Secure and Efficient Delegation of Pairings with Online Inputs

Giovanni Di Crescenzo[1], Matluba Khodjaeva[2], Delaram Kahrobaei[3], and Vladimir Shpilrain[4]

[1] Perspecta Labs Inc. Basking Ridge, NJ, USA.
Email: `gdicrescenzo@perspectalabs.com`
[2] CUNY John Jay College of Criminal Justice. New York, NY, USA.
Email: `mkhodjaeva@jjay.cuny.edu`
[3] University of York. Heslington, York, UK.
Email: `delaram.kahrobaei@york.ac.uk`
[4] City University of New York. New York, NY, USA.
Email: `shpil@groups.sci.ccny.cuny.edu`

Abstract. Delegation of pairings from a computationally weaker client to a computationally stronger server has been advocated to expand the applicability of pairing-based cryptographic protocols to computation paradigms with resource-constrained devices. Important requirements for such delegation protocols include privacy of the client's inputs and security of the client's output, in the sense of detecting, with high probability, any malicious server's attempt to convince the client of an incorrect pairing result. In this paper we show that pairings with inputs only available in the online phase can be efficiently, privately and securely delegated to a single, possibly malicious, server. We present new protocols in 2 different scenarios: (1) the two pairing inputs are publicly known; (2) privacy of both pairing inputs needs to be maintained (left open in previous papers; e.g., [23]). In both cases, we improve the online-phase client's runtime with respect to previous work. In the latter case, we show the first protocol where the client's online-phase runtime is faster than non-delegated computation for all of the most practical known curves. In previous work, the client's runtime was worse, especially for one of the most practical elliptic curves underlying the pairing function (i.e., BN-12).

Keywords: Applied Cryptography · Secure Delegation · Pairings · Bilinear Maps · Elliptic Curves · Group Theory

1 Introduction

Server-aided cryptography is an active research direction addressing the problem of computationally weaker clients delegating the most expensive cryptographic computations to more powerful servers. Recently, this area has seen an increased interest in reducing computation in client devices, including smart cards, smart phones, and even more resource-constrained devices, motivated by shifts in modern computation paradigms, towards more smart-card computing, cloud/edge/fog computing, etc.

The first formal model for secure delegation protocols was presented in [15], where a secure delegation protocol is formally defined as a secure function evaluation (a concept first proposed in [24]) of the client's function delegated to the server. Follow-up models include the model from [11] which defines separate requirements of correctness, (input) privacy and (output) security. In this paper we use a model of the latter type, since it allows a more granular parameterized description of solution properties (but note that our solutions can be proved in both model types). In this model, we have a client C, with inputs A, B, who delegates to one or more servers the computation of a pairing function e on the client's input, and the main desired requirements are:

1. *δ-correctness*: if both parties follow the protocol, C's output y at the end of the protocol, is equal to the value obtained by evaluating pairing e on its input (A, B), with probability at least δ (for some high δ value);
2. *input ϵ_p-privacy*: for any malicious algorithm Adv returning possible input pairs (A_0, B_0) or (A_1, B_1), playing as S in the protocol with C, and trying to guess which of these two pairs is being used by C as input, Adv's guess is only correct with probability $1/2 \pm \epsilon_p$ (for some small ϵ_p value);
3. *output ϵ_s-security*: for any malicious algorithm Adv returning input pair (A, B), playing as S in the protocol with C on input (A, B), and trying to convince C to return an incorrect output y', the probability that $y' \neq e(A, B)$ is at most ϵ_s (for some small ϵ_s value);
4. *efficiency with parameters* $(t_F, t_C, t_S, t_P, cc, mc)$: the protocol performance is upper-bounded by these functions: t_F, the runtime to compute pairing e without delegation; t_C, C's runtime in the online phase; t_S, S's runtime; t_P, C's runtime in the offline phase; cc, the communication exchanged between C and S; and mc, the number of messages exchanged between C and S.

As in previous work in the area, somewhat expensive computation can be performed in an *offline phase* (i.e., before the pairing result is used in a cryptographic protocol) and stored on the client's device (say, at device deployment time), and the reduction in the client's computation is mainly required in the *online phase* (i.e., when the pairing result is used in a cryptographic protocol). In this paper we focus on input scenarios where inputs A, B to pairing e are only available in the online phase, which is the case of interest in some important cryptographic protocols (most notably, Joux's 3-party key agreement protocol [16]). This paper can be seen as a follow-up to [10], where we mainly studied the technically simpler input scenarios where at least one of inputs A, B is also available in the offline phase, and briefly discussed that direct compositions of those protocols can be used to design protocols in technically more complex scenarios of online inputs. Here, we use different techniques, such as new probabilistic tests, and design 2 new protocols in the scenarios of online inputs, which improve over both previous work and direct compositions of protocols for other input scenarios.

Our Contributions. In this paper we show that when both inputs are only available in the online phase, bilinear-map pairings can be efficiently, privately and securely delegated to a single, possibly malicious, server. Our results include 2 new protocols, each for a different input scenario, improving the main performance metric (client's online runtime), with respect to all 4 of the recently proposed practical elliptic curves with security levels between 128 and 256 bits, as benchmarked in [6]. In both our protocols, the client's online program only performs 1 exponentiation to a short (e.g., 128-bit) exponent in the most computationally intensive curve's target group. This improves over all previous protocols, where the client required either a larger number of exponentiations to short exponents or exponentiations to longer exponents, or more expensive pairing operations.

Our first protocol, presented in Section 3, considers the case where both inputs A, B are publicly available. Here, in the online phase, the client only performs 1 short-exponent exponentiation in the target pairing group, 1 scalar multiplication in A's group and 1 short-scalar multiplication in B's group (and other lower-order operations). Previously, a protocol with at least 1 long-exponent exponentiation was proved in [8, 7], and a protocol with 2 short-exponent exponentiations was briefly discussed in [10], as a direct composition of protocols in scenarios where at least one input is known in the offline phase.

Our second protocol, presented in Section 4, considers the case where privacy of A, B needs to be guaranteed. Here, in the online phase, the client only performs 1 short-exponent exponentiation in the target pairing group, 3 scalar multiplications in A's group and 2 scalar multiplications (of which 1 short) in B's group (and other lower-order operations). This protocol is the first where the client's online runtime is faster than non-delegated computation with respect to all 4 of the elliptic curves benchmarked in [6]. Previously, this was provably achieved for 1 curve in [7] and an approach was briefly discussed in [10] achieving the same for 3 curves, using a direct composition of protocols in scenarios where at least one input is known in the offline phase.

Both our protocols only consist of a single communication exchange between client and server, and only require client and server to communicate a constant (specifically, 6 and 11) number of group elements.

Related Work. Pairing delegation to a single server was first studied by Girault et al. [12]. However, they only considered input secrecy but no output security against a malicious server. Guillevic et al. [13] proposed a more efficient scheme but their method increases communication complexity between client and server and their scheme does not provide security against a malicious server. Single-server protocols with this latter property for delegating $e(A, B)$ have first been provided by Chevallier-Mames et al. [8] and later by Kang et al. [18]. Canard et al. [7] improved these constructions and proposed more efficient and secure pairing delegation protocols. In particular, in [7] the authors showed that in their protocols the client's runtime is strictly lower than non-delegated computation of a pairing on the so-called KSS-18 curve [17]. Later, Guillevic et al. [13] showed that in protocols in [7] the client is actually less efficient than in a non-delegated computation of the pairing for the state-of-the-art optimal ate pairing on the Barreto-Naehrig curve (BN-12).

2 Pairing Definitions

In this section we recall the definition and a number of useful facts about pairings, including current most practical realizations, as well as definitions and benchmark values to evaluate the efficiency of our protocols.

Bilinear Maps. Let $\mathcal{G}_1$, $\mathcal{G}_2$ be additive cyclic groups of order l and $\mathcal{G}_T$ be a multiplicative cyclic group of the same order l, for some large prime l. A *bilinear map* (also called *pairing* and so called from now on) is an efficiently computable map $e : \mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}_T$ with the following properties:

1. *Bilinearity:* for all $A \in \mathcal{G}_1$, $B \in \mathcal{G}_2$ and 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 $\mathcal{G}_1$ and V is a generator for $\mathcal{G}_2$ then $e(U, V)$ is a generator for $\mathcal{G}_T$

The last property is there to rule out the trivial case where e maps all of its inputs to 1. We denote a conventional description of the bilinear map e as $desc(e)$.

The currently most practical *pairing realizations* use an ordinary elliptic curve E defined over a field $\mathbb{F}_p$, for some large prime p, as follows. Group $\mathcal{G}_1$ is the l-order additive subgroup of $E(\mathbb{F}_p)$; group $\mathcal{G}_2$ is a specific l-order additive subgroup of $E(\mathbb{F}_{p^k})$ contained in $E(\mathbb{F}_{p^k}) \setminus E(\mathbb{F}_p)$; and group $\mathcal{G}_T$ is the l-order multiplicative subgroup of $\mathbb{F}_{p^k}$. Here, k is the embedding degree; i.e., the smallest positive integer such that $l|(p^k - 1)$. After the Weil pairing was considered in [3], more efficient constructions have been proposed as variants of the Tate pairing, including the more recent ate pairing variants (see, e.g., [22] for more details on the currently most practical pairing realizations).

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

- a_1 (resp. a_2) denotes the runtime for addition in $\mathcal{G}_1$ (resp. $\mathcal{G}_2$);
- $m_1(\ell)$ (resp. $m_2(\ell)$) denotes the runtime for scalar multiplication of a group value in $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) with an ℓ-bit scalar value;
- m_T denotes the runtime for multiplication of group values in $\mathcal{G}_T$;
- $e_T(\ell)$ denotes the runtime for an exponentiation in $\mathcal{G}_T$ to an ℓ-bit exponent;
- p_T denotes the runtime for the bilinear pairing e;
- i_l denotes the runtime for multiplicative inversion in $\mathbb{Z}_l$;
- t_M denotes the runtime for testing membership of a value to $\mathcal{G}_T$;
- σ denotes the computational security parameter (i.e., the parameter derived from hardness studies of the underlying computational problem);
- λ denote the statistical security parameter (i.e., a parameter such that events with probability $2^{-\lambda}$ are extremely rare).

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). Finally, membership of a value w in $\mathcal{G}_T$ can be tested using one exponentiation in $\mathcal{G}_T$ to the l-th power (i.e., checking that $w^l = 1$), or, for some specific elliptic curves, including some of the most recommended in practice, using about 1 multiplication in $\mathcal{G}_T$ and lower-order Frobenius-based simplifications (see, e.g., [21, 2]).

For *concrete efficiency* estimate of our protocols, we will set $\lambda = 128$ and use benchmark results from [6] for the runtime of an optimal ate pairing and of the other most expensive operations (i.e., scalar multiplication in groups $\mathcal{G}_1$, $\mathcal{G}_2$ and exponentiation in $\mathcal{G}_T$) for the some of the most practical curve families, also recalled in Table 1 below, as well as for the values of σ for each curve family, recalled in Tables 2 and 3. We will also neglect lower-order operations such as equality testing, assignments, Frobenius-based simplifications, etc.

Table 1. Benchmark results (obtained by [6] on an Intel Core i7-3520M CPU averaged over thousands of random instances) for scalar multiplications in $\mathcal{G}_1, \mathcal{G}_2$ and exponentiations in $\mathcal{G}_T$ relative to an optimal ate pairing based on some of the some of the most practical curve families, measured in millions (M) of clock cycles.

Security level	**Family-k**	**Pairing e**	**Scal. mul. in $\mathcal{G}_1$**	**Scal. mul. in $\mathcal{G}_2$**	**Exp. in $\mathcal{G}_T$**
128-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

3 Delegating Pairings with Online Public Inputs

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 not before 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.

Our first protocol satisfies the following

Theorem 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, when inputs A and B are both publicly known in the online phase which satisfies 1-correctness, $2^{-\lambda}$-security, 0-privacy, and efficiency with parameters $(t_F, t_S, t_P, t_C, cc, mc)$, where

- $t_F = p_T$, $t_S = 3\,p_T$ and $t_P = p_T + m_2(\sigma) + i_l$;
- $t_C \leq a_1 + a_2 + m_1(\sigma) + m_2(\lambda) + 2\,m_T + e_T(\lambda) + 2\,t_M$;
- $cc = 1$ value in $\mathcal{G}_1$ + 2 values in $\mathcal{G}_2$ + 3 values in $\mathcal{G}_T$ and $mc = 2$.

The main takeaway from this theorem is that C can securely and efficiently delegate to S the computation of a bilinear pairing where both inputs A and B are publicly known but only available in the online phase. In particular, in the online phase C only performs 1 exponentiation to a λ-bit (thus, shorter) exponent in $\mathcal{G}_T$, 1 scalar multiplication in $\mathcal{G}_1$, 1 multiplication in $\mathcal{G}_2$ by a short, λ-bit scalar, and other lower-order operations. See Table 2 for a comparison with closest previous work, also showing estimated ratios of C's online runtime to a non-delegated pairing calculation ranging between 0.158 and 0.326 depending on the curve used. Additionally, C only computes 1 pairing in the offline phase, S only computes 3 pairings, and C and S only exchange 2 messages containing 6 group values.

Table 2. Protocols comparison in the input scenario with public A and B. (The expression of t_C only includes higher-order functions e_T, m_1, m_2. The estimated ratio t_C/t_F also counts terms based on functions a_1, a_2, m_T, t_M and uses $\lambda = 128$.)

Protocols	t_C	**Ratio:** t_C/t_F			
		BN-12 $\sigma = 461$	**BLS-12** $\sigma = 635$	**KSS-18** $\sigma = 508$	**BLS-24** $\sigma = 629$
[8] §5.2	$e_T(\sigma) + m_1(\sigma) + m_2(\sigma)$	1.719	1.439	0.956	1.517
[7] §4.1	$e_T(\sigma) + m_1(\sigma)$	0.832	0.697	0.460	0.697
[10] §4.1	$2\,e_T(\lambda) + m_2(\lambda) + m_1(\sigma) + m_1(\lambda)$	0.485	0.310	0.235	0.272
This paper §3	$e_T(\lambda) + m_1(\sigma) + m_2(\lambda)$	0.326	0.216	0.158	0.179

Protocol description. Note that in the input scenario where both A and B are public, a protocol satisfying correctness is trivial: C sends A, B, and S replies with $e(A, B)$. Also, there is no privacy property to satisfy, and so the challenge remains to modify this protocol so to efficiently satisfy output security. Our approach for that is based on a new probabilistic test which involves computation of 2 additional pairings from S, the first with inputs A and a masked version of B, and the second with inputs a masked version of A and a differently masked version of B. Masks are chosen carefully so to achieve some cancellation effect and to allow efficient verification from C, while especially minimizing the use of exponentiations in $\mathcal{G}_T$ to only 1 exponentiation to a short, λ-bit, exponent.

A formal description follows.

Offline Input to C and S: $1^\sigma, 1^\lambda$, $desc(e)$

Offline phase instructions:

1. C randomly chooses $U \in \mathcal{G}_1$, $P \in \mathcal{G}_2$, $c \in \{1, \ldots, 2^\lambda\}$ and $r \in \mathbb{Z}_l^*$
2. C sets $\hat{r} = r^{-1} \mod l$, $Q_0 := \hat{r} \cdot P$, $v := e(U, P)$ and $ov = (c, r, U, P, Q_0, v)$

Online Inputs: $A \in \mathcal{G}_1$ and $B \in \mathcal{G}_2$ to both C and S, and ov to C

Online phase instructions:

1. C sets $Z := r(A - U)$, $Q_1 := c \cdot B + P$ and sends Z, Q_0, Q_1 to S
2. S computes $w_0 := e(A, B)$, $w_1 := e(A, Q_1)$, $w_2 := e(Z, Q_0)$
   S sends w_0, w_1, w_2 to C
3. (Membership Test:) C checks that $w_0, w_2 \in \mathcal{G}_T$
   (Probabilistic Test:) C checks that $w_1 = (w_0)^c \cdot w_2 \cdot v$
   (with this test, C implicitly checks that $w_1 \in \mathcal{G}_T$)
   If any of these tests fails, C **returns** $\perp$ and the protocol halts
   C **returns** $y = w_0$

Properties of the protocol (C, S)**:** *The efficiency properties* are verified by protocol inspection. In particular, as for C's online runtime, note that the probabilistic test requires 1 exponentiation to a short, λ-bit, exponent in $\mathcal{G}_T$; the computation of Z requires 1 scalar multiplication in $\mathcal{G}_1$; the computation of Q_1 requires 1 multiplication in $\mathcal{G}_2$ by a short, λ-bit scalar. Additional lower-order operations include 2 addition/subtractions in $\mathcal{G}_1$ or $\mathcal{G}_2$, 2 multiplications and a small number of Frobenius-based simplifications in $\mathcal{G}_T$.

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

$$\begin{aligned}
w_0^c \cdot w_2 \cdot v &= e(A, B)^c \cdot e(Z, Q_0) \cdot e(U, P) \\
&= e(A, B)^c \cdot e(r(A - U), r^{-1} \cdot P) \cdot e(U, P) \\
&= e(A, B)^c \cdot e(A - U, P) \cdot e(U, P) \\
&= e(A, c \cdot B) \cdot e(A, P) \cdot e(U, P)^{-1} \cdot e(U, P) \\
&= e(A, c \cdot B + P) = e(A, Q_1) = w_1
\end{aligned}$$

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 $\epsilon_s \leq 2^{-\lambda}$ as a consequence of the following 3 facts, which we later prove:

1. the tuple (Z, Q_0, Q_1) leaks no information about c 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 c for which the tuple (w_0, w_1, w_2) satisfies both membership and probabilistic tests in step 3;
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, recall that $\mathcal{G}_1, \mathcal{G}_2$ are cyclic groups, and let G_1 be a generator of $\mathcal{G}_1$, and G_2 be a generator of $\mathcal{G}_2$. Also, let a, b be values such that $A = aG_1, B = bG_2$, and let u, z, p, q_0, q_1 be values such that

$$U = uG_1, Z = zG_1, P = pG_2, Q_0 = q_0G_2, Q_1 = q_1G_2.$$

Note that $(Z, Q_0, Q_1) = (zG_1, q_0G_2, q_1G_2)$, and, since $\mathcal{G}_1, \mathcal{G}_2$ are cyclic groups of prime order l, to prove Fact 1, it suffices to show that the distribution of the triple (z, q_0, q_1) is independent on c, for all $a, b \in \mathbb{Z}_l$ and any $c \in \{0, \ldots, 2^\lambda\}$. The latter is proved as follows: first observe that q_1 is uniformly distributed in $\mathbb{Z}_l$ since so is p and $q_1 = cb + p \mod l$; next, observe that q_0 is either $= 0$ when $p = 0$ or is uniformly distributed in $\mathbb{Z}_l^*$, even conditioned on q_1, since so is r and $q_0 = r^{-1}p \mod l$; and finally, observe that z is uniformly distributed in $\mathbb{Z}_l$, even conditioned on q_0, q_1, since so is u and $z = r(a - u) \mod l$.

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'). Because $\mathcal{G}_T$ is cyclic, we can consider a generator g for $\mathcal{G}_T$ and write $w_i = g^{a_i}$, for $i = 0, 1, 2$. Note that if the membership and probabilistic test are passed by (w_0', w_1', w_2'), all values w_0', w_1', w_2' are verified to be in $\mathcal{G}_T$. Then we can write

$$\forall i = 0, 1, 2,\ \exists u_i \in \mathbb{Z}_l \quad w_i' = g^{u_i} \cdot w_i \text{ such that for some } u_i \neq 0.$$

Now, note that in Fact 2 we study the case $u_0 \neq 0 \mod l$ (or else $y = w_0' = w_0 = e(A, B)$), and consider the following equivalent rewritings of the probabilistic test, obtained by variable substitutions and simplifications:

$$\begin{aligned}
w_1' &= (w_0')^c \cdot w_2' \cdot v \\
g^{u_1} \cdot w_1 &= (g^{u_0} \cdot w_0)^c \cdot g^{u_2} \cdot w_2 \cdot v \\
g^{u_1} \cdot w_1 &= g^{cu_0 + u_2} \cdot w_0^c \cdot w_2 \cdot v \\
g^{u_1} &= g^{cu_0 + u_2} \\
u_1 &= cu_0 + u_2 \mod l,
\end{aligned}$$

where the 4th equality follows since $w_1 = w_0^c \cdot w_2 \cdot v$. Now, if there exist two distinct c_1 and c_2 such that

$$u_1 = c_1u_0 + u_2 \mod l, \text{ and } u_1 = c_2u_0 + u_2 \mod l$$

then $u_0(c_1 - c_2) = 0 \mod l$, and finally $c_1 - c_2 = 0 \mod l$ (i.e $c_1 = c_2$), since $u_0 \neq 0 \mod l$. This shows that c is unique when $u_0 \neq 0 \mod l$, which proves Fact 2.

Towards proving Fact 3, note that, by Fact 1, C's message Z, Q_0, Q_1 does not leak any information about c. This implies that all values in $\{1, \ldots, 2^\lambda\}$ are still equally likely for c even when conditioning over messages Z, Q_0, Q_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 c that are still equally likely when conditioning over message Z, Q_0, Q_1. This proves Fact 3.

4 Delegating Pairings with Online Private Inputs

In this section we investigate client-server protocols for secure pairing delegation, in the scenario where both of the pairing inputs are only known to the client in the online phase, and need to remain private from the server. Our main result is a new protocol with desirable privacy, security and efficiency properties. In what follows, we give a formal statement of our result, an asymptotic and a concrete efficiency comparison with previous 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, privacy and security properties. Formally, our second protocol satisfies the following

Theorem 2. 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, when inputs A and B are both privately known to C in the online phase which satisfies 1-correctness, $2^{-\lambda}$-security, 0-privacy, and efficiency with parameters $(t_F, t_S, t_P, t_C, cc, mc)$, where
- $t_F = p_T$, $t_S = 4\,p_T$ and $t_P = 2\,p_T + 4\,m_2(\sigma) + 3\,i_l$;
- $t_C \leq 2\,a_1 + 2\,a_2 + 3\,m_1(\sigma) + m_2(\sigma) + m_2(\lambda) + 4\,m_T + e_T(\lambda) + 3\,t_M$;
- $cc = 3$ values in $\mathcal{G}_1 + 4$ values in $\mathcal{G}_2 + 4$ values in $\mathcal{G}_T$ and $mc = 2$.

The main takeaway from this theorem is that C can privately, securely and efficiently delegate to S the computation of a bilinear pairing in the input scenario where both A and B are only available to C, not before the online phase. In particular, in the online phase C only performs 1 exponentiation to a λ-bit (thus, shorter) exponent in $\mathcal{G}_T$, 3 scalar multiplications in $\mathcal{G}_1$, 2 scalar multiplications in $\mathcal{G}_2$, of which 1 with a λ-bit scalar, and other lower-order operations. See Table 3 for a concrete comparison with closest previous work, also showing estimated ratios of C's online runtime to a non-delegated pairing calculation ranging between 0.425 and 0.843 depending on the curve used. Additionally, C only computes 2 pairings in the offline phase, S only computes 4 pairings, and C and S only exchange 2 messages containing 11 group values.

Protocol(s) description. In our investigation, we actually evaluated 5 protocols, by past works, including ours, possibly combined with our result in Section 3, and 1 being new. In what follows, we briefly and informally describe the first 4 protocols, and then formally describe the new protocol, which happens to be the only one where C's online runtime is smaller than a non-delegated computation, and which proves Theorem 2. In Table 3 we give a detailed comparison of all protocols, with respect to C's online runtime.

Protocol Π_0. This protocol combines a randomization technique appeared, for instance, in [7], with our pairing delegation protocol from Section 3. C randomly chooses $r, s \in \mathbb{Z}_l$, computes rA, sB and sends them to S. Then C and S run a protocol such as the one from Section 3 in the (A' public online, B' public online) input scenario, where $A' = rA$ and $B' = sB$. Finally, A computes the desired result $e(A, B)$ as $e(A', B')^{1/rs}$.

Protocol Π_1. This protocol has been mentioned in our previous paper [10] and combines a simple randomization technique with pairing delegation protocols for different input scenarios in that same paper. In the offline phase, C randomly chooses $r \in \mathbb{Z}_l$ and $U \in \mathcal{G}_1$ and set $s = r^{-1} \mod l$. Then C and S run a protocol in the (A' public online, B' public online) input scenario, where $A' = rA$ and $B' = r^{-1}(B - U)$, and a protocol in the (A'' private online, B'' private offline) input scenario, where $A'' = A$ and $B'' = U$. Finally, A computes the desired result $e(A, B)$ as $e(A', B')/(e(A'', B''))$.

Protocol Π_2. This protocol combines known input randomization techniques used, for instance, in [8, 18], with pairing delegation protocols for different input scenarios in [10]. In the offline phase, C randomly chooses points R_a, R_b and computes $e(R_a, R_b)$. In the online phase, C sets $A_r = A + R_a, B_r = B + R_b$ and sends A_r, B_r to S, which computes and sends $e(A_r, B_r)$ to C. Then, C uses the state-of-the-art protocol in [10] for the input scenario (A' private offline, B' private online), where $A' = R_a, B' = B$, to delegate the computation of $e(R_a, B)$, and the state-of-the-art protocol in [10] for the input scenario (A'' private online, B'' private offline), where $A'' = A, B'' = R_b$, to delegate the computation of $e(A, R_b)$. Finally, C computes the desired result $e(A, B)$ as $e(A_r, B_r)/(e(R_a, B) \cdot e(A, R_b) \cdot e(R_a, R_b))$.

Protocol Π_3. This protocol is a variant of protocol Π_1 where the subprotocol for the (A' public online, B' public online) is realized using our improved protocol in Section 3.

Our final and most efficient protocol. Note that all of the previously discussed 4 protocols combine some input randomization technique with one or more subprotocols for different input scenarios. We observe that there is some inherent inefficiency in such approaches, as the subprotocols typically compute further input randomizations. In our final protocol, we bypass inefficiencies due to such compositions, and design a new probabilistic test, tailored to this specific input scenario, which involves computation of 4 pairings from S, all with efficiently masked variants of inputs A, B. Masks are chosen carefully so to achieve various cancellation effects and to allow efficient verification from C, while especially reducing the use of exponentiations in $\mathcal{G}_T$ to only 1 exponentiation to a short, λ-bit, exponent, as well as scalar multiplications in $\mathcal{G}_2$, to only 1.

A formal description follows.

Offline Input to C and S: $1^\sigma, 1^\lambda$, $desc(e)$

Offline phase instructions:

1. C randomly chooses $U_0, U_1 \in \mathcal{G}_1$, $P_0, P_1 \in \mathcal{G}_2$, $c \in \{1, \ldots, 2^\lambda\}$, $r_0, r_1, r_2 \in \mathbb{Z}_l^*$
2. C sets
   - $v_i := e(U_i, P_i)$, $Q_i := \hat{r}_i \cdot P_i$ where $\hat{r}_i = r_i^{-1} \mod l$, for $i = 0, 1$
   - $\hat{r}_2 := r_2^{-1}$, $Q_{2,1} = -r_2 \cdot P_0$ and $Q_{3,1} = r_2 \cdot P_1$
3. C sets $ov = (c, r_0, r_1, r_2, \hat{r}_2, U_0, U_1, P_0, P_1, Q_0, Q_1, Q_{2,1}, Q_{3,1}, v_0, v_1)$

Online Input to C: $A \in \mathcal{G}_1$, $B \in \mathcal{G}_2$, and ov

Online phase instructions:

1. C sets
   - $Z_0 := r_0(A - U_0)$, $Z_1 := r_1(A - U_1)$, $Z_2 := \hat{r}_2 \cdot A$ and
   - $Q_{2,0} = Q_{3,0} := r_2 \cdot B$, $Q_2 := Q_{2,0} + Q_{2,1}$, $Q_3 := c \cdot Q_{3,0} + Q_{3,1}$

   C sends $Z_0, Z_1, Z_2, Q_0, Q_1, Q_2, Q_3$ to S
2. S computes

   $$w_0 := e(Z_0, Q_0),\ w_1 := e(Z_1, Q_1),\ w_2 := e(Z_2, Q_2),\ w_3 := e(Z_2, Q_3)$$

   S sends w_0, w_1, w_2, w_3 to C
3. (Membership Test:) C checks that $w_0, w_1, w_2 \in \mathcal{G}_T$
   C computes $y = w_0 \cdot w_2 \cdot v_0$
   (Probabilistic Test:) C checks that $w_3 = (y)^c \cdot w_1 \cdot v_1$
   (with this test, C implicitly checks that $w_3 \in \mathcal{G}_T$)
   If any of these tests fails, C **returns** $\perp$ and the protocol halts
   C **returns** y

Properties of the protocol (C, S)**:** *The efficiency properties* are verified by protocol inspection. In particular, as for C's online runtime, note that the probabilistic test requires 1 exponentiation in $\mathcal{G}_T$ to a short, λ-bit, exponent; the computation of Z_0, Z_1, Z_2 requires 3 scalar multiplications in $\mathcal{G}_1$; the computation of Q_0, Q_1, Q_2, Q_3 requires 2 multiplications in $\mathcal{G}_2$, of which 1 is by a short, λ-bit, scalar. Other parts of these computations involve lower-order operations, such as 4 additions/subtractions in $\mathcal{G}_1$ or $\mathcal{G}_2$, 4 multiplications and a small number of Frobenius-based simplifications in $\mathcal{G}_T$. In Table 3 we compare our main protocol with protocols $\Pi_0, \ldots, \Pi_3$ as well as previous work. There, the estimated ratios of C's online runtime to a non-delegated pairing calculation is shown to range between 0.425 and 0.843 depending on the curve used.

Table 3. Protocols comparison in the input scenario with private A and B. (The expression of t_C only includes higher-order functions e_T, m_1, m_2. The estimated ratio t_C/t_F also counts terms based on functions a_1, a_2, m_T, t_M and uses $\lambda = 128$.)

Protocols	t_C	**Ratio:** t_C/t_F			
		BN-12	**BLS-12**	**KSS-18**	**BLS-24**
		$\sigma = 461$	$\sigma = 635$	$\sigma = 508$	$\sigma = 629$
[8] §4.1	$5\, e_T(\sigma) + m_2(\sigma)$	2.606	2.182	1.453	2.337
[18] §3	$3\, e_T(\sigma) + m_2(\sigma) + m_1(\sigma)$	1.719	1.439	0.956	1.517
[7] §5.1	$2\, e_T(\sigma) + 2\, m_2(\sigma) + 2\, m_1(\sigma)$	1.658	1.391	0.917	1.390
[10] Π_1	$3\, e_T(\lambda) + m_2(\sigma) + m_2(\lambda)$ $+3\, m_1(\sigma) + 2\, m_1(\lambda)$	1.161	0.823	0.578	0.697
This paper: Π_0	$e_T(\sigma) + e_T(\lambda) + m_2(\sigma)$ $+m_2(\lambda) + 2\, m_1(\sigma)$	1.155	0.911	0.617	0.874
This paper: Π_2	$3\, e_T(\lambda) + m_2(\sigma) + 2\, m_2(\lambda)$ $+2\, m_1(\sigma) + m_1(\lambda)$	1.072	0.760	0.550	0.694
This paper: Π_3	$2\, e_T(\lambda) + m_2(\sigma) + 2\, m_2(\lambda)$ $+1\, m_1(\sigma) + m_1(\lambda)$	1.002	0.729	0.502	0.604
This paper §4	$e_T(\lambda) + m_2(\sigma)$ $+m_2(\lambda) + 3\, m_1(\sigma)$	0.843	0.635	0.425	0.511

The *correctness* property follows by showing that if C and S follow the protocol, C always outputs $y = e(A, B)$. We first show that if C returns y, then this returned value y is the correct output $e(A, B)$. This is proved as follows:

$$\begin{aligned}
y = w_0 \cdot w_2 \cdot v_0 &= e(Z_0, Q_0) \cdot e(Z_2, Q_2) \cdot e(U_0, P_0) \\
&= e(r_0(A - U_0), r_0^{-1} P_0) \cdot e(r_2^{-1} A, r_2(B - P_0)) \cdot e(U_0, P_0) \\
&= e(A - U_0, P_0) \cdot e(A, B - P_0) \cdot e(U_0, P_0) \\
&= e(A, P_0) \cdot e(U_0, P_0)^{-1} \cdot e(A, B) \cdot e(A, P_0)^{-1} \cdot e(U_0, P_0) \ = \ e(A, B).
\end{aligned}$$

We now show that the 2 tests performed by C are always passed. The membership test is always passed by the pairing definition. To see that the probabilistic test is always passed, first note that $Q_2 = Q_{2,0} + Q_{2,1} = r_2(B - P_0)$, and $Q_3 = Q_{3,0} + Q_{3,1} = r_2(c \cdot B + P_1)$. Then we have that

$$\begin{aligned}
y^c \cdot w_1 \cdot v_1 &= e(A, B)^c \cdot e(Z_1, Q_1) \cdot e(U_1, P_1) \\
&= e(A, B)^c \cdot e(r_1(A - U_1), r_1^{-1} \cdot P_1) \cdot e(U_1, P_1) \\
&= e(A, B)^c \cdot e(A - U_1, P_1) \cdot e(U_1, P_1) \\
&= e(A, c \cdot B) \cdot e(A, P_1) \cdot e(U_1, P_1)^{-1} \cdot e(U_1, P_1) \\
&= e(r_2^{-1} \cdot A, r_2(c \cdot B + P_1)) = e(Z_2, Q_3) = w_3
\end{aligned}$$

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

The *privacy* property of the protocol against any malicious S follows by observing that C's only message $(Z_0, Z_1, Z_2, Q_0, Q_1, Q_2, Q_3)$ to S does not leak any information about C's inputs A, B. To prove this, we show the stronger property (also used later to show the security property of (C, S)) that for any A, B, c, the message sent by C in the protocol is a uniformly distributed 7-tuple in $\mathcal{G}_1^3 \times \mathcal{G}_2^4$. Towards proving this latter property, recall that $\mathcal{G}_1, \mathcal{G}_2$ are cyclic groups, and denoting as G_1 a generator for $\mathcal{G}_1$, and as G_2 a generator for $\mathcal{G}_2$, we can denote as $a, b, u, z_0, z_1, z_2, p_0, p_1, q_0, q_1, q_2, q_3$ the values in $\mathbb{Z}_l$ such that

- $A = aG_1, B = bG_2$
- $P_0 = p_0 G_2, P_1 = p_1 G_2$
- $Z_0 = z_0 G_1, Z_1 = z_1 G_1, Z_2 = z_2 G_1$
- $Q_0 = q_0 G_2, Q_1 = q_1 G_2, Q_2 = q_2 G_3, Q_3 = q_3 G_3$.

Note that C's message can be then written as

$$(Z_0, Z_1, Z_2, Q_0, Q_1, Q_2, Q_3) = (z_0 G_1, z_1 G_1, z_2 G_2, q_0 G_2, q_1 G_2, q_2 G_3, q_3 G_3),$$

and, since $\mathcal{G}_1, \mathcal{G}_2$ are cyclic groups of prime order l, to prove the above stronger property, it suffices to show that the 7-tuple $(z_0, z_1, z_2, q_0, q_1, q_2, q_3)$ is uniformly distributed in $(\mathbb{Z}_l^*)^3 \times \mathbb{Z}_l^4$, for all $a, b \in \mathbb{Z}_l$ and any $c \in \{0, \ldots, 2^\lambda\}$. The latter is proved as follows:

- z_0 is uniformly distributed in $\mathbb{Z}_l$ since so is u_0 and $z_0 = r_0(a - u_0) \mod l$;

– z_1 is uniformly distributed in $\mathbb{Z}_l$, even conditioned on z_0, since so is u_1 and $z_1 = r_1(a - u_1) \mod l$;
– z_2 is $= 0$ if $a = 0$ or else is uniformly distributed in $\mathbb{Z}_l^*$, even conditioned on z_0, z_1, since so is r_2 and $z_2 = r_2^{-1}a \mod l$;
– q_0 is $= 0$ if $p_0 = 0$ or else is uniformly distributed in $\mathbb{Z}_l^*$, even conditioned on z_0, z_1, z_2, since so is r_0 and $q_0 = r_0^{-1}p_0 \mod l$
– q_1 is $= 0$ if $p_1 = 0$ or else is uniformly distributed in $\mathbb{Z}_l^*$, even conditioned on q_0, z_0, z_1, z_2, since so is r_1 and $q_1 = r_1^{-1}p_1 \mod l$;
– q_2 is uniformly distributed in $\mathbb{Z}_l$, even conditioned on q_0, q_1, z_2, z_0, z_1, since so is p_0 and $q_2 = r_2(b - p_0) \mod l$;
– q_3 is uniformly distributed in $\mathbb{Z}_l$, even conditioned on $q_0, q_1, q_2, z_2, z_0, z_1$, since so is p_1 and $q_3 = r_2(cb + p_1) \mod l$.

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 $\epsilon_s \leq 2^{-\lambda}$ as a consequence of the following 3 facts, which we later prove:

1. the tuple $(Z_0, Z_1, Z_2, Q_0, Q_1, Q_2, Q_3)$ leaks no information about c to S;
2. for any S's message (w_0, w_1, w_2, w_3) different than what would be returned according to the protocol instructions, there is only one value of c for which (w_0, w_1, w_2, w_3) satisfies both the membership and the probabilistic test in step 3 of the protocol;
3. for any S's message (w_0, w_1, w_2, w_3) different than what would be returned according to the protocol instructions, the probability that (w_0, w_1, w_2, w_3) satisfies the probabilistic test in step 3 of the protocol is $\leq 2^{-\lambda}$.

The proof of Fact 1 follows from the stronger property established when proving the protocol's privacy property.

Towards proving Fact 2, let (w_0, w_1, w_2, w_3) be the values that would be returned by S according to the protocol, and assume a malicious algorithm Adv, corrupting S returns a different pair (w_0', w_1', w_2', w_3'). Because $\mathcal{G}_T$ is cyclic, we can consider a generator g for $\mathcal{G}_T$ and write $w_i = g^{a_i}$, for $i = 0, 1, 2, 3$. Note that if the membership and probabilistic tests are satisfied, all values in (w_0', w_1', w_2', w_3') are verified to be in $\mathcal{G}_T$. Then we can write

$$\forall\, i = 0, 1, 2, 3,\ \exists\, u_i \in \mathbb{Z}_l \quad w_i' = g^{u_i} \cdot w_i \text{ such that for some } u_i \neq 0.$$

Now, assume wlog that $u_i \neq 0 \mod l$ and consider the following equivalent re-writings of the probabilistic test, via variable substitutions and simplifications:

$$\begin{aligned}
w_3' &= y^c \cdot w_1' \cdot v_1 \\
w_3' &= (w_0' \cdot w_2' \cdot v_0)^c \cdot w_1' \cdot v_1 \\
g^{u_3} \cdot w_3 &= (g^{u_0} \cdot w_0 \cdot g^{u_2} \cdot w_2 \cdot v_0)^c \cdot g^{u_1} \cdot w_1 \cdot v_1 \\
g^{u_3} \cdot w_3 &= g^{c(u_0+u_2)+u_1} \cdot (w_0 \cdot w_2 \cdot v_0)^c \cdot w_1 \cdot v_1 \\
g^{u_3} \cdot w_3 &= g^{c(u_0+u_2)+u_1} \cdot y^c \cdot w_1 \cdot v_1
\end{aligned}$$

$$g^{u_3} = g^{c(u_0+u_2)+u_1}$$
$$u_3 = c(u_0+u_2)+u_1 \mod l,$$

where the 4th equality follows since $w_3 = y^c \cdot w_1 \cdot v_1$. Now, if there exist two distinct c_1 and c_2 such that

$$u_3 = c_1(u_0+u_2)+u_1 \mod l \text{ and } u_3 = c_2(u_0+u_2)+u_1 \mod l$$

then $(u_0+u_2)(c_1-c_2) = 0 \mod l$ which implies either $c_1 - c_2 = 0 \mod l$ (i.e. $c_1 = c_2$ and c is unique) or $u_0 + u_2 = 0 \mod l$ (i.e. $u_0 = -u_2 \mod l$). If $u_0 = -u_2 \mod l$ then $y = w_0' \cdot w_2' \cdot v_0 = g^{u_0} \cdot w_0 \cdot g^{u_1} \cdot w_2 \cdot v_0 = w_0 \cdot w_2 \cdot v_0 = e(A,B)$, and thus S is honest. If S is not honest then $c_1 = c_2$ and this proves Fact 2.

Towards proving Fact 3, note that, by Fact 1, no information about c is leaked by C's message $(Z_0, Z_1, Z_2, Q_0, Q_1, Q_2, Q_3)$. This implies that all values in $\{1, \ldots, 2^\lambda\}$ are still equally likely for c even when conditioning over C's message. Then, by using Fact 2, the probability that S's message (w_0, w_1, w_2, w_3) satisfies the probabilistic test, is 1 divided by the number 2^λ of values of c that are still equally likely even after conditioning over C's message. This proves Fact 3.

5 Conclusions

Pairings are important primitive operations in many public-key cryptosystems and, more generally, cryptographic protocols (see, e.g., [16, 4, 1, 3, 5, 14, 19]). In this paper we studied pairing delegation to a single, possibly malicious, server, in the input scenario where both inputs are not available until the online phase. We proposed new protocols in the scenario where (a) both inputs are publicly available; and (b) both inputs are known to the client but should remain private from the server. We showed the first protocol in case (b) which improves the client's online runtime with respect to non-delegated computation for all 4 practical curves for which pairing benchmark runtimes are reported in [6]. Both our protocols only require client and server to communicate a constant number (specifically, 6 and 11) of group elements. Indeed, exchanged communication is another important performance metric when analyzing the efficiency of security protocols (see, e.g., [20]). More generally, when implementing such protocols in practical applications, an overall analysis of the device's energy expenditure should at least take into account device characteristics and usage patterns in the intended application, together with protocol runtime and communication.

References

1. S.S. Al-Riyami, K.G. Paterson, *Certificateless Public Key Cryptography.* In: Laih C.S. (eds) Advances in Cryptology - ASIACRYPT 2003.
2. 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) Progress in Cryptology – LATINCRYPT 2015.

3. D. Boneh, M. Franklin, *Identity-based Encryption from the Weil Pairing.* In: Proc. of CRYPTO 2001. LNCS vol. 2139, Springer.
4. D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, *Public Key Encryption with Keyword Search.* In: Proc. of EUROCRYPT 2004. LNCS vol. 3027, Springer.
5. D. Boneh, B. Lynn, H. Shacham, *Short Signatures from the Weil Pairing.* In: Boyd C. (eds) Advances in Cryptology — ASIACRYPT 2001. LNCS vol 2248. Springer.
6. 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.
7. 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.
8. B. Chevallier-Mames, J.S. Coron, N. McCullagh, D. Naccache, M. Scott, *Secure delegation of elliptic-curve pairing. Cryptology ePrint Archive.* In: Proc. of CARDIS 2010. LNCS vol 6035. Springer. Also IACR EPrint 2005/150.
9. C. Chevalier, F. Laguillaumie, D. Vergnaud, *Privately Outsourcing Exponentiation to a Single Server: Cryptanalysis and Optimal Constructions.* In: Proc. of Computer Security – ESORICS 2016. LNCS vol. 9878. Springer.
10. 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.
11. 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.
12. M. Girault, D. Lefranc, *Server-aided verification: Theory and practice.* In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623.
13. Guillevic, A., Vergnaud, D., *Algorithms for outsourcing pairing computation.* In: Proc. of CARDIS 2014, LNCS vol. 8968. Springer.
14. F. Hess, *Efficient Identity Based Signature Schemes Based on Pairings.* In: Nyberg K., Heys H. (eds) Selected Areas in Cryptography. SAC 2002.
15. S. Hohenberger, A. Lysyanskaya, *How to securely outsource cryptographic computations.* In: Proc. of TCC 2005, pp. 264–282, Springer.
16. Antoine Joux, *A One Round Protocol for Tripartite Diffie-Hellman.* In: Proc. of ANTS 2000, pp. 385-394.
17. 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.
18. B.G. Kang, M.S. Lee, J.H. Park, *Efficient delegation of pairing computation.* In: IACR Cryptology ePrint Archive, n. 259, 2005.
19. J.K. Liu, M.H. Au, W. Susilo, *Self-generated-certificate publickey cryptography and certificateless signature/encryption schemein the standard model.* In: Proc. ACM Symp. on Information, Computer and Communications Security. ACM Press (2007).
20. C. Markantonakis, *Is the Performance of Smart Card Cryptographic Functions the Real Bottleneck?*, in Proc. of IFIP/SEC 2001: 77-92
21. M. Scott, *Unbalancing pairing-based key exchange protocols.* In: IACR Cryptology ePrint Archive, n. 688, 2013.
22. F. Vercauteren, *Optimal Pairings.* In: IEEE Transactions on Information Theory, vol. 56, no. 1, pp. 455–461, Jan. 2010.
23. D. Vergnaud, *Secure Outsourcing in Discrete-Logarithm-Based and Pairing-Based Cryptography.* In Proc. of WISTP 2018: 7-11.
24. A. Yao, *Protocols for secure computations.* In: Proc. of 23rd IEEE FOCS, pp. 160–168, 1982.