On the Implementation of a Pairing-based Cryptographic Protocol in a Constrained Device

Sébastien Canard[1], Nicolas Desmoulins[1], Julien Devigne[1,2], and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] UCBN, GREYC, Caen, France

Abstract. In this paper, we consider a pairing-based cryptographic protocol and the way to implement it on a restricted device such as a mobile phone or a smart card. Our aim is to show the different ways to do it, regarding (i) the capacity for the restricted device to implement a bilinear pairing and/or (ii) the performance regarding the implemented bilinear pairing. We show that there are different possibilities and study the security and efficiency of each of them. To illustrate our purpose, we make use of the Boneh-Boyen-Shacham group signature, which needs one on-line pairing computation.

1 Introduction

When operating in devices with restricted capabilities w.r.t. space, memory and computing performance, the implementation of some cryptographic algorithms sometimes need to be further studied. In these cases, it is important to find tricks to optimize the implementation until performance is acceptable by the customer. This is in particular the case when the studied cryptographic algorithm includes the use of one or several bilinear pairings.

In fact, bilinear pairings are today not studied enough to be embedded into any mobile phone or smart card, as it is the case for *e.g.* RSA or EC-DSA. Then, when one has to embed a pairing-based cryptographic algorithm onto *e.g.* a SIM card for mobile phones, one has to make some choices on the way to implement the whole algorithm, and in particular the bilinear pairing itself. In this paper, we study several possibilities, giving for each of them *pros and cons*.

To illustrate the different possibilities we have studied, we take in this paper the case of the implementation of the BBS group signature scheme [2] either in a mobile phone connected to the outside world, or in a SIM card connected to a mobile phone. Informally, in a group signature scheme [5], any member of the group can sign a document and any verifier can confirm that the signature has been computed by a member of the group. Moreover, group signatures are anonymous and unlinkable for every verifier except, in case of a dispute, for a given authority that knows some special information. In 2004, Boneh, Boyen and Shacham [2] have proposed a short group signature based on the use of a bilinear pairing (especially by the group member, during the group signature process).

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

The group member is now represented by a mobile phone or a SIM card and is connected to some device (a PC or a mobile phone resp.).

The paper is organized as follows. In Section 2, we introduce our study by giving the description of the BBS group signature and a summary of the performances one can obtain regarding mathematical operations related to a bilinear pairing. In Section 3, we explain how one can prevent the group member (the mobile phone or the SIM card) to produce a bilinear pairing by replacing such operation by exponentiation in $\mathbb{G}_T$ and the expected results. In Section 4, we show how the computation of a bilinear pairing can be delegated to some more powerful delegate (the connected device).

2 Introduction to Our Study

In this section, we introduce our study by describing the BBS group signature scheme and then giving some implementation results regarding pairings and related groups.

2.1 The BBS Group Signature Scheme

Boneh, Boyen and Shacham have proposed at CRYPTO 2004 a short group signature scheme [2] based on the Strong Diffie-Hellman and the Decisional Linear assumptions[3].

KEY GENERATION. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p and let g_1 (resp. g_2) be a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear map such that for all $(a, b) \in \mathbb{G}_1 \times \mathbb{G}_2$ and all $\alpha, \beta \in \mathbb{Z}$, $e([\alpha]a, [\beta]b) = e(a, b)^{\alpha\beta}$ and $e(g_1, g_2) \neq 1$.

Let $h \in \mathbb{G}_1$, $\zeta_1, \zeta_2 \in \mathbb{Z}_p^*$ and $u, v \in \mathbb{G}_1$ such that $[\zeta_1]u = [\zeta_2]v = h$. Let $\gamma \in \mathbb{Z}_p^*$ and $w = [\gamma]g_2$. Then, the tuple (ζ_1, ζ_2) composes the secret values to open (*.i.e* revoke the anonymity) a signature, γ is the secret key to add group members and $(p, g_1, g_2, h, u, v, w)$ is the whole group public key.

Each group member obtains from the group manager a tuple $(A, x) \in \mathbb{G}_1 \times \mathbb{Z}_p^*$ such that $A = [1/(\gamma + x)]g_1$. This couple verifies $e(A, w + [x]g_2) = e(g_1, g_2)$. The value x is the member's secret and A is the key used to retrieve the identity of a member in case of opening (*i.e.* anonymat revocation).

GROUP SIGNATURE GENERATION. On input a message m and a tuple (A, x), a group signature is executed as follows:

- choose at random $\alpha, \beta \in \mathbb{Z}_p$ and compute $T_1 = [\alpha]u$, $T_2 = [\beta]v$ and $T_3 = A + [\alpha + \beta]h$;
- compute $\delta_1 = x\alpha$ and $\delta_2 = x\beta$;
- choose at random $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p$;

[3] We used the additive notation for group laws in $\mathbb{G}_1$ and $\mathbb{G}_2$, which are elliptic curve groups.

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

– compute $t_1 = [r_\alpha]u$, $t_2 = [r_\beta]v$, $t_3 = [r_x]T_1 + [-r_{\delta_1}]u$, $t_4 = [r_x]T_2 + [-r_{\delta_2}]v$ and

$$t_5 = e(T_3, g_2)^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}; \quad (1)$$

– compute $c = \mathcal{H}(m\|T_1\|T_2\|T_3\|t_1\|t_2\|t_3\|t_4\|t_5)$;
– compute $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_x = r_x + cx$, $s_{\delta_1} = r_{\delta_1} + c\delta_1$ and $s_{\delta_2} = r_{\delta_2} + c\delta_2$.

A group signature is $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. Here (T_1, T_2, T_3) is a linear encryption of A (see [2] for such encryption scheme) and $(c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ is a proof of knowledge of a valid certificate (using the Fiat-Shamir heuristic [7]). The verification step consists then in verifying this proof of knowledge, using standard techniques, and the open procedure is the decryption of the linear encryption.

Implementation. We now focus on the group signature procedure and the way to implement it on *e.g.* the mobile phone of the group member.

As $e(h, w)$ and $e(h, g_2)$ depend only on public values, these pairings can be precomputed by *e.g.* the group manager (and directly put in the whole group public key), it only remains one additional bilinear pairing to compute: $e(T_3, g_2)$. As a conclusion, the group member should perform 7 random generations, 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, 1 pairing evaluation and a few operations in $\mathbb{G}_1$, $\mathbb{G}_T$ and $\mathbb{Z}_p$ (which will be neglected in the following).

2.2 Implementation of a Bilinear Pairing

The security level implies minimal sizes for r (the size of the elliptic curve subgroup in which pairing operands live) and p^k (the size of the finite field which receives pairing outputs). The integers r and p are prime numbers, and k is the *embedding degree*. We have developed our own bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with a Barreto-Naerhig elliptic curve E of equation $Y^2 = X^3 + 5$ over $\mathbb{F}_p$ (with p a prime number). More precisely, we have:

– $\mathbb{G}_1 = E[r](\mathbb{F}_p)$, where $E[r]$ denotes the r-torsion group;
– $\mathbb{G}_2 = E[r](\mathbb{F}_p) \cup \mathrm{Ker}(\pi_p - [p]) \subseteq E[r](\mathbb{F}_{p^k})$, where $\pi_p : E \longrightarrow E$ is the Frobenius endomorphism; and
– $\mathbb{G}_T = \mu_r \subset \mathbb{F}_{p^k}^*$ where μ_r is the group of r-th roots of unity.

We have then chosen a 128-bits security level, which gives us $\log_2 r = 256$ and $\log_2 p^k = 3248$. For $k = 12$, we obtain $|p| = |r| = 256$.

Some optimizations. To accelerate the computation of pairings, some optimizations were used. In particular, we make use of the results given in [6] for some general low level optimizations (in particular the use of Jacobian coordinates and the joint point-and-line computation).

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

We have then implemented the Ate pairing as described in [1], which permits us to boost the pairing computation. Our results are given in Table 1 for the Samsung Galaxy S2 smartphone with a Dual-core Exynos 4210 1.2GHz processor ARM Cortex-A9 with the Android OS, v2.3 (Gingerbread).

operation	notation	time computation (in ms)
Scalar multiplication in $\mathbb{G}_1 = E[r](\mathbb{F}_p)$	ϵ	5.7
Exponentiation in $\mathbb{G}_T = \mathbb{F}^*_{p^{12}}$	ζ	42
Ate pairing e	ψ	63

Table 1. Our implementation benchmark

Dealing with multi-scalar multiplications. The BBS scheme moreover needs to implement the multi-scalar multiplication in $\mathbb{G}_1$ (resp. multi-exponentiation in $\mathbb{G}_T$), which are the most costly operations. One solution to improve multi-scalar multiplication (resp. multi-exponentiation) is to use the generalization of the Shamir's trick which is presented in [9]. In that case, the computation of $c = \sum_{i=1}^{\ell} [e_i] g_i$ (resp. $c = \prod_{i=1}^{\ell} g_i^{e_i}$) is improved since it is not necessary to compute each scalar multiplication (resp. modular exponentiation) and add (resp. multiply) the results since c can be computed globally. Using such trick, the computation of c necessitates approximatively $\frac{2^{\ell+1}-1}{3\times 2^{\ell-1}}$ times the cost of a scalar multiplication (resp. modular exponentiation).

Thus, for a triple scalar multiplication in $\mathbb{G}_1$ (resp. modular exponentiation in $\mathbb{G}_T$), the expected time complexity is approximately 1.25ϵ (resp. 1.25ζ).

2.3 BBS on a Restricted Device

Using the above benchmark for pairings and operations in the different used groups, the whole BBS group signature represents the following complexity (neglecting random generation and scalar multiplications in $\mathbb{Z}_p$): 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and 1 pairing, that is approximately (using the generalization of Shamir's trick) $7.3\epsilon + 1.25\zeta + \psi$. Using our above benchmark, we obtain an estimate of approximately 157 ms for this solution.

3 From a Bilinear Pairing to an Exponentiation in $\mathbb{G}_T$

We now give one possibility to implement such group signature, which is in particular explained in the paragraph "Performance" of Section 6 in [2].

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

3.1 Removing the Bilinear Pairing

One possibility is to consider that the group manager, when generating the tuple (A, x) for the group member, already pre-computes $\Lambda = e(A, g_2)$ and gives it to the group member. Then, using Λ and assuming that $\tilde{w} = e(h, w)$ and $\tilde{h} = e(h, g_2)$ are already precomputed (see above), we see easily that

$$\Lambda \tilde{h}^{\alpha+\beta} = e(A, g_2)e(h, g_2)^{\alpha+\beta} = e(A + [\alpha + \beta]h, g_2) = e(T_3, g_2), \qquad (2)$$

which corresponds to the result we need. Then, Equation (1) becomes (in $\mathbb{G}_T$)

$$t_5 = \Lambda^{r_x} \tilde{w}^{-r_\alpha - r_\beta} \tilde{h}^{r_x(\alpha+\beta) - r_{\delta_1} - r_{\delta_2}}$$

which is a new way for the group member (the mobile phone) to compute t_5.

As a result, using such technique, the group member has now to perform 7 random generations, 5 exponentiations in $\mathbb{G}_1$, 2 double exponentiations in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and *no pairing evaluations*.

3.2 Pros and Cons

Implementation. The main advantage of this method is that it is not necessary to embed a pairing on the mobile phone, since the phone does not have to compute any more pairing. However, in practice, this advantage is not as important as it is since the mobile needs to perform multi-exponentiations in $\mathbb{G}_T$ and in $\mathbb{G}_1$. Thus, it is necessary to implement the algebraic structure of a bilinear pairing, without implementing a bilinear pairing. Thus, regarding the pure implementation aspects, the gain is not very important.

Efficiency. Regarding efficiency, the computation of the bilinear pairing plus a triple exponentiation is replaced by only a triple exponentiation (and no pairing!). The other operations are unchanged, except for some extra operations in $\mathbb{Z}_p$ which we neglect throughout the text (e.g. computing $r_x(\alpha + \beta) - r_{\delta_1} - r_{\delta_2}$).

In total, we obtain: 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$ and 1 triple exponentiation in $\mathbb{G}_T$, that is $7.3\epsilon + 1.25\zeta$. Using our above benchmark, we obtain an estimate of 94 ms for this solution.

Remark 1. As the most costly operation for a pairing is the final exponentiation in $\mathbb{G}_T$, the gain is not always as important as it can be (for other schemes than BBS). The different optimizations for this final exponentiation need to be compared to the existing methods regarding multi-exponentiation (see above). Then, depending on the number of components in the multi-exponentiations, the results regarding efficiency can be different.

4 Delegating the Pairing Computation

We here give another possibility which consists in delegating the computation of the bilinear pairings to a more powerful entity.

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

4.1 How to Delegate

We first remark that the pairing computation which we focus on is $e(T_3, g_2)$, where T_3 is part of the output group signature (and is consequently a public value since the group signature is public) and g_2 is a public parameter. As our pairing needs no secret key and as it takes on input public values, the output is also public (any verifier can compute it after the reception of a group signature). Our idea is then to delegate this computation to another entity. This entity can correspond to *e.g.* a more powerful laptop, or some kind of dedicated server (*e.g.* a *cloud* for cryptographic operations) where it is easier to implement a fast bilinear pairing. Another example is when the true group member corresponds to the SIM card while the external helper is the mobile phone.

After having computed T_3 (see above in Section 2.1), the mobile phone can send it to this powerful entity which computes $\tilde{t} = e(T_3, g_2)$ and sends the results to the mobile phone. In this case, Equation (1) becomes (in $\mathbb{G}_T$)

$$t_5 = \tilde{t}^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}. \tag{3}$$

In this case, the mobile phone has to perform 7 random generations, 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and *no pairing evaluations*.

4.2 Pros and Cons

Implementation. Again, in this solution, there is no need to implement a bilinear pairing in the mobile phone. Again also, it is still necessary to implement most of the algebraic structure of the bilinear pairing. Thus, regarding the pure implementation aspects, the gain is still not very important.

Efficiency. Regarding efficiency, the computation of the bilinear pairing plus a triple exponentiation is replaced by the sole triple exponentiation. However, we need to take into account the additional communication steps of such method (the computation of the bilinear pairing by the powerful entity is not taken into account as it can be executed in parallel with other computations performed by the mobile phone). In the previous methods, there is only one communication step of the whole group signature. Here, we add an additional communication for sending and receiving T_3 and $\tilde{t}$ respectively.

Practically speaking, we obtain (again neglecting random generation and scalar multiplications): 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$ and 1 triple exponentiation in $\mathbb{G}_T$, which corresponds exactly to the above time complexity. This time, we also only need to compute scalar addition (and no scalar multiplication as for the previous solution).

However, this does not include the additional communication. In practice, we can approximate the communication between a mobile phone and the exterior or between the SIM card and the mobile phone to 200 kbits/s (and thus approximately 17 ms for the communication in our case). Using an 3G/UMTS

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

communication, the resulting rate can reach 2 Mbits/s, which makes the communication step negligible.

FACING CORRUPTED DELEGATE. One additional problem with this method is that the powerful delegate can send to the mobile phone a wrong value $\tilde{t}$ so that the resulting group signature will be rejected. There exists in the literature verifiable delegation of cryptographic operation, but, to the best of our knowledge, no work has been done regarding pairings.

However, in some practical cases, this "attack" is not useful. In particular, if the group signature generated by the mobile signature necessarily goes throw this delegate for the final sending to the true verifier, this one can easily send to the verifier anything it wants to make reject the group signature. If such group signature is used for *e.g.* access control, the customer will see that she can not access the place she wants and thus detect that something is wrong.

Remark 2. In some cases, such as for the identity-based encryption scheme of Boneh-Franklin [3], the pairing evaluation includes a secret value, which may make this method impossible. In fact, as proposed by Lefranc and Girault [8], there exists some delegation technique for this case. For example, if one wants to compute $e(a, b)$ where a is secret and b is public, one possibility is for the mobile phone to compute $c = [\alpha]a$, where α is random, and for the delegate $d = e(c, b)$. The result $e(a, b)$ is then $d^{1/\alpha}$.

5 Conclusion

With our practical results, it seems that the second solution, which consists in replacing the bilinear pairing by operations in $\mathbb{G}_T$, is the best one. This also shows that the optimizations regarding operations in $\mathbb{G}_T$ are very important, as well as the communication rate between *e.g.* the mobile phone and the exterior. Note finally that the delegation technique can be extended to other operations related to the BBS signature scheme, such as proposed in [4], which can make the last solution better in some particular cases.

Acknowledgments

We are very grateful to Tanja Lange for her useful comments and suggestions.

References

1. Jean-Luc Beuchat, Jorge Enrique González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves. In *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2010.
2. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.

3. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
4. Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer, and Olivier Pereira. Group Signatures are Suitable for Constrained Devices. In *ICISC*, volume 6829 of *Lecture Notes in Computer Science*, pages 133–150, 2010.
5. David Chaum and Eugène van Heyst. Group Signatures. In *EUROCRYPT'91*, pages 257–265, 1991.
6. Zhaohui Cheng and Manos Nistazakis. Implementing Pairing-Based Cryptosystems. In *Proceedings of IWWST-2005*, 2005.
7. Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
8. Marc Girault and David Lefranc. Server-Aided Verification: Theory and Practice. In *ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 605–623. Springer, 2005.
9. Bodo Möller. Algorithms for Multi-Exponentiation. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 165–180, London, UK, 2001. Springer-Verlag.

Appeared in M. Abdalla and T. Lange (Eds.): Pairing 2012 – industrial track, volume to appear of LNCS.