Delegating a Pairing Can Be Both Secure and Efficient

Sébastien Canard[1], Julien Devigne[1,2], and Olivier Sanders[1,3]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] UCBN, GREYC, Caen, France
[3] École normale supérieure, CNRS & INRIA, Paris, France

Abstract. Bilinear pairings have been widely used in cryptographic protocols since they provide very interesting functionalities in regard of identity based cryptography, short signatures or cryptographic tools with complex properties. Unfortunately their implementation on limited devices remains complex and even if a lot of work has been done on the subject, the current results in terms of computational complexity may still be prohibitive. This is clearly not for today to find the implementation of a bilinear pairing in every smart card. One possibility to avoid this problem of efficiency is to delegate the pairing computation to a third party. The result should clearly be both secure and efficient. Regarding security, the resulting computation of a pairing $e(A, B)$ by the third party should be verifiable by the smart card. Moreover, if the points A and/or B are secret at the beginning of the protocol, they should also be secret after its execution. Regarding efficiency, besides some specific cases, existing protocols for delegating a pairing are costlier than a true embedded computation inside the smart card. This is due to the fact that they require several exponentiations to check the validity of the result.

In this paper we first propose a formal security model for the delegation of pairings that fixes some weakness of the previous models. We also provide efficient ways to delegate the computation of a pairing $e(A, B)$, depending on the status of A and B. Our protocols enable the limited device to verify the value received from the third party with mostly one exponentiation and can be improved to also ensure secrecy of $e(A, B)$.

Keywords: pairings, secure delegation, elliptic curve.

1 Introduction

Pairings. Since the publication of the paper by Joux [23], elliptic-curve bilinear pairings have been frequently used in cryptography because they offer more functionalities than RSA groups while keeping a lower size. One of their most famous application was due to Boneh and Franklin [8] who used them to solve the open problem of constructing an efficient identity-based encryption scheme. Other known usefulness of pairings is their capacity to shorten signatures [9,10,6] and to obtain constructions in the standard model for cryptographic tools with complex properties (see *e.g.* [5]).

© Springer International Publishing Switzerland 2014

In the following, we more generally consider a bilinear environment, which corresponds to a set of two additive groups $\mathbb{G}_1$, $\mathbb{G}_2$ and one multiplicative group $\mathbb{G}_T$, all of known prime order l, along with an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The way to describe such bilinear map has first been proposed by Miller in his unpublished paper. This work was next improved in many other ones, especially regarding the way to efficiently compute such mathematical function. Unfortunately, despite several improvements (one can find some of them in [2,3]), the computational cost of a pairing still remains high and may be prohibitive for restricted devices such as smart cards or RFID tags/sensor nodes. This may be a problem in some practical applications where the properties of a pairing are very useful.

Delegation. One way to solve this problem is to delegate the pairing computation to a more powerful entity such as a mobile phone (which is now possible in an efficient way [14,28]), a computer or a server. But this cannot be done at the detriment of the security. Indeed, if a smart card may today be considered as secure, this does not remain true for a mobile phone, a computer or a server that may be controlled by some dishonest entity (by the way of *e.g.* a malware).

In the setting of the delegation of a pairing, two security properties could be taken into account. The first one is *secrecy*, requiring that the more powerful entity should not learn anything about the inputs of the pairings (unless they are public). The second one is *verifiability* (also called correctness) and requires that the restricted device cannot accept a wrong value for the pairing which is delegated to the more powerful entity. We emphasize that a delegation protocol that does not ensure verifiability may cause severe security problems. As evidence, if the pairing occurs in the verification algorithm of some digital signature scheme, as the ones from [9,7], an incorrect value may lead for the restricted device to accept an invalid signature.

Related Work. The delegation of cryptographic operations is a technique which has been known since a long time (see *e.g.* [27,15,26]) and it now exists some constructions in the generic case [32,13]. The particular case of the pairing computation has first been studied, as far as we know, in a work by Girault and Lefranc [20]. But they have only considered the secrecy of the computation. Verifiable protocols for delegating a pairing $e(A, B)$ have first been provided by Chevallier-Mames *et al.* ([16,17]) and later by Kang *et al.* [25]. However their efficiency depends on the status of A and B. More specifically, their protocols remain rather efficient as long as one of the involved points is constant. Indeed, the *online* phase of the proposal for *variable* A and B in [16] (resp. [25]) necessitates 5 (resp. 3) exponentiations in $\mathbb{G}_T$, 3 tests of membership in $\mathbb{G}_T$ (resp. 3), plus additional scalar multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$, for the case of secret values. The proposal in [16] for a verifiable (but for public variable values) pairing delegation requires 3 exponentiations in $\mathbb{G}_T$ and 3 tests of membership in $\mathbb{G}_T$. Even if the computational costs of a pairing and of group operations depend on the choice of the parameters, the recent results of [28], implementing an optimal-Ate pairing over Barreto-Naehrig curves [4] (the most popular choice), seem to confirm that, regarding efficiency, it is better to directly embed the pairing computation inside the restricted device than using these solutions. Otherwise, the main contribution will be the save of area that is required to implement a pairing in a smart card, which is certainly a good point, but not enough in most contexts.

In [31], the authors have considered the more general case of delegating several pairings all at once. They pointed out the lack of formal security models in the previous works and therefore proposed a candidate for it. They have also described a protocol fulfilling this model but which can only handle one variable point (the other one has to be constant). Unfortunately, as illustrated by the results of the previous works, the trickiest case is the one where both A and B are variable which is commonly found in the verification of signature schemes [9,7] where verifiability is particularly relevant.

Our Methodology. For variable A and B, both papers [16,25] make use of the same methodology. They started by providing a protocol for secret points A and B, and they then convert it into a protocol where A and/or B are publicly known. We argue that it is far more interesting to convert a protocol for public A and B into a protocol for secret A and B. Indeed, if we assume the existence of a verifiable delegation protocol $\mathcal{P}$ to compute the pairing $e(A, B)$ for public A and B, then the case of the protocol $\mathcal{P}'$ for secret A and B can be treated, as in [20], by simply computing $A' \leftarrow [u]A$ and $B' \leftarrow [v]B$ for random u and v, and then running $\mathcal{P}$ with A' and B'. From $e(A', B')$, it is then easy to recover $e(A, B) \leftarrow e(A', B')^{(uv)^{-1}}$, and A' and B' need not to be secret. Then, the conversion from $\mathcal{P}$ into $\mathcal{P}'$ mostly requires one additional exponentiation in $\mathbb{G}_T$. Moreover, the result is obviously secure.

Curiously, the opposite is not true. If we consider a secure (verifiable and secret) pairing delegation protocol $\mathcal{P}'$ for secret values, the execution of this protocol for public values of A and B is not necessarily true, in particular depending on the kind of verifiability fulfilled by $\mathcal{P}'$. As evidence, we show in Section 3, using the protocol in [25], that this may permit the adversary to break the verifiability property from the knowledge of the secret points. We consequently describe a complete security model for pairing delegation. Our model is close to the one of [31] but with some necessary modifications, especially for the secrecy property.

In this paper, we will then focus on the design of an efficient scheme for public variable points since it leads to a scheme for secret ones. We will also study a specific case where we can ensure both secrecy and verifiability with better efficiency than the previous generic conversion.

Organization of the Paper. Section 2 gives some preliminaries for our study and Section 3 provides a security model for the delegation of a pairing. We describe in Section 4 verifiable protocols for public variable A and B, which necessitate the low-power entity to only compute one exponentiation (in the most important group). We explain in Section 5 how to achieve secrecy using the above idea but also provide an improved protocol for a specific case. Eventually, our last section compares our work with related ones and gives an example of the expected gain for a specific family of curve.

2 Preliminaries

In this section we recall some necessary definitions that will be used throughout the document.

Some Notations. In the following, $\overset{\$}{\leftarrow}$ corresponds to a random choice, while $\leftarrow$ is used to indicate an assignment of a variable.

Bilinear Groups. Bilinear groups are a set of three groups $\mathbb{G}_1, \mathbb{G}_2$ (with additive notations, as for elliptic curves) and $\mathbb{G}_T$ (with multiplicative notation), all of prime order l, along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. for all $X_1 \in \mathbb{G}_1, X_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_l$ we have $e([a]X_1, [b]X_2) = e(X_1, X_2)^{ab}$;
2. for $X_1 \neq 1_{\mathbb{G}_1}$ and $X_2 \neq 1_{\mathbb{G}_2}$, $e(X_1, X_2) \neq 1_{\mathbb{G}_T}$;
3. e is efficiently computable;

where $1_{\mathbb{G}_1}$ (resp. $1_{\mathbb{G}_2}$ and $1_{\mathbb{G}_T}$) is the neutral element of the group $\mathbb{G}_1$ (resp. $\mathbb{G}_2$ and $\mathbb{G}_T$).

Elliptic Curve Pairings. Our first protocol (see section 4.1) works for any bilinear groups while the second one (see section 4.2) works only for pairing computations on elliptic curves (which is currently the common case) and thus requires some additional definitions. We refer to [19,18] for a more extensive background on pairings.

Let p be a prime number. Let $E(\mathbb{F}_p)$ be an elliptic curve over the field $\mathbb{F}_p$. We usually define $\mathbb{G}_1$ as a subgroup of $E(\mathbb{F}_p)$, $\mathbb{G}_2$ as a subgroup of $E(\mathbb{F}_{p^k})$ and $\mathbb{G}_T$ as a subgroup of $(\mathbb{F}_{p^k})^*$, where k, called the embedding degree, is the smallest integer such that l divides $p^k - 1$.

Remark 1. In cryptography, the family of Tate's algorithms [19] is most of the time used to compute a pairing. These algorithms are divided in two parts: the Miller loop and the final exponentiation. This last step makes use of the exponent defined as $c = \frac{p^k-1}{l}$. In the following, $\forall\ \alpha \in \mathbb{G}_T$, $\widetilde{\alpha}$ will denote an element of $(\mathbb{F}_{p^k})^*$ such that $\widetilde{\alpha}^c = \alpha$. Thus, considering that $\widetilde{\alpha}$ is the output of the Miller's algorithm, α corresponds to the expected pairing value.

Testing Membership in $\mathbb{G}_T$. As said in the introduction, existing works [16,25,31] necessitate to test whether a value α belongs to $\mathbb{G}_T$ or not. The simplest way to test this membership is to check whether $\alpha^l = 1$ or not. However, this method requests one exponentiation in $\mathbb{F}_{p^k}$. As we will explain later (Remark 6), the only purpose of this check in our protocols (or the ones of [16,25,31]) is to ensure that the server does not return an element of $\mathbb{F}_{p^k}$ of small order. In [29], the author provides a very efficient way to avoid such a case. For example, for $k = 12$ (the usual choice for a 128-bit security), since pairing values are elements of the cyclotomic subgroup of order $\phi_{12}(p) = p^4 - p^2 + 1$, one may check membership of α to this subgroup by testing if $\alpha \cdot \alpha^{p^4} = \alpha^{p^2}$. Using the Frobenius action this can be done almost for free. However, this is useful as long as the cofactor $h := \frac{\phi_{12}(p)}{l}$ does not have small factors (for example, if h is prime and greater than l) which requires a special care when choosing the curve parameters. In the following, we will thus distinguish a test of membership from an exponentiation in $\mathbb{F}_{p^k}$.

3 Security Model

In this section, we give the security properties that we require for a secure pairing delegation. Let $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$. We consider a restricted device, usually called a client, wanting to obtain the output of $e(A, B)$. For this purpose, the client interacts with a more powerful device, usually called the server, which is not necessarily trusted. We thus need to describe an interactive protocol between the client and the server where the output for the client is $e(A, B)$.

3.1 Syntax

A pairing delegation scheme consists of the three algorithms defined below, where *params* are some public parameters (see Remark 2 below).

- `Init`$(params, A, B)$: this probabilistic algorithm takes as inputs two points A and B and outputs σ, sent to the server to compute with, and τ, kept secret by the client.
- `Compute`$(params, \sigma)$: this deterministic algorithm is run by the server to compute α, which value is sent to the client.
- `Extract`$(params, \sigma, \tau, \alpha)$: this algorithm is run by the client which uses the known secret (τ) and public values (σ) to check whether the computations (α) performed by the server are valid or not. The client finally outputs either a value μ (equals to $e(A, B)$ in the former case) or an error message $\perp$ (in the latter case).

Remark 2. We do not add a `Setup` algorithm since we assume that our delegation scheme will be used to compute pairings in cryptographic protocols where the public parameters *params* (containing, for example, a description of the bilinear groups) are already defined.

In practice, there are mainly two cases, considering the values A and B. In the first case, A or B is a *constant* value that never changes from one pairing computation to another, while the other is said *variable.* The other case is when both A and B are variables.

3.2 Security Notions

Regarding security, the authors of [16] and [25] have considered the three following informal security notions: (i) *completeness* (an honest client, interacting with an honest server, obtains $e(A, B)$ after completion of the protocol), (ii) *correctness* (a client interacting with a cheating server will output $\perp$ with overwhelming probability) and (iii) *secrecy* (even a dishonest server cannot learn any information about A and B).

As in [31], we rather define our security notions through experiments since they describe more precisely the power and knowledge of the adversary. Our correctness/verifiability (see the remark below) experiment is similar with the one from [31]. However, we propose a stronger definition of secrecy. Our security notions make use of the following oracle.

– $\mathcal{O}\texttt{Sim}(params, A, B)$: is an oracle that executes the client's side of the protocol. In this case, the adversary plays the role of the corrupted server.

Remark 3. In this paper we will talk about *verifiability* rather than *correctness* since the latter is frequently used to denote *completeness* in cryptographic protocols.

Completeness. Informally, our definition of completeness is the same as the one provided by [16]. More formally, we define the completeness experiment $\mathbf{Exp}_{\mathcal{A}}^{comp}(params)$ as follows, where $\mathcal{A}^{\mathcal{O}\texttt{Sim}}$ denotes an adversary $\mathcal{A}$ having an unconditional access to the $\mathcal{O}\texttt{Sim}$ oracle, in an interactive way.

1. $(A, B) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Sim}}(params)$.
2. $(\sigma, \tau) \leftarrow \texttt{Init}(params, A, B)$.
3. $\alpha \leftarrow \texttt{Compute}(params, \sigma)$.
4. $\mu \leftarrow \texttt{Extract}(params, \sigma, \tau, \alpha)$.
5. If $\mu = e(A, B)$ then return 1.

A pairing protocol is *complete* if the probability $\texttt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{comp}(params) = 1]$ is overwhelming for all $\mathcal{A}$.

Verifiability. Regarding the literature on the subject, the definitions that one can find in [16,25] on the verifiability property are not very satisfying. In fact, they do not clearly specify the status (known or unknown) of A and B *w.r.t.* the server. Indeed, it seems that the status of A and B *w.r.t.* the adversary in the experiment related to the verifiability property depends, in their definition, of their status *w.r.t.* the server in the real protocol. Then, the probability of success of an adversary against the verifiability property may depend on the one against the secrecy, which is not very common in security where property definitions are usually independent one with each other.

A REMARK ON RELATED WORK SECURITY. In fact, this may even lead to some defaults related to security, and we can illustrate that using [25]. Let us consider a server being able to recover the secret points A and B with non-negligible probability λ for the protocol described in [25] (see Figure 1). It is then possible to show that such adversary is able to break the verifiability of this protocol with the same probability λ.

Indeed, if the server sends (instead of specified values), $\alpha_3 = e(R_1, R_2)^{1+z}$ and $\alpha_4 = e(T_1, T_2).e(A, B)^z$, for a randomly chosen z, then the client will output $e(A, B)^{1+z}$ instead of $e(A, B)$ since $\alpha_1, \alpha_2, \alpha_3$ and α_4 still satisfy the last equality test on α_4. The adversary will then succeed against the verifiability property with probability at least λ.

As a conclusion, we think that it is better to consider another definition for the verifiability property, which does not depend on the status (known or unknown) of A and B.

```
client $(A, B, G_1, G_2, G_T = e(G_1, G_2))$                server$(G_1, G_2)$
$g_1, g_2 \xleftarrow{\$} \mathbb{Z}_l$
$R_1 \leftarrow [g_1]A$; $R_2 \leftarrow [g_2]B$
                                        $\xrightarrow{R_1, R_2}$
                                                  $\alpha_1 \leftarrow (R_1, G_2)$; $\alpha_2 \leftarrow (G_1, R_2)$;
                                                  $\alpha_3 \leftarrow e(R_1, R_2)$
                                        $\xleftarrow{\alpha_1, \alpha_2, \alpha_3}$
For $i \in \{1, 2, 3\}$:
    If $\alpha_i \notin \mathbb{G}_T$, return $\perp$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_l$
$T_1 \leftarrow A + [r_1]G_1$; $T_2 \leftarrow B + [r_2]G_2$
                                        $\xrightarrow{T_1, T_2}$
                                                  $\alpha_4 \leftarrow e(T_1, T_2)$
                                        $\xleftarrow{\alpha_4}$
If $\alpha_4 = \alpha_3^{(g_1 g_2)^{-1}} \cdot \alpha_1^{g_1^{-1} r_2} \cdot \alpha_2^{g_2^{-1} r_1} \cdot G_T^{r_1 r_2}$
    Return $\mu := \alpha_3^{(g_1 g_2)^{-1}}$
Else, return $\perp$
```

Fig. 1. The Kang *et al* protocol [25] for secret A and B

Formal definition of verifiability. Informally, verifiability requires that the client, even interacting with a dishonest server, will not output a wrong value for $e(A, B)$. We define the verifiability experiment $\mathbf{Exp}_{\mathcal{A}}^{verif}(params)$ as follows.

1. $(A, B, st) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sim}}(params)$.
2. $(\sigma, \tau) \leftarrow \mathtt{Init}(params, A, B)$.
3. $\alpha \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sim}}(params, \sigma, st)$.
4. $\mu \leftarrow \mathtt{Extract}(params, \sigma, \tau, \alpha)$.
5. If $\mu = \perp$ or $\mu = e(A, B)$ then return 0.
6. Else return 1.

We define $\mathbf{Adv}_{\mathcal{A}}^{verif}(params) = \mathtt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{verif}(params) = 1]$. A pairing delegation protocol is *verifiable* if, for any probabilistic polynomial time adversary, this advantage is negligible.

Secrecy. Informally, secrecy requires that the server cannot learn any information about A or B. We define the secrecy experiment $\mathbf{Exp}_{\mathcal{A}}^{sec}(params)$ as follows.

1. $(A_0, B_0, A_1, B_1, st) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sim}}(params)$.
2. $b \overset{\$}{\leftarrow} \{0, 1\}$.
3. $(\sigma, \tau) \leftarrow \mathtt{Init}(params, A_b, B_b)$.
4. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sim}}(params, \sigma, st)$.
5. If $b^* = b$ then return 1. Else return 0.

We define $\mathbf{Adv}_{\mathcal{A}}^{sec}(params) = |\mathtt{Pr}[\mathbf{Exp}_{\mathcal{A}}^{sec}(params) = 1] - \frac{1}{2}|$. A pairing delegation protocol is *secret* if, for any probabilistic polynomial time adversary, this advantage is negligible.

In [31], the adversary against the secrecy property must distinguish a valid transcript from a simulated one without knowing the secret points. Our model is then stronger since it allows the adversary to choose the challenge points A and B. It is similar to the IND-CPA notion for public key encryption schemes.

4 Protocols with Public A and B

We provide in this section two efficient protocols to delegate the computation of public A and B, even if both of them are variable. For clarity's sake we first describe a protocol whose efficiency is equivalent to one exponentiation and one test of membership and then show how to modify it to suit the case where this last operation cannot be performed cheaply.

4.1 A Protocol with Test of Membership

We assume, as the authors of [16], [25] and [31], that the public parameters contain 3 elements: $G_1 \in \mathbb{G}_1$, $G_2 \in \mathbb{G}_2$ and $\rho = e(G_1, G_2)$.

Remark 4. Papers [16,25] do not explain how the client obtains the values G_1, G_2 and ρ. As mentioned in [31], there are two ways to treat this. For example, the client could generate G_1 and G_2 and compute ρ once for all. This computation can also be done by a trusted authority, which one could then embed the values in the client. In the latter case, there is no longer need for implementing the whole pairing computation algorithm in the client since our protocols only require group operations in the bilinear groups. This may justify, besides the efficiency, the use of our solutions since it saves some area needed to implement cryptographic operations. Indeed, there exist several different pairings, such as the Weil pairing, the Tate pairing or one of its variants [1,22]. Our protocols are then compatible with all of them as long as the values $e(G_1, G_2)$ (one for each type of pairing) are loaded in the client's memory.

The three algorithms defining our pairing delegation scheme are described in Figure 2 and enable the client to delegate the computation of $e(A, B)$ with public $A \neq 1_{\mathbb{G}_1}$ and $B \neq 1_{\mathbb{G}_2}$.

```
client $(A, B, G_1, G_2, \rho)$                                   server$(A, B)$
Init$(params, A, B)$:
  $x_1, x_2 \overset{\$}{\leftarrow} \mathbb{Z}_l$
  $X_1 \leftarrow [x_1]G_1$; $X_2 \leftarrow [x_2]G_2$
  $\chi \leftarrow \rho^{x_1 \cdot x_2}$
  $T_1 \leftarrow [x_2^{-1}]A + X_1$; $T_2 \leftarrow [x_1^{-1}]B + X_2$
  Return $(\sigma, \tau) := ((T_1, T_2), (x_1, x_2))$
                                  $\xrightarrow{\sigma}$
                                                  Compute$(params, \sigma)$:
                                                    $\alpha_1 \leftarrow e(T_1, T_2)[e(G_1, B)e(A, G_2)]^{-1}$
                                                    $\alpha_2 \leftarrow e(A, B)$
                                                    Return $\alpha := (\alpha_1, \alpha_2)$
                                  $\xleftarrow{\alpha}$
Extract$(params, \sigma, \tau, \alpha)$:
  If $\alpha_1 = \chi \cdot \alpha_2^{(x_1 . x_2)^{-1}}$ and $\alpha_2 \in \mathbb{G}_T$
    Return $\mu := \alpha_2$
  Else, return $\perp$
```

Fig. 2. Delegation protocol for public A and B

Computational Cost. Since X_1, X_2 and χ are easily pre-computable (they do not need the knowledge of A and B), the client only has to compute *online* a scalar multiplication in $\mathbb{G}_1$, another one in $\mathbb{G}_2$, an exponentiation in $\mathbb{G}_T$ and a test of membership in $\mathbb{G}_T$. The efficiency of our method strongly depends on the parameters of the bilinear groups, especially if they allow us to use the idea from [29] to avoid an exponentiation in $\mathbb{F}_{p^k}$ for the test of membership (see end of Section 2). To get an idea of the order of magnitude of the computational cost, one may look at the results from [12,11]. For every family of curves, their timings indicate that the cost of our protocol (assuming that the test of membership in $\mathbb{G}_T$ is cheap) is significantly smaller than the one of a pairing. One example (for an optimal ate pairing on a KSS-18 curve [24]) is given in Table 1 at the end of this paper.

Security. As A and B are public, we only need to verify that our protocol ensures the completeness and the verifiability properties.

Completeness. The protocol is complete since:

$$\begin{aligned}\alpha_1 &= e([x_2^{-1}]A + X_1, [x_1^{-1}]B + X_2)[e(G_1, B)e(A, G_2)]^{-1} \\ &= e(A, B)^{(x_1 . x_2)^{-1}} e(X_1, X_2) = \chi \cdot \alpha_2^{(x_1 \cdot x_2)^{-1}}.\end{aligned}$$

Verifiability. The main idea of our protocol is to request from the server the computations of α_1 and α_2, involved in a relation with the secret value τ. So, an adversary trying to cheat the client has to provide α_1' and α_2' satisfying the same relation. In the following, we argue that he is unable to do so, which ensures the verifiability of our protocol.

Remark 5. Our following proof is verified in the generic group model (extended to the bilinear setting). Even if we do not really provide a formal theorem that the underlying new assumption is valid, the methodology we adopt in the sequel is quite similar to a proof in the generic group model.

In the verifiability experiment, the server is controlled by the adversary who wants to convince the client to accept a wrong value for $e(A, B)$. This means that the adversary sends an element $\alpha_2' \neq \alpha_2 = e(A, B)$ belonging to $\mathbb{G}_T$ (since we test membership in this subgroup). So we have $\alpha_2' = \alpha_2.\delta$ for some $\delta \in \mathbb{G}_T$. It follows that the server has to send $\alpha_1' = \alpha_1 \cdot \gamma$ verifying:

$$\alpha_1' = \chi \cdot (\alpha_2')^{(x_1 \cdot x_2)^{-1}} \iff \alpha_1 \cdot \gamma = \chi \cdot (\alpha_2 \cdot \delta)^{(x_1 \cdot x_2)^{-1}} \iff \gamma = \delta^{(x_1 \cdot x_2)^{-1}}.$$

For the adversary, breaking the verifiability is then equivalent to find any two values $\gamma, \delta \in \mathbb{G}_T$ such that $\gamma = \delta^{(x_1 \cdot x_2)^{-1}}$. However, finding such a pair $(\delta, \delta^{(x_1 \cdot x_2)^{-1}}) \in \mathbb{G}_T^2$ does not match any standard computational assumption. So we cannot directly conclude. We then study the probability of recovering $(\delta, \delta^{(x_1 \cdot x_2)^{-1}})$ by using combinations of elements involved in the protocol. We consider the case of type 3 pairings (*i.e.* there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$) in order to reduce the number of possible combinations. However, our proof can also be done for other types of pairings.

Let $a, b, x_1, x_2 \in \mathbb{Z}_l$ be such that:

$$A = [a]G_1,\ B = [b]G_2,\ X_1 = [x_1]G_1 \text{ and } X_2 = [x_2]G_2.$$

Our security model defined in the previous section allows the adversary to choose A and B and we consequently assume that he knows a and b. Since we work with bilinear groups, we assume that the adversary is only able to compute pairings or algebraic combinations in $\mathbb{G}_1$ or $\mathbb{G}_2$, *i.e.* the adversary is only able to choose $a_1, a_2, a_3, a_4 \in \mathbb{Z}_l$ and computes:

$$e([a_1]G_1 + [a_2]T_1, [a_3]G_2 + [a_4]T_2) = e(G_1, G_2)^{s \cdot (x_1 \cdot x_2)^{-1} + t},$$

with $s = a_2 a_4 ab$ and $t = a_1 a_3 + a_1 a_4 b(x_1)^{-1} + a_1 a_4 x_2 + a_2 a_3 a(x_2)^{-1} + a_2 a_4 a + a_2 a_3 x_1 + a_2 a_4 b + a_2 a_4 x_1 x_2$. The only way (unless to guess $(x_1 \cdot x_2)^{-1}$ with probability $\frac{1}{l}$) for the server to find a suitable pair $(\delta, \delta^{(x_1 \cdot x_2)^{-1}})$ is then to recover:

$$(e(G_1, G_2)^s, e(G_1, G_2)^{s \cdot (x_1 \cdot x_2)^{-1}})$$

which means that it must find a_1, a_2, a_3, a_4 cancelling t but not s. The map

$$\psi : (\mathbb{Z}_l)^4 \to \mathbb{Z}_l$$
$$(a_1, ..., a_4) \mapsto t$$

is a quadratic form, its matrix M is:

$$2^{-1}\begin{pmatrix} 0 & 0 & 1 & b(x_1)^{-1}+x_2 \\ 0 & 0 & a(x_2)^{-1}+x_1 & a+b+x_1x_2 \\ 1 & a(x_2)^{-1}+x_1 & 0 & 0 \\ b(x_1)^{-1}+x_2 & a+b+x_1x_2 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & N \\ N^T & 0 \end{pmatrix}$$

where N^T is the transpose of N. $\forall\ a, b \in \mathbb{Z}_l^*$ the rank of $N_{a,b}$ is 2, the number of zeroes of ψ is then:

$$|\{(a_1, \cdots, a_4) \in \mathbb{Z}_l^4 : \psi(a_1, \cdots, a_4) = 0\}| \le l^3 + l^2 - l \le 2l^3.$$

Since the server does not know x_1 or x_2, he must guess suitable values for $v = (a_1, a_2, a_3, a_4)$. However, the probability that v is an isotropic vector (*i.e.* $t = 0$) is negligible ($\frac{l^3+l^2-l}{l^4} \le \frac{2}{l}$). Then, if $\alpha_2 \ne e(A, B)$, the client outputs $\perp$ with overwhelming probability, which concludes the fact that our protocol is verifiable.

Remark 6. In our protocol, as in previous works, the client has to test membership in $\mathbb{G}_T$ of some values returned by the server. However, the purpose of such test is to ensure that the server does not return elements of $(\mathbb{F}_{p^k})^*$ of small orders. Indeed, as shown in the above security study, the adversary has to find a pair $(\delta, \delta^{(x_1 \cdot x_2)^{-1}})$ to break the verifiability property. If the order of α_2 is not checked, then the adversary can choose an element of $(\mathbb{F}_{p^k})^*$ of order 2, and would then succeed with probability $\frac{1}{2}$, since it just has to guess the parity of $(x_1 \cdot x_2)^{-1}$.

4.2 Efficient Variant with One Exponentiation

As explained above, the efficiency of our protocol mainly depends on the computational cost of the test of membership. If the curve parameters do not allow the client to use the idea from [29], then the test of membership in $\mathbb{G}_T$ will require a costly exponentiation in $(\mathbb{F}_{p^k})^*$, making the speed-up of the delegation less obvious. Our aim in this section is to remove this test while ensuring verifiability.

In a nutshell, we will make use of Remark 1 given in Section 2 so that the fact that the order of α_2 is implicitly l, without the necessity to verify such fact. More precisely, the client will now compute $\widetilde{\chi}$ (instead of χ) such that $\chi = \rho^{x_1 x_2}$. As explained in Remark 1, $\widetilde{\chi}^c = \chi$, where $c = \frac{p^k-1}{l}$. Then, the client and the server proceed as in the protocol of Figure 2, except that the server now returns $\widetilde{\alpha}_1$ (where $\widetilde{\alpha}_1^c = \alpha_1$) and α_2. Then, the client needs to check that

$$\alpha_2 = (\widetilde{\alpha}_1 \cdot \widetilde{\chi}^{-1})^{c \cdot x_1 \cdot x_2}.$$

Obviously, checking that α_2 is equal to an element of $(\mathbb{F}_{p^k})^*$, raised to the power c, necessarily ensures that it belongs to $\mathbb{G}_T$ (since this group contains all the elements of order l). As a conclusion, we no longer need to verify that α_2 belongs to $\mathbb{G}_T$. It follows that no adversary is able to cheat unless to provide a pair $(\delta, \delta^{(x_1 \cdot x_2)^{-1}})$ with δ of order l. The security of this variant is thus the same as the one of the original protocol.

Regarding efficiency, our protocol now requires 1 scalar multiplication in $\mathbb{G}_1$, 1 scalar multiplication in $\mathbb{G}_2$ and only one exponentiation in $(\mathbb{F}_{p^k})^*$. Since the exponent involved in this last operation is close to the one (namely c) involved in the last step of the Tate pairing, usually called the final exponentiation, we may use a similar methodology as the one in [30]. It thus remains to compare the computational cost of a Miller loop against 1 scalar multiplication in $\mathbb{G}_1$ and 1 scalar multiplication in $\mathbb{G}_2$. Using the timings from [28], we may conclude that our protocol is still more efficient than computing the pairing. Again, an estimated ratio is given in Table 1 for this variant.

4.3 Batch Delegation

In [31], the authors have considered the delegation of several pairings all at once but they have only proposed protocols with one constant point to each pairing. There are two reasons why batch delegation does not suit our protocol. First, with one constant point A, one may efficiently check the validity of the requested $e(A, B_1)$,...,$e(A, B_n)$ by using the bilinearity of the pairing since $e(A, B_1 + B_2 + ... + B_n) = e(A, B_1) \cdot ... \cdot e(A, B_n)$. However this is not possible with our protocol since we do not assume that one of the pairing's input is constant. More specifically, assuming that we want to delegate $e(A_1, B_1), ..., e(A_n, B_n)$, the computation of $e(A_1 + ... + A_n, B_1 + ... + B_n)$ is useless because it also involves several unknown values (the values $e(A_i, B_j)$ for $i \neq j$) that the client will have to cancel, which leads to additional computations. Second, the goal of batch delegation is to check validity of the n delegated pairings with less than n equality tests. However, when two pairings $\alpha_i = e(A_i, B_i)$ and $\alpha_j = e(A_j, B_j)$ are involved in the same equality test, they must be raised to different powers, else, an adversary could return $\alpha_i \cdot \delta$ and $\alpha_j \cdot \delta^{-1}$, for some $\delta \in \mathbb{G}_T$, and still satisfy the test. It then seems hard to construct a protocol for delegating n pairings with less than n exponentiations in $\mathbb{G}_T$ which is roughly the cost of n runs of our protocol (in the case of a cheap test of membership in $\mathbb{G}_T$).

5 Ensuring both Verifiability and Secrecy

We now consider the case where the points A and/or B are/is secret. We first explain how to modify our previous protocols to achieve secrecy and then propose an improved protocol which suits the case where B is a constant public point.

5.1 A Generic Conversion

There is an easy way to reach the secrecy property from the protocols described in the previous section, using the ideas given in [20]. If A and B are secret, one can simply compute $A' \leftarrow [u]A$ and $B' \leftarrow [v]B$ for randomly chosen u and v in $\mathbb{Z}_l$. Then, the client and the server play one of the protocols given in the previous section to get $e(A', B')$. Finally, $e(A, B)$ is obtained by simply computing $e(A', B')^{(uv)^{-1}}$. The completeness and verifiability of this protocol directly follow from the ones of the protocols given in the previous section. The secrecy is then obvious since A' and B' are seen as random elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. This leads to a secure protocol requiring mainly two exponentiations in $\mathbb{G}_T$ and either one test of membership if we use the protocol described in Section 4.1, or one exponentiation in $\mathbb{G}_T$ and another one in $(\mathbb{F}_{p^k})^*$ if we use the protocol described in Section 4.2. Since all of these exponentiations require the knowledge of A and B, they have to be performed *online*.

5.2 A Protocol with Public Constant *B*

The case where A is a secret variable point and B is a constant public one can be found in some cryptographic protocols such as the one of Hess [21]. It was consider in [25] and [31]. But, on one hand, the solution provided in the former is not enough secure (see Section 3.2), since the verifiability depends on the secrecy. On the other hand, the solution of the latter requires two exponentiations in $\mathbb{G}_T$, one test of membership and additional computations in $\mathbb{G}_1$ and $\mathbb{G}_2$ during the *online* phase.

We here assume that the client already knows $\varrho \leftarrow e(G_1, B)$ (G_1 is a parameter and B is constant). We then provide a more efficient protocol, which is described in Figure 3.

```
client $(A, B, G_1, \varrho)$                                    server$(B)$
Init$(params, A, B)$:
  $x, y, u \overset{\$}{\leftarrow} \mathbb{Z}_l$
  $X \leftarrow [x]G_1$; $Y \leftarrow [y]G_1$
  $(\chi_1, \chi_2) \leftarrow (\varrho^x, \varrho^y)$
  $T_1 \leftarrow A + X$; $T_2 \leftarrow [u]A + Y$
  Return $(\sigma, \tau) := ((T_1, T_2), (u))$
                                  $\xrightarrow{\sigma}$
                                                     Compute$(params, \sigma)$:
                                                       $\alpha_1 \leftarrow e(T_1, B)$
                                                       $\alpha_2 \leftarrow e(T_2, B)$
                                                       Return $\alpha := (\alpha_1, \alpha_2)$
                                  $\xleftarrow{\alpha}$
Extract$(params, \sigma, \tau, \alpha)$:
  If $\alpha_2 = \chi_2 \cdot (\alpha_1 \cdot \chi_1^{-1})^u$ and $\alpha_1 \in \mathbb{G}_T$
    Return $\mu := \alpha_1 \cdot \chi_1^{-1}$
  Else, return $\perp$
```

Fig. 3. Delegation protocol for secret A and public constant B

Table 1. Efficiency and security comparison, where m_1 (resp. m_2) stands for a scalar multiplication in $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), e_T stands for an exponentiation in $\mathbb{G}_T$, e_F for an exponentiation in $(\mathbb{F}_{p^k})^*$, t_T for a test of membership in $\mathbb{G}_T$ and p_T stands for a pairing. We say that the verifiability is "conditional" when it depends on the secrecy (see section 3.2). Provided ratios assume that the test of membership can be performed cheaply [29]. The amount of storage required to store the pre-computed values is implicitly given in the column "offline client". Indeed, if an operation in a group is pre-computed, then the result, which is an element of this group, must be stored.

Protocols with variable A and B

	secrecy	verifiability	offline client	online client	server	Ratios
[20]	yes	no	-	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$1\mathsf{e}_T$	$1\mathsf{p}_T$	0.46
[16,17] [Sect 4.1]	yes	yes	$2\mathsf{m}_1$,$2\mathsf{m}_2$,$2\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$5\mathsf{e}_T$,$3\mathsf{t}_T$	$4\mathsf{p}_T$	1.46
[16,17] [Sect 5.2]	no	yes	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$1\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$3\mathsf{e}_T$,$3\mathsf{t}_T$	$4\mathsf{p}_T$	0.96
[25]	yes	conditional	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$1\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$3\mathsf{e}_T$,$3\mathsf{t}_T$	$4\mathsf{p}_T$	0.96
Ours [Sect 4.1]	no	yes	$1\mathsf{m}_1$,$1\mathsf{m}_2$, $1\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$1\mathsf{e}_T$,$1\mathsf{t}_T$	$4\mathsf{p}_T$	0.46
Ours [Sect 4.2]	no	yes	$1\mathsf{m}_1$,$1\mathsf{m}_2$, $1\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{m}_2$,$1\mathsf{e}_F$	$4\mathsf{p}_T$	0.84
Ours + [20][Sect 5.1]	yes	yes	$1\mathsf{m}_1$,$1\mathsf{m}_2$, $1\mathsf{e}_T$	$2\mathsf{m}_1$,$2\mathsf{m}_2$,$2\mathsf{e}_T$,$1\mathsf{t}_T$	$4\mathsf{p}_T$	0.92

Protocols with variable secret A and constant public B

	secrecy	verifiability	offline client	online client	server	Ratios
[25][Sect 4.3]	yes	conditional	$1\mathsf{m}_1$,$1\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{e}_T$,$1\mathsf{t}_T$	$2\mathsf{p}_T$	0.30
[31][SVPC]	yes	yes	$1\mathsf{m}_1$,$1\mathsf{e}_T$	$2\mathsf{m}_1$,$2\mathsf{e}_T$,$1\mathsf{t}_T$	$2\mathsf{p}_T$	0.60
Ours [Sect 5.2]	yes	yes	$2\mathsf{m}_1$,$2\mathsf{e}_T$	$1\mathsf{m}_1$,$1\mathsf{e}_T$,$1\mathsf{t}_T$	$2\mathsf{p}_T$	0.30

Computational Cost. Since χ_1 and χ_2 can be pre-computed, our protocol requires one exponentiation in $\mathbb{G}_T$, one test of membership in $\mathbb{G}_T$ and one scalar multiplication in $\mathbb{G}_1$.

Security. The protocol is complete since $e(A,B) = \alpha_1 \cdot \chi_1^{-1}$ and:

$$\begin{aligned}\chi_2 \cdot (\alpha_1 \cdot \chi_1^{-1})^u &= e(Y,B) \cdot (e(A+X,B) \cdot e(X,B)^{-1})^u \\ &= e(Y,B) \cdot e(A,B)^u \\ &= \alpha_2.\end{aligned}$$

T_1 and T_2 are random elements of $\mathbb{G}_1$ and thus do not reveal any information about A. As in the previous section, a pair (α'_1, α'_2) will satisfy the equality test if and only if $\alpha'_1 = \alpha_1 \cdot \delta$ and $\alpha'_2 = \alpha_2 \cdot \delta^u$. Since u is only involved in the computation of T_2, an adversary, even knowing A, will not be able to find a couple $(\delta, \delta^u) \in \mathbb{G}_T^2$ unless to guess Y. Our protocol ensures then both secrecy and verifiability with less computations than the one from [31], as we will see in the next section.

6 Conclusion and Efficiency Comparison

In this paper, we have provided several delegation processes for a bilinear pairing. We argue that our results are much more efficient than the state-of-the-art, for a comparable or improved security. As evidence, we provide in Table 1 a global comparison between our results and related works.

We use in this table the timings from [12,11] since this paper precisely describes the computational cost of operations in each group. Moreover, the authors have implemented their algorithms so that ratios between their different benchmark results do not depend on the platforms. They therefore remain relevant even considering an implementation on a smart card.

We emphasize that the efficiency of our protocols depends on the chosen pairing and curve. We do not claim that our protocols are more efficient than any implementation of pairing on any curve. However, there are some curves for which the efficiency gain is significant. As evidence, we give in the last column of Table 1 the estimated ratios between the online computational cost of our protocols and the one of a pairing for the KSS-18 [24] family of curves.

Acknowledgments. This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project. We are also grateful to anonymous referees for their valuable comments.

References

1. Paulo, S.L.M., Barreto, S.D., Galbraith, C.O.: hEigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. IACR Cryptology ePrint Archive, 375 (2004)
2. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
3. Barreto, P.S.L.M., Lynn, B., Scott, M.: On the selection of pairing-friendly groups. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 17–25. Springer, Heidelberg (2004)
4. 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 (2006)

5. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. IACR Cryptology ePrint Archive, 658 (2011)
6. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get shorty via group signatures without encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 381–398. Springer, Heidelberg (2010)
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
10. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security 2004, pp. 168–177. ACM (2004)
11. Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in pairing groups. In: Selected Areas in Cryptography (2013) (to appear)
12. Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in pairing groups. IACR Cryptology ePrint Archive, 458 (2013)
13. Canard, S., Coisel, I., Devigne, J., Gallais, C., Peters, T., Sanders, O.: Toward Generic Method for Server-Aided Cryptography. In: Qing, S., Zhou, J., Liu, D. (eds.) ICICS 2013. LNCS, vol. 8233, pp. 373–392. Springer, Heidelberg (2013)
14. Canard, S., Desmoulins, N., Devigne, J., Traoré, J.: On the implementation of a pairing-based cryptographic protocol in a constrained device. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 210–217. Springer, Heidelberg (2013)
15. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
16. Chevallier-Mames, B., Coron, J.-S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. IACR Cryptology ePrint Archive, 150 (2005)
17. 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.) CARDIS 2010. LNCS, vol. 6035, pp. 24–35. Springer, Heidelberg (2010)
18. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
19. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
20. Girault, M., Lefranc, D.: Server-aided verification: Theory and practice. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623. Springer, Heidelberg (2005)
21. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
22. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
23. Joux, A.: A one round protocol for tripartite diffie-hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)

24. 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 (2008)
25. Kang, B.G., Lee, M.S., Park, J.H.: Efficient delegation of pairing computation. IACR Cryptology ePrint Archive, 259 (2005)
26. Lim, C.H., Lee, P.J.: Server (Prover/Signer)-aided verification of identity proofs and signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 64–78. Springer, Heidelberg (1995)
27. Matsumoto, T., Kato, K., Imai, H.: Speeding up secret computations with insecure auxiliary devices. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 497–506. Springer, Heidelberg (1990)
28. Sánchez, A.H., Rodríguez-Henríquez, F.: NEON implementation of an attribute-based encryption scheme. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 322–338. Springer, Heidelberg (2013)
29. Scott, M.: Unbalancing pairing-based key exchange protocols. Cryptology ePrint Archive, Report 2013/688 (2013), `http://eprint.iacr.org/`
30. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the final exponentiation for calculating pairings on ordinary elliptic curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
31. Tsang, P.P., Chow, S.S.M., Smith, S.W.: Batch pairing delegation. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 74–90. Springer, Heidelberg (2007)
32. Yao, A.C.-C.: Protocols for Secure Computations (extended abstract). In: FOCS, pp. 160–164. IEEE Computer Society (1982)