly *annihilate* the purpose of pairing delegation, its *raison d'étre*. The computational complexity would then be very similar to the schemes of Chaevallier-Mames *et al.*, [20], Kang *et al.* [28], and Canard *et al.* [15] as outlined in Introduction. Note also that the schemes in [20,28,15] have less *communication overhead* than the scheme in [23].

3.3 Ren *et al.*'s Scheme from SCIENCE CHINA, Inf. Sciences [39]

Ren *et al.* proposed the following scheme: The Rand algorithm outputs the following values

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

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

where P is the generator of $\mathbb{G}_1$ (of prime order q), $\hat{P}$ is the generator of $\mathbb{G}_2$ (of prime order q), $a, b, a_i, b_i \in_R \mathbb{Z}_q^*$, $1 \leq i \leq 3$.
Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The scheme in [39] is as follows:

1. T queries U_1 in random order as follows:
   . $U_1(A - aP, B - b\hat{P}) \longleftarrow \alpha_{11} = e(A - aP, B - b\hat{P})$,
   . $U_1(A - aP + a_3P, b_1\hat{P}) \longleftarrow \alpha_{12} = e(A - aP + a_3P, b_1\hat{P})$,
   . $U_1(a_1P, B - b\hat{P} + b_3\hat{P}) \longleftarrow \alpha_{13} = e(a_1P, B - b\hat{P} + b_3\hat{P})$.
2. Then similarly, T queries U_2 in random order as follows:
   . $U_2(A-aP+a_1P, B-b\hat{P}+b_1\hat{P}) \longleftarrow \alpha_{21} = e(A-aP+a_1P, B-b\hat{P}+b_1\hat{P})$,
   . $U_2(A - aP + a_2P, b\hat{P}) \longleftarrow \alpha_{22} = e(A - aP + a_2P, b\hat{P})$,
   . $U_2(aP, B - b\hat{P} + b_2\hat{P}) \longleftarrow \alpha_{23} = e(aP, B - b\hat{P} + b_2\hat{P})$.
3. T computes
   . $\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1} = e(A - aP, b_1\hat{P})$,
   . $\alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1} = e(a_1P, B - b\hat{P})$,
   . $\alpha_{22} \cdot e(a_2P, b\hat{P})^{-1} = e(A - aP, b\hat{P})$,
   . $\alpha_{23} \cdot e(aP, b_2\hat{P})^{-1} = e(aP, B - b\hat{P})$.
4. T choices $t_1, t_2 \in_R \mathbb{Z}_q^*$ and queries U_1 in random order as follows:
   . $U_1(e(A - aP, b\hat{P}), t_1b^{-1}) \longleftarrow \alpha_{14} = e(A - aP, t_1\hat{P})$,
   . $U_1(e(aP, B - b\hat{P}), t_2a^{-1}) \longleftarrow \alpha_{15} = e(t_2P, B - b\hat{P})$,
5. Similarly, T queries U_2 in random order as follows:
   . $U_2(e(A - aP, b_1\hat{P}), t_1b_1^{-1}) \longleftarrow \alpha_{24} = e(A - aP, t_1\hat{P})$,
   . $U_2(e(a_1P, B - b\hat{P}), t_2a_1^{-1}) \longleftarrow \alpha_{25} = e(t_2P, B - b\hat{P})$,
6. T verifies
   . $\alpha_{14} \stackrel{?}{=} \alpha_{24}$,
   . $\alpha_{21} \stackrel{?}{=} \alpha_{11} \cdot e(A - aP, b_1\hat{P}) \cdot e(a_1P, B - b\hat{P}) \cdot e(a_1P, b_1\hat{P})$.
7. If the verification step fails T outputs $\perp$.

8. Else, T outputs

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

An Attack on The Verifiability of [39]: Suppose U_1 is a malicious and U_2 is an honest server. Firstly, it could successfully guess the correct positions of α_{1i}, $i = 1, 2, 3$ with probability $1/6$. After a successful guess, U_1 knows a_1P and $b_1\hat{P}$, and could easily compute a_3P by subtracting the first component of α_{12} from the first component of α_{11} as well. Similarly, by subtracting the second component of α_{13} from the second component of α_{11}, the point $b_3\hat{P}$ could be computed by U_1. Then, U_1 could compute $e(a_3P, b_1\hat{P})$ and $e(a_1P, b_3\hat{P})$. Moreover, the it could compute values

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

Then, U_1 could simply send to the delegator T the following bogus values instead of α_{1i}, $i = 1, 2, 3$:

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

After receiving the values θ_{1i}, α_{2i}, $i = 1, 2, 3$, the delegator T would compute the following values following the scheme specification:

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

In the second round, the malicious server U_1 could send

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

instead of α_{14} and α_{14}, respectively. Note that U_1 could manipulate α_{14} and α_{15} with θ_{14} and θ_{15} with probability 1 since only squares are taken which are independent of the correct positions of α_{14} and α_{15}. Then, following the protocol honestly, the second server U_2 would compute

. $U_2(e(A - aP, 2b_1\hat{P}), t_1b_1^{-1}) \longleftarrow \alpha_{24} = e(A - aP, 2t_1\hat{P})$,
. $U_2(e(2a_1P, B - b\hat{P}), t_2a_1^{-1}) \longleftarrow \alpha_{25} = e(2t_2P, B - b\hat{P})$,

implying that

. $\theta_{14} = \alpha_{24}$,
. $\alpha_{21} = \theta_{11} \cdot e(A - aP, 2b_1\hat{P}) \cdot e(2a_1P, B - b\hat{P}) \cdot e(a_1P, b_1\hat{P})$.

After passing the verification step with these bogus values, the output would be the bogus value

$$e(A,B)e(A-aP,b_1\hat{P})^{-1}e(a_1P,b_3\hat{P})^{-1} = \theta_{11}\cdot e(A-aP,b\hat{P}), e(aP,B-b\hat{P})\cdot e(aP,b\hat{P}).$$

instead of $e(A, B)$. Note that the values $e(A - aP, b_1\hat{P})$ and $e(a_1P, B - b\hat{P})$ are computed by the honest server U_2, hence these values remain unchanged.
This attack shows that the scheme in [39] does *not satisfy* the *full verifiability*, and it is a scheme in which a malicious U_1 could pass the verification step with bogus values with probability at least 1/6. Hence, U_1 could manipulate the output with probability at least 1/6.

3.4 Luo *et al.*'s Scheme from IEEE TrustCom 2016 [35]

Luo *et al.* proposed a scheme for delegation of generic batch pairings [35]. They use a multiplicative notation for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. To be consistent with the rest of the paper, we summarize their scheme for a single delegation $e(A, B)$, where $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$ in the usual additive notation. The Rand algorithm outputs the following values

$$uP, 2uP, vQ, 2vQ, -2vQ, xP, 2xP, yQ, 2yQ, -2yQ,$$
$$e(P,Q)^{2uv}, e(P,Q)^{2xy},$$

where P is the generator of $\mathbb{G}_1$ (of prime order q) and Q is the generator of $\mathbb{G}_2$ (of prime order q) and $u, v, x, y \in_R \mathbb{Z}_q^*$.
Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The scheme in [35] is as follows:

1. T queries U_1 in random order as follows:

   . $U_1(A + uP, B + vQ) \longleftarrow \alpha_1 = e(A + uP, B + vQ)$,
   . $U_1(A + 2xP, -B - 2yQ) \longleftarrow \alpha_2 = e(A + 2xP, -B - 2yQ)$.
2. Then similarly, T queries U_2 in random order as follows:
   . $U_2(A + xP, B + yQ) \longleftarrow \beta_1 = e(A + xP, B + yQ)$,
   . $U_2(A + 2uP, -B - 2vQ) \longleftarrow \beta_2 = e(A + 2uP, -B - 2vQ)$.
3. T verifies
$$\alpha_1^2 \cdot \beta_2 \cdot e(P,Q)^{2uv} \stackrel{?}{=} \beta_1^2 \cdot \alpha_2 \cdot e(P,Q)^{2xy}.$$
4. If the verification step fails T outputs $\perp$.
5. Else, T outputs
$$e(A,B) = \alpha_1^2 \cdot \beta_2 \cdot e(P,Q)^{2uv}.$$

An Attack on The Verifiability of [35]: Assume U_1 is a malicious server. It could successfully guess the positions of α_1 and α_2 with probability 1/2. Instead of sending α_1 and α_2, U_1 could send the bogus values $\theta_1 = \alpha_1 \cdot C$ and $\theta_2 = \alpha_2 \cdot C^2$ and would pass the verification step with probability at least 1/2, where $C \in \mathbb{G}_3$ is any arbitrary bogus value. Then, the scheme outputs $C^2 e(A,B)$ instead of $e(A,B)$ with probability at least 1/2. Obviously, a malicious server U_2 could mount a similar simple attack.
This fairly simple attack shows that the authors' claim in [35] of having a *fully verifiable scheme* is unfortunately *false*.

3.5 Luo *et al.*'s Scheme from Advances in Internetworking, Data & Web Technologies, 2018 [34]

Luo *et al.* proposed the following scheme for delegation of pairing computation under OUP model [34]: The Rand algorithm outputs the following values

$$a, b, k_1, k_2, aP_1, bP_2, ak_2^{-1}P_1, bk_1^{-1}P_2, e(P_1,P_2)^{-ab(k_1k_2)^{-1}},$$

where P_1 is the generator of $\mathbb{G}_1$ (of prime order q) and P_2 is the generator of $\mathbb{G}_2$ (of prime order q) and $a, b, k_1, k_2 \in_R \mathbb{Z}_q^*$.
Let T denote the delegator and U be the untrusted server. The scheme in [34] is as follows:

1. T queries U_1 in random order as follows:

   . $U(A + aP_1, B + bP_2) \longleftarrow V_1 = e(A + aP_1, B + bP_2)$,
   . $U(k_1A + ak_2^{-1}P_1, k_2B + bk_1^{-1}P_2)$
   $\longleftarrow V_2 = e(k_1A + ak_2^{-1}P_1, k_2B + bk_1^{-1}P_2)$,
   . $U(A, B + bP_2) \longleftarrow V_3 = e(A, B + bP_2)$,
   . $U(A + aP_1, B) \longleftarrow V_4 = e(A + aP_1, B)$,
   . $U(k_1A, k_2B + bk_1^{-1}P_2) \longleftarrow V_5 = e(k_1A, k_2B + bk_1^{-1}P_2)$,
   . $U(k_1A + ak_2^{-1}P_1, k_2B) \longleftarrow V_6 = e(k_1A + ak_2^{-1}P_1, k_2B)$.
2. T verifies
$$V_3 \cdot V_4^{-1} \stackrel{?}{=} V_5 \cdot V_6^{-1}.$$
3. If the verification step fails T outputs $\perp$.
4. Else, T outputs
$$e(A,B) = \left(\frac{V_2 \cdot e(P_1,P_2)^{-ab(k_1k_2)^{-1}}}{V_1}\right)^{(k_1k_2-1)^{-1}}.$$

Attacks on The Security and Verifiability of [35]: A malicious server U could store 6 possible values of the first components of V_i (resp. U could store probability 6 values of the second components of V_i). If the delegator T uses either $A \in \mathbb{G}_1$ or $B \in \mathbb{G}_2$ to delegate a pairing computation of the form $e(A,Y)$ with $Y \in \mathbb{G}_2$ or to delegate a pairing computation of the form

$e(X, B)$ with $X \in \mathbb{G}_1$, then a malicious server U_1 could successfully find A or B. Obviously, any delegation of both $e(A, Y)$ and $e(X, B)$ would leak the output value $e(A, B) \in \mathbb{G}_3$, too. Hence, the scheme in [34] is *completely insecure*.
On the other side, the authors' claim for *full verifiability* of the scheme in [34] does *not hold* either. A malicious server U could guess the correct position of V_1 with probability at least $1/6$ (resp. the correct position of V_2 with probability at least $1/6$) and outputs an arbitrary bogus value $C_1 \in \mathbb{G}_3$ (resp. $C_2 \in \mathbb{G}_3$) instead of V_1 (resp. V_2), and pass the verification step with C_1 (resp. C_2). This results in a bogus value instead of $e(A, B)$ with probability at least $1/6$.
In addition to having neither secure nor fully verifiable scheme, the scheme in [34] is also *totally inefficient*, since the values $k_1 A$ and $k_2 B$ need to be computed interactively. In particular, the scheme would be at least as computationally inefficient as a direct computation of $e(A, B)$ by the delegator T, which could directly *annihilate* the purpose of pairing delegation, its *raison d'étre*.

3.6 Kalkar *et al.*'s Scheme from International Conference on Information Security Theory and Practice (WISTP'2017) [27]

The following scheme is proposed by Kalkar *et al.*: The Rand algorithm outputs the following values

$$\alpha, \beta, x, y, m, n, \alpha P_1, xP_1, yP_1, \alpha P_1 - xP_1, \alpha P_1 - yP_1, mP_2, nP_2, \beta P_2, e(\alpha P_1, \beta P_2),$$

where P_1 is the generator of $\mathbb{G}_1$ (of prime order q) and P_2 is the generator of $\mathbb{G}_2$ (of prime order q) and $\alpha, \beta, x, y, m, n \in_R \mathbb{Z}_q^*$.

1. T queries U_1 in random order as follows:
   - $U_1(A - \alpha P_1, mP_2) \longleftarrow A_{11} = e(A - \alpha P_1, mP_2)$,
   - $U_1(A - \alpha P_1, B - nP_2) \longleftarrow A_{12} = e(A - \alpha P_1, B - nP_2)$,
   - $U_1(\alpha P_1 - xP_1, B - \beta P_2) \longleftarrow A_{13} = e(\alpha P_1 - xP_1, B - \beta P_2)$,
   - $U_1(yP_1, B - \beta P_2) \longleftarrow A_{14} = e(yP_1, B - \beta P_2)$.
2. Similarly, T queries U_2 in random order as follows:
   - $U_2(A - \alpha P_1, nP_2) \longleftarrow A_{21} = e(A - \alpha P_1, nP_2)$,
   - $U_2(A - \alpha P_1, B - mP_2) \longleftarrow A_{22} = e(A - \alpha P_1, B - mP_2)$,
   - $U_2(\alpha P_1 - yP_1, B - \beta P_2) \longleftarrow A_{23} = e(\alpha P_1 - yP_1, B - \beta P_2)$,
   - $U_2(xP_1, B - \beta P_2) \longleftarrow A_{24} = e(xP_1, B - \beta P_2)$.
3. T verifies
   - $A_{11}A_{22} \stackrel{?}{=} A_{21}A_{12}$,
   - $A_{13}A_{24} \stackrel{?}{=} A_{23}A_{14}$.
4. If the verification step fails T outputs $\perp$. T computes and outputs
5. Else, T outputs
$$A_{11}A_{22}A_{13}A_{24}e(P_1, P_2)^{\alpha\beta} \;\; .$$

An Attack on The Verifiability of [27]: A malicious server U_1 could send the bogus values CA_{1i} instead of A_{1i}, $i = 1, 2, 3, 4$, and could successfully pass the verification step always, i.e. with probability 1. Then, the delegator computes the bogus output $C^2 e(A, B)$ instead of $e(A, B)$. Obviously, a malicious server U_2 could also mount a similar attack, and could always pass the verification step with bogus values.
This fairly simple attack shows that the the claim in [35] of having a *fully verifiable* scheme is unfortunately *false*. In particular, *no verifiability* is provided in [27].

4 VerPair: An Efficient Fully Verifiable Secure Delegation Scheme For Pairing Computation

In this section, we first propose an efficient fully verifiable secure partial delegation scheme for the precomputation step Rand. Secondly, we introduce VerPair which is a fully verifiable efficient secure delegation scheme for general pairing computation under the OMTUP assumption.

4.1 Rand: A Fully Verifiable Secure Partial Delegation Scheme of The Precomputation Step

The precomputation scheme Rand consists of a precomputation step realized by one of the existing techniques [13,12,45]. The other part consists of a delegation scheme. Before initializing the subroutine Rand, a global security parameter κ is chosen which outputs the *global parameters*

1. the prime number q,
2. the groups $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, and $(\mathbb{G}_3, \cdot)$ of order q,
3. the pairing map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$,
4. the generators P_1, P_2, and $g := e(P_1, P_2)$ of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$, respectively.

Together with these global parameters, the *static* tuples

$$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2) \tag{1}$$

are computed at the initialization of the subroutine Rand, and loaded to the delegator T (by a trusted party, e.g. by using HSM, TPM, etc.), where α_1 and α_2 are random elements in $\mathbb{Z}_q^*$. Note that $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is *secret* and *protected* from U_2, but not necessarily from U_1, and $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is *secret* and *protected* from U_1, but not necessarily from U_2. Rand takes no input except global parameters and static tuples (1).
After calling Rand, the first part chooses random values $a, b, s, , t \in_R \mathbb{Z}_q^*$ and outputs

$$(a, aP_1),\ (b, bP_2),\ (s, sP_1, sP_2) \text{ and } (t, g^t). \tag{2}$$

In the delegated second part of Rand, the delegator T chooses first randomly $t_1, t_2, c_1, c_2 \in_R \mathbb{Z}_q^*$. Then,

1. T queries U_1 in random order as follows:

   . $U_1(a \cdot b - t, g) \longleftarrow \gamma_1 = g^{ab-t}$,
   . $U_1(t_1 \cdot s - t, g) \longleftarrow \gamma_2 = g^{t_1 s - t}$,
   . $U_1(t_2 \cdot s - t, g) \longleftarrow \gamma_3 = g^{t_2 s - t}$,
   . $U_1(t_1 - \alpha_1, P_1) \longleftarrow \gamma_4 = (t_1 - \alpha_1)P_1$,
   . $U_1(t_1 - \alpha_1, P_2) \longleftarrow \gamma_5 = (t_1 - \alpha_1)P_2$,
   . $U_1(t_2 - \alpha_2, P_1) \longleftarrow \gamma_6 = (t_2 - \alpha_2)P_1$,
   . $U_1(t_2 - \alpha_2, P_2) \longleftarrow \gamma_7 = (t_2 - \alpha_2)P_2$,
   . $U_1(c_1 \cdot t_1, \cdot^{-1}) \longleftarrow \theta_{11} = c_1^{-1} t_1^{-1}$,
   . $U_1(c_2 \cdot t_2, \cdot^{-1}) \longleftarrow \theta_{12} = c_2^{-1} t_2^{-1}$.
2. T queries U_2 in random order as follows:

   . $U_2(ab - t, g) \longleftarrow \beta_1 = g^{ab-t}$,
   . $U_2(t_1 s - t, g) \longleftarrow \beta_2 = g^{t_1 s - t}$,
   . $U_2(t_2 s - t, g) \longleftarrow \beta_3 = g^{t_2 s - t}$,
   . $U_2(t_1 - \alpha_1, P_1) \longleftarrow \beta_4 = (t_1 - \alpha_1)P_1$,
   . $U_2(t_1 - \alpha_1, P_2) \longleftarrow \beta_5 = (t_1 - \alpha_1)P_2$,
   . $U_2(t_2 - \alpha_2, P_1) \longleftarrow \beta_6 = (t_2 - \alpha_2)P_1$,
   . $U_2(t_2 - \alpha_2, P_2) \longleftarrow \beta_7 = (t_2 - \alpha_2)P_2$,
   . $U_2(c_1 t_1, \cdot^{-1}) \longleftarrow \theta_{21} = c_1^{-1} t_1^{-1}$,
   . $U_2(c_2 t_2, \cdot^{-1}) \longleftarrow \theta_{22} = c_2^{-1} t_2^{-1}$.
3. After receiving γ_i, θ_{1j} from U_1 and β_i, θ_{2j} from U_2, $1 \leq i \leq 7, \leq j \leq 2$, T verifies
$$\gamma_i \stackrel{?}{=} \beta_i \text{ and } \theta_{1j} \stackrel{?}{=} \theta_{2j}. \tag{3}$$
4. If Equations (3) do not hold simultaneously, then T outputs $\perp$.
5. Else, T outputs
   . $((a, aP_1),\ (b, bP_2),\ (s, sP_1, sP_2),\ (t, g^t),$
   $(t_1,\ t_1 P_1 = \gamma_4 + \alpha_1 P_1,\ t_1 P_2 = \gamma_5 + \alpha_1 P_2),$
   $(t_2,\ t_2 P_1 = \gamma_6 + \alpha_2 P_1, t_2 P_2 = \gamma_7 + \alpha_2 P_2),$
   $g^{ab} = \gamma_1 \cdot g^t, g^{t_1 s} = \gamma_2 \cdot g^t, g^{t_2 s} = \gamma_3 \cdot g^t,$
   $t_1^{-1} = c_1 \cdot \theta_{11},\ t_2^{-1} = c_2 \cdot \theta_{12},\ a \cdot t_1^{-1},\ a \cdot t_2^{-1},\ b \cdot t_1^{-1},\ b \cdot t_2^{-1})$.

Remark 3. We note that the outputs of the delegated part of Rand is always secret or (honest/adversarial) protected except *possibly* $(t_1, t_1 P_1, t_1 P_2)$ from U_1 and $(t_2, t_2 P_1, t_2 P_2)$ from U_2. We refer to Section 5 for the further details. Furthermore, we here give a simple but more efficient two-server version for secure delegation of the modular inversion $t^{-1} \bmod q$ which is first introduced in [16].

4.2 VerPair: A Fully Verifiable Secure Delegation Scheme

The Rand scheme outputs the following values

$$((a, aP_1),\ (b, bP_2),\ (s_1, s_1 P_1),\ (s_2, s_2 P_2),\ (t, g^t),$$
$$(t_1,\ t_1 P_1,\ t_1 P_2), (t_2,\ t_2 P_1,\ t_2 P_2),$$

$$g^{ab},\ g^{t_1 s},\ g^{t_2 s},$$

$$t_1^{-1},\ t_2^{-1},\ at_1^{-1},\ at_2^{-1},\ bt_1^{-1},\ bt_2^{-1}),$$

where P_1 is the generator of $\mathbb{G}_1$ (of prime order q) and P_2 is the generator of $\mathbb{G}_2$ (of prime order q) and $a, b, t_1, t_2, s \in_R \mathbb{Z}_q^*$.
Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The inputs of $\mathsf{VerPair}$ are the outputs of Rand scheme and the secret, private inputs $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.
The steps of $\mathsf{VerPair}$ are given as follows:

1. T queries U_1 in random order as follows:

   . $U_1(t_1P_1, B - bP_2 - sP_2) \longleftarrow D_{11} = e(t_1P_1, B - bP_2 - sP_2)$,
   . $U_1(A - aP_1 - sP_1, t_1P_2) \longleftarrow D_{12} = e(A - aP_1 - sP_1, t_1P_2)$.

2. Then similarly, T queries U_2 in random order as follows:

   . $U_2(t_2P_1, B - bP_2 - sP_2) \longleftarrow D_{21} = e(t_2P_1, B - bP_2 - sP_2)$,
   . $U_2(A - aP_1 - sP_1, t_2P_2) \longleftarrow D_{22} = e(A - aP_1 - sP_1, t_2P_2)$.

3. After receiving D_{11}, D_{12} from U_1 and D_{21}, D_{22} from U_2, T computes
   . $\beta_1 = D_{11} \cdot g^{t_1 s} = e(t_1P_1, B - bP_2)$,
   . $\beta_2 = D_{12} \cdot g^{t_1 s} = e(A - aP_1, t_1P_2)$,
   . $\beta_3 = D_{21} \cdot g^{t_2 s} = e(t_2P_1, B - bP_2)$,
   . $\beta_4 = D_{22} \cdot g^{t_2 s} = e(A - aP_1, t_2P_2)$.
4. Then, T queries U_1 in random order as follows:

   . $U_1(\beta_3, at_2^{-1}) \longleftarrow D_{13} = \beta_3^{at_2^{-1}} = e(aP_1, B - bP_2)$,
   . $U_1(\beta_4, bt_2^{-1}) \longleftarrow D_{14} = \beta_4^{bt_2^{-1}} = e(A - aP_1, bP_2)$,
   . $U_1(A - aP_1, B - bP_2) \longleftarrow D_{15} = e(A - aP_1, B - bP_2)$.

5. Similarly, T queries U_2 in random order as follows:

   . $U_2(\beta_1, at_1^{-1}) \longleftarrow D_{23} = \beta_1^{at_1^{-1}} = e(aP_1, B - bP_2)$,
   . $U_2(\beta_2, bt_1^{-1}) \longleftarrow D_{24} = \beta_2^{bt_1^{-1}} = e(A - aP_1, bP_2)$,
   . $U_2(A - aP_1, B - bP_2) \longleftarrow D_{25} = e(A - aP_1, B - bP_2)$.
6. T verifies
$$D_{13} \stackrel{?}{=} D_{23},\ D_{14} \stackrel{?}{=} D_{24},\ D_{15} \stackrel{?}{=} D_{25}. \tag{4}$$
7. If Equations (4) do not hold simultaneously, then T outputs $\perp$.
8. Else, T outputs
$$e(A, B) = D_{13} \cdot D_{14} \cdot D_{15} \cdot g^{ab}.$$

5 Efficiency & Security Analysis of Rand and VerPair

In this section, we analyze the security, verifiability and efficiency properties of the delegated part of Rand and VerPair. Moreover, we compare VerPair with the previous proposals with respect to its computational and communication complexities for both the delegator T and the services U_i, $i = 1, 2$, i.e. the overall communication overhead, number of rounds, memory requirements of the delegator T, and the computational complexity for T.

Theorem 1. *In the one-malicious version of a two-untrusted program model (OMTUP), the algorithms (T, U_1, U_2) are a fully verifiable $\mathcal{O}(\frac{1}{\log q})$-efficient delegated-secure implementations of the delegated part of* Rand *scheme, where the static inputs*

$$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$$

of Rand *maybe honest, secret; or $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ honest, unprotected from U_1; or $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ honest, unprotected from U_2.*

Proof. We prove completeness, security, full verifiability and efficiency of Rand as follows:
Completeness. Assume that the servers U_1 and U_2 run Rand honestly. Since we delegate the same computations to both U_1 and U_2, we clearly have

$$\gamma_i = \beta_i,\ \theta_{1j} = \theta_{2j}, \text{ for } 1 \leq i \leq 7, j = 1, 2.$$

Hence, the first verification step of Rand is complete. Now, the following equalities hold:

- $\gamma_1 \cdot g^t = g^{ab-t} g^t = g^{ab}$,
- $\gamma_2 \cdot g^t = g^{t_1 s - t} g^t = g^{t_1 s}$,
- $\gamma_3 \cdot g^t = g^{t_2 s - t} g^t = g^{t_2 s}$,
- $\gamma_4 + \alpha_1 P_1 = (t_1 - \alpha_1) P_1 + \alpha_1 P_1 = t_1 P_1$,
- $\gamma_5 + \alpha_1 P_2 = (t_1 - \alpha_1) P_2 + \alpha_1 P_2 = t_1 P_2$,
- $\gamma_6 + \alpha_2 P_1 = (t_2 - \alpha_2) P_1 + \alpha_2 P_1 = t_2 P_1$,
- $\gamma_7 + \alpha_2 P_2 = (t_2 - \alpha_2) P_2 + \alpha_2 P_2 = t_2 P_2$,
- $c_1 \theta_{11} = c_1 (c_1^{-1} t_1^{-1}) = t_1^{-1}$,
- $c_2 \theta_{12} = c_2 (c_2^{-1} t_2^{-1}) = t_2^{-1}$.

Since the values $a, b, t \in \mathbb{Z}_q^*$ with $(aP_1, bP_2, sP_1, sP_1, g^t)$ are computed in the first part of Rand using a precomputation subroutine, and the outputs at_i^{-1}, bt_i^{-1}, $i = 1, 2$, are solely computed by T itself, we are also done with completeness of Rand. □
Security & **Full verifiability.** The proof is similar to [25]. We assume now that $\mathcal{A} = (E, U_1', U_2')$ is a probabilistic polynomial-time (PPT) adversary interacting with a PPT-based algorithm T in the delegated-security model of Section (2). Our fist claim is

$$EVIEW_{real} \sim EVIEW_{ideal},$$

e.g. Pair One in the security model that the *external* adversary environment E learns nothing useful. Note that *all static inputs*

$$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2),\ (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$$

are assumed to be honest, secret for an environmental adversary since neither E and U_1' nor E and U_2' cannot communicate directly to develop a joint strategy after interacting with T. Then, ignoring the ith round, a simulator S_1 first chooses elements x_i, $1 \le i \le 9$ randomly, and makes 18 random queries to U_1' and U_2'

. $U_1'(x_i, P_1) \longleftarrow \gamma_i$, $U_2'(x_i, P_1) \longleftarrow \beta_i$ for $i = 4, 6$,
. $U_1'(x_i, P_2) \longleftarrow \gamma_i$, $U_2'(x_i, P_2) \longleftarrow \beta_i$ for $i = 5, 7$,
. $U_1'(x_i, g) \longleftarrow \gamma_i$, $U_2'(x_i, g) \longleftarrow \beta_i$ for $i = 1, 2, 3$,
. $U_1'(x_8, \cdot^{-1}) \longleftarrow \theta_{11}$, $U_2'(x_8, \cdot^{-1}) \longleftarrow \theta_{21}$,
. $U_1'(x_9, \cdot^{-1}) \longleftarrow \theta_{12}$, $U_2'(x_9, \cdot^{-1}) \longleftarrow \theta_{22}$.

Note that we do not consider the outputs at_i^{-1}, bt_i^{-1}, $i = 1, 2$, to prove the result since it is solely computed by T without any interaction with E, U_1' or U_2'. Then, S_1 behaves

. if the outputs of U_1' and U_2' are not equal for a randomly selected i, $1 \le i \le 9$, then the values $Y_p^i ='' error''$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, '' error'', \emptyset)$ in the ideal process) are produced by S_1,
. if no $''error''$ is detected , then then the values $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, and $replace^i = 0$ (corresponding to the output $(estate^i, Y_p^i, Y_u^i)$ in the ideal process) are produced by S_1,
. otherwise, S_1 selects a random element r and outputs $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, r, Y_u^i)$ in the ideal process).

In either cases, S_1 saves the appropriate states.
The distributions of inputs in the real and ideal experiments are computationally indistinguishable. In the ideal experiment, the inputs are chosen uniformly at random. In the real experiment, all inputs of the delegated part of Rand are independently randomized by the choice of uniformly distributed random elements $t_1, t_2, c_1, c_2 \in_R \mathbb{Z}_q^*$. Note that, by each invocation of Rand, new random values are generated which are different from other invocations. Then, there are two cases

. if U_1' and U_2' behave honestly both in the real and the ideal experiments in the round i, then we have $EVIEW_{real}^i \sim EVIEW_{ideal}^i$ since in the real execution $T^{U_1', U_2'}$ perfectly runs Rand and in the ideal execution S_1 does not change the output,
. If one of U_1' or U_2' behaves dishonestly in the round i, than this can be detected by both T and S_1 with probability 1. The reason is that one server is *always* honest under OMTUP, and only the equality of the *same* delegated inputs are compared coming from an honest and a potentially dishonest

server. Then, any misbehavior could always be detected, and this will result an output of an $''error''$. This argument also shows that Rand is fully verifiable.

Note that it is impossible that Rand could be corrupted implying that S_1 never executes the case of selecting a random element r and returning $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ in the ideal experiment since Rand is fully verifiable, thence it is impossible for both U_1' and U_2' to deviate from their functionalities. Thus, we have

$$EVIEW_{real}^i \sim EVIEW_{ideal}^i$$

even in the case of a dishonest server U_1' or U_2'. By the hybrid argument, we conclude that

$$EVIEW_{real} \sim EVIEW_{ideal}.$$

Secondly, we claim that $UVIEW_{real} \sim UVIEW_{ideal}$, i.e. Pair Two of the delegated-security model that the untrusted server U_i, $i = 1$ or $i = 2$, learns nothing useful. By ignoring the ith round, a simulator S_2 produces random queries for both U_1' and U_2', behaving exactly like S_1, and saves its states. Furthermore, it saves the states of (U_1', U_2'). Due to OMTUP assumption, an external environment adversary cannot tell U_1' or U_2' that the simulator S_2 produces bogus outputs since neither E and U_1' nor E and U_2' can communicate directly to develop a joint strategy after interacting with T. Similarly, U_1' and U_2' cannot communicate directly to collaborate to test the random inputs. Now, we have the following possibilities:

. $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ are honest, secret for both U_1' and U_2',
. $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is honest, unprotected from U_1', and/or $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is honest, unprotected from U_2'.

If $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ are honest, secret for both U_1' and U_2', then U_1' and U_2' cannot distinguish the real queries from the random ones due to the exactly same reason when interacting with S_1, whence $UVIEW_{real}^i \sim UVIEW_{ideal}^i$. Hence, by a hybrid argument

$$UVIEW_{real} \sim UVIEW_{ideal}.$$

If $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is honest, unprotected from U_1', then $t_1, t_1 P_1, t_1 P_2$ are unprotected from U_1' but honest, secret for U_2'. If further $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is honest, unprotected from U_2', then $t_1, t_1 P_1, t_1 P_2$ are unprotected from U_2' but honest, secret for U_1'. Then, the simulation S_2 is trivial for unprotected values, i.e. S_2 behaves the same way as in the real execution. In this case, the rest of the proof follows exactly as above for honest, secret static inputs, whence

$$UVIEW_{real} \sim UVIEW_{ideal}. \square$$

Efficiency. Since Rand needs

. two modular multiplications (MM's) to prepare $c_1 t_1$ and $c_2 t_2$,

. four elliptic curve point additions (PA's) to compute

$$t_1P_1, t_1P_2, t_2P_1, t_2P_2,$$

. three MM's to compute $g^{ab}, g^{t_1 s}, g^{t_2 s}$,
. two MM's to compute t_1^{-1} and t_2^{-1}, and
. four MM's to compute at_1^{-1}, bt_1^{-1}, at_2^{-1}, and bt_2^{-1}.

Moreover, computation of modular exponents and elliptic curve scalar multiplications take $\mathcal{O}(\log q)$ steps (e.g. by square-and-multiply and double-and-add methods or their variants). Therefore, (T, U_1, U_2) is an $\mathcal{O}(1/\log q)$-efficient implementation of Rand.□

Theorem 2. *In the one-malicious version of a two-untrusted program model, the algorithms $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$ are a fully verifiable $\mathcal{O}(\frac{1}{\log q})$-efficient delegated-secure implementations of* VerPair*, where the inputs (A, B) may be honest, secret; or honest, protected; or adversarial protected.*

Proof. We prove completeness, security, full verifiability and efficiency of VerPair as follows:
Completeness. Assume that the servers U_1 and U_2 run VerPair honestly. It is not difficult to see that

$$\begin{aligned} D_{13} = \beta_3^{at_2^{-1}} &= e(aP_1, B - bP_2) \\ = \beta_1^{at_1^{-1}} &= D_{23}, \end{aligned}$$

and

$$\begin{aligned} D_{14} = \beta_4^{bt_2^{-1}} &= e(A - aP_1, bP_2) \\ = \beta_2^{bt_1^{-1}} &= D_{24}. \end{aligned}$$

Similarly,

$$D_{15} = e(A - aP_1, B - bP_2) = D_{25}.$$

Hence, the verification step of VerPair is complete. Now,

$$\begin{aligned} D_{13} \cdot D_{14} \cdot D_{15} \cdot g^{ab} &= \beta_3^{at_2^{-1}} \cdot \beta_4^{bt_2^{-1}} \cdot e(A - aP_1, B - bP_2) \cdot g^{ab} \\ &= e(aP_1, B - bP_2) \cdot e(A - aP_1, bP_2) \cdot e(A - aP_1, B - bP_2) \cdot e(aP_1, bP_2) \\ &= e(aP_1, B - bP_2) \cdot e(aP_1, bP_2) \cdot e(A - aP_1, bP_2) \cdot e(A - aP_1, B - bP_2) \\ &= e(aP_1, B) \cdot e(A - aP_1, B) \\ &= e(A, B). \square \end{aligned}$$

Full Verifiability. Assume without loss of generality that U_1 is a malicious server capable of cheating the delegator T with non-negligible probability. Let

$h = g^{\omega} \in \mathbb{G}_3$ be given. Now, we consider the algorithms T^{U_1,U_2} implementing $\mathsf{VerPair}$ for which $aP_1 = \omega P_1$ is chosen. The delegator T verifies at the end of the scheme

$$D_{13} = \beta_3^{\omega t_2^{-1}} = e(\omega P_1, B - bP_2) = \beta_1^{\omega t_1^{-1}} = D_{23}, \tag{5}$$

$$D_{14} = \beta_4^{b t_2^{-1}} = e(A - \omega P_1, bP_2) = \beta_2^{b t_1^{-1}} = D_{24}, \tag{6}$$

and

$$D_{15} = e(A - \omega P_1, B - bP_2) = D_{25}. \tag{7}$$

Since both D_{15} and D_{25} are used to delegate $e(A - \omega P_1, B - bP_2)$ and U_2 is honest, U_1 can only cheat T during the verification formulas (5) and (6). Let $x, y \in \mathbb{Z}_q^*$ with $A - \omega P_1 - sP_1 = xP_1,\ B - bP_2 - sP_2 = yP_2$ be given. Instead of sending $D_{11} = g^{xt_1}$ (resp. $D_{12} = g^{yt_1}$), U_1 chooses bogus values $\theta_1, \theta_2 \in \mathbb{Z}_q^*$ and send $\Gamma_{11} = g^{\theta_1}$ and $\Gamma_{12} = g^{\theta_2}$ to the delegator. Then, T computes

- $\varphi_1 = \Gamma_{11} \cdot g^{t_1 s} = g^{\theta_1 + t_1 s}$,
- $\varphi_2 = \Gamma_{12} \cdot g^{t_1 s} = g^{\theta_2 + t_1 s}$,
- $\beta_3 = D_{21} \cdot g^{t_2 s} = e(t_2 P_1, B - bP_2)$,
- $\beta_4 = D_{22} \cdot g^{t_2 s} = e(A - \omega P_1, t_2 P_2)$.

instead of β_1 and β_2. Note that β_3 and β_4 are correct values since U_2 is honest. Then, U_2 computes in the second round

- $\phi_{23} = (g^{\theta_1 + t_1 s})^{\omega t_1^{-1}} = g^{\theta_1 \omega t_1^{-1} + s\omega}$, and
- $\phi_{24} = (g^{\theta_2 + t_1 s})^{b t_1^{-1}} = g^{\theta_2 b t_1^{-1} + sb}$

instead of D_{23} and D_{24} following T^{U_1,U_2} honestly. Hence, in order to pass the verification steps (5) and (6), U_1 must know exactly the values of ϕ_{23} and ϕ_{24}. Note that if $B - bP_2 = y_2 P_2$ and $A - \omega P_1 = x_1 P_1$, then U_1 knows further $\beta_3 = g^{x_1 t_2}$ and $\beta_4 = g^{y_1 t_2}$, and the values ωt_2^{-1} and $b t_2^{-1}$ from the scheme specification. Furthermore, sP_1 (resp. sP_2) is also known by U_1 in the second round; since by subtracting $A - \omega P_1$ (resp. $B - bP_2$) from the first component of D_{12} (resp. from the second component of D_{11}), U_1 can easily obtain sP_1 (resp. sP_2). Then, in order to compute the values

- $\phi_{23} = g^{\theta_1 \omega t_1^{-1} + sa} = g^{\omega(\theta_1 t_1^{-1} + s)} = (g^{\theta_1 t_1^{-1} + s})^{\omega}$, and
- $\phi_{24} = g^{\theta_2 b t_1^{-1} + sb} = g^{b(\theta_2 t_1^{-1} + s)} = (g^{\theta_2 t_1^{-1} + s})^{b}$.

U_1 needs to know the exponents ω and b from h_1^{ω} and h_2^{b} with non-negligible probability due the fact that $h_1 = g^{\theta_1 t_1^{-1} + s},\ h_2 = g^{\theta_2 t_1^{-1} + s} \in \mathbb{G}_3$ are known to U_1. Notice that, ω, b and ωb cannot also be computed from ωt_2^{-1}, $b t_2^{-1}$ and $\omega b - t$ by the proof of secrecy of the delegated part of Rand, i.e. t_2 is only available to U_2. Therefore, if U_1 can compute ω from $h^{\theta_1 t_1^{-1} + s} = h_1^{\omega}$, thence solves the discrete logarithm problem (DLP) to the base g, with non-negligible probability. Since, U_1 is a polynomially bounded adversary, this gives a contradiction. $\square$

Security. The proof is similar to the proof of Theorem (1). We assume now that $\mathcal{A} = (E, U_1', U_2')$ is a probabilistic polynomial-time (PPT) adversary interacting with a PPT-based algorithm T in the delegated-security model of Section (2). Our fist claim is

$$EVIEW_{real} \sim EVIEW_{ideal},$$

e.g. Pair One in the security model that the *external* adversary environment E learns nothing useful. If inputs (A, B) are either honest, protected or adversarial protected, then a simulator S_1 behaves exactly as in the real execution, i.e. it never requires to access (A, B) since both of them are not secret to the adversary E. We now assume that (A, B) are honest, secret inputs. Then, ignoring the ith round, S_1 first chooses elements $\ell_i \in \mathbb{Z}_q^*$, $1 \leq i \leq 8$ randomly, computes $(\ell_1 P_1, \ell_2 P_2, \ell_3 P_1, \ell_4 P_2)$ for U_1' and $(\ell_4 P_1, \ell_6 P_2, \ell_7 P_1, \ell_8 P_2)$, and makes 2 random queries to U_1'

. $U_1'(\ell_1 P_1, \ell_2 P_2) \longleftarrow D_{11}$,
. $U_1'(\ell_3 P_1, \ell_4 P_2) \longleftarrow D_{12}$,

and 2 random queries to U_2'

. $U_2'(\ell_5 P_1, \ell_6 P_2) \longleftarrow D_{21}$,
. $U_2'(\ell_7 P_1, \ell_8 P_2) \longleftarrow D_{22}$.

After receiving the outputs of U_1' and U_2', the simulator S_1 chooses random elements (g_1, γ_1), $(g_2, \gamma_2) \in \mathbb{G}_3 \times \mathbb{Z}_q^*$ and random elements $\ell_9, \ell_{10} \in \mathbb{Z}_q^*$, compute $(\ell_9 P_1, \ell_{10} P_2)$, and queries randomly to U_1'

. $U_1'(g_1, \gamma_1) \longleftarrow D_{13}$,
. $U_1'(g_2, \gamma_2) \longleftarrow D_{14}$,
. $U_1'(\ell_9 P_1, \ell_{10} P_2) \longleftarrow D_{15}$,

similarly, S_1 chooses random elements $(g_3, \gamma_3), (g_4, \gamma_4) \in \mathbb{G}_3 \times \mathbb{Z}_q^*$ and random elements $\ell_{11}, \ell_{12} \in \mathbb{Z}_q^*$, compute $(\ell_{11} P_1, \ell_{12} P_2)$, and queries randomly to U_2'

. $U_2'(g_3, \gamma_3) \longleftarrow D_{23}$,
. $U_1'(g_4, \gamma_4) \longleftarrow D_{24}$,
. $U_1'(\ell_{11} P_1, \ell_{12} P_2) \longleftarrow D_{25}$.

Then, S_1 behaves

. if the outputs D_{1i} of U_1' and D_{2i} U_2' are not equal for a randomly selected i, $3 \leq i \leq 5$, then the values $Y_p^i ='' error''$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, '' error'', \emptyset)$ in the ideal process) are produced by S_1,
. if no $''error''$ is detected , then then the values $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, and $replace^i = 0$ (corresponding to the output $(estate^i, Y_p^i, Y_u^i)$ in the ideal process) are produced by S_1,
. otherwise, S_1 selects a random element r and outputs $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, r, Y_u^i)$ in the ideal process).

In either cases, S_1 saves the appropriate states.
The distributions of inputs in the real and ideal experiments are computationally indistinguishable. In the ideal experiment, the inputs are chosen uniformly at random. In the real experiment, all inputs of VerPair are independently randomized by the choice of uniformly distributed random elements. Note that, by each invocation of VerPair, new random values are generated by Rand which are different from other invocations, and computationally indistinguishable from random elements. Since VerPair is a fully verifiable secure-delegated scheme, we only have two cases

- if U_1' and U_2' behave honestly both in the real and the ideal experiments in the round i, then we have $EVIEW_{real}^i \sim EVIEW_{ideal}^i$ since in the real execution $T^{U_1',U_2'}$ perfectly runs VerPair, and in the ideal execution S_1 does not change the output,
- If one of U_1' or U_2' behaves dishonestly in the round i, than this can be detected by both T and S_1 with probability 1, see full verifiability.

In particular, it is impossible that VerPair could be corrupted implying that S_1 never executes the case of selecting a random element r and returning $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ in the ideal experiment, see Remark (4.2) for further details. This implies that it is impossible for both U_1' and U_2' to deviate from their functionalities. Thus, we have

$$EVIEW_{real}^i \sim EVIEW_{ideal}^i$$

even in the case that one of U_i', $i = 1, 2$, misbehaves. By the hybrid argument, we conclude that

$$EVIEW_{real} \sim EVIEW_{ideal}.$$

It is clear that this argument only works if *only* one server misbehaves (under OMTUP model), i.e. if both U_1 and U_2 are malicious simultaneously, then the misbehavior in this case is not independent of the inputs (A, B) whereas the misbehavior of only one of U_i, i=1,2, is independent of the inputs (A, B).
Secondly, we claim that

$$UVIEW_{real} \sim UVIEW_{ideal},$$

i.e. Pair Two of the delegated-security model that the untrusted server U_i, $i = 1$ or $i = 2$, learns nothing useful. For a round i, a simulator S_2 behaves exactly like S_1 to produce random queries by ignoring the ith round for both U_1' and U_2', and saves its states. Furthermore, it saves the states of (U_1', U_2'). Due to OMTUP assumption, an external environment adversary can tell neither to U_1' nor to U_2' that the simulator S_2 produces bogus outputs since the output in the real experiment is not corrupted, and neither E and U_1' nor E and U_2' can communicate directly in order to develop a joint strategy after interacting with T. Hence, honest, secret; honest, protected; or adversarial protected inputs are all private for both U_1' and U_2', although E could easily distinguish between these real and ideal experiments. The reason, exactly as in the case of interacting with

S_1, is that in the ith round of the real experiment, the values given to either U_1' or U_2' are completely re-randomized by Rand, and S_2 generates random, independent queries for both U_1' and U_2' in the ideal experiment. Thus, we have

$$UVIEW^i_{real} \sim UVIEW^i_{ideal}$$

for each round i. It follows then by a hybrid argument

$$UVIEW_{real} \sim UVIEW_{ideal}. \square$$

Efficiency. Since VerPair needs

. four elliptic curve point additions (PA's) to compute

$$A - aP_1,\ B - bP_2,\ A - aP_1 - sP_1,\ B - bP_2 - sP_2,$$

. four MM's to compute β_i for $1 \leq i \leq 4$,
. three MM's to compute $e(A, B)$.

Furthermore, computation of modular exponents and elliptic curve scalar multiplications take $\mathcal{O}(\log q)$ steps (e.g. by square-and-multiply and double-and-add methods or their variants). Hence, (T, U_1, U_2) is an $\mathcal{O}(1/\log q)$-efficient implementation of VerPair.$\square$

Remark 4. 1. Note that revealing information about α_1 to U_1' (resp. α_2 to U_2') in Theorem (1) corresponds for instance to the case that U_1' (resp. U_2') is a computationally unbounded adversary that can use an effective discrete logarithm solver (DLP-solver) to retrieve t_1 from t_1P_1 or t_1P_2 (resp. t_2 from t_2P_1 or t_2P_2), e.g. by means of an efficient quantum algorithm, and subsequently using t_1 (resp. t_2), to obtain the values α_1 (resp. α_2) of Rand. Hence, Theorem (1) implies that Rand offers security even in the presence of a *computationally unrestricted adversary* corrupting one of U_1 or U_2.

2. On the other hand, security proof of Theorem (2) relies on the proof of its full verifiability. Since, full verifiability is only guaranteed in the presence of polynomially bounded adversaries, VerPair offers security against polynomially bounded adversarial server U_i, $i = 1, 2$, whereas a computationally unbounded adversary could cheat the delegator T by means of a DLP-solver such that the third case in the proof of Theorem (2) could happen that VerPair could be corrupted implying that a simulator S_1 executes in the ith round the case of selecting a random element r and returning $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ in the ideal experiment. Hence, the proof of Theorem (2) implies that VerPair offers security in the presence of a *polynomially bounded adversary* corrupting one of U_1 or U_2.

6 Comparison

Delegation Scheme	Secrecy (output)	Verifiability (real)	Client's workload	Servers' workload	#Rounds
[34]	no	0	4PA, 4MM, 2SM, 3MI, 1ME	6P	1
[23]	no	1	6PA, 19MM	10P	1
[27]	yes	0	4PA, 6MM	8P	1
[35]	yes	1/2	8PA, 6MM	4P	1
[39]	yes	5/6	8PA, 14MM	6P, 4ME	2
[41] $(s = 4)$	yes	0.967	8PA, 19MM	10P	1
[20]	yes	1	4PA, 6MM, 6SM, 10ME	4P	2
[28]	yes	1	2PA, 3MM, 4SM, 7ME	4P	2
[15]	yes	1	2PA, 1MM, 1TM, 4SM, 2ME	4P	1
VerPair	yes	1	4PA, 7MM	6P, 4ME	2

Table 1. Comparison of the Delegator's Computational Costs and Communication Complexities.

In this section, we compare VerPair with the previous results claiming full verifiability. Let **SM** represent scalar multiplication in $\mathbb{G}_1, \mathbb{G}_2$, **ME** modular exponentiation in $\mathbb{G}_3$, **MI** modular inverse in $\mathbb{G}_3$, **PA** point addition in $\mathbb{G}_1, \mathbb{G}_2$, **TM** test membership in $\mathbb{G}_T$, and **P** a pairing computation.
For the efficiency comparison, we mainly focus on fully verifiable pairing delegation schemes, and do not focus on the schemes that does not hold their premises by either leaking $e(A, B)$ or not being fully verifiable. This process leaves us three schemes beside VerPair, [20], [28], and [15]. These are unfortunately the most inefficient algorithms since they contain modular exponentiations and membership testing operations; VerPair achieves much better performance results as expected. Furthermore, VerPair has also much better performance even when compared with the delegation schemes which are not fully verifiable. For example, if we consider the schemes with highest verifiability guarantees [39], [41], we can also see from Table (1) that VerPair is much more efficient. In Table (1), we write security and verifiability issues and inefficient parts of the schemes in red to emphasize the problems of each delegation scheme. Hence, VerPair is the first efficient fully verifiable pairing delegation scheme requiring neither costly **SM** nor **ME** online operations on the delegator's side. Additionally, partial delegation of Rand scheme eliminates also the requirement of the offline computation of costly **SM**, **ME**, and **MI** operations on the delegator's side. Therefore, partially delegated Rand and VerPair enables for the first time a complete general delegation mechanism without any offline and online computation of **SM**, **ME** and **MI** operations.

	160-bit MNT	256-bit BN	512-bit KSS	640-bit BLS
Pairing	0.0032647	0.0049727	0.0440744	0.0754905
[15]	0.0042152	0.00656252	0.0447128	0.161772
[39]	0.0001405	0.0002814	0.0012098	0.0021017
VerPair	0.0000837	0.0001726	0.0007897	0.0013833

Table 2. Comparison of VerPair with pairing calculation for different choices of curves

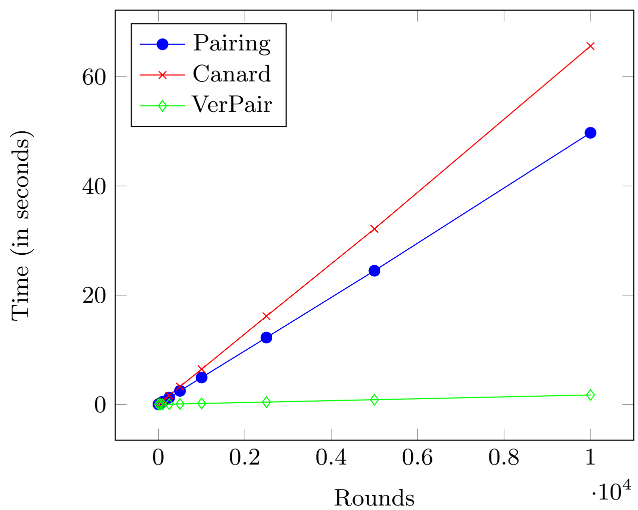

Fig. 1. Timing for different number of rounds on a 256-bit BN curve

Numerical Results. From fully verifiable schemes apart from VerPair, we choose the one that requires least computational overhead [15]. Again from [39] [41], we choose the one that requires less information [40]. These schemes are implemented together with VerPair using the MIRACL library [38] on a 3.40 GHz Intel Core i7-3770 processor, compiled with GCC, with standard /O2 compiler optimization. One can find the average results for 10000 trials on Table (2). For [15], the values that can be precomputed are assumed to be computed offline, and also membership test operation is not included in timed section.
Using a 256-bit BN curve, computation times of a pairing, the scheme in [15], and VerPair are compared. The results can be seen in Figure (1).

Communication Cost. In order to be able to propose a scheme with full verifiability, we required two rounds in VerPair. We left it as an open question either to propose a non-interactive fully verifiable secure delegation of pairings without any online computation of **SM**, **ME**, and **MI** operations, or to prove its impossibility. We however conjecture that it is impossible to have a non-interactive fully verifiable delegation scheme as long as the description of the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ are known by the servers. Hence, the tradeoff of achieving the full verifiability is either to perform costly online operations like **SM**, **ME**, and **MI**, or to add another round to the delegation. In practice however, Meulenaer *et al.* in their seminal work [37] give a model regarding the total energy consumption of cryptographic operations in wireless sensor networks by measuring the energy consumptions in MICAz and TelosB sensor nodes. In particular, this analysis shows that the computation of a single **SM** (or a **ME** operation) requires considerably more energy than a single round of communication (considering the total communication overhead including Transmit, Listen, Receive, Compute, Sleep). Hence, the risk of causing single point of failure is considerably higher in the schemes in [20] and [15] than VerPair since they require several **SM** and **ME** operations while VerPair requires only an additional round in order to achieve the full verifiability.

7 Conclusion

Main focus of this study is to deal with the problem of fully verifiable secure delegation of general pairing computation. By presenting the concrete attack scenarios, we show that several pairing delegation schemes do not satisfy the claimed verifiability and/or security guarantees. Then, we propose an efficient and fully verifiable secure delegation scheme VerPair under one-malicious version of a two-untrusted-program model (OMTUP). The proposed scheme involves a precomputation step Rand and pairing delegation scheme VerPair. We also point out that it is also possible to reduce the overall scheme computation overhead by partially delegating Rand. Later, we give a detailed security analysis of VerPair using a variant of the Hohenberger and Lysyanskaya's simulation-based security model. Using the MIRACL library[38], we implement VerPair on different paring-friendly elliptic curves, present implementation results, and compare these results with the previous schemes. Even if the network and communication costs, and the cost of actual computation of the costly precomputation step is not included in performance tests, VerPair scheme runs considerably more efficient than all the previous schemes. As possible future work, it is highly desirable either **(a)** to propose fully verifiable secure delegation schemes for pairing computation under the TUP assumption, or even more interesting under the OUP assumption, which do not require any online computation of costly modular exponentiations and elliptic curve scalar multiplications **(b)** to show impossibility results. As another future work for the practical deployment of the pairing delegation, studying intensively the trade-offs between computational efficiency, memory requirement of the delegator, concrete cryptographic protocols to be delegated, and secure implementation aspects is highly required.

References

1. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Advances in Cryptology – EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 – May 4, 2017, Proceedings, Part I. pp. 601–626. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978-3-319-56620-7_21
2. Öznur Arabacı, Kiraz, M.S., İsa Sertkaya, Uzunkol, O.: More efficient secure outsourcing methods for bilinear maps. Cryptology ePrint Archive, Report 2015/960 (2015), https://eprint.iacr.org/2015/960
3. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334 (2017), http://eprint.iacr.org/2017/334
4. Barreto, P.S.L.M., Galbraith, S.D., hÉigeartaigh, C.Ó., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (Mar 2007), https://doi.org/10.1007/s10623-006-9033-6
5. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II. pp. 276–294. Springer Berlin Heidelberg, Berlin, Heidelberg (2014), https://doi.org/10.1007/978-3-662-44381-1_16
6. Beuchat, J.L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography - Pairing 2010: 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings. pp. 21–39. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-17455-1_2
7. Blake, I., Seroussi, G., Smart, N.: Advances in Elliptic Curve Cryptography. Cambridge University Press, New York, NY, USA (2005)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings. pp. 41–55. Springer Berlin Heidelberg, Berlin, Heidelberg (2004), https://doi.org/10.1007/978-3-540-28628-8_3
9. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: 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
10. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Advances in Cryptology — EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings. pp. 416–432. Springer Berlin Heidelberg, Berlin, Heidelberg (2003), https://doi.org/10.1007/3-540-39200-9_26
11. 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
12. Boyko, V., Peinado, M., Venkatesan, R.: Speeding up discrete log and factoring based schemes via precomputations. In: Advances in Cryptology — EURO-

CRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31 – June 4, 1998 Proceedings. pp. 221–235. Springer Berlin Heidelberg, Berlin, Heidelberg (1998), https://doi.org/10.1007/BFb0054129

13. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation. In: Advances in Cryptology — EUROCRYPT' 92: Workshop on the Theory and Application of Cryptographic Techniques Balatonfüred, Hungary, May 24–28, 1992 Proceedings. pp. 200–207. Springer Berlin Heidelberg, Berlin, Heidelberg (1993), https://doi.org/10.1007/3-540-47555-9_18
14. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Advances in Cryptology, CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings. pp. 61–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2002), https://doi.org/10.1007/3-540-45708-9_5
15. 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: 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings. pp. 549–565. Springer International Publishing, Cham (2014), https://doi.org/10.1007/978-3-319-07536-5_32
16. Cavallo, B., Di Crescenzo, G., Kahrobaei, D., Shpilrain, V.: Efficient and secure delegation of group exponentiation to a single server. In: Radio Frequency Identification: 11th International Workshop, RFIDsec 2015, New York, NY, USA, June 23-24, 2015, Revised Selected Papers. pp. 156–173. Springer International Publishing, Cham (2015), https://doi.org/10.1007/978-3-319-24837-0_10
17. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) Computer Security – ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings. pp. 541–556. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-33167-1_31
18. Chen, X., Susilo, W., Li, J., Wong, D.S., Ma, J., Tang, S., Tang, Q.: Efficient algorithms for secure outsourcing of bilinear pairings. Theoretical Computer Science 562(Supplement C), 112 – 121 (2015), http://www.sciencedirect.com/science/article/pii/S0304397514007282
19. Chevalier, C., Laguillaumie, F., Vergnaud, D.: Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In: Computer Security – ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I. pp. 261–278. Springer International Publishing, Cham (2016), https://doi.org/10.1007/978-3-319-45744-4_13
20. 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), http://eprint.iacr.org/2005/150
21. Chow, S.S.M., Yiu, S.M., Hui, L.C.K.: Efficient identity based ring signature. In: Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings. pp. 499–512. Springer Berlin Heidelberg, Berlin, Heidelberg (2005), https://doi.org/10.1007/11496137_34

22. van Dijk, M., Juels, A.: On the impossibility of cryptography alone for privacy-preserving cloud computing. Cryptology ePrint Archive, Report 2010/305 (2010), https://eprint.iacr.org/2010/305
23. Dong, M., Ren, Y., Zhang, X.: Fully verifiable algorithm for secure outsourcing of bilinear pairing in cloud computing. KSII Transactions on Internet and Information Systems pp. 3648–3663 (2017), http://www.itiis.org/digital-library/manuscript/file/1753/TIIS+Vol+11,+No+7-19.pdf
24. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. Information Theory, IEEE Transactions on 52(10), 4595–4602 (Oct 2006), http://dx.doi.org/10.1109/TIT.2006.881709
25. 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
26. Joux, A.: A one round protocol for tripartite diffie-hellman. Journal of Cryptology 17(4), 263–276 (2004), http://dx.doi.org/10.1007/s00145-004-0312-y
27. Kalkar, Ö., Kiraz, M.S., Sertkaya, İ., Uzunkol, O.: A more efficient 1-checkable secure outsourcing algorithm for bilinear maps. to appear in Proceedings of The 11th WISTP International Conference on Information Security Theory and Practice (WISTP'2017) (2018)
28. Kang, B.G., Lee, M.S., Park, J.H.: Efficient delegation of pairing computation. Cryptology ePrint Archive, Report 2005/259 (2005), https://eprint.iacr.org/2005/259
29. Kiraz, M.S., Uzunkol, O.: Efficient and verifiable algorithms for secure outsourcing of cryptographic computations. International Journal of Information Security 15(5), 519–537 (Oct 2016), https://doi.org/10.1007/s10207-015-0308-7
30. Kiraz, M.S., Uzunkol, O.: Still wrong use of pairings in cryptography. Cryptology ePrint Archive, Report 2016/223 (2016), https://eprint.iacr.org/2016/223
31. Kiyomura, Y., Inoue, A., Kawahara, Y., Yasuda, M., Takagi, T., Kobayashi, T.: Secure and efficient pairing at 256-bit security level. In: Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings. pp. 59–79. Springer International Publishing, Applied Cryptography and Network Security: 15th International Conference, ACNS 2017 (2017), https://doi.org/10.1007/978-3-319-61204-1_4
32. Koblitz, N., Menezes, A.: Pairing-Based Cryptography at High Security Levels. In: 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
33. Libert, B., Quisquater, J.J.: New identity based signcryption schemes from pairings. Cryptology ePrint Archive, Report 2003/023 (2003), https://eprint.iacr.org/2003/023
34. Luo, X., Yang, X., Niu, X.: An efficient and secure outsourcing algorithm for bilinear pairing computation. In: Advances in Internetworking, Data & Web Technologies: The 5th International Conference on Emerging Internetworking, Data & Web Technologies (EIDWT-2017). pp. 328–339. Springer International Publishing, Cham (2018), https://doi.org/10.1007/978-3-319-59463-7_33
35. Luo, Y., Fu, S., Huang, K., Wang, D., Xu, M.: Securely outsourcing of bilinear pairings with untrusted servers for cloud storage. In: 2016 IEEE Trustcom/BigDataSE/ISPA. pp. 623–629 (Aug 2016), https://doi.org/10.1109/TrustCom.2016.0118

36. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (2011)
37. de Meulenaer, G., Gosset, F., Standaert, F.X., Pereira, O.: On the energy cost of communication and cryptography in wireless sensor networks. In: 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. pp. 580–585 (Oct 2008), https://doi.org/10.1109/WiMob.2008.16
38. MIRACL: Multiprecision integer and rational arithmetic C/C++ library. CertiVox UK Ltd., (accessed 2017-09-07) (2016), https://github.com/miracl/MIRACL
39. Ren, Y., Ding, N., Wang, T., Lu, H., Gu, D.: New algorithms for verifiable outsourcing of bilinear pairings. Science China Information Sciences 59(9), 99103 (Aug 2016), https://doi.org/10.1007/s11432-016-5550-8
40. Ren, Y., Ding, N., Wang, T., Lu, H., Gu, D.: New algorithms for verifiable outsourcing of bilinear pairings. Science China Information Sciences 59(9), 99103 (2016), http://dx.doi.org/10.1007/s11432-016-5550-8
41. Ren, Y., Dong, M., Niu, Z., Du, X.: Non-interactive verifiable outsourcing algorithm for bilinear pairing with improved checkability. Security and Communication Networks pp. 1–9 (2017), http://downloads.hindawi.com/journals/scn/aip/4892814.pdf
42. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: 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
43. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Advances in Cryptology - ASIACRYPT 2008: 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings. pp. 90–107. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), https://doi.org/10.1007/978-3-540-89255-7_7
44. Tian, H., Zhang, F., Ren, K.: Secure bilinear pairing outsourcing made more efficient and flexible. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. pp. 417–426. ASIA CCS '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2714576.2714615
45. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Chow, S.S.M., Liu, Z., Tan, X.: Securely outsourcing exponentiations with single untrusted program for cloud storage. In: Computer Security - ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I. pp. 326–343. Springer International Publishing, Cham (2014), https://doi.org/10.1007/978-3-319-11203-9_19