Secure Delegation of Elliptic-Curve Pairing

Benoît Chevallier-Mames[1], Jean-Sébastien Coron[2], Noel McCullagh[3⋆], David Naccache[2], and Michael Scott[3⋆]

[1] Gemplus Card International
Applied Research & Security Centre
Avenue des Jujubiers, La Ciotat, F-13705, France
benoit.chevallier-mames@gemplus.com
[2] Gemplus Card International
Applied Research & Security Centre 34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, david.naccache}@gemplus.com
[3] School of Computing
Dublin City University
Glasnevin
Dublin 9, Ireland
{noel.mccullagh, mike}@computing.dcu.ie

Abstract. In this paper we describe a simple protocol for securely delegating elliptic-curve pairings. A computationally limited device (typically a smart-card) will delegate the computation of the pairing $e(A, B)$ to a more powerful device (for example a PC), in such a way that:

1. the powerful device learns nothing about the points being paired (A and B), nor about the pairing's result $e(A, B)$,
2. and the limited device is able to detect when the powerful device is cheating.

We also describe more efficient variants of our protocol when one of the points or both are already known, and further efficiency gains when constant points are used.

1 Introduction

Since the discovery of the first practical identity-based cryptosystem based on the elliptic-curve pairing [1], pairing-based cryptography has become a very active research area. To date, many pairing-based protocols have been proposed with novel and attractive properties, for example for key-exchange [5] and digital signatures [3].

The increasing popularity of pairing-based cryptosystems and their foreseeable deployment in computationally constrained devices such as smart-cards and dongles spurred recent research in the implementation of pairing (*e.g.* [7]). Unfortunately, although pairing is a cubic-time operation, pairing implementation

⋆ These authors are presently also at NoreTech.

attempts in limited devices such as smart-cards reveal that the embedded code may be slow, resource-consuming and tricky to program.

Given that several PC-based pairing libraries exist, it seems natural to find-out whether a smart-card could interact with such packages to privately compute the elliptic-curve pairing. Note that beyond preserving operands and results from preying eyes, the card must also ascertain that bogus libraries don't mislead it into generating wrong results.

In this paper, we propose a simple protocol for the secure delegation of elliptic-curve pairing. A computationally limited device (for example a smart-card) will delegate the computation of the elliptic-curve pairing $e(A, B)$ to a more powerful device (for example a PC), in such a way that:

1. the powerful device learns nothing about the points being paired (A and B) nor about the pairing's result $e(A, B)$,
2. and the limited device is able to detect when the powerful device is cheating.

The limited device will restrict itself to simple curve or field operations. We also describe efficient variants of our protocol applicable when one of the points A and B or both are already publicly known.

2 Preliminaries

Our protocol for secure pairing delegation is actually more general than just elliptic-curve pairing: as most pairing-based cryptosystems, it works for any bilinear map. Therefore, we briefly review the necessary facts about bilinear maps. We follow the notations of [2], except that we use the additive notation for the groups $\mathcal{G}_1$ and $\mathcal{G}_2$. We refer the reader to [6] for an extensive background on elliptic-curve pairing.

1. $\mathcal{G}_1$ and $\mathcal{G}_2$ are two (additive) cyclic groups of prime order p;
2. G_1 is a generator of $\mathcal{G}_1$ and G_2 is a generator of $\mathcal{G}_2$;
3. ψ is a computable isomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$ with $\psi(G_1) = G_2$;
4. e is a computable bilinear map $e : \mathcal{G}_1 \cdot \mathcal{G}_2 \to \mathcal{G}_T$;
5. $\mathcal{G}_T$ is a multiplicative cyclic group of order p.

A bilinear map is a map $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ with the following properties:

1. Bilinear: for all $U \in \mathcal{G}_1, V \in \mathcal{G}_2$ and $a, b \in \mathbb{Z}$, $e(a.U, b.V) = e(U, V)^{ab}$
2. Non-degenerate: $e(G_1, G_2) \neq 1$

Note that the previous conditions imply that $e(G_1, G_2)$ is a generator of $\mathcal{G}_T$.

3 Secure Pairing Delegation

In this section, we formalize the security notions for secure pairing delegation. Our setting is the following: a computationally limited device, called the card and denoted by $\mathcal{C}$, will delegate the computation of $e(A, B)$ to a more powerful device, called the terminal and denoted $\mathcal{T}$. Both devices are actually probabilistic polynomial-time Turing machines.

The security notions could be formalized in the general framework of secure multiparty computation (for standard definitions, see for example [4]). However, we observe that our setting is much simpler than general secure two-party computation: the terminal has no secret and outputs nothing; only the terminal can be malicious. Therefore, we say that a protocol for pairing delegation is secure if it satisfies the three following security notions:

Completeness: after protocol completion with an honest terminal, $\mathcal{C}$ obtains $e(A, B)$, except with negligible probability.

Secrecy: a (possibly cheating) terminal should not learn any information about the points A and B. Formally, for any malicious $\mathcal{T}$, there exists a simulator $\mathcal{S}$ such that for any A, B, the output of S is computationally indistinguishable from $\mathcal{T}$'s view:

$$S \stackrel{c}{\equiv} \text{View}_{\mathcal{T}}(A, B)$$

Correctness: $\mathcal{C}$ should be able to detect a cheating $\mathcal{T}$, except with negligible probability. Formally, for any cheating $\mathcal{T}$ and for any A, B, $\mathcal{C}$ either outputs $\perp$ or determines $e(A, B)$, except with negligible probability.

4 Our Protocol

4.1 Description

In the following, we describe our protocol for securing pairing delegation. $\mathcal{C}$ and $\mathcal{T}$ are given as input a description of the groups $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$, and a description of the bilinear map $e : \mathcal{G}_1 \cdot \mathcal{G}_2 \to \mathcal{G}_T$. $\mathcal{C}$ and $\mathcal{T}$ receives the generators G_1 and G_2; moreover we assume that $\mathcal{C}$ receives $e(G_1, G_2)$. $\mathcal{C}$ is given as input the points A and B and must eventually determine $e(A, B)$.

1. $\mathcal{C}$ generates a random $g_1 \in \mathbb{Z}_p$ and a random $g_2 \in \mathbb{Z}_p$, and queries the three following pairings from $\mathcal{T}$:

$$\alpha_1 = e(A + g_1.G_1, G_2), \quad \alpha_2 = e(G_1, B + g_2.G_2)$$

$$\alpha_3 = e(A + g_1.G_1, B + g_2.G_2)$$

2. $\mathcal{C}$ checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Should this test fail, $\mathcal{C}$ outputs $\perp$ and halts.
3. $\mathcal{C}$ computes a purported value for $e(A, B)$:

$$e_{AB} = \alpha_1^{-g_2} \cdot \alpha_2^{-g_1} \cdot \alpha_3 \cdot e(G_1, G_2)^{g_1 g_2} \tag{1}$$

4. $\mathcal{C}$ generates four random values $a_1, r_1, a_2, r_2 \in \mathbb{Z}_p$ and queries the pairing:

$$\alpha_4 = e(a_1.A + r_1.G_1, a_2.B + r_2.G_2)$$

5. $\mathcal{C}$ computes:

$$\alpha'_4 = (e_{AB})^{a_1 a_2} \cdot (\alpha_1)^{a_1 r_2} \cdot (\alpha_2)^{a_2 r_1} \cdot e(G_1, G_2)^{r_1 r_2 - a_1 g_1 r_2 - a_2 g_2 r_1} \quad (2)$$

and checks that $\alpha'_4 = \alpha_4$. In this case, $\mathcal{C}$ accepts e_{AB} as the genuine value of $e(A, B)$; otherwise it outputs $\perp$.

4.2 Security Proof

The following theorem shows that our protocol is secure:

Theorem 1. *The previous protocol is a secure pairing delegation protocol.*

Proof. The completeness property is easily established. From bilinearity:

$$e(A + g_1.G_1, B + g_2.G_2) = e(A, B) \cdot e(A, G_2)^{g_2} \cdot e(G_1, B)^{g_1} \cdot e(G_1, G_2)^{g_1 g_2}$$

Then, for an honest $\mathcal{T}$, we have:

$$\alpha_1 = e(A + g_1.G_1, G_2) = e(A, G_2) \cdot e(G_1, G_2)^{g_1} \quad (3)$$
$$\alpha_2 = e(G_1, B + g_2.G_2) = e(G_1, B) \cdot e(G_1, G_2)^{g_2} \quad (4)$$
$$\alpha_3 = e(A + g_1.G_1, B + g_2.G_2) \quad (5)$$

Combining the four previous equations, we obtain:

$$\alpha_3 = e(A, B) \cdot (\alpha_1)^{g_2} \cdot (\alpha_2)^{g_1} \cdot e(G_1, G_2)^{-g_1 g_2}$$

which, using (1), shows that $\mathcal{C}$ computes the correct $e_{AB} = e(A, B)$. Moreover, using:

$$\begin{aligned} \alpha_4 &= e(a_1.A + r_1.G_1, a_2.B + r_2.G_2) \\ &= e(A, B)^{a_1 a_2} \cdot e(A, G_2)^{a_1 r_2} \cdot e(G_1, B)^{r_1 a_2} \cdot e(G_1, G_2)^{r_1 r_2} \end{aligned}$$

we obtain from equations (3) and (4):

$$\alpha_4 = (e_{AB})^{a_1 a_2} \cdot (\alpha_1)^{a_1 r_2} \cdot (\alpha_2)^{r_1 a_2} e(G_1, G_2)^{r_1 r_2 - a_1 g_1 r_2 - a_2 g_2 r_1}$$

which, using (2), gives $\alpha_4 = \alpha'_4$ and shows that $\mathcal{C}$ eventually outputs the correct $e_{AB} = e(A, B)$.

The secrecy property follows from the fact that $\mathcal{T}$ receives only random, independently distributed points in the groups $\mathcal{G}_1$ and $\mathcal{G}_2$. Therefore, the simulator $\mathcal{S}$ simply consists in running $\mathcal{T}$ with randomly generated points. The simulator's output and $\mathcal{T}$'s view when interacting with $\mathcal{C}$ are then identically distributed.

The correctness property is established as follows: we show that if the value e_{AB} computed by $\mathcal{C}$ at step 3 is unequal to $e(A, B)$, then the element α'_4 computed by $\mathcal{C}$ at step 5 has a nearly uniform distribution in $\mathcal{G}_T$, independent of $\mathcal{T}$'s view. Then, the probability that $\alpha_4 = \alpha'_4$ at step 5 will be roughly $1/p$. Therefore, $\mathcal{C}$ will output $\perp$, except with negligible probability.

We let $U = a_1.A + r_1.G_1$ and $V = a_2.B + r_2.G_2$. Moreover, we let $a, b, u, v \in \mathbb{Z}_p$ be such that $A = a.G_1$, $B = b.G_2$, $U = u.G_1$, $V = v.G_2$, which gives:

$$u = a_1 \cdot a + r_1 \tag{6}$$

$$v = a_2 \cdot b + r_2 \tag{7}$$

$\mathcal{C}$ checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{G}_T$. Therefore, we must have $e_{AB} \in \mathcal{G}_T$, and since $e(G_1, G_2)$ is a generator of $\mathcal{G}_T$, we can let $\beta_1, \beta_2, \beta_3 \in \mathbb{Z}_p$ be such that:

$$\alpha_1 = e(A, G_2) \cdot e(G_1, G_2)^{g_1+\beta_1} \tag{8}$$

$$\alpha_2 = e(G_1, B) \cdot e(G_1, G_2)^{g_2+\beta_2} \tag{9}$$

$$e_{AB} = e(A, B) \cdot e(G_1, G_2)^{\beta_3} \tag{10}$$

Therefore, the value e_{AB} is correct iff $\beta_3 = 0$.

From the previous observation, we also have $\alpha'_4 \in \mathcal{G}_T$. Therefore, we can assume that $\alpha_4 \in \mathcal{G}_T$, since otherwise $\alpha'_4 \neq \alpha_4$ and $\mathcal{C}$ outputs $\perp$. Then we can let $\beta_4, \beta'_4 \in \mathbb{Z}_p$ be such that:

$$\alpha_4 = e(U, V) \cdot e(G_1, G_2)^{\beta_4} \tag{11}$$

$$\alpha'_4 = e(U, V) \cdot e(G_1, G_2)^{\beta'_4} \tag{12}$$

Therefore, $\mathcal{C}$ outputs e_{AB} iff $\beta_4 = \beta'_4$.

In the following, we assume that $u \neq 0$ and $v \neq 0$. Since (u, v) is uniformly distributed in $\mathbb{Z}_p$, this happens with probability $(1 - 1/p)^2 \geq 1 - 2/p$.

We show that if $\beta_3 \neq 0$, then β'_4 has a nearly uniform distribution in $\mathbb{Z}_p$, independent of $\mathcal{T}$'s view, and therefore $\beta_4 = \beta'_4$ happens with negligible probability.

From equations (2), (8), (9), (10) and (12), we obtain:

$$\beta'_4 = a_1 a_2 \beta_3 + a_1 r_2 \beta_1 + a_2 r_1 \beta_2 \tag{13}$$

$\mathcal{T}$'s view includes the points $A + g_1.G_1$, $B + g_2.G_2$, U and V and the group elements $\alpha_1, \alpha_2, \alpha_3$ and α_4. Therefore, $\mathcal{T}$'s view is entirely determined by $(\beta_1, \beta_2, \beta_3, \beta_4, u, v, r)$, where r is the randomness used by $\mathcal{T}$. Moreover, given $(\beta_1, \beta_2, \beta_3, \beta_4, u, v, r)$, the element (a_1, a_2) is uniformly distributed over $\mathbb{Z}_p^2$.

From equations (6), (7) and (13), we obtain:

$$\beta'_4 = a_1 a_2 (\beta_3 - b\beta_1 - a\beta_2) + a_1(v\beta_1) + a_2(u\beta_2)$$

Lemma 1. *Let p be a prime integer and let $a, b, c, d \in \mathbb{Z}$ such that $(a, b, c) \neq (0, 0, 0)$. Then the number of solutions $(x, y) \in \mathbb{Z}_p^2$ to the polynomial equation $a \cdot xy + b \cdot x + c \cdot y + d = 0 \mod p$ is at most $2p - 1$.*

Proof. The proof is straightforward and is therefore omitted.

Since $u, v \neq 0$, then $\beta_3 \neq 0$ implies $(\beta_3 - b\beta_1 - a\beta_2, v\beta_1, u\beta_2) \neq (0,0,0)$. Then using the previous lemma, for any $\gamma \in \mathbb{Z}_p$, the probability over $(a_1, a_2) \in \mathbb{Z}_p^2$ that $\beta_4' = \gamma$ is such that:

$$\Pr[\beta_4' = \gamma] \leq \frac{2p-1}{p^2} \leq \frac{2}{p}$$

Therefore, if $\beta_3 \neq 0$, the probability that $\beta_4' = \beta_4$ is at most $2/p$.

Since $u = 0$ or $v = 0$ with probability at most $2/p$, we conclude that if $e_{AB} \neq e(A, B)$, then $\mathcal{C}$ outputs $\perp$, except with probability at most $4/p$. □

Note that the security of the protocol is not based on any computational assumptions; namely the protocol achieves *unconditional security*.

4.3 Efficiency

Our protocol requires a total of four scalar multiplications in $\mathcal{G}_1$ and four in $\mathcal{G}_2$, and a total of ten exponentiations in $\mathcal{G}_T$. Our protocol is actually a one-round protocol since the four pairing queries can be performed in the same round.

5 Efficient Variants

In this section, we describe more efficient variants of our protocol, when one of the points A and B or both are already publicly known.

For example, when decrypting a Boneh-Franklin ciphertext [1], the point A is the user's private key, and B is some part of the ciphertext. Therefore, B is already publicly known and does not need to be protected. Moreover, when encrypting with Boneh and Franklin's scheme, A is the trusted party's public-key, and B is the recipient's identity. Therefore, both A and B are already publicly known and don't need to be protected.

When B is publicly known, the definition of the secrecy property is modified by simply giving B to the simulator. When both A and B are public, the secrecy property is not necessary anymore.

5.1 Secure Pairing Delegation with Public B

The protocol is the same as the protocol described in the previous section, except that we can take $g_2 = 0$ since point B does not need to be protected.

1. $\mathcal{C}$ generates a random $g_1 \in \mathbb{Z}_p$ and queries the three following pairings from $\mathcal{T}$:

$$\alpha_1 = e(A + g_1.G_1, G_2), \quad \alpha_2 = e(G_1, B), \quad \alpha_3 = e(A + g_1.G_1, B)$$

2. $\mathcal{C}$ checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Should this test fail, $\mathcal{C}$ outputs $\perp$ and halts.
3. $\mathcal{C}$ computes a purported value for $e(A, B)$:
$$e_{AB} = (\alpha_2)^{-g_1} \cdot \alpha_3 \tag{14}$$
4. $\mathcal{C}$ generates four random values $a_1, r_1, a_2, r_2 \in \mathbb{Z}_p$ and queries the pairing:
$$\alpha_4 = e(a_1.A + r_1.G_1, a_2.B + r_2.G_2)$$
5. $\mathcal{C}$ computes:
$$\alpha'_4 = (e_{AB})^{a_1 a_2} \cdot (\alpha_1)^{a_1 r_2} \cdot (\alpha_2)^{a_2 r_1} \cdot e(G_1, G_2)^{r_1 r_2 - a_1 g_1 r_2} \tag{15}$$
and checks that $\alpha'_4 = \alpha_4$. In this case, $\mathcal{C}$ outputs e_{AB}; otherwise it outputs $\perp$.

The protocol is more efficient than the protocol of Section 4 since only three scalar multiplications in $\mathcal{G}_1$ and $\mathcal{G}_2$, and eight exponentiations in $\mathcal{G}_T$ are required.

Theorem 2. *The previous protocol with public B is a secure pairing delegation protocol.*

Proof. The proof is similar to the proof of theorem 1 and is therefore omitted.

5.2 Secure Pairing Delegation with Public A and B

The protocol is similar to the previous protocol except that we can also take $g_1 = 0$ since A does not need to be protected.

1. $\mathcal{C}$ queries the three following pairings from $\mathcal{T}$:
$$\alpha_1 = e(A, G_2), \quad \alpha_2 = e(G_1, B), \quad \alpha_3 = e(A, B)$$
2. $\mathcal{C}$ checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Should this test fail, $\mathcal{C}$ outputs $\perp$ and halts.
3. $\mathcal{C}$ computes a purported value for $e(A, B)$:
$$e_{AB} = \alpha_3$$
4. $\mathcal{C}$ generates four random values $a_1, r_1, a_2, r_2 \in \mathbb{Z}_p$ and queries the pairing:
$$\alpha_4 = e(a_1.A + r_1.G_1, a_2.B + r_2.G_2)$$
5. $\mathcal{C}$ computes:
$$\alpha'_4 = (e_{AB})^{a_1 a_2} \cdot (\alpha_1)^{a_1 r_2} \cdot (\alpha_2)^{a_2 r_1} \cdot e(G_1, G_2)^{r_1 r_2}$$
and checks that $\alpha'_4 = \alpha_4$. In this case, $\mathcal{C}$ outputs e_{AB}; otherwise it outputs $\perp$.

The protocol is more efficient than the protocol of Section 4 since only two scalar multiplications in $\mathcal{G}_1$ and $\mathcal{G}_2$, and seven exponentiations in $\mathcal{G}_T$ are required.

Theorem 3. *The previous protocol with public A and B is a secure pairing delegation protocol.*

Proof. The proof is similar to the proof of theorem 1 and is therefore omitted.

6 A Different Strategy for Constant Known Points

As pointed out in section 5 it is common to calculate the IBE encryption pairing as $e(A,B)^r$, where A is the trusted parties public key and B is the recipients public key, rather than $e(rA,B)$ since pairing exponentiation is quicker than point scalar multiplication. We noted earlier that $e(A,B)$ is not a secret value, both of the points A and B are public knowledge. Furthermore, the point A is used to encrypt to any member of this IBE domain. Because of this, we call it a constant point.

We do not have to protect these points or the result of the pairing - if we off-load this computation to a terminal we only have to verify that the correct value has been returned. A random point $Q \in \mathcal{G}_2$ and a value $\alpha_c = e(A,Q)$ are stored on $\mathcal{C}$. It should be noted here that A, the constant point, is used in the construction of α_c. Q and α_c are not available outside $\mathcal{C}$.

1. $\mathcal{C}$ generates a random $x \in \mathbb{Z}_p^*$ and sends $\mathcal{T}$ the element of $\mathcal{G}_2, xS$, where $S = B - Q$.
2. $\mathcal{T}$ computes the purported values

$$\alpha_1 = e(A, xS)$$
$$\alpha_2 = e(A, B)$$

   and returns these to $\mathcal{C}$
3. $\mathcal{C}$ computes:

$$\alpha_{1c} = \alpha_1^{x^{-1}} \tag{16}$$

4. $\mathcal{C}$ does the equality checks $\alpha_{1c} \cdot \alpha_c \stackrel{?}{=} \alpha_2$ and $\alpha_2^p \stackrel{?}{=} 1$. If these checks are true then $\mathcal{C}$ returns α_2 as the result of the pairing e_{AB}, otherwise it outputs $\perp$ and halts.

Efficiency: The above protocol only requires one point scalar multiplication, two pairing exponentiations and one pairing multiplication.

6.1 Pairing with one constant known secret point

A special case - Boneh & Franklin IBE decryption: In the previous section we looked at pairings whereby all of the points were public, and we gave a real-life application, B&F IBE encryption. We now look at B&F IBE decryption. In IBE decryption we have to keep one point secret, the recipients private key, which, for consistancy with the previous section we will denote B. This pairing also includes a public point, moreover, there is redundancy in the ciphertext to make sure that the ciphertext was not tampered with in transit. We can use this to our advantage - it can be used to determine a cheating $\mathcal{T}$. We

call this a special case, as it allows us to do a reduced amount of computation. We still however need to prevent $\mathcal{T}$ from determining the result of the pairing. A is the $\mathcal{G}_1$ element of the B&F IBE ciphertext.

1. $\mathcal{C}$ generates a random value $x \in \mathbb{Z}_p^*$.
2. $\mathcal{C}$ computes xB where B is the private key, and passes this value to $\mathcal{T}$.
3. $\mathcal{T}$ computes the following:
$$\alpha = e(A, xB) \tag{17}$$
4. $\mathcal{C}$ computes the following:
$$\alpha_c = \alpha^{x^{-1}} \tag{18}$$
5. $\mathcal{C}$ now, instead of doing any validation check on the actual pairing, just uses this value in traditional IBE decryption. If the pairing value supplied was not correct (i.e. the ciphertext was tampered with) then this simply results in IBE decryption failing.

Efficiency: The above protocol only requires one point scalar multiplication and two pairing exponentiations.

6.2 Other Scenarios

We saw in the previous section that we were able to exploit the redundancy in the IBE ciphertext to allow us to determine if $\mathcal{T}$ had cheated in it dealings with $\mathcal{C}$. However, we are not always this fortunate. We now look at situations where we have to determine using the pairing values alone, whether $\mathcal{T}$ has cheated. We look at the scenario above where the element of $\mathcal{G}_1$ is public, the element of $\mathcal{G}_2$ is secret and we may or may not want to keep the result of the pairing secret. The two schemes of section 6 and subsection 6.1 can be easily modified to create a new scheme that has the properties of both. This protocol, for the pairing of the points A and B would then proceed as follows, with the values $\alpha_c = e(Q, B)$ stored on $\mathcal{C}$. Again α_c is generated from the known constant point and a random point. α_c, Q and B are not available off the card.

1. $\mathcal{C}$ calculates the value $S = A - Q$. $\mathcal{C}$ then generates the random values $x, y, z \in \mathbb{Z}_p^*$ and calculates the xS, yB and zB. These values are passed to $\mathcal{T}$.
2. $\mathcal{T}$ uses these values to calculate the following pairings:
$$\alpha_1 = e(xS, yB)$$
$$\alpha_2 = e(A, zB)$$
3. $\mathcal{T}$ now returns the calculated values α_1 and α_2 to $\mathcal{C}$.
4. $\mathcal{C}$ now calculates:
$$\alpha_{1c} = \alpha_1^{xy^{-1}}$$
$$\alpha_{2c} = \alpha_2^{z^{-1}}$$

5. $\mathcal{C}$ now does the following validation checks, $\alpha_{1c} \cdot \alpha_c \stackrel{?}{=} \alpha_{2c}$ and $\alpha_{2c}^p \stackrel{?}{=} 1$. If this test is passed, $\mathcal{C}$ returns α_{2c} as the result of the pairing e_{AB}, otherwise it returns $\perp$ and halts.

It's obvious how the above protocol could be adapted to blind both points by $\mathcal{C}$ doing a random multiplication of A, then passing this blinded value to T, and then a doing a subsequent pairing exponentiation by the inverse when the pairings are recieved back from $\mathcal{T}$.

Efficiency: The above protocol only requires three point scalar multiplication, three pairing exponentiations and one pairing multiplication (or 4, 3 and 1 for the variation with both points blinded).

7 Security Proof

Theorem 4. *The scheme proposed in section 6.2 is a secure pairing delegation.*

Proof. The scheme has the properties of completeness, secrecy and correctness as outlined in section 3.

Completeness: After completing the protocol with an honest terminal $\mathcal{C}$ obtains the values e_{AB}. This follows from the bilinearity of the pairing, $\alpha_c = e(Q, B)$ and the fact that $\mathcal{C}$ deliberately constructs A as $A = S + Q$, that is $(S = A - Q)$:

$$\begin{aligned}
\alpha_1^{(xy)^{-1}} \cdot \alpha_c &= \alpha_2^{z^{-1}} \\
e(xS, yB)^{(xy)^{-1}} \cdot \alpha_c &= e(A, zB)^{z^{-1}} \\
e(S, B) \cdot \alpha_c &= e(A, B) \\
e(S, B) \cdot e(Q, B) &= e(A, B) \\
e(S + Q, B) &= e(A, B)
\end{aligned}$$

Secrecy: As in section 5.1 we assume that one of the points in the pairing is public knowledge.
The terminal $\mathcal{T}$ views the points xS, yB and zB. However since S is a generator of $\mathcal{G}_1$ and B is a generator of $\mathcal{G}_2$, and the nonces x, y and z are random $\in \mathbb{Z}_p^*$, then xS, yB and zB are randomly distributed in $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. The simulator $\mathcal{S}$ simply consists of running $\mathcal{T}$ with randomly generated points in $\mathcal{G}_1$ and $\mathcal{G}_2$. The simulators output and $\mathcal{T}$'s view when interacting with $\mathcal{C}$ are indistinguishable.

Correctness: $\mathcal{C}$ will return a result e_{AB} if the validation checks succeed, otherwise it will return $\perp$. For a cheating $\mathcal{T}$, $\mathcal{C}$ will return a result e_{AB} with the same probability that $\mathcal{T}$ can give to $\mathcal{C}$ two elements of $\mathcal{G}_T$ such that the validity check 19 holds.

$$\alpha_2^{z^{-1}} / \alpha_1^{(xy)^{-1}} \stackrel{?}{=} \alpha_c \tag{19}$$

If we write α_1 as p^γ, α_2 as p^κ and α_c as p^ξ for unknown γ,κ and ξ, and p a generator of $\mathcal{G}_T$. Then we have the following relationship:

$$\begin{aligned} p^{\kappa z^{-1}} / p^{\gamma (xy)^{-1}} &\stackrel{?}{=} p^\xi \\ p^{\kappa z^{-1} - \gamma (xy)^{-1}} &\stackrel{?}{=} p^\xi \\ \kappa z^{-1} - \gamma (xy)^{-1} &\stackrel{?}{=} \xi \\ \kappa &\stackrel{?}{=} (\xi + \gamma (xy)^{-1}) z \\ \gamma &\stackrel{?}{=} (\kappa z^{-1} - \xi)(xy) \\ 0 &\stackrel{?}{=} \gamma - (\kappa z^{-1} - \xi)(xy) \end{aligned} \tag{20}$$

Since Q is random and unknown to the attacker, then p^ξ is random and unknown to the attacker. Also, we know that the equation evaluates to an element $\in \mathcal{G}_T$. Therefore, for each value of p^κ there is exactly one value for p^γ for which the equation holds (as shown above). This value is entirely determined by the random nonces in the system (x, y, z) and therefore is uniformly distributed in $\mathcal{G}_T$.

Lemma 2. *The total number of valid solutions to an equation of the form* $0 = \gamma - (\kappa z^{-1} - \xi)(xy)$*, for fixed* x, y, z *and* $\xi \in \mathbb{Z}_p^*$ *is* $(p-1)$*.*

Proof. γ *can take any value in* $\mathbb{Z}_p^*$*, that is* $(p-1)$ *different values. The corresponding value of* κ *is given by* $\kappa = (\xi + \gamma(xy)^{-1})z$*, see equation 20.*

The number of valid solutions to the above equation is $(p-1)$. The total number of possible solutions for $(p^{\kappa'}, p^{\gamma'}) \in \mathcal{G}_T^2$ is $(p-1)^2$ (γ taking $(p-1)$ different values and κ taking $(p-1)$ different values). This leads to the following, where $(p^{\kappa'}, p^{\gamma'})$ is an attempt to break the system, but where x, y, z and ξ are unknown.

$$Pr[\kappa' = (\xi + \gamma'(xy)^{-1})z] \leq \frac{(p-1)}{(p-1)^2} \leq \frac{1}{(p-1)}$$

The probability of $\mathcal{T}$ returning to $\mathcal{C}$ a tuple $(p^{\kappa'}, p^{\gamma'})$ that was not generated according to the protocol, such that $\mathcal{C}$ does not return $\perp$ is negligible. □

8 Conclusion

In this paper we described a simple protocol for secure delegation of elliptic-curve pairing. Our protocol allows a computationally limited device (for example a smart-card) to delegate the computation of the pairing $e(A, B)$ to a more powerful device (for example a PC), in such a way that:

1. the powerful device learns nothing about the points being paired (A and B) nor the pairing's result $e(A, B)$,
2. and the limited device is able to detect when the powerful device is cheating.

We have also described more efficient variants of our protocol when one of the points or both are already known, and further efficiency gains when constant points are used.

Our protocols achieve unconditional security. An interesting research direction would be to speed-up the protocols by trading-off unconditional security against computational security.

References

1. D. Boneh and M. Franklin, *Identity based encryption from the Weil pairing*, SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003.Proceeding of Crypto '2001, Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, pp. 213-229, 2001.
2. D. Boneh, H. Shacham and B. Lynn, *Short signatures from the Weil pairing*. Proceedings of Asiacrypt '01, Lecture Notes in Computer Science, vol. 2248, Springer-Verlag, pp. 514-532, 2001.
3. D. Boneh and X. Boyen, *Short Signatures Without Random Oracles*. Proceedings of Eurocrypt 2004, Lecture Notes in Computer Science, vol. 3027, pp. 56-73, 2004.
4. R. Canetti, *Security and Composition of Multiparty Cryptographic Protocols*, Journal of Cryptology, (2000) 13: pp. 143–202.
5. A. Joux, *A one round protocol for tripartite Diffie-Hellman*. Proceedings of ANTS IV, Lecture Notes in Computer Science, vol. 1838, pp. 385–394. Springer-Verlag, 2000.
6. A. Menezes, *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
7. M. Scott and P. Barreto, *Compressed Pairings*, Proceedings of Crypto 2004, Lecture Notes in Computer Science, vol. 3152, pp. 140–156 2004.