Verifiable Outsourcing of Pairing Computations

Thierry Mefenza[1,2] and Damien Vergnaud[3,4]

[1] DI/ENS, École normale supérieure, CNRS, PSL University
F-75005 Paris, France
[2] INRIA
[3] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6
Équipe Almasty, F-75005 Paris, France
[4] Institut Universitaire de France, Paris, France

Abstract. Since 2000, *pairings* have found many applications in cryptography. Pairing computation is very costly on limited devices such as smart cards but it could be made more efficient if the limited device may outsource some computation to a more computationally powerful (but potentially malicious) device. In this setting, it is important to ensure the limited device that the computation was carried out correctly. Known *verifiable* pairing delegation protocols still have a prohibitive cost for limited devices.
We study the question of how a limited device can delegate many pairing computations at once to a potentially malicious and powerful device. We propose four efficient batch pairing delegation protocols which are much more efficient than previous constructions (on the state-of-the art optimal Ate pairing on a Barreto-Naehrig curve). We notably propose the first batch pairing delegation protocols for variable and public left-side and right-side inputs for the pairing and we apply this scheme to improve the efficiency of batch verification of popular short signatures schemes (namely, Boneh-Lynn-Shacham and Pointcheval-Sanders signatures).

Keywords. Pairings; Verifiable outsourcing; Pairing delegation protocols; Batch pairing delegation protocols; Elliptic Curves; Optimal Ate pairing ; Short Signatures.

1 Introduction

We address the practical problem of speeding up pairing computations in cryptography using an untrusted computational resource. In particular, we introduce new efficient protocols for outsourcing pairing computations from a computationally limited device to an untrusted helper in such a way that the limited device has some insurance that the computation was carried out correctly.

Outsourcing Pairing Computation. Since their introduction in cryptography [20, 7], *pairings* have found many applications (for instance they are used to design efficient identity-based encryption schemes [7] or short signatures schemes [8]). A pairing is a bilinear, non-degenerate and (efficiently) computable

map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In practice, the first two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are prime-order ℓ subgroups (denoted additively) of the group of points $E(\mathbb{F}_q)$ of an elliptic curve E defined over a finite field $\mathbb{F}_q$ and the so-called *target* group $\mathbb{G}_T$ (denoted multiplicatively) is the order ℓ subgroup of a finite field extension $\mathbb{F}_{q^k}$.

Due to their importance in cryptography and since pairings are resource consuming, a lot of work have been devoted to speed up their computation. In practice, pairings can indeed be computed on computationally constrained devices such as smart-cards and the pairing computation on such a device can be very slow. Nowadays a computationally limited device – that we will call the *client* – can be connected to a more powerful device – that we will call the *server* – (for example a smart-card can be connected to a phone or a phone connected to a computer). In this setting, the client could compute a pairing with the help of the server. The client will then interact with the server in a protocol and outsource some costly operations to the server.

We consider several scenario for the delegation of the pairing computation $e(P, Q)$ where $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$. We would like to satisfy two security notions. The first security notion called the *secrecy or privacy* is that during the protocol, the server should not learn any information about the secret input of the pairing (if P, Q or both are secret inputs known only by the client). The second notion called the *verifiability* is that the client should be able to detect any malicious server and reject any wrong value output by the server with overwhelming probability. Obviously, a delegation protocol that does not ensure verifiability may cause severe security problems (in particular if the pairing computation occurs in the verification algorithm of some authentication protocol).

In 2005, Hohenberger and Lysyanskaya [19] provided a formal security definition for securely outsourcing computations from a computationally limited device to untrusted helpers. They introduced protocols where the client outsources computation to two, possibly dishonest, servers that are physically separated (and do not communicate). Many verifiable pairing delegation protocols were proposed in this setting (see [28] and references therein for a recent survey). This separation of the two servers is actually a strong assumption hard to be met in practice and it is more realistic to consider scenarios where the client delegate some computation to a single server. Unfortunately, in the present state-of-the-art, the different proposals for verifiable pairing delegation to a single server are inefficient and it is even sometimes better in practice to directly embed the pairing computation inside the restricted device than using these solutions.

In computer science, a *direct sum problem* asks whether solving n independent copies of a given task requires n times the amount of resources needed to solve a single copy. This fundamental question has been studied in many cryptographic scenarios. Due to the inefficiency of the known protocols for delegation of a unique pairing, it is an interesting problem to propose efficient protocols when the client wants to compute several pairings at the same time. Improving prior work from the literature, we consider this *batch pairing delegation* scenario in this paper.

Prior work. Girault and Lefranc [17] proposed in 2005 the first pairing delegation protocol that achieves secrecy but not verifiability. In 2014, Guillevic and Vergnaud [18] proposed a more efficient scheme but their method increases communication complexity between the client and the server (and their scheme does not provide verifiability).

In 2005, Chevallier-Mames, Coron, McCullagh, Naccache and Scott [15, 16] introduced the security notions of verifiable pairing delegation protocol and proposed the first verifiable pairing delegation protocol. The drawback of their protocol is that it is more costly for the client than computing the pairing himself. Later in 2014, Canard, Devigne and Sanders [12] improved their construction and proposed a much more efficient verifiable delegation protocol. Canard, Devigne and Sanders showed that their construction is more efficient for the client than computing a pairing himself on the so-called KSS-18 curve [21]. Later, Guillevic and Vergnaud [18] showed that Canard, Devigne and Sanders protocol is actually less efficient than computing a pairing for the state of the art optimal Ate pairing on a Barreto-Naehrig curve [5].

In 2007, Tsang, Chow and Smith [27] introduced the security notion of batch pairing delegation protocols and propose the first batch pairing delegation protocols when the client wants to compute several pairings $e(P_i, Q_i)$ where $P_i \in \mathbb{G}_1$ and $Q_i \in \mathbb{G}_2$ for $i \in \{1, \ldots, n\}$. Their main protocol works only when one of the input is constant (*i.e.* when $Q_1 = \cdots = Q_n = Q \in \mathbb{G}_2$). Moreover, in the setting where the constant input is secret, their protocols use costly exponentiations in the target group of the pairing and is then more resource consuming for the client than computing himself the pairings independently.

Contributions of the paper. In this paper, we propose four new efficient batch pairing delegation protocols. In the case where the right-side input is constant and public and the left-side inputs are variable and public, we proposed a protocol similar to the one of Tsang, Chow and Smith [27] but which is much more efficient by using endomorphisms as it was done by Guillevic and Vergnaud [18] and small exponents. We estimate the efficiency of the proposed protocol on the optimal Ate pairing on a Barreto-Naehrig curve. For a 128-bit security, we obtain a 74% improvement for the client compared to independent computations of many pairings by the client while the protocol of Tsang, Chow and Smith gives a 5% improvement. In the case where the right-side input is constant and secret and the left-side inputs are public and variable, we proposed a more efficiently protocol than the one proposed by Tsang, Chow and Smith. Our constructed protocol uses less costly scalar multiplications in the inputs groups of the pairing rather than costly exponentiations in the target group. For a 128-bit security, we obtain a 52% improvement for the client compared to independent computations of many pairings by the client while the protocol of Tsang, Chow and Smith is more costly for the client than if he computes the pairings himself. Finally, in the setting where the left and right-side inputs are variable, Canard, Devigne and Sanders said that they do not see how it is possible to construct a batch pairing delegation protocol without costly exponentiations in the target group. We show

that it is possible to do so and we propose the first batch pairing delegation protocol in this setting. For a 128-bit security, we obtain a 40% improvement for the client compared to independent computations of many pairings by the client. We show how such a protocol can be used to increase the efficiency of batch signature verification of some popular short signatures schemes (Boneh-Lynn-Shacham [8] and Pointcheval-Sanders signatures scheme [25]).

2 Preliminaries

2.1 Batch Verification

Many of the computational tasks in cryptography are for the purpose of verifying some property: signatures, message authentication codes or identification schemes involve verification procedures that assure the verifier of the validity of the input. This property is usually defined by a set of mathematical equations and the verification process involves algebraic operations to check whether these relations holds. One approach formalized by Bellare, Garay and Rabin [6] is the use of batching for the purpose of speeding up several verifications. The idea, called the *small exponents test* is to use randomization to verify multiple equations simultaneously at the cost of a single equation verification. This technique has found various applications due to its low computation load and time at the verifier.

The soundness of this batching technique relies on the following celebrated lemma from Schwartz and Zippel [30, 26].

Lemma 1. *(Schwartz-Zippel [30, 26]) Let $\mathbb{F}$ be a finite field and $P(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of degree d. Let $r_1, \ldots r_n$ be chosen independently and uniformly at random from a subset S of $\mathbb{F}$. If P is a nonzero polynomial, then*

$$\Pr[P(r_1, \ldots r_n) = 0] \leq \frac{d}{\sharp S}.$$

2.2 Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be two additive cyclic groups of prime order ℓ and $\mathbb{G}_T$ be a multiplicative cyclic group of the same order ℓ. An *admissible bilinear map* or *pairing* is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which satisfies the following properties:

1. **Bilinearity:** for all $a, b \in \mathbb{F}_\ell$, $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, $e(aP, bQ) = e(P, Q)^{ab}$;
2. **Non-degeneracy:** for $P \neq 1_{\mathbb{G}_1}$, $Q \neq 1_{\mathbb{G}_2}$, we have $e(P, Q) \neq 1_{\mathbb{G}_T}$;
3. **Efficiency:** e is efficiently computable.

In practice, $\mathbb{G}_1$ is a subgroup of the group of rational points $E(\mathbb{F}_p)$ of an elliptic curve E over a prime finite field $\mathbb{F}_p$, $\mathbb{G}_2$ is a subgroup of the group of rational points $E(\mathbb{F}_{p^k})$ of the elliptic curve E defined over finite field extension $\mathbb{F}_{p^k}$ (where k is called the *embedding degree* and is the smallest integer such that ℓ divides $p^k - 1$) and $\mathbb{G}_T$ is a subgroup of $\mathbb{F}_{p^k}$. In practice, the value $e(P, Q)$ is computed

in two steps using Miller's algorithm [24]: in the first step, called the Miller loop, the algorithm outputs some value f and in the second step, called the final exponentiation, $e(P,Q)$ is obtained by computing $f^{\frac{p^k-1}{\ell}}$.

In this paper, for practical purposes we consider the Optimal Ate pairing on a Barreto-Naehrig Curve with embedding degree $k = 12$. These curves indeed provide one of the most efficient instantiations of bilinear maps in cryptography (but it is worth mentioning that our protocols are generic and can be used on all pairing instantiations and for instance on Barreto-Lynn-Scott curves [4])). Due to the recent attacks on the discrete logarithm problem in subgroups of the multiplicative group of finite field extension $\mathbb{F}_{p^k}$ (see [22] and references therein), to achieve a 128-bit security level, users have to use a prime number p of 461 bits (in order to have finite field extension size of 5534 bits) [2].

Arithmetic Efficiency on Barreto-Naehrig Curves. We recall some estimations provided by [18] using the Relic Library of Aranha [1] on a Barreto-Naehrig Curve at the 128-bit security level. As mentioned above, p has to be at least a 461-bit prime number due to the most efficient known attack to solve the discrete logarithm problem in $\mathbb{F}_{p^k}$. We will use these estimations in the forthcoming sections to compare the efficiency of our pairing delegation protocols with the existing protocols.

The average cost for a scalar multiplication with an exponent a of bit-length $\log_2 a$ in $\mathbb{G}_1$ is $10.7 \log_2 a$ multiplications[1] in $\mathbb{F}_p$ (*i.e.* $4933M_p$ if $\log_2 a = 461$ where M_p denotes the cost of one multiplication in the finite field $\mathbb{F}_p$), $25.7 \log_2 a$ for a scalar multiplication in $\mathbb{G}_2$ (*i.e.* $11848M_p$ if $\log_2 a = 461$) and $36 \log_2 a$ for an exponentiation in $\mathbb{G}_T$ (*i.e.* $16596M_p$ if $\log_2 a = 461$). The cost for an addition in $\mathbb{G}_1$ is $11M_p$, $29M_p$ for an addition in $\mathbb{G}_2$ and $54M_p$ for a multiplication in $\mathbb{G}_T$. The average cost for an optimal Ate pairing computation on a Barreto-Naehrig curve with $k = 12$ is $(102. \log_2 s + 80/3. \log_2 s + 8048)M_p$ (the total cost for the Miller loop is $(102. \log_2 s + 80/3. \log_2 s + 137)M_p$ and the total cost for the final exponentiation is $7911M_p$ see [18] for details), where $s = 6t + 2$ and p is a polynomial of degree 4 in t, namely $\log_2 s \approx \log_2 p/4$. Then for a 461-bit prime p, the average average cost for an optimal ate pairing computation on a Barreto-Naehrig curve with $k = 12$ is $22876M_p$. These estimates can be found in Tab. 3 (see Appendix A).

Boyd and Pavlovski [9] proposed efficient attacks on many batch verification protocols if the elements output by the potentially malicious server are not check to belong to the appropriate group. In our batch pairing delegation protocols, we have to verify that the elements output by the server belong to the target group $\mathbb{G}_T$ of the pairing. Otherwise, the efficient attacks proposed by Boyd and Pavlovski can be adapted to break our constructions. Recently, Barreto, Costello, Misoczki, Naehrig, Pereira and Zanon [3] proposed a solution to avoid subgroups attacks on some pairings used in practice in cryptography. Their solution gives

[1] This cost assumes that scalar multiplications are performed with a binary signed representation of a, see [18] for details.

a way to test group membership in $\mathbb{G}_T$ (see [3] for details and proofs). On a Barreto-Naehrig curve with $k = 12$, $\mathbb{G}_T$ is the cyclotomic subgroup of order $\Phi_k(p)$ in $\mathbb{F}^{\star}_{p^k}$, and it is then easy to check that an element $g \in \mathbb{G}_T$ (just check that $g^{p^4} \cdot g = g^{p^2}$). This membership check is almost free and costs one multiplication in $\mathbb{F}_{p^k}$.

2.3 Boneh-Lynn-Shacham and Pointcheval-Sanders Signatures

In this section we recall the Boneh, Lynn and Shacham signature scheme and Pointcheval and Sanders signature scheme that we use in the forthcoming sections for the applications of our proposed batch pairing delegation protocols.

Boneh-Lynn-Shacham signature scheme. This short signature scheme was proposed by Boneh, Lynn and Shacham [8]. This signature scheme is provably secure in the random oracle model under the so-called Computational Diffie-Hellman assumption. The scheme is defined by the four following algorithms

- **Setup.** The user generates three groups $\mathbb{G}_1$, $\mathbb{G}_2$ (with additive notation) and $\mathbb{G}_T$ (with multiplicative notation) of the same prime order ℓ having an efficiently computable pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_2$ is generated by some element P. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ be a collision-resistant hash function (modeled as a random oracle in the security reduction).
- **Key generation.** The user picks uniformly at random $x \in \mathbb{Z}_\ell$, computes $Q = xP$ (with the notation $xP = \underbrace{P + \cdots + P}_{x \text{ times}}$), sets $\mathrm{sk} = x$ and $\mathrm{pk} = Q$.
- **Signature generation.** Given a message $m \in \{0,1\}^*$ and the secret key $\mathrm{sk} = x$, the user computes the hash value $\mathcal{H}(m) \in \mathbb{G}_1$ and outputs the signature $\sigma = x\mathcal{H}(m)$.
- **Signature verification.** Given $\sigma \in \mathbb{G}_1$ and the public key $\mathrm{pk} = Q$, a verifier accepts it as a signature on $\mathcal{H}(m) \in \mathbb{G}_1$ if and only if:

$$e(\sigma, P) = e(\mathcal{H}(m), Q)$$

Pointcheval-Sanders signature scheme. Camenisch and Lysyanskaya [11] proposed a signature scheme provably secure in the standard model under the so-called LRSW assumption [23]. Their scheme can be used to construct efficient anonymous credential systems, efficient group signature and efficient identity-based scheme. One of the drawbacks of the Camenisch-Lysyanskaya signature scheme is its size linear in the number of messages to be signed and the use of (inefficient) symmetric pairing. In order to deal with this drawbacks, Pointcheval and Sanders [25] proposed a signature scheme which has the same features as Camenisch-Lysyanskaya signature scheme. Their signature scheme is provably secure in the standard model under the LRSW assumption and is defined by the four following algorithms

– **Setup.** The user generates three groups $\mathbb{G}_1, \mathbb{G}_2$ (with additive notation), $\mathbb{G}_T$ (with multiplicative notation) of prime order ℓ having an efficiently computable pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
– **Key generation.** The user picks uniformly at random $x, y \in \mathbb{Z}_\ell$, $P \in \mathbb{G}_2$ and computes $X = xP$, $Y = yP$ and sets $\mathsf{sk} = (x, y)$ and $\mathsf{pk} = (P, X, Y)$.
– **Signature generation.** Given a message $m \in \mathbb{Z}_p$ and the secret key $\mathsf{sk} = (x, y)$, the user chooses a random element $R \in \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$ and outputs the signature $\sigma = (R, (x + my)R)$.
– **Signature verification.** Given $\sigma = (R, S) \in \mathbb{G}_1^2$ and the public key $\mathsf{pk} = (P, X, Y)$, a verifier accepts it as a signature on $m \in \mathbb{Z}_\ell$ if and only if :

$$R \neq 1_{\mathbb{G}_1} \quad \text{and} \quad e(R, X + mY) = e(S, P).$$

3 Pairing Delegation Protocol – State-of-the-Art

3.1 Definitions

As mentioned in the introduction, a secure computation outsourcing scheme mainly addresses two issues: the privacy of the inputs/output of the outsourced computational problem and the validity of the returned results. Following [19, 29], we briefly recall the definition of a secure outsourcing scheme as a 4-tuple $(\mathcal{T}, \mathcal{S}, \mathcal{R}, \mathcal{V})$ consisting of four different probabilistic algorithms:

1. **Problem Transformation** $\mathcal{T} : F(\mathbf{x}) \to G(\mathbf{y})$. The client locally transforms the problem $F(\mathbf{x})$ to a new form $G(\mathbf{y})$, where $\mathbf{y}$ is the new input and G is the new problem description. The client then outsources $G(\mathbf{y})$ to the server.
2. **Server Computation** $\mathcal{S} : G(\mathbf{y}) \to (\Omega, \Gamma)$. The server solves the transformed problem $G(\mathbf{y})$ to obtain the corresponding result Ω. At the same time, S returns Γ that is a proof of the validity of the result.
3. **Result Recovery** $\mathcal{R} : \Omega \to \omega$. Based on the returned result Ω, the client recovers the result ω of the original problem $F(\mathbf{x})$.
4. **Result Verification** $\mathcal{V} : (\Omega, \Gamma, \omega) \to \top = \{\mathsf{True}, \mathsf{False}\}$. Based on ω, Ω and the proof Γ, the client verifies the validity of the result.

The following requirements are desirable properties for secure outsourcing schemes:

1. **Correctness**: given that the server is honest, the client can successfully recover the correct result ω from the returned result Ω. That is $\mathcal{R}(\Omega) = \omega$.
2. **Privacy**: the server is unable to derive any key information about the original input $\mathbf{x}$ and output ω from the transformed problem G, the new input $\mathbf{y}$ and the new output Ω.
3. **Verifiability**: an honest client interacting with a dishonest server (which would like the client to accept a wrong value ω) will output an error message $\mathcal{V}(\Omega, \Gamma, \omega) = \mathsf{False}$ with overwhelming probability.

In this paper, our main goal is to achieve verifiability and to measure the performance of such a scheme, we adopt the definitions proposed in [19]:

Definition 1 (α-efficient). *Suppose the running time of a task F for the client is t_0 and the running time of local processing $(\mathcal{T}, \mathcal{R}, \mathcal{V})$ for the client is t_p. Then the outsourcing scheme is α-efficient if $\frac{t_0}{t_p} \geq \alpha$.*

Definition 2 (β-verifiable). *Given the returned output Ω and the proof Γ, denote the probability that the client is able to verify the validity of the result ω as ρ. Then the outsourcing scheme is β-verifiable if $\rho \geq \beta$.*

From these definitions, we can see that a larger α indicates a better performance of a secure outsourcing scheme, while a larger β means a better verifiability.

3.2 Tsang-Chow-Smith Batch Pairing Delegation Protocol

In this section we recall the batch pairing delegation protocol proposed by Tsang, Chow and Smith [27] presented for the sake of simplicity as Algorithm 1. They proposed a protocol for a client who wants to delegate n pairing computations $e(P_i, A)$ for $i \in \{1, \ldots, n\}$ for variable and secret points $P_i's$ and a constant and secret point A.

Algorithm 1: Tsang-Chow-Smith batch pairing delegation protocol

Input: n secret and variable points $P_i's$ and a secret point A
Output: $e(P_1, A), \ldots, e(P_n, A)$

1 The client selects a random point $\tilde{Q}$, a random element $r_A \in \mathbb{Z}_\ell$ and precomputes $\tilde{A} = r_A A$ and $\chi = e(\tilde{Q}, \tilde{A})$
2 The client selects n random elements $r_1, \ldots, r_n \in \mathbb{Z}_\ell$ and computes $\tilde{P}_i = r_i P_i$, for $i \in \{1, \ldots, n\}$
3 The client selects n random elements $b_1, \ldots, b_n \in \mathbb{Z}_\ell$ and computes $\tilde{P}_0 = \tilde{Q} + \sum_{i=1}^{n} b_i \tilde{P}_i$
4 The client sends $\tilde{P}_0, \ldots, \tilde{P}_n$ and $\tilde{A}$ to the server
5 The server computes $\alpha_i = e(\tilde{P}_i, \tilde{A})$, for $i \in \{0, \ldots, n\}$
6 Verification:
7 The client verifies that $\alpha_0, \ldots, \alpha_n \in \mathbb{G}_T$
8 The client computes $\alpha' = \chi \cdot \prod_{i=1}^{n} (\alpha_i)^{b_i}$
9 **if** $\alpha_0 = \alpha'$ **then**
10 outputs $(\alpha_1^{\frac{1}{r_A r_1}}, \ldots, \alpha_n^{\frac{1}{r_A r_n}})$
11 **else**
12 outputs $\perp$ and halts

With the formalism from the previous section, the Problem Transformation $\mathcal{T}$ corresponds to step 1-4, the Server Computation corresponds $\mathcal{S}$ corresponds to step 5, the Result Verification $\mathcal{V}$ corresponds to step 8-9 and the Result Recovery $\mathcal{R}$) corresponds to step 10.

We now analyze the overall computational cost for the client with precomputations in Algorithm 1 with 128-bit security on the Optimal Ate pairing

on Barreto-Naehrig with 461-bit primes p and ℓ. Our analysis is done for public and variable points $P_i's$ and public and constant point A and for public and variable points $P_i's$ and secret and constant point A:

- For public and variable points $P_i's$ and public and constant point A, Tsang, Chow and Smith proposed Algorithm 1 with $r_1 = \cdots = r_n = 1$ and $r_A = 1$. The overall computational cost for the client is $(461 \times 10.7n + 11n + 54(n + 1) + 461 \times 36n + 54n)M_p = (21647.7n + 54)M_p$ (corresponding in step 3 to n scalar multiplications with exponents of bit-length at most 461 and n point additions in $\mathbb{G}_1$, in step 7 to $n + 1$ membership check in $\mathbb{G}_T$ and in step 8 to n exponentiations with exponents of bit-length at most 461 and n multiplications in $\mathbb{G}_T$).
  On a Barreto-Naehrig curve with a 461-bit prime, the overall cost for the client to compute the n pairings in the naive way is $22876nM_p$. The computational ratio of the protocol is $(21647.7n + 54)/(22876n) \approx 0.95$ (or equivalently the scheme is 1.06-efficient).
- For secret and variable points $P_i's$ and secret and constant point A, Tsang, Chow and Smith proposed Algorithm 1 with $r_1 = \cdots = r_n = 1$. The overall computational cost for the client is $(461 \times 10.7n + 461 \times 10.7n + 11n + 54(n + 1) + 461 \times 36n + 54n + 461 \times 36n)M_p = (43122.4n + 54)M_p$ (corresponding in step 2 to n scalar multiplications with exponents of bit-length at most 461, in step 3 to n scalar multiplications with exponents of bit-length at most 461 and n point additions in $\mathbb{G}_1$, in step 7 to $n + 1$ membership check in $\mathbb{G}_T$, in step 8 to n exponentiations with exponents of bit-length at most 461 and n multiplications in $\mathbb{G}_T$, and in step 10 to n exponentiations with exponents of bit-length at most 461 in $\mathbb{G}_T$). In this case, the protocol is more costly than computing the pairings directly for the client and the computational ratio of the protocol in this setting is $(43122.4n + 54)/(22876n) \approx 1.89$.

4 Batch Pairing Delegation Protocols

It seems difficult to provide verifiable pairing delegation protocols without expensive exponentiation in $\mathbb{G}_T$ by the client. A solution is to allow the client to delegate many pairing computations at once with lower costs than delegating independently each pairing computation. We provide four verifiable pairing delegation protocols that allow the client to compute n pairings $e(P_1, Q_1), \ldots, e(P_n, Q_n)$, where the points P_i, Q_i are variable and public or secret.

4.1 Case where P_i, Q_i are public and $Q_1 = Q_2 = \cdots = Q_n = Q$

In this section, we suppose that the points $P_i's$ are variable and public and the points $Q_i's$ are constant, public. Our proposed batch pairing delegation protocol (Algorithm 2) is similar to the batch delegation protocol proposed in [27]. The difference with their protocol is that we use small coefficients in our scheme in order to deal with applications with specific verifiability and we use endomorphisms as it was done in [18, 13] for improving efficiency for large n.

Algorithm 2: Batch Pairing delegation with P_i, Q_i public and Q_i constant

Input: n public and variable P_i, Q_i and $Q_1 = Q_2 = \cdots = Q_n = Q$
Output: $e(P_1, Q_1), \ldots, e(P_n, Q_n)$

The client selects a random point P_0 and precomputes $\chi = e(P_0, Q)$
The client sends $P_1, \ldots, P_n$ and Q to the server
The server computes $\alpha_i = e(P_i, Q)$, for $i \in \{1, \ldots, n\}$
Verification:
The client selects n random elements $a_1, \ldots, a_n \in \{0, 1, \ldots, 2^t - 1\}$ (for some integer t)
The client computes $P'_i = \sigma_i(P_i)$, for an endomorphism $\sigma_i \in S$ and for $i \in \{1, \ldots, n\}$
The client computes $P = P_0 + \sum_{i=1}^{n} a_i P'_i$
The server computes $\alpha_0 = e(P, Q)$
The client verifies that $\alpha_0, \ldots, \alpha_n \in \mathbb{G}_T$
The client computes $\alpha' = \chi . \prod_{i=1}^{n} (\alpha_i^{\sigma_i})^{a_i}$
if $\alpha_0 = \alpha'$ **then**
    outputs $\alpha_1, \ldots, \alpha_n$
else
    outputs $\perp$ and halts

Remark 1. During the computation of α', $\alpha_i^{\sigma_i} = e(\sigma_i(P_i), Q)$ can be computed with a cheap endomorphism σ_i in $\mathbb{F}^{\star}_{p^k}$ and costs 8 multiplications in $\mathbb{F}_p$. S is the set of endomorphisms on $E(\mathbb{F}_p)$ and for the optimal Ate pairing on a Barreto-Naehrig curve, $S = \{Id, -Id, \phi, \phi^2, -\phi, -\phi^2\}$, where ϕ is the endomorphism computed from the complex multiplication on the curve. For $\sigma \in S$, the computation of the image of a point by σ is almost free and costs at most one multiplication and one subtraction in $\mathbb{F}_p$.

We obtain readily the following security theorem using the well-known techniques of batch verification [6, 10]:

Theorem 1. *The batch pairing delegation protocol described in Algorithm 2 is β-verifiable with $\beta = 1 - 1/(2^t \cdot \#S)$.*

We provide a sketch of proof in the following (but the complete proof is postponed to the full version of the paper).

Proof (Sketch). The small exponents $(a_1, \ldots, a_n)$ are information-theoretically hidden from the server and if the server sends correct values $\alpha_i = e(P_i, Q)$, we indeed have $\alpha_0 = \alpha'$ with $\alpha' = \chi . \prod_{i=1}^{n} (\alpha_i^{\sigma_i})^{a_i}$.

Let us assume that a malicious server sends to the client at least one value α_i such that $\alpha_i \neq e(P_i, Q)$ for some $i \in ?\{1, \ldots, n\}$. Since the client checks that the elements α_i actually belongs to $\mathbb{G}_T$, the probability that $\alpha' = \alpha_0$ with at least one wrong α_i is upper-bounded by $(2^{-t}/\#S)$ (by the Schwartz-Zippel Lemma and the soundness of the small-exponent verification test [6, 13]) and we obtain the claimed security. □

Table 1. Efficiency of our protocol

t	n	Security	Computational Cost	Ratio Cost	α	β
58	5	60	$14237M_p$	0.12	8.03	$1-2^{-60}$
78	10	80	$37760M_p$	0.16	6.06	$1-2^{-80}$
98	20	100	$94146M_p$	0.2	4.86	$1-2^{-100}$
118	50	120	$281984M_p$	0.24	4.06	$1-2^{-120}$
126	100	128	$601274M_p$	0.26	3.80	$1-2^{-128}$

We now analyze the overall computational cost for the client with precomputations in Algorithm 2. Step 6 costs at most nM_p (at most $1M_p$ for each σ_i), step 7 costs at most $(10.7tn + 11n)M_p$ (n scalar multiplications with exponents of bit-length at most t and n additions in $\mathbb{G}_1$), step 9 costs $54(n+1)M_p$(n+1 membership check in $\mathbb{G}_T$), step 10 costs $((8n+36nt)+54n)M_p$(n endomorphisms exponentiations, n exponentiations with exponents of bit-length at most t and n additions $\mathbb{G}_T$).

The overall cost for the client is therefore $46.7tn + 128n + 54$. Note that on a Barreto-Naehrig curve with a 461-bit prime, the overall cost for the client to compute the n pairings in the naive way is $22876nM_p$. The computational ratio is $(46.7tn + 128n + 54)(22876n)$. For 128-bit security (*i.e.* for $(1-2^{-128})$-verifiability), our computational cost is about 0.26 of the cost of a n pairings and our protocol is 3.84-efficient and is much better than the one proposed by Tsang, Chow and Smith whose computational cost is 0.95. In our scheme, the client sends $(n+2)$ points to the server and the server computes $(n+1)$ pairings. In Table 1, we summarize the computational cost for the client and the computational ratio for various values of n, t.

4.2 Case where P_i, Q_i are public and variable

In this section, we propose two batch pairing delegation protocols (Algorithm 3 and 4) when $P_i's$ and $Q_i's$ are variable and public. They are the first batch pairing delegation protocols where all inputs are variable. We analyze the efficiency of these protocols and compare them for a fixed β-verifiability (for instance 128-bit security, *i.e.* $\beta = 1-2^{-128}$) when the number n of pairings to be computed grows.

These two delegation protocols make use of a parameter t and we obtain readily as above the following security theorem (whose simple proof is postponed to the full version of this paper):

Theorem 2. *The batch pairing delegation protocols described in Algorithm 3 and 4 are β-verifiable with $\beta = 1-2^{-t}$.*

The overall computational cost for the client in Algorithm 3 is $(36tn^2 + 36.4tn + 108n^2 + 40n + 22782)M_p$. In fact, step 5 costs at most $54n^2M_p$ (n^2 membership check in $\mathbb{G}_T$), step 6 costs at most $(10.7tn + 11(n-1) + 25.7tn +$

Algorithm 3: Batch Pairing delegation with P_i, Q_i public and variable

Input: n public and variable P_i, Q_i
Output: $e(P_1, Q_1), \ldots, e(P_n, Q_n)$

The client sends $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_n$ to the server
The server computes $\alpha_{i,j} = e(P_i, Q_j)$, for $i, j \in \{1, \ldots, n\}$
Verification:
The client verifies that $\alpha_{i,j} \in \mathbb{G}_T$, for $i, j \in \{1, \ldots, n\}$
The client selects $2n$ random elements $a_1, b_1 \ldots, a_n, b_n \in \{0, 1, \ldots, 2^t - 1\}$
The client computes $P = \sum_{i=1}^{n} a_i P_i$ and $Q = \sum_{i=1}^{n} b_i Q_i$
The client computes $\alpha = e(P, Q)$
The client computes $\alpha' = \prod_{i,j=1}^{n} \alpha_{i,j}^{a_i b_j}$
if $\alpha = \alpha'$ **then**
    outputs $\alpha_{1,1}, \ldots, \alpha_{n,n}$
else
    outputs $\perp$ and halts

$29(n-1))M_p$ (n scalar multiplications with exponents of bit-length at most t and n additions $\mathbb{G}_1$ and $\mathbb{G}_2$), step 7 costs $22876M_p$ (one pairing computation), step 8 costs at most $(36tn^2 + 54(n^2 - 1))M_p$ (n^2 exponentiations with exponents of bit-length at most t and $n^2 - 1$ multiplications in $\mathbb{G}_T$). The computational ratio is $(36tn^2 + 36.4tn + 108n^2 + 40n + 22782)/(22876n)$. In Algorithm 3, the client does not have pre-computations to do and sends $2n$ points to the server and the server computes n^2 pairings.

Algorithm 4: Batch Pairing delegation with P_i, Q_i public and variable

Input: n public and variable P_i, Q_i
Output: $e(P_1, Q_1), \ldots, e(P_n, Q_n)$

The client selects n random elements $\lambda_1, \ldots, \lambda_n \in \{0, 1, \ldots, \ell - 1\}$ and a random point $R \in \mathbb{G}_2$
The client precomputes $R_i = \lambda_i R$ for $i \in \{1, \ldots, n\}$
The client selects n random elements $b_1, \ldots, b_n \in \{0, 1, \ldots, 2^t - 1\}$
The client computes $Q'_i = b_i Q_i + R_i$ for $i \in \{1, \ldots, n\}$
The client sends $P_1, \ldots, P_n$, $Q_1, \ldots, Q_n$ and $Q'_1, \ldots, Q'_n$ to the server
The server computes $\alpha_i = e(P_i, Q_i)$ and $\alpha'_i = e(P_i, Q'_i)$, for $i \in \{1, \ldots, n\}$
Verification:
The client verifies that $\alpha_i, \alpha'_i \in \mathbb{G}_T$, for $i \in \{1, \ldots, n\}$
The client computes $P = \sum_{i=1}^{n} \lambda_i P_i$
The client computes $\alpha = e(P, R)$
The client computes $\alpha' = \prod_{i=1}^{n} \alpha'_i$ and $\alpha'' = \prod_{i=1}^{n} \alpha_i^{b_i}$
if $\alpha' = \alpha'' \times \alpha$ **then**
    outputs $\alpha_1, \ldots, \alpha_n$
else
    outputs $\perp$ and halts

Table 2. Efficiency of Algorithms 3 and 4

t	n	Security	Cost Alg. 3	Cost Alg. 4	Ratio Alg. 3	Ratio Alg. 4
32	2	32	$30231M_p$	$37137.2M_p$	0.66	0.81
32	4	32	$47761.2M_p$	$51463.4M_p$	0.52	0.56
32	16	32	$364618.8M_p$	$137420M_p$	0.99	0.37
32	32	32	$1351575.6M_p$	$252030.2M_p$	1.84	0.34
64	2	64	$37169.2M_p$	$41086M_p$	0.81	0.89
64	4	64	$70852.4M_p$	$59361M_p$	0.77	0.64
64	16	64	$678167.6M_p$	$169011M_p$	1.85	0.46
64	32	64	$2568497.2M_p$	$315211M_p$	3.5	0.43
128	2	128	$51044.4M_p$	$48983.6M_p$	1.11	1.07
128	4	128	$117034.8M_p$	$75156.2M_p$	1.27	0.82
128	16	128	$1305265.2M_p$	$232191.8M_p$	3.56	0.63
128	32	128	$5002340.4M_p$	$441572.6M_p$	6.83	0.6

In Algorithm 4, step 4 costs at most $n(25.7t+29)M_p$ (n scalar multiplications with exponents of bit-length at most t and additions in $\mathbb{G}_2$), step 8 costs at most $108nM_p$ ($2n$ membership check in $\mathbb{G}_T$), step 9 costs at most $(10.7n\log_2\ell+11(n-1))M_p$ (n scalar multiplications with exponents of bit-length at most $\log_2\ell$ and $n-1$ additions in $\mathbb{G}_1$), step 10 costs $22876M_p$ (one pairing computation), step 11 costs at most $(54(n-1)+36nt+54(n-1))M_p$ ($2(n-1)$ multiplications in $\mathbb{G}_T$ and n exponentiations with exponents of bit-length at most t in $\mathbb{G}_T$), step 12 costs $54M_p$ (one multiplication in $\mathbb{G}_T$).

The overall computational cost for the client with pre-computations in Algorithm 4 is $61.7tn+5188.7n+22811$. In Algorithm 4, the client pre-computes n scalar multiplications in $\mathbb{G}_2$ and sends $3n$ points to the server and the server computes $2n$ pairings. In Table 2, we summarize the computational cost for the client and the computational ratio in Algorithms 3 and 4 for various values of n and t.

One can see that for β-verifiability with large $1-\beta$ and for small number of pairing computations, Algorithm 3 is more efficient than Algorithm 4. Algorithm 4 is more suitable for stronger security (for instance $t=128$) and for a fixed β-verifiability Algorithm 4 is much more efficient than the naive method as the number of pairing computations grows.

4.3 Case where $P_i's$ are secret and $Q_1=\cdots=Q_n=Q$ is secret

Tsang, Chow and Smith batch pairing delegation protocol is not efficient for public $P_i's$ and secret and constant Q (since the client has to compute n costly exponentiations in $\mathbb{G}_T$). In this section, we propose a more efficient batch pairing delegation protocol (see Algorithm 5) and we compare the efficiency of our protocol with Tsang, Chow and Smith batch pairing delegation protocol for 128-security bit and for various n.

In Algorithm 5 with the β-verifiability with $\beta=1-1/(6.2^t)$ and $t=126$, the overall computational cost for the client is $(461\times 10.7n+11n+46.7tn+$

Algorithm 5: Batch Pairing delegation with public and variable P_i and secret and constant Q

Input: n public and variable P_i,secret and constant Q
Output: $e(P_1, Q), \ldots, e(P_n, Q)$

1 The client chooses n random points $X_1, \ldots, X_n \in \mathbb{G}_1$, a random element $r \in \mathbb{Z}_\ell$ and precomputes $Q_0 = r^{-1}Q$, $\chi_i = e(X_i, Q_0)^{-1}$, for $i \in \{1, \ldots, n\}$
2 The client computes $P_i' = rP_i + X_i$, for $i \in \{1, \ldots, n\}$
3 The client uses Algorithm 2 to compute $\alpha_i = e(P_i', Q_0)$, for $i \in \{1, \ldots, n\}$
4 The client computes $e(P_i, Q)$ as $\alpha_i.\chi_i$, for $i \in \{1, \ldots, n\}$

$128n + 54 + 54n)M_p = (11009.9n + 54)M_p$ (corresponding in step 2 to n scalar multiplications with exponents of bit-length at most 461 and n point additions in $\mathbb{G}_1$, in step 3 to computations of n pairing using Algorithm 2, and in step 4 to n multiplications in $\mathbb{G}_T$). The computational ratio is $(11009.9n + 54)/(22876n)$ and is about 0.48. One can see that our batch pairing delegation protocol is more efficient than the one proposed by Tsang, Chow and Smith whose computational cost is 1.89.

We obtain again readily the following security theorem (whose simple proof is postponed to the full version of this paper):

Theorem 3. *The batch pairing delegation protocol described in Algorithm 5 is private and β-verifiable with $\beta = 1 - 1/(6.2^t)$.*

5 Applications

In this section we present some applications of our batch pairing delegation protocol (see Algorithm 4) for variable and public inputs to make the batch verification of Boneh-Lynn-Shacham and Pointcheval-Sanders signatures schemes more efficient

5.1 Batch Verification of Boneh-Lynn-Shacham Signatures

Batch verification of a signature is a technique to reduce the commputational cost for the verifier who wants to verify many signatures for an user at once or many signatures for many different users. In the Boneh-Lynn-Shacham signature scheme, the verifier has to compute two pairings for a signature verification and if it wants to verify independently n signatures σ_i, for $i \in \{1, \ldots, n\}$ for an user or for n different users (for some integer n), the verifier will need $2n$ pairing computations. This verification can be made more efficient with batch verification as it was shown by Camenisch, Hohenberger and Pedersen [10].

1. For one user with pairs of keys $(\mathrm{pk}, \mathrm{sk})$, the verifier will accept n signatures σ_i on n messages m_i, for $i \in \{1, \ldots, n\}$ if the following equality holds:

$$e\left(\sum_{i=1}^{n} a_i \sigma_i, P\right) = e\left(\sum_{i=1}^{n} a_i \mathcal{H}(m_i), \mathrm{pk}\right),$$

for some small and random integers a_i. The batch verification of the n signatures then needs 2 pairing computations rather than $2n$ (together with $2n$ scalar multiplications in $\mathbb{G}_1$ with small exponents).
Using our batch pairing delegation protocols Algorithms 3 and 4 for public and variable inputs, a verifier can further improve this batch verification by delegating the computation of these two pairings and saves using Table 2 34% of the computational cost (namely the verifier will compute 1.32 pairings rather than 2 pairings) for 32-bit security.

2. For n users with pairs of keys $(\mathrm{pk}_i, \mathrm{sk}_i)$, for $i \in \{1, \ldots, n\}$, the verifier will accept n signatures σ_i on the n messages m_i, for $i \in \{1, \ldots, n\}$ (where σ_i is the signature of the message m_i under the key $(\mathrm{pk}_i, \mathrm{sk}_i)$) if the following equality holds:

$$e\left(\sum_{i=1}^{n} a_i \sigma_i, P\right) = \prod_{i=1}^{n} e\left(a_i \mathcal{H}(m_i), \mathrm{pk}_i\right),$$

for some small and random integers a_i. The batch verification of the n signatures then needs $n+1$ pairing computations rather than $2n$ (together with $2n$ scalar multiplications in $\mathbb{G}_1$ with small exponents). A verifier can further improve this verification with Algorithm 4 and computes using Table 2 $0.43(n+1)$ for 64-bit security or $0.6(n+1)$ pairings for 128-bit security when n grows.

5.2 Batch Verification of Pointcheval-Sanders Signatures

In the Pointcheval-Sanders signature scheme, the verifier has to compute two pairings for a signature verification and if he wants to verify independently n signatures σ_i, for $i \in \{1, \ldots, n\}$ for an user or for n different users (for some integer n), he will need $2n$ pairing computations. This verification can be made more efficient with batch verification as follows:

1. For one user with pairs of keys $(\mathrm{pk} = (P, X, Y), \mathrm{sk})$, the verifier will accept n signatures $\sigma_i = (R_i, S_i)$ on n messages m_i, for $i \in \{1, \ldots, n\}$ if the following equality holds:

$$e\left(\sum_{i=1}^{n} \alpha_i R_i, X\right) e\left(\sum_{i=1}^{n} (\alpha_i m_i) R_i, Y\right) = e\left(\sum_{i=1}^{n} \alpha_i S_i, P\right),$$

for some small and random integers α_i. In addition to the scalar multiplications by the messages m_i, the verification of the n signatures then needs 3 pairing computations rather than $2n$ (together with $3n$ scalar multiplications in $\mathbb{G}_1$ with small exponents). The verifier can further improve this batch verification with Algorithms 3 and 4 and computes 1.62 pairings for 32-bit security.

2. For n users with pairs of keys $(\mathrm{pk}_i = (P, X_i, Y_i), \mathrm{sk}_i)$, for $i \in \{1, \ldots, n\}$, the verifier will accept n signatures σ_i on the n messages m_i, for $i \in \{1, \ldots, n\}$ if the following equality holds:

$$\prod_{i=1}^{n} e(\alpha_i R_i, X_i + m_i Y_i) = e\left(\sum_{i=1}^{n} \alpha_i S_i, P\right),$$

for some small and random integers α_i. In addition to the scalar multiplications by the messages m_i, the batch verification of the n signatures then needs $n + 1$ pairing computations rather than $2n$ (together with $3n$ scalar multiplications in $\mathbb{G}_1$ with small exponents). The verifier can further improve this batch verification with Algorithms 3 and 4 and computes using Table 2 $0.43(n+1)$ for 64-bit security or $0.6(n+1)$ pairings for 128-bit security when n grows.

6 Conclusion

In this paper, we proposed four efficient batch pairing delegation protocols in many settings which outperformed the previous proposals. We proposed the first batch pairing delegation protocol in the setting where the left and right-side inputs are variable and public and showed its applications in cryptography to improve the efficiency of batch verification signatures of popular short signatures schemes for instance Boneh-Lynn-Shacham and Pointcheval-Sanders signatures scheme. It remains an open question to construct a generic batch pairing delegation protocol for variable and secret left and right-side inputs. Another interesting open problem is to provide lower bounds on the efficiency of verifiable pairing delegation protocols in these various settings (as it was done in [14] for private delegation of group exponentiation).

References

1. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIbrary for Cryptography. http://code.google.com/p/relic-toolkit/ (September 2013)
2. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334 (2017), http://eprint.iacr.org/2017/334
3. Barreto, P.S.L.M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G.C.C.F., Zanon, G.: Subgroup security in pairing-based cryptography. In: Lauter, K.E., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 245–265. Springer, Heidelberg, Germany, Guadalajara, Mexico (Aug 23–26, 2015)
4. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg, Germany, Amalfi, Italy (Sep 12–13, 2003)
5. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg, Germany, Kingston, Ontario, Canada (Aug 11–12, 2006)
6. Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 170–191. Springer, Heidelberg, Germany, Campinas, Brazil (Apr 20–24, 1998)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (Sep 2004)
9. Boyd, C., Pavlovski, C.: Attacking and repairing batch verification schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 58–71. Springer, Heidelberg, Germany, Kyoto, Japan (Dec 3–7, 2000)
10. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. Journal of Cryptology 25(4), 723–747 (Oct 2012)
11. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004)
12. Canard, S., Devigne, J., Sanders, O.: Delegating a pairing can be both secure and efficient. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 14. LNCS, vol. 8479, pp. 549–565. Springer, Heidelberg, Germany, Lausanne, Switzerland (Jun 10–13, 2014)
13. Cheon, J.H., Lee, M.: Improved batch verification of signatures using generalized sparse exponents. Computer Standards & Interfaces 40, 42–52 (2015)
14. Chevalier, C., Laguillaumie, F., Vergnaud, D.: Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In: Askoxylakis, I.G., Ioannidis, S., Katsikas, S.K., Meadows, C.A. (eds.) ESORICS 2016, Part I. LNCS, vol. 9878, pp. 261–278. Springer, Heidelberg, Germany, Heraklion, Greece (Sep 26–30, 2016)
15. 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
16. Chevallier-Mames, B., Coron, J.S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. In: Gollmann, D., Lanet, J.L., Iguchi-Cartigny,

J. (eds.) Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010. pp. 24–35. Lecture Notes in Computer Science, Springer (2010)

17. Girault, M., Lefranc, D.: Server-aided verification: Theory and practice. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623. Springer, Heidelberg, Germany, Chennai, India (Dec 4–8, 2005)
18. Guillevic, A., Vergnaud, D.: Algorithms for outsourcing pairing computation. In: Joye, M., Moradi, A. (eds.) Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8968, pp. 193–211. Springer (2014)
19. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg, Germany, Cambridge, MA, USA (Feb 10–12, 2005)
20. Joux, A.: A one round protocol for tripartite diffie-hellman. In: Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings. pp. 385–394 (2000)
21. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In: Galbraith, S.D., Paterson, K.G. (eds.) PAIRING 2008. LNCS, vol. 5209, pp. 126–135. Springer, Heidelberg, Germany, Egham, UK (Sep 1–3, 2008)
22. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016)
23. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg, Germany, Kingston, Ontario, Canada (Aug 9–10, 1999)
24. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17(4), 235–261 (Sep 2004)
25. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 29 – Mar 4, 2016)
26. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4), 701–717 (1980)
27. Tsang, P.P., Chow, S.S.M., Smith, S.W.: Batch pairing delegation. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 07. LNCS, vol. 4752, pp. 74–90. Springer, Heidelberg, Germany, Nara, Japan (Oct 29–31, 2007)
28. Uzunkol, O., znur Kalkar, ?sa Sertkaya: Fully verifiable secure delegation of pairing computation: Cryptanalysis and an efficient construction. Cryptology ePrint Archive, Report 2017/1173 (2017), https://eprint.iacr.org/2017/1173
29. Zhou, K., Afifi, M.H., Ren, J.: Expsos: Secure and verifiable outsourcing of exponentiation operations for mobile cloud computing. IEEE Trans. Information Forensics and Security 12(11), 2518–2531 (2017), https://doi.org/10.1109/TIFS.2017.2710941
30. Zippel, R.: An explicit separation of relativised random polynomial time and relativised deterministic polynomial time. Inf. Process. Lett. 33(4), 207–212 (1989)

Table 3. Estimations for common operations and for a Barreto-Naehrig curve with $k = 12$ and $\log p = 461$ bits.

Operation	cost	total over $\mathbb{F}_p$
$\mathbb{F}_{p^k}$ arithmetic		
		M_p
M_{p^2}	$3M_p$	$3M_p$
S_{p^2}	$2M_p$	$2M_p$
M_{p^6}	$6M_{p^2}$	$18M_p$
S_{p^6}	$2M_{p^2} + 3S_{p^2}$	$12M_p$
$M_{p^{12}}$	$3M_{p^6}$	$54M_p$
$S_{p^{12}}$	$2M_{p^6}$	$36M_p$
$S_{\phi_{12}(p)}$	z^2, $z \in \mathbb{F}_{p^{12}}$, $\mathrm{Norm}(z) = 1$	$18M_p$
z^a, for any z, a	$\log a\ S_{p^{12}} + \log a\ /3\ M_{p^{12}}$	$54 \log a\ M_p$
z^a, $\mathrm{Norm}_{\mathbb{F}_{p^{12}}/\mathbb{F}_p}(z) = 1$	$\log a\ S_{\phi_{12}(p)} + \log a\ /3\ M_{p^{12}}$	$36 \log a\ M_p$
$\mathrm{Norm}_{\mathbb{F}_{p^{12}}/\mathbb{F}_p}(z)$, for any z	$\mathrm{Norm}_{\mathbb{F}_{p^{12}}/\mathbb{F}_{p^6}/\mathbb{F}_{p^2}/\mathbb{F}_p}(z)$	$59\ M_p$
z^r, $\mathrm{Norm}_{\mathbb{F}_{p^{12}}/\mathbb{F}_p}(z) = 1$	$z^p z^{1-t} = z^p (z^{p^6})^{t-1}$	$4616\ M_p$
check order$(z) = r$ in $\mathbb{F}_{p^k}$	$\mathrm{Norm}_{\mathbb{F}_{p^{12}}/\mathbb{F}_p}(z) = 1$; $z^r = 1$	$4675\ M_p$
$E(\mathbb{F}_p)$ arithmetic		
Doubling (Dbl_p)	$2M_p + 5S_p$	$7M_p$
Addition (Add_p)	$7M_p + 4S_p$	$11M_p$
Scalar mult. $[a]P$	$\log a$ Dbl $+ \log a$ /3Add	$10.7 \log a\ M_p$
$[a_1]P_1 + [a_2]P_2$	$\max(\log a_1, \log a_2)$ (Dbl $+$ 2/3Add)	$\max(\log a_1, \log a_2)$ $14.33M_p$
$E(\mathbb{F}_{p^2})$ arithmetic		
Doubling (Dbl_{p^2})	$2M_{p^2} + 5S_{p^2}$	$16M_p$
Addition (Add_{p^2})	$7M_{p^2} + 4S_{p^2}$	$29M_p$
Scalar mult. $[b]Q$	$\log b$ $\mathrm{Dbl}_{p^2} + \log b$ /3Add_{p^2}	$25.7 \log b\ M_p$
$[b_1]Q_1 + [b_2]Q_2$	$\max(\log b_1, \log b_2)$ (Dbl_{p^2} + 2/3Add_{p^2})	$\max(\log b_1, \log b_2)$ $35.33M_p$
Pairing on a Barreto-Naehrig curve with $\log_2 p = 461$		
Miller loop		$14965M_p$
Final powering		$7911M_p$
Pairing		$22876M_p$

A Estimations for Arithmetic and Pairing

Table 3 summarizes the estimates for arithmetic in the groups $\mathbb{G}_1$, $\mathbb{G}_1$, $\mathbb{G}_T$ and for the optimal ate pairing computation on a Barreto-Naehrig curve with $\log p = 461$ bits.