On Single-Server Delegation Without Precomputation

Matluba Khodjaeva[1] and Giovanni Di Crescenzo[2][a]
[1]*CUNY John Jay College of Criminal Justice, NY, U.S.A.*
[2]*Peraton Labs, Basking Ridge, NJ, U.S.A.*

Keywords: Cryptography, Secure Delegation, Pairings, Elliptic Curves.

Abstract: Many public-key cryptosystems use pairings as important primitive operations. To expand the applicability of these solutions to computationally weaker devices, it has been advocated that a computationally weaker client delegates such primitive operations to a computationally stronger server. Important requirements for such delegation protocols include privacy of the client's pairing inputs and security of the client's output, in the sense of detecting, except for very small probability, any malicious server's attempt to convince the client of an incorrect pairing result. Except for less than a handful of results, all single-server delegation protocols in the literature are structured into an offline phase, where precomputation can be performed, and an online phase, where the client has resource constraints. Designing single-server delegation protocols without precomputation is naturally harder. In this paper, we show that the computation of a pairing with non-private inputs can be efficiently delegated to a single server, without need for precomputation. We also discuss the failure of a previously published attempt, and note the inefficiency of natural extensions of our protocol to more demanding input cases.

1 INTRODUCTION

The area of server-aided cryptography investigates the problem of computationally weaker clients delegating the most expensive cryptographic computations to computationally powerful servers. Interest in this area is recently increasing because of computation paradigms shifts, including cloud/fog/edge computing, large-scale computations over big data, and computations with resource-constrained devices, as in the Internet of Things.

This problem of delegating (aka outsourcing) computation in cryptography was first discussed in (Feigenbaum, 1985; Abadi et al., 1989; Matsumoto et al., 1988), first modeled in (Hohenberger and Lysyanskaya, 2005). Consistently with these and follow-up papers in the area (see, e.g., (Gennaro et al., 2010; Cavallo et al., 2015; Di Crescenzo et al., 2022)), we consider a model where a client, denoted as C, with an input x, delegates to a server, denoted as S, the computation of a function F on the client's input, and the main desired requirements are:

1. *result correctness*: if C and S honestly run the protocol, at the end of the protocol C returns $F(x)$;
2. *input privacy*: no new information about x should be revealed to S;
3. *result security*: S should not be able, except possibly with very small probability, to convince C to return a result different than $F(x)$ at the end of the protocol; and
4. *efficiency*: C's runtime, denoted as t_C, should be much smaller than the runtime, denoted as t_F, of computing $F(x)$ without delegation; moreover, it is of interest to minimize S's runtime t_S, the communication complexity cc, and the number of messages mc.

In almost all previous work in single-server delegation, protocols can be partitioned into (a) an offline phase, where input x is not yet known, but somewhat expensive pre-computation, performed by the client or a client deployer, is stored on the client's device, and (b) an online phase, where client's runtime is limited, and thus help by the server is needed to compute $F(x)$. This partition has proved very useful to obtain delegation protocols for many operations often used in cryptographic protocols (see, e.g., (Di Crescenzo et al., 2022) for a survey in the area). The problem of designing delegation protocols without offline phase precomputation, which we consider in this paper (see also Figure 1), is of interest for both theoretical and

[a] https://orcid.org/0000-0002-5138-1144

Khodjaeva, M. and Di Crescenzo, G.
On Single-Server Delegation Without Precomputation.
DOI: 10.5220/0012140100003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 540-547
ISBN: 978-989-758-666-8; ISSN: 2184-7711

practical reasons. From a theoretical point of view, a delegation protocol with precomputation typically involves precomputation in the offline phase of the same function that is being delegated on different inputs, while a delegation protocol without precomputation would realize a more demanding type of delegated computation, giving different insights on the problem that admit such solutions. From a practical point of view, at the end of the offline phase, typically secret values are stored on the client's resource-constrained device, and the rest of the protocol has to significantly rely on such values to remain secrets, thus increasing the storage and trust requirements on the system.

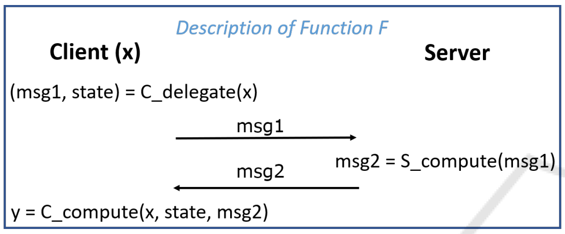
Description of Function F
Client (x)
Server
(msg1, state) = C_delegate(x)
msg1
msg2 = S_compute(msg1)
msg2
y = C_compute(x, state, msg2)

Figure 1: Delegated computation of $y = F(x)$.

Our Contributions. In this paper we show that a pairing (aka bilinear map) with publicly known inputs can be efficiently and securely delegated to a single, possibly malicious, server, without need for offline phase precomputations. We show that our protocol is efficient with respect to a number of metrics; most notably, the client performs strictly less computation than in non-delegated computation, for all 4 classes of practical elliptic curve families underlying the pairing definition. We also discuss the failure of a very recent attempt (Kalkar et al., 2022), and observe the inefficiency of natural extensions of our protocol to more demanding input scenario, where one of the two inputs or both have to remain private. Previously, delegation protocols without precomputation were given for group inverses (Cavallo et al., 2015), for a batch of group exponentiations with a single public base and multiple public exponents (Di Crescenzo et al., 2017), and for a batch of pairings with one public fixed input and one public variable input from (Tsang et al., 2007).

Related Work. Pairing-based cryptography, starting with (Joux, 2000; Sakai et al., 2000; Boneh and Franklin, 2001), has attracted much research in the past 2-3 decades (see, e.g., (Moody et al., 2015)). Single-server delegation protocols with precomputation for pairings have been proposed in (Girault and Lefranc, 2005; Guillevic and Vergnaud, 2014; Chevallier-Mames et al., 2010; Kang et al., 2005; Canard et al., 2014; Kachisa et al., 2008; Guillevic and Vergnaud, 2014; Canard et al., 2014; Di Crescenzo et al., 2020a; Di Crescenzo et al., 2020b). A survey on delegated computation of specific operations beyond cryptography can be found in (Shan et al., 2018). A survey on delegated computation of arbitrary functions, with clients more computationally powerful than considered here, can be found in (Ahmad et al., 2018).

2 DEFINITIONS

Let $\mathbb{G}_1$, $\mathbb{G}_2$ be additive cyclic groups of order l and $\mathbb{G}_T$ be a multiplicative cyclic group of the same order l, for some large prime l. A *pairing* is an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with description denoted as $desc(e)$, with the following properties:

1. *Bilinearity:* for all $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, and for any $r, s \in \mathbb{Z}_l$, it holds that $e(rA, sB) = e(A,B)^{rs}$
2. *Non-triviality:* if U is a generator for $\mathbb{G}_1$ and V is a generator for $\mathbb{G}_2$ then $e(U,V)$ is a generator for $\mathbb{G}_T$ (this property rules out the trivial scenario where e maps all of its inputs to 1).

The currently most practical *pairing realizations* use an ordinary elliptic curve E defined over a field $\mathcal{F}_p$, for some large prime p, as follows. Group $\mathbb{G}_1$ is the l-order additive subgroup of $E(\mathcal{F}_p)$; group $\mathbb{G}_2$ is a specific l-order additive subgroup of $E(\mathcal{F}_{p^k})$ contained in $E(\mathcal{F}_{p^k}) \setminus E(\mathcal{F}_p)$; and group $\mathbb{G}_T$ is the l-order multiplicative subgroup of $\mathcal{F}^*_{p^k}$. Here, k is the embedding degree; i.e., the smallest positive integer such that $l | (p^k - 1)$; $\mathcal{F}_{p^k}$ is the extension field of $\mathcal{F}_p$ of degree k; and $\mathcal{F}^*_{p^k}$ is the field composed of non-zero elements of $\mathcal{F}_{p^k}$. After the Weil pairing was considered in (Boneh and Franklin, 2001), more efficient constructions were proposed as variants of the Tate pairing, including the more recent ate pairing variants (see, e.g., (Vercauteren, 2010; Scott, 2013; Barreto et al., 2015) for details on the currently most practical constructions).

For our results, we further assume that $\mathbb{G}_T$ is a subgroup of a group $\mathcal{G}_T$, also contained in $\mathcal{F}^*_{p^k}$, with the following two properties:

1. testing membership in $\mathcal{G}_T$ is more efficient than testing membership in $\mathbb{G}_T$;
2. all elements of $\mathcal{G}_T$ have order $\geq l$.

This assumption is satisfied by a recently proposed security strengthening of the most practical pairing realizations. Motivated by reducing the chances of low-order attacks in cryptographic protocols, in (Barreto et al., 2015) the authors proposed the notion of *subgroup-secure* elliptic curves underlying a pairing, in turn extending the notion of $\mathbb{G}_T$-*strong* curves from (Scott, 2013). As a critical step in achieving these notions, both of these papers set $\mathcal{G}_T = G_{\Phi_k(p)}$, where $G_{\Phi_k(p)}$ is the cyclotomic subgroups of order $\Phi_k(p)$ in $\mathcal{F}^*_{p^k}$, and where $\Phi_k(p)$ denotes the k-th cyclotomic polynomial. Satisfaction of above property (2) is directly implied by the definitions of both $\mathbb{G}_T$-strong and subgroup-secure curves. Satisfaction of above property (1) when $\mathcal{G}_T = G_{\Phi_k(p)}$ is detailed in Section 5.2 of (Barreto et al., 2015) for the curve families BN-12, BLS-12, KSS-18, and BLS-24, in turn elaborating on Section 8.2 of (Scott, 2013). There, testing membership in $\mathcal{G}_T$ is shown to only require one multiplication in $\mathbb{G}_T$ and a few lower-order Frobenius-based simplifications. As a comparison, currently the best methods for testing membership in $\mathbb{G}_T$ involve a large-exponent exponentiation in $\mathbb{G}_T$ (see, e.g., discussions in (Scott, 2021)). Thus, in our protocols, instead of an expensive membership test for $\mathbb{G}_T$, we use a much cheaper membership test for $\mathcal{G}_T$, and prove that it suffices for our purposes, when used in conjunction with our probabilistic correctness tests.

For *parameterized efficiency* evaluation of our protocols, we will use the following definitions:

- a_i: runtime for addition in $\mathbb{G}_i$, for $i = 1,2$;
- $m_i(\ell)$: runtime for scalar multiplication of a group value in $\mathbb{G}_i$ with an ℓ-bit scalar value, for $i = 1,2$;
- m_T: runtime for multiplication of group values in $\mathbb{G}_T$;
- $e_T(\ell)$: runtime for an exponentiation in $\mathbb{G}_T$ to an ℓ-bit exponent;
- p_T: runtime for the bilinear pairing e;
- i_l denotes the runtime for multiplicative inversion in $\mathbb{Z}_l$;
- t_M: runtime for testing membership of a value to $\mathcal{G}_T = G_{\Phi_k(p)}$.

We recall some well-known facts about these quantities, of interest when evaluating the efficiency of our protocols. First, for large enough ℓ, $a_1 << m_1(\ell)$, $a_2 << m_2(\ell)$, $m_T(\ell) << e_T(\ell)$, and $e_T(\ell) < p_T$. Also, using a double-and-add (resp., square-and-multiply) algorithm, one can realize scalar multiplication (resp., exponentiation) in additive (resp., multiplicative) groups using, for random scalars (resp., random exponents), about 1.5ℓ additions (resp., multiplications). Membership of a value w in $\mathbb{G}_T$ can be computed using one exponentiation in $\mathbb{G}_T$ to the l-th power (i.e., checking that $w^l = 1$), but we avoid this or any other expensive group membership tests in our protocols.

For *numeric efficiency* evaluation of our protocols, we will use benchmark results from (Bos et al., 2013) on runtime of an optimal ate pairing and of other most expensive operations (i.e., scalar multiplication in groups $\mathbb{G}_1$, $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_T$) for some of the best curve families, also recalled in Table 2. For both parameterized and numeric evaluation of our protocols, we will neglect lower-order operations such as equality testing, assignments, Frobenius calculations, etc.

3 A FLAWED PROTOCOL

In this section we review a protocol underlying the first 3 delegation schemes in (Kalkar et al., 2022), and study its properties. We show that this protocol satisfies result correctness without need for precomputation but does not satisfy result security, by describing an adversary who, while acting as S, can force C to return an incorrect output with probability 1. We also discuss the flaw in the proof for the result security property which was described in (Kalkar et al., 2022).

Informal Description: We consider the task of delegating, without pre-computation, the computation of a pairing e on input $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, for the input scenario where both A and B are publicly known (thus, no input privacy is required). A very natural approach consists of C asking S to produce both the desired pairing $e(A,B)$ and a pairing for a related input pair $e(A',B')$, and use the relationship between (A,B) and (A',B') to probabilistically check the correctness of the first pairing value. We formally describe one instance of this approach from (Kalkar et al., 2022).

Formal Description: A formal description of Algorithm 1 (PVPV) in (Kalkar et al., 2022) for $i = 1$.

Input scenario: A and B are public online.

Online phase instructions:

```
1. $C$ randomly chooses $a,b \in \mathbb{Z}_l$
   $C$ computes $A' := aA$ and $B' = bB$
   $C$ sends $A',B'$ to $S$
2. $S$ computes $\alpha := e(A,B)$ and $\alpha' := e(A',B')$
   $S$ sends $\alpha,\alpha'$ to $C$
3. $C$ checks that $\alpha,\alpha' \in \mathcal{G}_T$
   $C$ checks that $\alpha' = \alpha^{ab}$
   If any of these tests fails,
       $C$ **returns** $\perp$ and the protocol halts
   $C$ **returns** $y := \alpha$
```

Remarks. In (Kalkar et al., 2022), the authors use a particular case of this protocol as part of their delegation protocol for a batch of pairing computations. We note that their batch delegation protocol is a direct n-fold repetition of the same subprotocol for each input pair (A_i, B_i) in the batch. This subprotocol, in turn, can be seen as a particular case of the protocol above described; specifically the pair of values (a,b) is chosen in a small set (i.e., $[0,2^t] \times \{1\}$) in their protocol, for some small value t, and in a much larger set (i.e., $\mathbb{Z}_l \times \mathbb{Z}_l$) here, where l is the (large) order of pairing groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. This generalization increases the entropy of C's message and thus even strengthens our observation below that this approach is not successful.

Result Correctness Is Satisfied. To see that the result correctness property is satisfied with probability 1, we observe that if C and S follow the protocol, then C's output y satisfies $y = e(A,B)$ by step 2 of the protocol, and C does not output $\perp$ since the verification check in step 3 is satisfied, as:

$$\alpha' = e(aA, bB) = e(A,B)^{ab} = \alpha^{ab}.$$

Result Security Is Not Satisfied. To see that the result security property is not satisfied with probability 1, we now show an attacker algorithm S' which, when playing as S, makes C return $y \neq e(A,B)$ with probability very close to 1.

S's instructions: On input C's message (A', B'), S' does the following:

1. randomly choose $u, v \in \mathbb{Z}_l$
2. S' sets $\beta := e(uA, vB)$ and $\beta' := e(uA', vB')$
3. S' sends β, β' to C.

We now show that both of C's verifications are satisfied. About the first verification, we observe that $A', uA, uA' \in \mathbb{G}_1$ since so does A, $B', vB, vB' \in \mathbb{G}_2$ since so does B, and thus $\beta, \beta' \in \mathcal{G}_T$ by definition of pairing. About the second verification, we observe that

$$\begin{aligned} \beta' &= e(uA', vB') = e(u(aA), v(bB)) \\ &= e(a(uA), b(vB)) = e(uA, vB)^{ab} = \beta^{ab}. \end{aligned}$$

Thus, C returns $y = \beta = e(uA, vB)$, which is $\neq e(A,B)$ whenever $u \neq 1$ and $v \neq 1$, and thus with probability $1 - 1/l^2$.

Flaws in the Proof from (Kalkar et al., 2022). The proof for the result security property, as written in (Kalkar et al., 2022), is based on the following 3 claims:

1. A' leaks no information about a
2. A' is uniformly distributed since a is chosen uniformly random
3. Claim 2 implies Claim 1.

We now analyze these 3 claims.

As written, Claim 2 is unspecified because there are no domains for the uniform distribution or set from which a is chosen. Given that the protocol uniformly chooses a from $[1,2^t]$, and sets $A' = a \cdot A$, a correct revision of this statement would say that A' is uniformly distributed in a 2^t-size subset of $\mathbb{G}_1$ since a is chosen uniformly from $[1,2^t]$.

To meaningful analyze Claim 3, we observe that the above updated variant of Claim 2 does not imply Claim 1. This is because there are only 2^t possible values for a, if not conditioning on A, A', and there is only 1 value of a, when conditioning on A, A'.

Indeed Claim 1 is also false because given A, A', and a, it is possible to test whether $A' = aA$. In other words, A' always leaks the output of a predicate of equality to a given value a.

4 A NEW PROTOCOL

In this section we investigate client-server protocols for pairing delegation, in the scenario where the two pairing inputs are known to both parties, and there is no offline phase or precomputation prior to the online phase. In other words, all calculation will be done only during the online phase. Our main result is a new protocol with desirable security and efficiency properties. In what follows, we give a formal statement of our result, an asymptotic and concrete efficiency comparison with the previous most efficient protocols in the same input scenario, an informal description of the ideas behind the protocol, a formal description of the protocol and a proof of the protocol's correctness and security properties.

Theorem 4.1. Let e be a pairing, as defined in Section 2, let σ be its computational security parameter, and let λ be a statistical security parameter. There exists (constructively) a client-server protocol (C,S) for delegating the computation of e without precomputation, when inputs A and B are both publicly known in the online phase, which satisfies 1-result correctness, $2^{-\lambda}$-result security, and efficiency with parameters (t_S, t_C, cc, mc), where

– $t_S = 3\,p_T$
– $t_C \leq a_1 + i_l + m_1(\sigma) + m_T + e_T(\lambda) + e_T(r) + 2t_M$
– $cc = 2$ values in $\mathbb{G}_1 + 3$ values in $\mathbb{G}_T$
– $mc = 2$.

The main takeaway from this theorem is that C can securely and efficiently delegate to S the computation of a bilinear pairing whose both inputs A and B are publicly known in the online phase and where there is no offline phase for C to precomputes anything. In particular, in the online phase C performs one group exponentiation and 1 exponentiation to a λ-bit exponent in $\mathbb{G}_T$, and 1 group multiplication and 1 multiplication to a λ-bit scalar in $\mathbb{G}_1$, as well as other lower-order operations; see also Table 3 for a more detailed analysis of the asymptotic performance of our protocol, even compared with previous work. The numeric efficiency improvement over non-delegated computation was estimated to range between 1.353 and 2.832 depending on the curve used; see also Table 3. Additionally, C does not precalculate any operations during offline phase, S only computes 3 pairings, and C and S only exchange 2 messages containing a small number of group values.

Protocol Description. The main idea in this protocol is that since both inputs A and B are publicly known, S can compute $w_0 = e(A,B)$ and send w_0 to C, along with some efficiently verifiable 'proof' that w_0 was correctly computed. This proof is realized by the following 3 steps: first, C sends to S a randomized version Z_0 and Z_1 of a randomly chosen value U and the input value A (masked using U); then S computes and sends to C pairing values $w_1 = e(Z_0,B)$ and $w_2 = e(Z_1,B)$; and finally C verifies that $w_0,w_1 \in \mathcal{G}_T$ and uses w_0,w_1 and w_2 in an efficient probabilistic verification for the correctness of S's message (w_0,w_1,w_2). We stress that instead of performing offline calculations, C performs 1 exponentiation with a full-domain exponent and 1 exponentiation with a 1 short, λ-bit exponent, in group $\mathbb{G}_T$. A formal description follows.

Formal Description of Protocol $\mathcal{P}_1(C,S)$.

Online Input to C and S: $1^\sigma, 1^\lambda, desc(e), A \in \mathbb{G}_1$, and $B \in \mathbb{G}_2$

Online phase instructions:

1. C randomly chooses $b \in \{1,\ldots,2^\lambda\}$, and $U \in \mathbb{G}_1$;
   C sets $u' := u^{-1} \mod l$, $Z_0 := u' \cdot U$, and $Z_1 := b \cdot A + U$
   C sends Z_0, Z_1 to S
2. S computes $w_0 := e(A,B)$, $w_1 := e(Z_0,B)$ and $w_2 := e(Z_1,B)$
   S sends w_0,w_1,w_2 to C
3. Membership Test: C checks that $w_0,w_1 \in \mathcal{G}_T$
   Probabilistic Test: C checks that $w_2 = w_0^b \cdot w_1^u$
   If any of these tests fails,
   C **returns** $\perp$ and the protocol halts
   C **returns** $y = w_0$

Properties of Protocol $\mathcal{P}_1(C,S)$: *The efficiency properties* are verified by protocol inspection. In particular:

- *Round complexity:* the online phase of the protocol only requires two messages: one from C to S, followed by one from S to C.
- *Communication complexity:* during the online phase, C sends 2 values in $\mathbb{G}_1$ and S sends 3 values in $\mathbb{G}_T$.
- *Runtime complexity:* the runtime property directly follows by protocol inspection. In particular, C's calculation of Z_0, Z_1 only requires 1 group multiplication, 1 multiplication to a short, λ-bit, scalar, and 1 scalar addition in a group $\mathbb{G}_1$. In C's $\mathcal{G}_T$-membership test only requires 1 multiplication in $\mathbb{G}_T$ for each membership test, as discussed in Section 2, total 1 multiplications in $\mathbb{G}_T$, and C's probabilistic test requires 1 multiplication, 1 group multiplication and 1 exponentiation in $\mathbb{G}_T$ to a short, λ-bit, exponent.

The *correctness* property follows by showing that if C and S follow the protocol, C always output $y = e(A,B)$. We show that the 2 tests performed by C are always passed. The membership test is always passed by pairing definition; the probabilistic test is always passed since

$$\begin{aligned} w_2 = e(Z_1,B) &= e(b \cdot A + U, B) \\ &= e(A,B)^b \cdot e(U,B) = e(A,B)^b \cdot e(u^{-1} \cdot U, B)^u \\ &= e(A,B)^b \cdot e(Z_0,B)^u = w_0^b \cdot w_1^u. \end{aligned}$$

This implies that C never returns $\perp$, and thus returns $y = w_0 = e(A,B)$.

To prove the *security* property against any malicious S we need to compute an upper bound ε_s on the security probability that S convinces C to output a y such that $y \neq e(A,B)$. We obtain that $\varepsilon_s \leq 2^{-\lambda}$ as a consequence of the following 3 facts, which we later prove:

1. (Z_0,Z_1) leaks no information about b to S;
2. for any S's message (w_0,w_1,w_2) different than what would be returned according to the protocol instructions, there is only one b for which the tuple (w_0,w_1,w_2) satisfies both membership and probabilistic tests in step 2;
3. for any S's message (w_0,w_1,w_2) different than what would be returned according to the protocol instructions, the probability that (w_0,w_1,w_2) satisfies the probabilistic test is $\leq 2^{-\lambda}$.

Towards proving Fact 1, we observe that: (a) $Z_0 = u^{-1} \cdot U$ is uniformly and indigently distributed in $\mathbb{G}_1$ since so is the u in $\mathbb{Z}_l$ and it does not leak any information about U in $\mathbb{G}_1$; (b) $Z_1 = b \cdot A + U$ is also uniformly distributed in $\mathbb{G}_1$ since so is U by (a) and Z_1 does not leak any information about b to S.

Table 1: Protocols comparison in the input scenario (A and B public online). The expressions for t_C only include higher-order functions p_T, e_T, m_1, m_2.

Protocols	**C's pre-calculation (t_P)**	**C's online calculation (t_C)**
(Chevallier-Mames et al., 2010) §5.2	$p_T + e_T(r) + m_1(r) + m_2(r)$	$3\,e_T(r) + m_1(r) + m_2(r)$
(Canard et al., 2014) §4.1	$p_T + e_T(r) + m_1(r) + m_2(r)$	$e_T(r) + m_1(r) + m_2(r)$
(Di Crescenzo et al., 2020a) §4.1	$2\,p_T + m_2(r) + 2\,m_1(r)$	$2\,e_T(\lambda) + m_2(\lambda) + m_1(r) + m_1(\lambda)$
(Di Crescenzo et al., 2020b) §3	$p_T + m_2(r)$	$e_T(\lambda) + m_2(\lambda) + m_1(r)$
Ours [§ 4]	0	$e_T(r) + e_T(\lambda) + m_1(r)$

Towards proving Fact 2, let (w_0, w_1, w_2) be the values that would be returned by S according to the protocol, and assume a malicious algorithm Adv, corrupting S returns a different triple (w_0', w_1', w_2'). Note that if $w_0' \notin \mathcal{G}_T$ or $w_1' \notin \mathcal{G}_T$, the triple (w_0', w_1', w_2') does not satisfy the group $\mathcal{G}_T$-membership test. Thus, we assume that both $w_0' \in \mathcal{G}_T$ and $w_1' \in \mathcal{G}_T$ and observe that if triple (w_0', w_1', w_2') satisfies the probabilistic correctness test, then $w_2' \in \mathcal{G}_T$. Because $\mathbb{G}_T$ is a multiplicative group, we can write $w_i' = d_i \cdot w_i$ for $i = 0, 1, 2$ and some $d_0, d_1, d_2 \in \mathbb{G}_T$ such that $d_0 \neq 1$ or $d_1 \neq 1$ or $d_2 \neq 1$. Now, assume wlog that $d_0 \neq 1$ and consider the following equivalent rewritings of the probabilistic test, obtained by variable substitutions and simplifications:

$$\begin{aligned} w_2' &= (w_0')^b \cdot (w_1')^u \\ d_2 \cdot w_2 &= (d_0 \cdot w_0)^b \cdot (d_1 \cdot w_1)^u \\ d_2 \cdot w_2 &= (d_0^b \cdot d_1^u) \cdot w_0^b \cdot w_1^u \\ d_2 &= d_0^b \cdot d_1^u \\ d_2 (d_1)^{-u} &= d_0^b, \end{aligned}$$

where the 4th equality follows from the correctness property implying that $w_2 = w_0^b \cdot w_1^u$.

Now, if there exist two distinct b_1 and b_2, assumed wlog to satisfy $b_1 > b_2$, and such that

$$d_2(d_1)^{-u} = d_0^{b_1} \text{ and } d_2(d_1)^{-u} = d_0^{b_2},$$

then $d_0^{b_1 - b_2} = 1$. By our assumption that every element in $\mathbb{G}_T$ has order $> l$, which is $> 2^\lambda$, and by observing that $b_1 - b_2 < 2^\lambda$, we derive that d_0 cannot have order $\leq b_1 - b_2$. Thus the equality $d_0^{b_1 - b_2} = 1$ can only hold when $b_1 = b_2$.

This proves Fact 2.

Towards proving Fact 3, note that, by Fact 1, C's message Z_0, Z_1 does not leak any information about b. This implies that all values in $\{1, \ldots, 2^\lambda\}$ are still equally likely for c even when conditioning over messages Z_0, Z_1. Then, by using Fact 2, the probability that S's message (w_0, w_1, w_2) satisfies the probabilistic test, is 1 divided by the number 2^λ of values of b that are still equally likely when conditioning over message Z_0, Z_1. This proves Fact 3.

4.1 Extension to Other Input Cases

***A* Private and *B* Public.** In this input scenario, we can define protocol $\mathcal{P}_2$ by some natural modifications to protocol $\mathcal{P}_1$, where C further apply a random mask to A before asking S to compute and return pairings, and later C performs one more exponentiation in $\mathbb{G}_T$ to undo the mask, and recover the desired pairing value.

***A* and *B* Private.** In this input scenario, we can define protocol $\mathcal{P}_3$ by some natural modifications to protocols $\mathcal{P}_1$ and $\mathcal{P}_2$, where C further applies a random mask to both A and B before asking S to compute and return pairings, and later C performs one more exponentiation in $\mathbb{G}_T$ to undo the masks, check the correctness of the received values and recover the desired pairing value.

Performance Analysis. In our performance analysis for protocols $\mathcal{P}_2$ and $\mathcal{P}_3$, C's runtime is only lower than non-delegated computation for one of the 4 considered curve families.

5 NUMERICAL PERFORMANCE ANALYSIS

We show a numerical performance analysis of our protocols from Section 4, as well as previous protocols from the literature in each of the considered input scenarios. Our numerical performance analysis consists of using benchmark results from (Bos et al., 2013) for the runtime of an optimal ate pairing and of the other most expensive operations (i.e., scalar multiplication in groups $\mathbb{G}_1$, $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_T$) for relative to an optimal ate pairing based on some of the currently most practical elliptic curve families (i.e., BN-12, BLS-12, KSS-18, BLS-24), also recalled in Table 2. In Table 3 we compare the performance of our protocols in Section 4 with past work for the same input scenario.

Table 2: Benchmark results (obtained by (Bos et al., 2013) on an Intel Core i7-3520M CPU averaged over thousands of random instances) for scalar multiplications in $\mathbb{G}_1, \mathbb{G}_2$ and exponentiations in $\mathbb{G}_T$ relative to an optimal ate pairing based on some of the best known curve families, measured in millions (M) of clock cycles. The security levels are from (Bos et al., 2013), except for BN-12, whose level was reduced because of recent attacks (Kim and Barbulescu, 2016).

Sec. level	**Family-k**	**Pairing e**	**Scal. mul. in $\mathbb{G}_1$**	**Scal. mul. in $\mathbb{G}_2$**	**Exp. in $\mathbb{G}_T$**
105-bits	BN-12	7.0	0.9	1.8	3.1
192-bits	BLS-12	47.2	4.4	10.9	17.5
	KSS-18	63.3	3.5	9.8	15.7
256-bits	BLS-24	115.0	5.2	27.6	47.1

Table 3: Protocols comparison in scenarios where both A and B are publicly known.

Protocols	**C's pre-calculation (t_P)**	**Ratio: p_T/t_C**			
		BN12 $r = 210$	**BLS12** $r = 424$	**KSS18** $r = 376$	**BLS24** $r = 504$
Input Scenario: (A and B are publicly known and thus no input privacy is required)					
(Chevallier-Mames et al., 2010) §5.2	$p_T + e_T(r)$	0.580	0.694	1.045	0.659
(Canard et al., 2014) §4.1	$p_T + e_T(r) + m_1(r) + m_2(r)$	1.197	1.433	2.173	1.434
(Di Crescenzo et al., 2020a) §4	$2\,p_T + m_2(r) + 2\,m_1(r)$	2.001	4.045	6.869	5.571
(Di Crescenzo et al., 2020b) §3	$p_T + m_2(r)$	2.981	5.519	8.216	7.994
This paper §4	0	**1.353**	**1.881**	**2.832**	**1.958**

6 CONCLUSIONS

In this paper we studied the problem of techniques for a computationally weaker client to efficiently, privately and securely delegate bilinear pairings to a single, possibly malicious, server, without precomputation. Previously to this paper, we only knew of two examples of operation commonly used in cryptographic protocols having such delegation protocols without precomputation: group inverses, and multiple group exponentiations with a public base and multiple public exponents. It remains of interest to extend our protocol to more elaborated input scenarios (i.e., keeping input privacy) while achieving client efficiency for many curve families of interest in practical applications of pairings. It also remains of interest to find more operations often used in cryptography constructions for which such delegation protocols exist.

ACKNOWLEDGEMENTS

Work by Matluba Khodjaeva was supported by a Faculty Scholarship grant from the Office for the Advancement of Research at John Jay College and PSC CUNY Cycles 53 and 54. Work by Giovanni Di Crescenzo was supported by the Defense Advanced Research Projects Agency (DARPA), contract n. HR001120C0156. Approved for Public Release, Distribution Unlimited. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or the U.S. Government.

REFERENCES

M. Abadi, J. Feigenbaum, J. Kilian, *On Hiding Information from an Oracle* In: J. Comput. Syst. Sci. 39(1): 21-50 (1989).

H. Ahmad, L. Wang, H. Hong, J. Li, H. Dawoo, M. Ahmed, Y. Yang, *Primitives towards verifiable computation: a survey*. In Front. Comput. Sci. 2018, Vol. 12, Issue (3) : 451-478.

P.S.L.M. Barreto, C. Costello, R. Misoczki, M. Naehrig, G.C.C.F. Pereira, G. Zanon, *Subgroup security in pairing-based cryptography.* In Lauter K., Rodríguez-Henríquez F. (eds) LATINCRYPT 2015. LCNS vol. 9230. Springer.

D. Boneh, M. Franklin, *Identity-based encryption from the weil pairing.* In Kilian J. (eds) Advances in Cryptology — CRYPTO 2001. LNCS vol 2139. Springer.

J.W. Bos, C. Costello, M. Naehrig, *Exponentiating in pairing groups.* In Lange T., Lauter K., Lisoněk P. (eds) SAC 2013. LNCS vol 8282. Springer.

S. Canard, J. Devigne, O. Sanders: *Delegating a pairing can be both secure and efficient.* In Boureanu I., Owesarski P., Vaudenay S. (eds) Applied Cryptography and Network Security. ACNS 2014. LNCS vol 8479. Springer

B. Cavallo, G. Di Crescenzo, D. Kahrobaei, V. Shpilrain, *Efficient and secure delegation of group exponentiation to a single server.* In Proc. of RFIDSec 2015: pp. 156–173, LNCS, Springer.

X. Chen, W. Susilo, J. Li, D.S. Wong, J. Ma, S. Tang, and Q. Tang, *Efficient algorithms for secure outsourcing of bilinear pairings.* In Theor. Comput. Sci., vol. 562, no. 9, pp. 112–121, 2015.

B. Chevallier-Mames, J.S. Coron, N. McCullagh, D. Naccache, M. Scott: *Secure delegation of elliptic-curve pairing. Cryptology ePrint Archive.* In Proc. of Smart Card Research and Advanced Application. CARDIS 2010. LNCS vol 6035. Springer. Also in http://eprint.iacr.org/2005/150.

G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, V. Shpilrain, *Secure and Efficient Delegation of Elliptic-Curve Pairing*. In: Proc. of ACNS 2020. LNCS, vol 12146. Springer, Cham.

G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, V. Shpilrain, *Secure and Efficient Delegation of Pairings with Online Inputs*. In: Proc. of CARDIS 2020. LNCS, vol. 12609. Springer.

G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, and V. Shpilrain, *Computing Multiple Exponentiations in Discrete Log and RSA Groups: From Batch Verification to Batch Delegation*. In Proc. of 3rd IEEE Workshop on Security and Privacy in the Cloud, IEEE, 2017.

G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, and V. Shpilrain: *A Survey on Delegated Computation*. In Proc. of DLT 2022: 33-53.

J. Feigenbaum, *Encrypting Problem Instances: Or ..., Can You Take Advantage of Someone Without Having to Trust Him?* In Proc. of CRYPTO 1985: 477-488.

R. Gennaro, C. Gentry, B. Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers.* In Proc. of CRYPTO 2010, LNCS 6223, pp. 465–482.

M. Girault, D. Lefranc, *Server-aided verification: Theory and practice.* In Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623.

A. Guillevic, D. Vergnaud, *Algorithms for outsourcing pairing computation.* In Joye M., Moradi A. (eds) Smart Card Research and Advanced Applications. CARDIS 2014. LNCS vol 8968. Springer.

S. Hohenberger, A. Lysyanskaya, *How to securely outsource cryptographic computations.* In Proc. of TCC 2005, pp. 264–282, Springer.

A. Joux, *A one round protocol for tripartite Diffie-Hellman*, in Proc. of ANTS IV, LNCS, vol. 1838, Springer, 2000, pp. 385–393.

E.J. Kachisa, E.F. Schaefer, M. Scott, *Constructing Brezing-Weng pairing friendly elliptic curves using elements in the cyclotomic field.* In Galbraith S.D., Paterson K.G. (eds) Pairing-Based Cryptography – Pairing 2008. LNCS vol. 5209. Springer.

B.G. Kang, M.S. Lee, J.H. Park, *Efficient delegation of pairing computation.* In IACR Cryptology ePrint Archive, n. 259, 2005.

O. Kalkar, I. Sertkaya, and S. Tutdere, *On The Batch Outsourcing Of Pairing Computations*, in: The Computer Journal, 2022.

T. Kim, R. Barbulescu, *Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case.* In Proc. of CRYPTO (1) 2016, pp. 543–571, Springer.

T. Matsumoto, K. Kato, H. Imai, *An improved algorithm for secure outsourcing of modular exponentiations*. In Proc. of CRYPTO 1988, pp. 497–506, LNCS, Springer.

D. Moody, R. Peralta, R. Perlner, A. Regenscheid, A. Roginsky, and L. Chen, *Report on Pairing-based Cryptography*, in J. Res. Natl. Inst. Stand. Technol. 120: 11–27, 2015.

M. Scott, *Unbalancing pairing-based key exchange protocols.* In IACR Cryptology ePrint Archive, n. 688, 2013.

M. Scott, *A note on group membership tests for* $\mathbb{G}_1$, $\mathbb{G}_2$ *and* $\mathbb{G}_T$ *on BLS pairing-friendly curves.* In IACR Cryptology ePrint Archive, n. 1130, 2021.

R. Sakai, K. Ohgishi, andM. Kasahara, *Cryptosystems based on pairing*, in Symposium on Cryptography and Information Security (SCIS), 2000.

Z. Shan, K. Ren, M. Blanton, C. Wang, *Practical Secure Computation Outsourcing: A Survey*. In ACM Comput. Surv. 51(2): 31:1-31:40, 2018.

P. Tsang, S. Chow, and S. Smith *Batch pairing delegation.* In: Proceedings of International Workshop on Security Nara, Japan, 29-31 October, 2007, pp.74–90.Springer,Berlin.

F. Vercauteren, *Optimal Pairings.* In IEEE Transactions on Information Theory, vol. 56, no. 1, pp. 455–461, Jan. 2010.