Improved Broadcast Encryption Scheme with Constant-Size Ciphertext

Renaud Dubois[1], Aurore Guillevic[1,2], and Marine Sengelin Le Breton[1]

[1] Thales Communications and Security
160 Boulevard de Valmy – BP 82 – 92704 Colombes Cedex – France
renaud.dubois@thalesgroup.com
aurore.guillevic@thalesgroup.com
marine.sengelin@gmail.com
[2] Crypto Team, DI, ENS – 45 rue d'Ulm – 75230 Paris Cedex 05 – France

Abstract. The Boneh-Gentry-Waters (BGW) [4] scheme is one of the most efficient broadcast encryption scheme regarding the overhead size. This performance relies on the use of a *pairing*. Hence this protocol can benefit from public key improvements. The ciphertext is of constant size, whatever the proportion of revoked users is. The main lasting constraint is the computation time at receiver end as it depends on the number of revoked users. In this paper we describe two modifications to improve the BGW bandwidth and time complexity. First we rewrite the protocol and its security proof with an asymmetric pairing over the Barreto-Naehrig (BN) curves instead of a symmetric one over supersingular curves. This modification leads to a practical gain of 60% in speed and 84% in bandwidth. The second tweaks allows to reduce the computation time from $O(n-r)$ to $\min(O(r), O(n-r))$ for the worst case (and better for the average case). We give performance measures of our implementation for a 128-bit security level of the modified protocol on a smartphone.

Keywords: Broadcast encryption, asymmetric pairings, Barreto-Naehrig curves, Android.

1 Introduction

A broadcast encryption scheme is a protocol allowing a broadcaster to send messages to a large set $\mathcal{U}$ of *users* or receivers, $n = \#\mathcal{U}$. The set evolves at each session in a dynamic way such that the broadcaster may choose any subset $\mathcal{S}$ of *privileged users* or *members* from $\mathcal{U}$. $\mathcal{R}$ is the set of revoked users (non-members), $r = \#\mathcal{R}$. A broadcast protocol is secure under (t, n)-collusion if for all subset $\mathcal{R} \subset \mathcal{U}$ with $r \leqslant t$, the revoked users from $\mathcal{R}$ are not able to decipher. Fully collusion-secure protocols are mostly appreciated. The classical application of broadcast encryption protocols are pay-TV systems, broadcast of content and over the air re-keying mechanisms (OTAR) in radio systems. The content or payload is encrypted under a private key. Then the private key is encrypted in a manner described by the chosen protocol. An *overhead* is added to the encrypted data. It contains the encrypted session key such that only the members can decipher it. In most broadcast protocols, a description of the members (or equivalently of the revoked users) is also added to the overhead. The system constraints are

- the **bandwidth consumption**, related to the overhead size ω,
- the sender computation time τ_s, public key (resp. secret key) memory PK_s (resp. SK_s),
- the **users** (receivers) **computation time** τ_u, public key (resp. secret key) memory PK_u (resp. SK_u).

The naive solution assigns a different private key to each user. The overhead contains the session key encrypted with each private key corresponding to each member. The bandwidth is linear in the number of members. Many schemes have been suggested to provide an efficient broadcast. Lots of them are combinatorial tree-based schemes using the subset cover framework [12]. However the overhead size is the minimal number of primary blocks used to cover the set. In particular, for the worst case of $r = \frac{n}{2}$, i.e. half the users are revoked ones, the others are members, the overhead size is the same as in the naive solution. Boneh, Gentry and Waters introduced in [4] two versions (denoted BGW_1 and BGW_2 in the following) of a pairing

based protocol which solved this problem. The overhead size is in $O(1)$ for BGW_1 and in $O(\sqrt{n})$ in BGW_2, for n users in the system. This came at a time complexity expanse, as given in the table 1. Indeed, this protocol uses asymmetric cryptography. Delerablée, Paillier and Pointcheval described another scheme in [7], reducing the time complexity. However the implementation is more complex, as it requires to handle formal sums of points. To our knowledge, there is very few commercial products using pairings (some for IBE, see [16]), and none for broadcast. Despite there are several software and hardware pairing implementations with precise benchmarks, to our knowledge, there are not yet an entire broadcast protocol based on pairings implemented and presented with precise timings.

Reference	ω	τ_r	PK_r	SK_r
Complete Subtree [12]	$O(r\log(\frac{n}{r}))$	$O(\log\log n)$	-	$O(\log(n))$
Subset difference [12]	$O(r)$	$O(\log(n))$	-	$O(\log^2(n))$
BGW_1 [4]	$O(1)$	$O(n-r)$	$O(n)$	$O(1)$
BGW_2 [4]	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(1)$
DPP_1 [7]	$O(1)$	$O(r^2)$	$O(n)$	$O(1)$
DPP_2 [7]	$O(r)$	$O(r)$	$O(1)$	$O(1)$
this paper, Sec. 2.1	$O(1)$	$\min(O(r), O(n-r))$	$O(n)$	$O(1)$
this paper, Sec. 2.2	$O(\sqrt{n})$	$\min(O(\frac{r}{\sqrt{n}}), O(\frac{n-r}{\sqrt{n}}))$	$O(\sqrt{n})$	$O(1)$

Table 1. Complexities of well known broadcast encryption schemes

Our contributions. A practical instantiation was not explained in [4]. A straightforward implementation of the protocol uses a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. This results in quite large size for the bandwidth elements. Each element (in $\mathbb{G}$) is of size half an RSA modulus size. For a 128-bit security level, this means 1536 bits per element instead of 128 in a combinatorial tree based protocol. We propose to design BGW with an appropriate asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In this way, the elements in $\mathbb{G}_1$ have a size close to the optimal case in public key cryptography, i.e. 256 bits for the example above, rather than half a RSA modulus size. We adapt the protocol and set in the right groups $\mathbb{G}_1$ or $\mathbb{G}_2$ the different elements (public and private keys, bandwidth elements), knowing that the elements in $\mathbb{G}_1$ have the smallest size, those of $\mathbb{G}_2$ have quite medium size (at most half an RSA modulus) and those of $\mathbb{G}_T$ are close to an RSA modulus size. The resulting bandwidth consumption is divided by 6 at a 128-bit security level. We adapt accordingly the security proof.

The protocol security relies on the difficulty of a non-standard problem, the ℓ-BDHE (ℓ-Bilinear Diffie-Hellman Exponent problem). About one year after the publication in 2005 of BGW, Cheon proposed attacks in [5, 6] against the family of Diffie-Hellman related problems used in the public key based protocols, including the ℓ-BDHE. More recently at the PKC 2012 conference, an implementation of such an attack was presented at a security level of 80 bits [14]. We analyse the impact of Cheon's attacks on the size of the three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. We propose a resistant elliptic curve.

The BGW scheme relies on public key tools. Hence the computation time is quite slower than in a symmetric key based protocol, especially for decryption. We provide an efficient trade-off between memory and precomputation. Finally our practical implementation on a smartphone shows that with all our improvements, this BGW broadcast encryption scheme can be efficiently used for commercial applications.

This paper is organized as follows: in Sec. 2 we describe how BGW can benefit from the use of an asymmetric pairing and adapt the security proof. In Sec. 3, we detail our choice of a pairing-friendly elliptic curve and consider modifications due to Cheon's attacks. In Sec. 4, we describe how to use well chosen pre-computation to reduce dramatically the computation cost. Finally, in Sec. 5 we give our results of a complete implementation of the protocol on a smartphone.

2 BGW with an asymmetric pairing

Boneh, Gentry and Waters [4] describe a scheme with a minimal overhead. The scheme use a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The pairing definition and properties can be found in [3, Ch. IX]. We will use the additive notation for both $\mathbb{G}_1$ and $\mathbb{G}_2$ and the multiplicative notation for $\mathbb{G}_T$. In the original paper, the scheme is described with a symmetric pairing: $\mathbb{G}_1 = \mathbb{G}_2$ that is, we can swap the inputs $e(P, Q) = e(Q, P)$. In practice, the third group $\mathbb{G}_T$ is a finite field extension of the form $(\mathbb{F}_{p^k})^*$, of size $k \log p$ an RSA modulus. To use a symmetric pairing, supersingular elliptic curves shall be used, which is inefficient. $\mathbb{G}_1$ and $\mathbb{G}_2$ have the same size, an explicit isomorphism exists between these two groups and their size is half the size of $\mathbb{G}_T$ (in large characteristic). For a justification, see Sec. 3. We propose to adapt the scheme to an asymmetric pairing in order to have a group $\mathbb{G}_1$ with smaller coefficients. We reorganise the elements and set in $\mathbb{G}_1$ those on which the bandwidth depends. Let n be the total number of users and r the number of revoked users. To remove confusion with the finite field characteristic (used later) commonly denoted p, we will denote by m the groups order.

2.1 First version of the scheme

We start by re-writing the special case where the ciphertexts and private keys are of constant size. The n users are considered globally. The number of revoked users is r hence $n - r$ users must be able to decipher.

Setup (n) Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order m with an asymmetric pairing e from $\mathbb{G}_1 \times \mathbb{G}_2$ into $\mathbb{G}_T$. Let P a random generator for $\mathbb{G}_1$ and Q for $\mathbb{G}_2$. Let α a random element in $\mathbb{Z}_m$. Compute $P_i = \alpha^i P \in \mathbb{G}_1$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Compute $Q_i = \alpha^i Q \in \mathbb{G}_2$ for $i = 1, 2, \ldots, n$. Pick a random $\gamma \leftarrow \mathbb{Z}_m$ and set $V = \gamma P \in \mathbb{G}_1$. The broadcaster public key is

$$\mathrm{PK}_s = (P, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, V, Q, Q_1) \in \mathbb{G}_1^{2n+1} \times \mathbb{G}_2^2 \ .$$

Each user i receives an added public key Q_i. The additional public key $(Q_1, \ldots, Q_n) \in \mathbb{G}_2^n$ is dispatched among all users. The complete public key PK is in $\mathbb{G}_1^{2n+1} \times \mathbb{G}_2^{n+1}$. The secret key for user i is $\mathrm{SK}_{u,i} = D_i = \gamma P_i \in \mathbb{G}_1$; its public key is $\mathrm{PK}_{u,i} = \big(Q_i,\ (P_i)_{1 \leqslant i \leqslant 2n,\ i \neq n+1}\big)$. Let $\mathcal{S} = \mathcal{U} \setminus \mathcal{R}$ the subset of authorized users, $\#\mathcal{S} = n - r$.

Encrypt $(\mathcal{S}, \mathrm{PK}_s)$ Pick a random $k \leftarrow \mathbb{Z}_m$ and set $K = e(P_{n+1}, Q)^k = e(P_n, Q_1)^k \in \mathbb{G}_T$. Set

$$\mathrm{Hdr} = \left(kQ,\ k\Big(V + \sum_{j \in \mathcal{S}} P_{n+1-j}\Big)\right) \in \mathbb{G}_2 \times \mathbb{G}_1$$

and output (Hdr, K).

Decrypt $(i, \mathcal{S}, \mathrm{Hdr}, D_i, \mathrm{PK}_{u,i})$ Let $\mathrm{Hdr} = (C_0, C_1)$. The i-th user computes

$$K = \frac{e(C_1,\ Q_i)}{e\Big(D_i + \sum_{\substack{j \in \mathcal{S} \\ j \neq i}} P_{n+1-j+i},\ C_0\Big)}$$

The verification uses the relation $e([i]P, [j]Q) = e(P, Q)^{ij} = e([j]P, [i]Q)$. We have chosen to set C_1 in $\mathbb{G}_1$ to save bandwidth, as the elements in $\mathbb{G}_1$ have coefficients a least twice as small as those in $\mathbb{G}_2$. It would be great to set C_0 in $\mathbb{G}_1$ as for C_1. Unfortunately in this case the user would have to compute the sum over all authorized users in $\mathbb{G}_2$ which is more time consuming than in $\mathbb{G}_1$. The storage size needed for a user i would be increased too. Our chosen trade-off will appear more natural through the generalized version of the scheme.

2.2 General scheme

To reduce the public key size, the n users are organized into A groups of B users with $AB \geqslant n$. In [4] the authors suggest to choose $B = \lfloor \sqrt{n} \rfloor$ and $A = \lceil \frac{n}{B} \rceil$. We can also divide users into groups according to their country, subscription or other criterion due to the system (Pay-TV, OTAR). A user i is referenced by its group number (say a) and its range in that group (say b). Hence $i = \{a, b\}$ with $1 \leqslant a \leqslant A$ and $1 \leqslant b \leqslant B$. The Hdr will contain A public elements (instead of a unique C_1), each one dedicated to a determined group of users. Here we see relevant to set all these elements in $\mathbb{G}_1$. There is still the C_0 element that we need to set in $\mathbb{G}_2$ in order to keep in $\mathbb{G}_1$ the user public and private keys and a part of the decryption.

Setup $_B(n)$ Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q$ as above. Let α a random element in $\mathbb{Z}_m$. Compute $P_i = \alpha^i P \in \mathbb{G}_1$ for $i = 1, 2, \ldots, B, B+2, \ldots, 2B$; this is the common public key. In each group of users, the user $i = \{a, b\}$ receives the set of (P_i) and an additional public key $Q_b = \alpha^b Q \in \mathbb{G}_2$. Then pick uniformly at random $\gamma_1, \gamma_2, \ldots, \gamma_A \leftarrow \mathbb{Z}_m$ and set $V_1 = \gamma_1 P, \ldots, V_A = \gamma_A P \in \mathbb{G}_1$. The centralized public key is $\mathrm{PK}_s = (P, P_1, P_2, \ldots, P_B, P_{B+2}, \ldots, P_{2B}, V_1, \ldots, V_A, Q, Q_1) \in \mathbb{G}_1^{2B+A} \times \mathbb{G}_2^2$. The secret key for the user number b in the group a is $\mathrm{SK}_{u,\{a,b\}} = D_{a,b} = \gamma_a P_b \in \mathbb{G}_1$. Its public key is $\mathrm{PK}_{u,\{a,b\}} = \big(Q_b, (P_i)_{1 \leqslant i \leqslant 2B,\ i \neq B+1}\big)$. The user doesn't need the others Q_ℓ hence to save memory on his constrained device (e.g. smartphone, set-up box) we don't add them. Note that this scheme is relevant even for unbalanced group sizes. For large groups, the time computation will increase, but the bandwidth consumption will be the same : one group element (in $\mathbb{G}_1$) per group of users, whatever the size of the group is.

Encrypt $(\mathcal{S}, \mathrm{PK}_s)$ For each group a of users, denote by $\mathcal{S}_a$ the set of authorized users in this group. Pick a random k in $\mathbb{Z}_m$ and set $K = e(P_{B+1}, Q)^k = e(P_B, Q_1)^k \in \mathbb{G}_T$. Set Hdr $(\in \mathbb{G}_2 \times \mathbb{G}_1^A) =$

$$\Big(kQ,\ k\big(V_1 + \sum_{j \in \mathcal{S}_1} P_{B+1-j}\big),\ k\big(V_2 + \sum_{j \in \mathcal{S}_2} P_{B+1-j}\big), \ldots,\ k\big(V_A + \sum_{j \in \mathcal{S}_A} P_{B+1-j}\big)\Big)$$

Decrypt $\big(i = \{a, b\}, \mathcal{S}_a, \mathrm{Hdr}, \mathrm{SK}_{u,\{a,b\}}, \mathrm{PK}_{u,\{a,b\}}\big)$ Let denote Hdr $= (C_0, C_1, \ldots, C_A)$ and recall that a user i is number b in the group a. The user $i = \{a, b\}$ computes

$$K = \frac{e(C_a, Q_b)}{e\big(D_{a,b} + \sum_{\substack{j \in \mathcal{S}_a \\ j \neq b}} P_{B+1-j+b},\ C_0\big)} \ .$$

The verification uses the same bilinearity property as previously:

$$e([i]P, [j]Q) = e(P, Q)^{ij} = e([j]P, [i]Q) \ .$$

Table 2 gives the protocol complexity with an asymmetric pairing, BGW_1 denotes the one instance version, BGW_2 denotes the parallel instance version. ω is the bandwidth consumption, PK_s denotes the sender's memory for the public key, τ_s the time computation and respectively PK_u, τ_u denote the receiver's ones. r_a is the number of revoked users in the group a. Note that they are at most B users in a group a.

Schema	ω	PK_s	τ_s	PK_r	τ_r
BGW_1	$\mathbb{G}_2 \times \mathbb{G}_1$	$\mathbb{G}_1^{2n+1} \times \mathbb{G}_2^{n+1}$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$	$\mathbb{G}_1^{2n-1} \times \mathbb{G}_2$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$
BGW_2	$\mathbb{G}_2 \times \mathbb{G}_1^A$	$\mathbb{G}_1^{2B+A} \times \mathbb{G}_2^{B+1}$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$	$\mathbb{G}_1^{2B-1} \times \mathbb{G}_2$	$(B-r_a)\mathrm{Add}_{\mathbb{G}_1}$

Table 2. Theoretical complexity for BGW protocol, asymmetric pairing

2.3 Security proof

In [4, §3.3], the authors prove the semantic security of the general system. We faced some trouble when adapting the security proof to an asymmetric pairing in the setting above. We need to add a copy in $\mathbb{G}_2$ of the inputs elements in $\mathbb{G}_1$ to the problem. This difficulty rises in the challenge phase. To generate a consistent input for the adversary, the challenger must have a copy in $\mathbb{G}_2$ of the inputs in $\mathbb{G}_1$. This is transparent with a symmetric pairing (in which case an isomorphism from $\mathbb{G}_1$ into $\mathbb{G}_2$ is available). This is also quite easy if an isomorphism from $\mathbb{G}_2$ into $\mathbb{G}_1$ is available. More precisely, let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ three cyclic groups of prime order together with an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let P a generator for $\mathbb{G}_1$ and Q for $\mathbb{G}_2$. Let $Q^{'}$ a random element in $\mathbb{G}_2$. In the challenge phase, the challenger must compute a corresponding $P^{'} \in \mathbb{G}_1$ such that $\log_P(P^{'}) = \log_Q(Q^{'})$ without knowing $\log_Q(Q^{'})$. In other words, in this construction there is some $k \in \mathbb{Z}_m$ such that $Q^{'} = [k]Q$ and we have to find a corresponding $P^{'} \in \mathbb{G}_1$ such that $P^{'} = [k]P$ with the same $k \in \mathbb{Z}_m$, without knowing k. Therefore we need an isomorphism ϕ which maps the generator $Q \in \mathbb{G}_2$ downto $P \in \mathbb{G}_1$. Hence $\phi(Q^{'}) = P^{'}$. In this way we can end the security proof as in the original paper. Such a map usually does not exists for ordinary pairing-friendly elliptic curves. For supersingular curves, there is the distortion map from $\mathbb{G}_1$ to $\mathbb{G}_2$ which provides a symmetric pairing. For ordinary elliptic curves, the trace map [3, IX.7.4] is degenerated, as $\mathbb{G}_2$ is commonly built as the trace-zero subgroup. With the notations from [10], the security proof must be written assuming that the pairing is of Type 3 : $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Hence the adversary needs to receive $P^{'}$, that is why it must appears in the challenger inputs.

Let start with an asymmetric variant of ℓ-BDHE problem:

Definition 1 (l-BDHEasym). *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ three cyclic groups of prime order together with an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Given $(P, P_1, \ldots, P_\ell, P_{\ell+2}, \ldots, P_{2\ell}) \in \mathbb{G}_1^{2\ell}$, $(Q, Q_1, \ldots, Q_\ell) \in \mathbb{G}_2^{\ell+1}$ such that $P_i = [\alpha^i]P$, $Q_i = [\alpha^i]Q$, and $(P^{'}, Q^{'}) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $\log_P P^{'} = \log_Q Q^{'}$, compute*

$$e(P_{\ell+1}, Q^{'}) \text{ which is the same as } e(P^{'}, Q_{\ell+1}) \ .$$

Definition 2 (Decisional l-BDHEasym). *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ three cyclic groups of prime order together with an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $\mathbf{y}_{P,Q,\alpha,\ell} = (P_1, P_2, \ldots, P_\ell, P_{\ell+2}, \ldots, P_{2\ell}, Q_1, Q_2, \ldots, Q_\ell)$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ε in solving the decision ℓ-BDHEasym in $\mathbb{G}_T$ if $\Big| Pr[\mathcal{B}(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,\ell}, e(P_{\ell+1}, Q^{'})) = 0] - Pr[\mathcal{B}(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,\ell}, T) = 0] \Big| \geqslant \varepsilon$ where the probability is over the random choice of generators $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, of random point $P^{'} \in \mathbb{G}_1$, the random choice of $\alpha \in \mathbb{Z}_m$, the random choice of $T \in \mathbb{G}_T$ and the random bits consumed by $\mathcal{B}$. The distribution on the left is denoted by $\mathcal{P}_{BDHEasym}$ and the distribution on the right by $\mathcal{R}_{BDHEasym}$.*

The decision (t, ε, ℓ)-BDHEasym assumption holds in $\mathbb{G}_T$ if no t-time algorithm has advantage at least ε in solving the decision ℓ-BDHE problem in $\mathbb{G}_T$.

According to the definitions in [13], BGW and the variants presented here are asymmetric broadcast encryption with a static set of users (the joint is made at setup only) and stateless users (the public and private keys do not evolve from a session to another). A selective security for key indistinguishability is proven (the target set is chosen before the setup phase).

Suppose there exists a t-time adversary, $\mathcal{A}$, who receives an instance of the protocol. The adversary is able to distinguish between a valid and a random session key with advantage $\mathrm{AdvBr}_{\mathcal{A},B} > \varepsilon$ for a system parametrized with a given B. One build an algorithm, $\mathcal{B}$, that has advantage ε in solving the decision $B-$BDHEasym problem in $\mathbb{G}_T$.

Algorithm $\mathcal{B}$ takes as input a random decision $B-$BDHEasym challenge $(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,B}, Z)$ where $\mathbf{y}_{P,Q,\alpha,B} = (P_1, P_2, \ldots, P_B, P_{B+2}, \ldots, P_{2B}, Q_1, Q_2, \ldots, Q_B)$ and Z is either $e(P_{B+1}, Q^{'})$ or a random element in $\mathbb{G}_T$. The aim of $\mathcal{B}$ is to decide if Z is valid or random. For doing that, $\mathcal{B}$ simulates a session of the broadcast protocol and submits it to $\mathcal{A}$. Then $\mathcal{B}$ uses $\mathcal{A}$'s answer to decide if Z is valid or random. Algorithm $\mathcal{B}$ proceeds as follows.

Init. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives the set $\mathcal{S} = \cup_{1 \leqslant a \leqslant A} \mathcal{S}_a$ of users that $\mathcal{A}$ wishes to be challenged on.

Setup. $\mathcal{B}$ needs to generate a public key PK and private keys D_i for users $i \notin \mathcal{S}$. We can use the same idea as in the original proof. Algorithm $\mathcal{B}$ chooses uniformly at random $u_a \in \mathbb{Z}_m$ for $1 \leqslant a \leqslant A$. The users are divided into A groups of at most B users. A user i is number b in a precise group a. For $a = 1, \ldots, A$, algorithm $\mathcal{B}$ sets $V_a = [u_a]P - \sum_{j \in \mathcal{S}_a} P_{B+1-j}$. It gives $\mathcal{A}$ the public key

$$\mathrm{PK} = (P_1, \ldots, P_B, P_{B+2}, \ldots, P_{2B},\ Q_1, \ldots, Q_B,\ V_1, \ldots, V_A)$$

which is in $\mathbb{G}_1^{2B-1} \times \mathbb{G}_2^B \times \mathbb{G}_1^A$.

Boneh, Gentry and Waters note in their paper [4] that since P, α and the u_a values are chosen uniformly at random, the public key

$$\mathrm{PK}_{original} = (P, P_1, \ldots, P_B, P_{B+2}, \ldots, P_{2B},\ V_1, \ldots, V_A) \in \mathbb{G}_1^{2B+A}$$

has an identical distribution to that in the actual construction. Here it is necessary to give $(Q_1, \ldots, Q_B) \in \mathbb{G}_2^B$ too. If we assume that P is a generator chosen at random in $\mathbb{G}_1$ and Q (which generates $\mathbb{G}_2$) is also chosen at random and independent from P, we can consider that all these elements are uniformly distributed at random.

Next the adversary needs all private keys that are not in the target set $\mathcal{S}$. For each user $i = \{a, b\} \notin \mathcal{S}$, algorithm $\mathcal{B}$ computes the corresponding private key

$$D_{a,b} = [u_a]P_b - \sum_{j \in \mathcal{S}_a} P_{B+1-j+b}\ .$$

The same equality holds as in the original proof

$$D_{a,b} = [u_a][\alpha^b]P - [\alpha^b] \sum_{j \in \mathcal{S}_a} P_{B+1-j} = [\alpha^b]V_a\ .$$

The authors in [4] note that the unknown value P_{B+1} is not involved in the sum, as i is a revoked user ($i = \{a, b\}$ with $b \notin \mathcal{S}_a$).

Challenge. To generate the challenge, $\mathcal{B}$ computes Hdr as

$$(Q^{'}, [u_1]P^{'}, \ldots, [u_A]P^{'})\ .$$

$\mathcal{B}$ then randomly chooses a bit $b \in \{0, 1\}$ and sets $K_b = Z$ and picks a random K_{1-b} in $\mathbb{G}_T$. It gives (Hdr, K_0, K_1) as the challenge to $\mathcal{A}$.

We use the same justification as in the above cited paper. The algorithm knows both $Q^{'}$ and $P^{'}$ such that $\log_P(P^{'}) = \log_Q(Q^{'})$ hence can compute a valid Hdr. When the input to $\mathcal{B}$ is a B-BDHEasym tuple, $Z = e(P_{B+1}, Q^{'})$ and (Hdr, K_0, K_1) is a valid challenge to $\mathcal{A}$ as in a real attack. Let k such that $P^{'} = [k]P$. $P^{'}$ and $Q^{'}$ are bound together in the sense that $P^{'} = [k]P$ and $Q^{'} = [k]Q$ with the same $k \in \mathbb{Z}_m$. $[u_a]P^{'} = [k][u_a]P = [k]\big([u_a]P - \sum_{j \in \mathcal{S}_a} P_{B+1-j} + \sum_{j \in \mathcal{S}_a} P_{B+1-j}\big) = [k]\big(V_a + \sum_{j \in \mathcal{S}_a} P_{B+1-j}\big)$. We can see in this form that $(Q^{'}, [u_1]P^{'}, \ldots, [u_A]P^{'})$ is a valid encryption of the key $e(P_{B+1}, Q)^k$. Then $e(P_{B+1}, Q)^k = e(P_{B+1}, Q^{'}) = Z = K_b$. Hence (Hdr, K_0, K_1) is a valid challenge to $\mathcal{A}$. On the other and, when the input to $\mathcal{B}$ is a random tuple, Z is a random element from $\mathbb{G}_T$, and K_0, K_1 are random elements from $\mathbb{G}_T$.

Guess This last step is the same as in the paper [4]. The adversary $\mathcal{A}$ outputs a guess $b^{'}$ of b. If $b = b^{'}$ the algorithm $\mathcal{B}$ outputs 0, i.e. it guesses that $Z = e(P_{B+1}, Q^{'})$. Otherwise, it outputs 1, i.e. Z is a random element in $\mathbb{G}_T$. If $(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,B}, Z)$ is sampled from $\mathcal{R}_{\mathrm{BDHEasym}}$ then $\Pr\big[\mathcal{B}(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,B}, Z) = 0\big] = 1/2$. If $(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,B}, Z)$ is sampled from $\mathcal{P}_{\mathrm{BDHEasym}}$ then

$$\Big|\Pr\Big[\mathcal{B}(P, Q, P^{'}, Q^{'}, \mathbf{y}_{P,Q,\alpha,B}, Z) = 0\Big] - 1/2\Big| = \mathrm{AvdBr}_{\mathcal{A},B} \geqslant \varepsilon\ .$$

It follows that $\mathcal{B}$ has advantage at least ε in solving B-BDHEasym problem in $\mathbb{G}_T$. This conclude the security proof.

2.4 Diffie-Hellman problem and variants

The security relies on the ℓ-Bilinear Diffie-Hellman Exponent assumption which is a weaker problem than the Diffie-Hellman one. The difficulty of this problem was first studied in [5]. See also improvements in [11, 6] and the implementation in [14].

Theorem 1 ([11, Theorem 1']). *Let P be an element of prime order m in an abelian group $\mathbb{G}$. Suppose that d is a positive divisor of $m-1$. If $P, [\alpha]P, [\alpha^d]P$ are given, α can be computed within $O(\sqrt{m/d}+\sqrt{d})$ group operations using space for $O(\max(\sqrt{m/d}, \sqrt{d}))$ groups elements.*

Theorem 2 ([11, Theorem 2']). *Let P be an element of prime order m in an abelian group $\mathbb{G}$. Suppose that d is a positive divisor of $m+1$ and $[\alpha^i]P$ are given for $1 \leqslant i \leqslant 2d$. Then α can be computed within $O(\sqrt{m/d}+d)$ group operations using space for $O(\max(\sqrt{m/d}, \sqrt{d}))$ groups elements.*

The main idea for the first theorem is to find a divisor d of $m-1$ in the range $2 \leqslant d \leqslant B$ or $B+2 \leqslant d \leqslant 2B$ to reduce the complexity from $O(\sqrt{m})$ to $O(\sqrt{m/d}+\sqrt{d})$. A decomposition of the classical Baby Step Giant Step (BSGS) algorithm in two phases reduces the complexity of BSGS from $O(\sqrt{m})$ to two BSGS running, the first in $O(\sqrt{m/d})$ and the second in $O(\sqrt{d})$. We have to take into account this attack to choose properly a convenient elliptic curve when setting the system parameters.

1. We can enlarge the parameters in order to prevent the system from these attacks and match the previously chosen security level. Assuming that $B \ll m$, we consider that the attack is in at most $O(\sqrt{m/2B})$. For a 128-bit security level, instead of a prime order group $\mathbb{G}_1$ of size $\log m = 256$, we have to set $\log m = 256 + \log(2B)$. If the system is designed for 10^6 users and $B \approx 10^3$, enlarging $\log m$ with at least 12 bits is enough and quite cheap if it does not affect considerably the size of $\mathbb{G}_T$.
2. If enlarging m with a few bits will enlarge the size of $\mathbb{G}_T$ of a few hundred bits, we may prefer to choose directly a safe prime order m, such that $m-1$ and $m+1$ are not divisible by factors smaller than $2B$. Of course either $m-1$ or $m+1$ will be a multiple of 4 but we loose only 2 bits.

3 Choice of the pairing-friendly elliptic curve

The two instantiations are the Weil pairing and the Tate pairing over elliptic curves (defined over finite fields). They can be quite efficiently computed with the algorithm due to Miller. Definitions and properties of Weil and Tate pairings can be found in the survey [3, Ch. IX]. Let p be a large prime and $E(\mathbb{F}_p)$ an elliptic curve defined by a reduced Weierstraß equation $y^2 = x^3 + ax + b$. Remember that $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of prime order m of the elliptic curve and $\mathbb{G}_T$ is the multiplicative group of an extension field $\mathbb{F}_{p^k}$. The main difficulty is to find suitable elliptic curves for pairings. An almost exhaustive study of known pairing-friendly elliptic curves can be found in [9]. If the protocol relies exclusively on the Diffie-Hellman problem, to achieve the same complexity in the three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ we must choose carefully the size of the groups as following. Up to now, only generic attacks such as Pohlig-Hellman exists for solving the Diffie-Hellman problem in an elliptic curve[3]. If the protocol relies exclusively on the Diffie-Hellman problem, for a N-bit security level, prime order groups $\mathbb{G}_1, \mathbb{G}_2$ of size $\log m = 2N$ bits are convenient. The group $\mathbb{G}_T \simeq \mathbb{F}_{p^k}^*$ is exposed to the less difficult index calculus attack. Hence the size $k \log p$ of $\mathbb{G}_T$ is greater than those of $\mathbb{G}_1$ and $\mathbb{G}_2$. An RSA modulus size is commonly considered to be safe. The following key-size int Tab. 3 are recommended by the ECRYPT II research group [8, Tab. 7.2].

We have chosen a 128-bit security level. A supersingular curve (over a prime field in large characteristic) has an embedding degree k at most 2 resulting in $\log p = 1624$, $\log m = 256 + \delta$ and $\rho = \log p / \log m \approx 6$. The notation $+\delta$ means that enlarging m by a few bits will not impact on $\log p$, hence on the size of $\mathbb{F}_{p^k}$. The well-known Barreto-Naehrig curves (BN, [2]) fit almost exactly the recommended sizes of $\mathbb{G}_1$ and $\mathbb{G}_T$, taken into account Cheon's attack. Indeed, for these curves, $k = 12$ and $\log m = \log p$. Hence with $k \log p = 3264$

[3] ordinary, over a prime field in large characteristic, of trace $\not\equiv 0 \mod p$ and $\neq 1$

Security (bits)	RSA	Discrete Logarithm		Elliptic curve
		field	subfield	
80	1248	1248	160	160
112	2432	2432	224	224
128	**3248**	**3248**	**256**	**256**
160	5312	5312	320	320
192	7936	7936	384	384
256	15424	15424	512	512

Table 3. Ecrypt II key-size recommendations

and $\log m = \log p = 272$, the parameters are strong enough against the ℓ-BDHE problem for a 128-bit security level and a BGW protocol with at most $2B$ users per group and $\log(2B) = 16$.

If we prefer to follow NIST recommendations, the $k = 12$ embedding degree is exactly what we need : $\log m = 256$ and as $\rho = \log p / \log m = 1.0$, $k \log p = 3072$ as expected. In particular, for a 128 bit security level, using an asymmetric pairing decreases the size of the element in the group $\mathbb{G}_1$ by a factor of 6. To prevent from Cheon's attacks, we can increase the size of m by 12 bits but it result in increasing the size of $\mathbb{F}_{p^k}$ by 144 bits. To avoid this, we must generate a strong BN curve, without any integer d dividing m and less than 2^{12}. We heard about this attack after launching the prototype development. Hence the benchmarks were computed for this curve.

$x =$ `- 0x4000000000000031C` (defines p, m and t)

$p =$ `0x24000000000006FE700000000082705C800000043937699E80000D20DA314BD9`

$m =$ `0x24000000000006FE700000000082705C200000043937604A80000D20D9F74979`

$t =$ `0x60000000000009540000000003A0261`

$b =$ `0x17`

The elliptic curve defined over the prime field $\mathbb{F}_p$ with parameter equation $a = 0$ and b above has prime order m and trace t. The three numbers x, $m-1$ and $m+1$ are smooth.

$x = 2^2 * 5^2 * 43 * 139 * 757 * 10192497083$,

$m - 1 = 2^3 * 3 * 5^2 * 23 * 43 * 71 * 139 * 757 * 338172217 * 10192497083$
$*1065629744969022147085838680434831409024186859$,

$m + 1 = 2 * 7 * 11 * 31 * 67 * 179 * 1297 * 839731 * 15999517 * 282551569$
$*35836294153183 * 251224184937629 * 6415963443272843$.

Assuming that there are around 2^{10} users per group,$\log(2B) \leqslant 12$ and the security for this curve is 116 bits instead of 128. Then we heard about Cheon's attack and tried to find a "strong" curve. Because of the parameter structure, the curve order m is such that 12 divides $m-1$ and 2 divides $m+1$. We ran a search over almost prime x to find an m such that no divisor less than 2^{12} divides either $m-1$ or $m+1$ (except 12 for $m-1$ and ones less than 16 for $m+1$). We found a few appropriate curves, for example

$x =$ `0x4000000000087F7F` $= 248861 * 18531172093771$

$p =$ `0x2400000000131EDE500003CEEC974A28964D2C8BEE1F7C511355420E690A2713`

$m =$ `0x2400000000131EDE500003CEEC974A28364D2C8BEE05FDD41355405D1C6EA10D`

$m - 1 = x*12 * 757798571 * 431644596110779526675237 * 899539747440060915487289$

$m + 1 = 2 * 480707 * 420180967 * 107234028019 * 1416027609325038349$
$*265454606642679936569002939766381$

$t =$ `0x6000000000197E7D000001B14C9B8607`

$b =$ `0xC`

For this curve, $12 \mid m-1$ and the next divisor is 248861; $2 \mid m+1$ and the next divisor is 480707. Because of the 12, we loose 4 bits. A bypass would be to index the users from 13 instead of 1. Our implementation doesn't depends on a particular p or m hence changing their value will not infer on the timings if their size remains the same.

The possible choice are presented in Tab. 4. The notation $+\delta$ means that enlarging m by a few bits will not impact on $\log p$, hence on the size of $\mathbb{F}_{p^k}$.

Recommendations	Curve	k	$k \log p$	$\log p$	$\rho = \log p / \log m$	$\log m$
Ecrypt II	Supersingular	2	3248	1624	not fixed, ≈ 6 here	$256 + \delta$
	Barreto-Naehrig	12	3264	272	1.0	272
NIST	Supersingular	2	3072	1536	not fixed, ≈ 6 here	$256 + \delta$
	Strong BN	12	3072	256	1.0	256

Table 4. Parameters size depending on the embedding degree

Improving pairing computation is not the scope of this paper, that is why we will not discuss on a precise choice of pairing over Tate, ate or twisted-ate ones. We are more interested in filling the gap between protocol descriptions and practical implementation of them. So we use the quite generic and well known Tate pairing. Another variant can be chosen without any change in the protocol settings. Our timings are not very fast but competitive, see Tab. 5. Our implementation is generic and this is very useful in this context: we can choose a strong order m, taken into account Cheon's attack, without needing to rewrite some parts of the pairing implementation.

Curve	k	$\log m$	$\log p$	Miller's Loop	Exponentiation	Pairing
Supersingular	2	256	1536	29.88 ms	25.99 ms	55.87 ms
Barreto-Naehrig	12	256	256	14.51 ms	5.18 ms	19.69 ms

Table 5. Our Implementation of pairing computation on a AMD64 3Ghz (Ubuntu 10.10)

4 Reducing Time Complexity

The public keys are points on an elliptic curve hence addition is as cheap as subtraction. If the number of revoked users is small ($r \ll n/2$), the initial computation in $O(n - r)$ is quite slow. We can instead consider that the value $\Sigma_{n,i} = \sum_{1 \leqslant j \neq i \leqslant n} P_{n+1-j+i}$ is precomputed for each user i. Then

$$S = \Sigma_{n,i} - \sum_{j \in \mathcal{R}} P_{n+1-j+i}$$

with $\mathcal{R}$ the set of revoked users. Now the complexity is $O(\min(r, n - r))$ (where O is the cost of a point addition, EllAdd). We can do better with a precomputed tree.

4.1 Binary public key tree precomputation

In this section we describe how to decrease the computation time from $O(\min(r, n - r))$ using only twice memory. The tweak consists in two modifications.

1. Modify the public key into a binary public key tree T twice long obtained by
   - sorting all users in a binary tree whose leaves are the users;
   - precomputing for each node the sum of each public key of the nodes below.

2. For each encryption and decryption, choose the optimal including/excluding tree to compute the sum. For example, for each decryption, use Alg. 1 if $r < n/2$ or its variant if $r \geqslant n/2$ to compute the value of the sum S.

Let consider a user i in a system of at most n users. This user needs the elements $P_{n+1-j+i}$, $1 \leqslant j \neq i \leqslant n$ of the public key $P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}$ that is, $n-1$ elements in $\mathbb{G}_1$. He needs also $Q_i \in \mathbb{G}_2$ (which does not need to appear in the tree). Each user computes a different (translated by i) tree. We assume that the nodes are labelled in the same way for each user. The difference from a user to another is only the initialization of the leaf values.

Example 1 (Precomputing the tree). Suppose that $n = 16$. The user $i = 9$ computes the tree represented in Fig. 1. The leaves are the $P_{n+1-j+9}$ with $1 \leqslant j \neq i \leqslant n$. We represent two lines : the users and the $n+1-j+9$ index. For each node in the tree, the user computes the sum of the two children. The value $P_{n+1} = P_{17}$ is missing, the user puts $\mathcal{O}$ instead on the corresponding leaf. The user 9 does not need the values $P_1, \ldots, P_9$ and $P_{26}, \ldots, P_{32}$. The value stored at node 31 is the sum of all public keys from P_{10} to P_{25}, except P_{17} (replaced by $\mathcal{O}$). The value stored at node 25 is the sum of the public keys P_{22} to P_{25}.

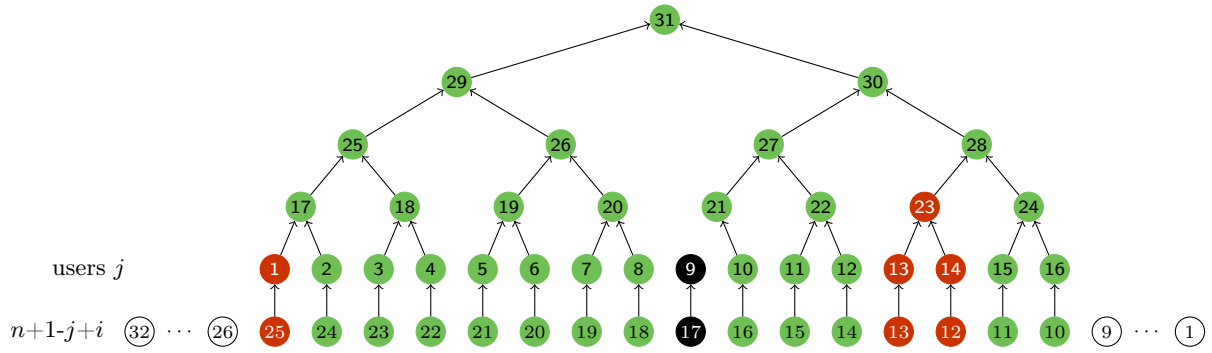


Fig. 1. Public keys and precomputation with $n = 16$, for user $i = 9$

Algorithm 1 Improved computation of S when $r < n/2$

INPUT: The user ID i, the set of privileged users $\mathcal{S}$, the precomputed public tree T (for user i)
OUTPUT: The sum of points on the elliptic curve $S = \sum_{j \in \mathcal{S}, j \neq i} P_{n+1-j+i}$

Let T' be the binary tree whose nodes are those of T.
for each node of T', from the leafs do the root **do**
    if it is the leaf of an authorized user or if there exists a green node below **then**
        color the node in green
    else
        color the node in red
    end if
end for
$S \leftarrow T_{root}$
for each red node with a green parent **do**
    subtract the related public value from S
end for
return S

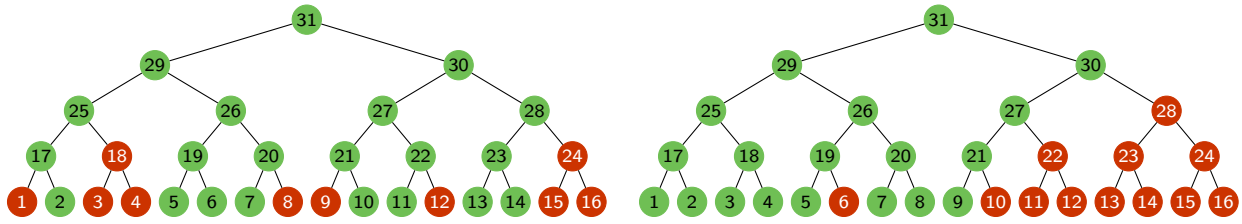
Example 2 (Computing S quickly with the tree). Consider the same set of $n = 16$ users, indexed from 1 to 16. Assume that at the current session, the users 1, 13 and 14 are revoked ($r = 3 < n/2 = 8$). They are in red on Fig. 1. For user 9, using algorithm 1, the sum S is computed by summing the following elements of T:

$$S = T_{31} - T_1 - T_{23}$$

Note that a subtraction is as cheap as an addition on an elliptic curve. The resulting cost is only 2 ElAdd, while it would have been 13 on the original scheme.

When $r > n/2$, we can do the same reasoning but instead of covering the revoked users and subtracting the corresponding public keys from $\Sigma_{n,i}$, we cover the members and add the corresponding public keys, starting with $\mathcal{O}$. In [4] the authors propose to store the previous sum S from a session to the next, subtract the new revoked users and add the no-longer revoked ones. This is efficient only if the proportion of newly revoked and re-authorized users is very small.

4.2 Complexity analysis



Random distribution Ordered distribution

Example 3 (Computing S quickly in a tree). Consider a set of $n = 16$ users, indexed from 1 to 16 as illustrated in two above figures (revoked users are black). For user '2' compute the key session by subtracting the value in node '1,3,4,8,9,12,24' thus the cost is 6 EllAdd. Note that in the figure on the right (sorted tree) the cost is only 3 EllAdd (subtract node '30' and '6', add node '9'). Cost would have been 8 in the original scheme.

Algorithm 1 has something common with the Subset Cover computation. However here there is no need to store extra secrets elements, as the difference of the subset is done by a simple elliptic point subtraction. It is obvious that the number of operations is always lower than the number r of revoked users in Alg. 1 and lower than the number of members $n - r$ in its variant ($r > n/2$). It can be equal in the worst case: in this case S is just a difference (r operations) or just a sum ($n - r$ operations).

The average case is hard to analyse [1] as it strongly depends on the distribution of revoked users in the tree. When r or $(n - r)$ is small, with a uniform distribution, the complexity will be close to it. In practice the users are sorted by behaviour so that nodes that are close are mostly to be revoked together. In a real world application the behaviour is the subscription date or product. However some random revocations (rare events) appear with compromising, expirations, etc.

5 Implementation on a smartphone

For any implementation a trade-off between specificity (using a sparse modulus for quick reduction, using very specific curves) and performances has to be done. We chose to develop a very generic library in C language which can use any modulus and any type of pairing-friendly elliptic curve in Weiertraß representation over a prime finite field (i.e. in large characteristic). The BN curves and supersingular curves have been implemented. The library LibCryptoLCH [15] is a proprietary industrial library using a modular approach as in OpenSSL. It implements arithmetic over $\mathbb{F}_p$ using Montgomery multiplication, elliptic curve computation over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ using the modified Jacobian coordinates. The pairing computation is specific for each $\mathbb{F}_{p^k}$ field. The construction of the extension field $\mathbb{F}_{p^k}$ and its arithmetic is quite automatised by using macros in C.

We now present some computational results of our improved implementation for 128-bit security level. Our proof of concept consists in a standard PC to represent the sender, and a smartphone to represent the receiver. The smartphone can be personalized with any secret key of the system. Thus the given results are the same as would be in a real system with a million smartphones. The smartphone is a dual core 1.2 Ghz Samsung Galaxy II with Android OS. The PC is a 3Ghz Intel(R) Core(TM)2 Duo CPU with 2.9 Gio RAM. The last improvements described in Sec. 4 where unfortunately not yet implemented. The broadcaster runs the system initialization, the key attribution to a new user and the session key encryption. First, we simulate the decryption time for an authorized user on the PC to estimate the growing cost of decryption with respect to the total number of users n, see Tab. 6.

Smartphones with Android platform use the Java programming language. Thanks to the Java Native Interface, we can load the library in C language, run the decryption on the smartphone and measure its timing. For doing that, we call the `currentTimeMillis()` function of the `system` class. Results are presented in Tab. 6 and Tab. 8. We measure the worst case $r = n/2$ of BGW_2 so the improvements described in Sec. 4 are not visible. The users are divided in A parallelized groups of B users with $B = \lceil \sqrt{n} \rceil$.

Number of users n	Setup	User init.	Encryption $r = n/2$	Decryption (simulation)	Decryption (smartphone)
50000	22.15 s	0.03 s	3.58 s	1.10 s	1.44 s
100000	40.45 s	0.03 s	7.03 s	1.13 s	1.79 s
200000	1 m 16 s	0.03 s	14.72 s	1.14 s	2.08 s
500000	3 m 07 s	0.05 s	32.97 s	1.16 s	2.65 s
1000000	6 m 09 s	0.07 s	1 m 04 s	1.18 s	3.33 s
3000000	18 m 24 s	0.12 s	3 m 07 s	1.23 s	4.96 s
5000000	30 m 42 s	0.16 s	5 m 11 s	1.27 s	6.09 s

Table 6. Computation time obtained on a 3 Ghz PC (encryption) and a smartphone Samsung Galaxy SII 1.20 Ghz Android (decryption)

The decryption time depends on the total number of users and on the ratio of revoked users. The Tab. 7 and Tab. 8 show the increasing encryption and decryption times when r decreases from 87.5% to 0%.

Members / Number n of users	12.5%	25%	50%	100%
50000	2.46 s	2.62 s	3.58 s	7.17 s
100000	3.11 s	4.10 s	7.03 s	13.84 s
200000	3.74 s	7.27 s	14.72 s	26.28 s
500000	9.65 s	16.46 s	32.97 s	1 m 03 s
1000000	16.99 s	33.46 s	1 m 04 s	2 m 06 s
3000000	49.67 s	1 m 36 s	3 m 07 s	6 m 11 s
5000000	1 m 20 s	2 m 37 s	5 m 11 s	10 m 18 s

Table 7. Encryption time with respect to the authorized user percentage obtained on the 3Ghz PC

An acceptable decryption time on the smartphone must be less than 2 seconds from our point of view. Here this correspond to less than 200 000 users according to Tab. 8. For larger n, we need to reduce this time. The pairing computation is not very time consuming. The sum $\sum_{j \in \mathcal{S}_a, j \neq b} P_{B+1-j+b}$ is the most important part of the computation time. With a first trick: addition over $\mathcal{S}_a$ when $n - r \ll n$ and subtraction over $\mathcal{R}_a$ (the revoked users of group a) when $r \ll n$, the worst case of $r = n/2$ become the upper bound. This means

still at most $3.33s$ when $r = n/2$. With a precomputed tree, the average case will have faster encryption and decryption times than those presented in Tab. 8.

Number n of users \ Members	12.5%	25%	50%	100%
50000	1.18 s	1.20 s	1.44 s	1.93 s
100000	1.28 s	1.46 s	1.79 s	2.46 s
200000	1.36 s	1.60 s	2.08 s	3.03 s
500000	1.55 s	1.91 s	2.65 s	4.15 s
1000000	1.75 s	2.25 s	3.33 s	5.46 s
3000000	2.23 s	3.15 s	4.96 s	8.63 s
5000000	2.65 s	3.78 s	6.09 s	10.84 s

Table 8. Decryption time with respect to the authorized user percentage obtained on the smartphone

We manage to develop a functional prototype based on improved state-of-the-art broadcast protocol with a relative effectiveness. This provides consistent simulation time. In a real system, a dedicated Android implementation of the finite field arithmetic, the elliptic curve arithmetic and the pairing computation will certainly improve by a factor 2 or 3 our results, leading to less than 2 seconds to decipher, even for 5 000 000 users in the worth case of $r = n/2$.

6 Conclusion

We presented an improved version of BGW suitable for use with a pairing on one of the fastest pairing-friendly elliptic curves. Our presentation can be easily adapted to other well-suited pairing friendly elliptic curves. We considered the attacks on the underlying non-standard problem. We also provided time computation on a prototype, the broadcaster hosted on a standard PC and each receiver hosted on a Samsung Galaxy II smartphone with Android operating system. For large groups of users (more than 200000), the decryption time is up to 2 seconds which can be too slow. Hence we proposed improvements based on a time-memory trade-off. Because of the use of an asymmetric pairing, the public key size remains reasonable, hence doubling this size is feasible in order to reduce under 2 seconds the decryption time. We then explained and justified all our choices to use in practice in a real Pay-TV or OTAR system the BGW protocol.

Acknowledgments

This work was supported in part by the French ANR-09-VERS-016 BEST Project. The authors express their gratitude to Philippe Painchault, Emeline Hufschmitt and Damien Vergnaud for helpful comments and careful proofreading. The authors thank the anonymous reviewers of the Pairing conference for their thorough reviews and useful comments.

References

[1] P. Austrin and G. Kreitz. Lower bounds for subset cover based broadcast encryption. In S. Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 343–356. Springer Berlin / Heidelberg, 2008.

[2] P. S. Barreto and M. Naehrig. Pairing friendly elliptic curves of prime order. In *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, 2006.

[3] I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography.* Number 317 in London Mathematical Society Lecture Note Series. Cambridge University Press, 2005.

[4] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer Berlin / Heidelberg, 2005.

[5] J. H. Cheon. Security analysis of the strong diffie-hellman problem. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer Berlin / Heidelberg, 2006.

[6] J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *J. Cryptology*, 23:457–476, 2010.

[7] C. Delerablée, P. Paillier, and D. Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007*, volume 4575 of *LNCS*, pages 39–59. Springer Berlin / Heidelberg, 2007.

[8] European Network of Excellence in Cryptology II. ECRYPT II Yearly Report on Algorithms and Keysizes, 2011. `http://www.ecrypt.eu.org/documents/D.SPA.17.pdf`.

[9] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.

[10] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[11] S. Kozaki, T. Kutsuma, and K. Matsuo. Remarks on Cheon's algorithms for pairing-related problems. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *LNCS*, pages 302–316. Springer Berlin / Heidelberg, 2007.

[12] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62. Springer Berlin / Heidelberg, 2001.

[13] D. Phan, D. Pointcheval, and M. Strefler. Security notions for broadcast encryption. In J. Lopez and G. Tsudik, editors, *Applied Cryptography and Network Security*, volume 6715 of *LNCS*, pages 377–394. Springer Berlin / Heidelberg, 2011.

[14] Y. Sakemi, G. Hanaoka, T. Izu, M. Takenaka, and M. Yasuda. Solving a discrete logarithm problem with auxiliary input on a 160-bit elliptic curve. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *Public Key Cryptography – PKC 2012*, volume 7293 of *LNCS*, pages 595–608. Springer Berlin / Heidelberg, 2012.

[15] Thales Communications & Security. LibCryptoLCH, librairie cryptographique du laboratoire chiffre, 2012.

[16] Voltage security. Voltage identity-based encryption. `http://www.voltage.com/technology/ibe.htm`.