Deterministic Identity-Based Signatures for Partial Aggregation

Javier Herranz

INRIA Futurs, Laboratoire d'Informatique,
École Polytechnique, 91128 Palaiseau cedex, France
e-mail: herranz@lix.polytechnique.fr

Abstract

Aggregate signatures are a useful primitive which allows to aggregate into a single and constant-length signature many signatures on different messages computed by different users. Specific proposals of aggregate signature schemes exist only for PKI-based scenarios. For identity-based scenarios, where public keys of the users are directly derived from their identities, the signature schemes proposed up to now do not seem to allow constant-length aggregation.

We provide an intermediate solution to this problem, by designing a new identity-based signature scheme which allows aggregation when the signatures to be aggregated come all from the same signer. The new scheme is deterministic and enjoys some better properties than the previous proposals. We formally prove that the scheme is unforgeable, in the random oracle model, assuming that the Computational co-Diffie-Hellman problem is hard to solve.

1 Introduction

Identity-based (from now on, ID-based) cryptography was introduced by Shamir in [15] as an alternative to traditional public key cryptography, based on infrastructures (PKI). In PKI-based cryptography, each user generates on his own his secret and public keys. A certification authority must sign a digital certificate which links the identity of the user and his public key. The validity of this certificate must be checked before using the public key of the user, when encrypting a message to him or when verifying a signature from him. Obviously, the management of digital certificates decreases the efficiency of practical implementations of public key cryptosystems.

The idea of ID-based cryptography is that the public key of any user directly infers from his identity (e-mail address, telephone number, etc.). Later, the user contacts with a master entity who uses some secret information to compute the secret key related to the identity of the user. This secret key is sent to the user throughout a secure channel. ID-based cryptography has been the object of a lot of research during the last years, specially since the discovery that efficient ID-based cryptosystems can be designed by using bilinear pairings, which can be implemented on some elliptic curves over a finite field (see [16] for a complete bibliography of cryptographic works based on pairings).

On the other hand, the concept of aggregate signature schemes was introduced by Boneh et al. in [4]. The idea is that many signatures on different messages computed by different users can be aggregated into a single signature. Later, the correctness of all the signatures can be verified from the aggregate signature. Ideally, the length of the aggregate signature (excluding the messages and the public keys of the signers) should be constant, independent of the number of signed messages. This concept is very useful in situations where a device must store many signatures, for example routing protocols in wireless networks requiring authentication.

The only known proposals of aggregate signature schemes work in PKI-based scenarios. Our initial goal was to design ID-based aggregate signature schemes. However, by using the ID-based signature schemes existing in the literature (see [15, 13, 8, 6]) this does not seem to be possible at all, because the length of the resulting aggregate signatures would be linear on the number of aggregated signatures. In order to (partially) solve this problem, we design in this work a new ID-based signature scheme, which allows a more compact aggregation, that we denote as *partial*: the length of the resulting aggregate signatures will not depend on the number of signed messages, but on the number of signers. This improvement, which can be considered in principle as a minor one, becomes very important in situations where a device must store many signatures coming from a small set of users.

The idea of the new ID-based signature scheme is quite simple. The phase where the master entity generates secret keys for users is probabilistic, contrary to what happens in previous schemes; in fact, it consists in computing a Schnorr [14] signature (R, σ) on the given identity. Later, the signature phase itself will be deterministic: a user employs part of his secret key, the value σ to compute a signature ω on the message, using the signature scheme of Boneh, Lynn and Shacham [5] (from now on, denoted as BLS signature scheme), and then he appends the other part of his secret key, R. Therefore, the final signature is the pair (R, ω). The scheme can be implemented in groups which admit bilinear pairings. We prove with detail that the resulting ID-based signature scheme is existentially unforgeable under adaptive chosen message attacks in an ID-based scenario (following the model introduced in [6]), in the random oracle model for the two employed hash functions, and assuming the hardness of the Computational co-Diffie-Hellman problem.

The new scheme is as efficient as the previously proposed ID-based signature schemes. Furthermore, it enjoys some new desirable properties, apart from the possibility of partial aggregation; for example, if the master entity wants to compute a new secret key for some user, he must not update the secret keys of the rest of users in the system, as it happens in other approaches to ID-based signatures.

The rest of the paper is organized as follows. In Section 2 we explain the protocols of an ID-based signature scheme and the security requirements that such schemes must satisfy. In Section 3 we detail the design of the new ID-based signature scheme and we formally prove its unforgeability. In Section 4 we compare the new scheme with previous ID-based signature schemes, we explain some properties enjoyed by the new scheme, and we show how the new one can be used to achieve partial aggregation of signatures in ID-based scenarios. Finally, we conclude with a summary of the work and some open problems in Section 5.

2 Identity-Based Signature Schemes

An ID-based signature scheme consists of the following probabilistic algorithms:

Setup: it takes as input a security parameter k and returns, on the one hand, the system public parameters `params` and, on the other hand, the value `master-key`, which is known only to the master entity.

Extract: it takes as inputs `params, master-key` and a string $ID \in \{0,1\}^*$ specifying some identity; the algorithm returns a private key SK_{ID} to the user with identity ID. This step must be done over a secure channel.

Signature: the signature algorithm takes as inputs `params`, a message $M \in \mathcal{M}$ (where $\mathcal{M}$ is the message space specified in `params`), an identity ID and a secret key SK_{ID}, and returns a signature θ for the message M.

Verification: finally, the verification algorithm takes as inputs `params`, a message M, a signature θ and an identity ID; it returns 1 if the verification is correct, and 0 if not.

2.1 Correctness

We say that an ID-based signature scheme satisfies the *correctness* property when, for every message $M \in \mathcal{M}$ and any signature $\theta =\texttt{Signature}(\texttt{params}, M, ID, SK_{ID})$, it holds $\texttt{Verification}(\texttt{params}, M, \theta, ID) = 1$, provided that $\texttt{Setup}(k) = (\texttt{params},\texttt{master-key})$ and $\texttt{Extract}(\texttt{params},\texttt{master-key}, ID) = SK_{ID}$.

2.2 Unforgeability

With respect to security, we follow the model introduced in [6], which extends to the ID-based scenario the standard security model for signature schemes introduced in [7]: an ID-based signature scheme is existentially unforgeable under chosen message attacks if any probabilistic polynomial time adversary $\mathcal{A}$ has a negligible advantage in the following game, that it plays against a challenger:

Setup: the challenger takes a security parameter k and runs the `Setup` algorithm of the ID-based signature scheme. It gives to the adversary the resulting `params`. The challenger keeps secret the `master-key`.

Queries: the adversary makes different queries to the challenger.

- Extraction queries $< ID_i >$. The challenger responds by running algorithm `Extract` of the scheme, to obtain the private key SK_i which corresponds to the identity ID_i. The value SK_i is sent to the adversary.
- Signature queries $< ID_i, M_i >$. The challenger first obtains the corresponding private key SK_i by executing the algorithm `Extract`, and then it executes $\texttt{Signature}(\texttt{params}, M_i, ID_i, SK_i) = \theta_i$. The resulting signature θ_i is given to the adversary.
- Hash queries. If the scheme involves some hash function H_i which is assumed to behave as a random oracle [2] in the security proof, then the challenger must answer queries of the adversary to this oracle, providing it with consistent and totally random values.

All these queries can be made in an adaptive way; that is, each query may depend on the answers obtained to the previous queries.

Forgery: the adversary $\mathcal{A}$ outputs a tuple (ID, M, θ). We say that $\mathcal{A}$ *succeeds* if:

- $\texttt{Verify}(\texttt{params}, M, \theta, ID) = 1$; and
- the adversary has not requested an extraction query for ID; and
- θ has not been obtained as an answer of the challenger to a signature query $< ID, M >$.

The *advantage* of such an adversary is defined as

$$\texttt{Adv}_{\mathcal{A}}^{CMA}(k) \;=\; \Pr[\mathcal{A} \text{ succeeds}].$$

Definition 1. *An adversary $\mathcal{A}$ is a $(T, \varepsilon, Q_i, Q_e, Q_s)$-forger against an ID-based signature scheme if it runs in time at most T, its advantage is at least ε, and it is allowed to make Q_i queries to the random oracle which models H_i, to make Q_e extraction queries, and to make Q_s signature queries.*

3 A New Deterministic ID-Based Signature Scheme

Our goal was to design aggregate signature schemes in ID-based scenarios. Since this does not seem to be possible starting from the existing ID-based signature schemes, we propose a new one which is suitable for aggregation of signatures coming from the same signer.

The idea of the new scheme is quite simple: in the extract phase, the master entity uses his secret key to compute a Schnorr [14] signature on the given identity. This signature is the secret key of the user, who employs a part of it to compute a BLS [5] signature on the desired message. In this way, the signature phase of the scheme is itself deterministic, as it is the BLS signature scheme. The protocols that take part in the new scheme are detailed below.

Setup: on input a security parameter k, the master entity generates two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $q > 2^k$, along with a generator g of $\mathbb{G}$, such that these groups admit a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, which must be efficiently computable, non-degenerate (that is, $e(g, g) \neq 1$) and bilinear (that is, $e(g^a, g^b) = e(g, g)^{ab}$, for any $a, b \in \mathbb{Z}_q$).

The master entity sets the message space $\mathcal{M} = \{0,1\}^*$ and chooses two hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \{0,1\}^* \to \mathbb{G}$.

Finally, the master entity chooses an element $x \in \mathbb{Z}_q^*$ at random and computes $Y = g^x$.

The public outputs of the protocol are $\texttt{params}= (k, q, \mathbb{G}, g, \mathbb{G}_T, e, H_1, H_2, Y)$. The secret information stored by the master entity is $\texttt{master-key}= x$.

Extract: when a user with identity $ID \in \{0,1\}^*$ requests for his secret key, the master entity computes a Schnorr signature on the message ID. That is:

1. he chooses uniformly at random an element $r \in \mathbb{Z}_q^*$;
2. he computes the value $R = g^r$;
3. finally, he computes the value $\sigma = r + xH_1(ID, R) \bmod q$.

The master entity privately sends the secret key $SK = (R, \sigma)$ to the user, who can verify the correctness of the received secret key by checking if $g^\sigma = R \cdot Y^{H_1(ID,R)}$.

In fact, the value R can be sent to the user throughout public channels, because it will be later part of the (public) signatures computed by the user. Therefore, the secret key of the user is actually limited to the value σ.

Signature: to sign a message $M \in \{0,1\}^*$, a user with identity ID and secret key $SK = (R, \sigma)$ computes the value $\omega = H_2(M, ID)^\sigma$. The signature is the pair $\theta = (R, \omega)$.

Verification: given a signature $\theta = (R, \omega)$ computed by a user with identity ID on a message M, the recipient verifies its correctness by checking if

$$e(\omega, g) \;=\; e\left(H_2(M, ID), R \cdot Y^{H_1(ID,R)}\right).$$

If the equality holds, then the output of the verification algorithm is 1 (valid signature). Otherwise, the output is 0 (invalid signature).

3.1 Correctness of the Scheme

It is quite easy to see that this property is achieved. In effect, if a user ID receives a correct secret key $SK = (R, \sigma)$ satisfying $g^\sigma = R \cdot Y^{H_1(ID,R)}$, and later this user proceeds, as specified above, to compute a signature $\theta = (R, \omega)$ on a message M as $\omega = H_2(M, ID)^\sigma$, then the verification equation is satisfied:

$$e(\omega, g) = e(H_2(M, ID)^\sigma, g) = e(H_2(M, ID), g^\sigma) = e\left(H_2(M, ID), R \cdot Y^{H_1(ID,R)}\right).$$

We have used the bilinearity property of the pairing e, which ensures in particular that $e(g^{ab}, g) = e(g^a, g^b)$ for all values $a, b \in \mathbb{Z}_q$.

3.2 Unforgeability of the Scheme

We will prove that the proposed scheme is unforgeable under adaptive chosen message attacks, in the random oracle model for the hash functions H_1 and H_2, assuming that the Computational co-Diffie-Hellman problem is hard to solve in the group $\mathbb{G}$.

Definition 2. *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups with prime order q, admitting a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$ chosen at random, and let a, b be two elements chosen independently and at random from $\mathbb{Z}_q^*$.*

We say that an algorithm $\mathcal{F}$ solves the Computational co-Diffie-Hellman (co-CDH) problem in $\mathbb{G}$ if it receives as input the tuple $(q, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^b)$ and outputs the element g^{ab}.

In the proof of unforgeability, we will need a well-known result of elementary probability (the proof can be found in [12]).

Lemma 1. *(The Splitting Lemma). Let X and Y be two finite sets where two probability distributions are considered. Let $A \subset X \times Y$ be a set such that $\Pr[A] \geq \gamma$, where the probability distribution in $X \times Y$ is the joint probability distribution induced by the distributions in X and Y. For any $\alpha < \gamma$, let us define*

$$B = \{(x, y) \in X \times Y | \Pr_{y' \in Y}\left[(x, y') \in A\right] \geq \gamma - \alpha\} \text{ and } \bar{B} = X \times Y \ - B,$$

then the following statements hold:

1. $\Pr[B] \geq \alpha$.
2. *for any $(x, y) \in B$, $\Pr_{y' \in Y}[(x, y') \in A] \geq \gamma - \alpha$.*
3. $\Pr[B|A] \geq \alpha/\gamma$.

We will prove that a hypothetic successful attack against our ID-based signature scheme could be used to construct an algorithm which solves the co-CDH problem with non-negligible probability and in polynomial time. Since this is assumed to be unfeasible, we conclude that there cannot exist successful attacks against our scheme, and so it is secure.

Theorem 1. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s)$-forger against our ID-based signature scheme, where the groups $\mathbb{G}$ and $\mathbb{G}_T$ have prime order q. Then the co-CDH problem can be solved in $\mathbb{G}$ with probability ε' and within time T' satisfying*

$$\varepsilon' \geq \left(\frac{q-1}{q}\right)^3 \frac{\varepsilon^2}{16\hat{e}^4 Q_1 (Q_e+1)^2 (Q_s+1)^2} \quad \text{and} \quad T' \leq 2T + T_{exp}\left(2Q_e + 4Q_s + 2Q_1 + 3Q_2\right) ,$$

where T_{exp} denotes the time needed to perform a modular exponentiation in $\mathbb{G}$ and $\hat{e}$ is the base of natural logarithms.

Proof. We are going to construct a probabilistic polynomial time Turing machine $\mathcal{F}$ which will use the attacker $\mathcal{A}$ as a sub-routine in order to solve a given instance of the Computational co-Diffie-Hellman problem. Therefore, $\mathcal{F}$ will try to perfectly simulate the environment of $\mathcal{A}$.

The machine $\mathcal{F}$ receives the public data $(q, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^b)$, and its goal is to compute the value g^{ab}. The public key of the master entity is defined to be $Y = g^a$ and is sent to the attacker $\mathcal{A}$. After that, $\mathcal{F}$ runs the attacker $\mathcal{A}$ against the ID-based signature scheme, answering to all the queries that $\mathcal{A}$ makes. To do this, $\mathcal{F}$ maintains three tables $\mathtt{TAB}_1$, $\mathtt{TAB}_2$ and $\mathtt{TAB}_{\mathtt{SK}}$, which are updated as explained below. Let μ and δ be two real parameters in the interval $(0, 1)$ that will be specified later.

First query involving ID_i. The first time $\mathcal{F}$ receives a query from $\mathcal{A}$ involving an identity ID_i, it chooses a bit $c_i \in \{0, 1\}$ such that $\Pr[c_i = 0] = \mu$. According to the result of this choice, $\mathcal{F}$ proceeds as follows:

- If $c_i = 0$, then $\mathcal{F}$ chooses $\sigma_i, h_{1i} \in \mathbb{Z}_q$ independently and at random; later, it computes $R_i = g^{\sigma_i} \cdot Y^{-h_{1i}}$. Then $\mathcal{F}$ stores the tuple (ID_i, R_i, h_{1i}) in the table $\mathtt{TAB}_1$, and stores the tuple $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table $\mathtt{TAB}_{\mathtt{SK}}$.

- If $c_i = 1$, then $\mathcal{F}$ chooses $R_i \in \mathbb{G}$ and $h_{1i} \in \mathbb{Z}_q$ independently and at random; then it stores the tuple (ID_i, R_i, h_{1i}) in the table $\mathtt{TAB}_1$, and stores the tuple $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table $\mathtt{TAB}_{\mathtt{SK}}$, where $\sigma_i = \perp$ meaning that this value is unknown by $\mathcal{F}$.

Hash queries to H_1. These queries are answered in the standard way when considering the random oracle model: when $\mathcal{A}$ makes a query (ID_j, R_j) to H_1, the machine $\mathcal{F}$ looks for (ID_j, R_j) in the table $\mathtt{TAB}_1$. If it is already there, then $\mathcal{F}$ answers the stored value h_{1j}. Otherwise, $\mathcal{F}$ chooses at random $h_{1j} \in \mathbb{Z}_q$, sends it to $\mathcal{A}$ and stores the new tuple (ID_j, R_j, h_{1j}) in $\mathtt{TAB}_1$.

Hash queries to H_2. When $\mathcal{A}$ makes a query (M_i, ID_i) to the oracle which models the behavior of the hash function H_2, the machine $\mathcal{F}$ first looks for (M_i, ID_i) in the table $\mathtt{TAB}_2$. If it is already there, then the stored value h_{2i} is answered to $\mathcal{A}$. Otherwise, $\mathcal{F}$ chooses $t_i \in \mathbb{Z}_q^*$ at random, and chooses a bit $d_i \in \{0, 1\}$ according to the distribution $\Pr[d_i = 0] = \delta$. If $d_i = 0$, then $\mathcal{F}$ defines $h_{2i} = g^{t_i}$. If $d_i = 1$, then $\mathcal{F}$ defines $h_{2i} = (g^b)^{t_i}$. The machine $\mathcal{F}$ stores the tuple $(d_i, M_i, ID_i, h_{2i}, t_i)$ in the table $\mathtt{TAB}_2$ and returns the value h_{2i} to $\mathcal{A}$.

Extraction queries. When $\mathcal{A}$ asks for the secret key corresponding to an identity ID_i, the machine $\mathcal{F}$ looks for ID_i in the table $\mathtt{TAB}_{\mathtt{SK}}$. If it is not already there, then this is the first query involving identity ID_i; in this case, $\mathcal{F}$ can impose $c_i = 0$, proceed as explained above and store the resulting tuple $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table.

Anyway, if $c_i = 0$, the pair $SK_i = (R_i, \sigma_i)$ is returned to $\mathcal{A}$. Otherwise, if $c_i = 1$, the machine $\mathcal{F}$ halts.

Signature queries. $\mathcal{A}$ can ask for a valid signature for pairs (M_i, ID_i) of its choice. Without loss of generality, we can assume that $\mathcal{A}$ has queried before this pair (M_i, ID_i) to the oracle for H_2. To answer the signature query, the machine $\mathcal{F}$ looks for the entry $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table $\mathtt{TAB_{SK}}$ and for the entry $(d_i, M_i, ID_i, h_{2i}, t_i)$ in the table $\mathtt{TAB}_2$. We consider three cases.

- If $c_i = 0$, then $\mathcal{F}$ can use the secret key pair $SK_i = (R_i, \sigma_i)$ to compute a valid signature for (M_i, ID_i) and return it to $\mathcal{A}$.
- If $c_i = 1$ and $d_i = 0$, then we have that $h_{2i} = g^{t_i}$. In this case, $\mathcal{F}$ computes the value
$$\omega_i = \left(R_i \cdot Y^{h_{1i}}\right)^{t_i}$$
and returns the valid signature $\theta_i = (R_i, \omega_i)$ to $\mathcal{A}$.
- Finally, if $c_i = 1$ and $d_i = 1$, which happens with probability $(1-\mu)(1-\delta)$, then the machine $\mathcal{F}$ halts.

A first forgery. Provided the machine $\mathcal{F}$ does not halt, the environment of $\mathcal{A}$ is perfectly simulated; in this case, the machine $\mathcal{A}$ will produce with probability at least ε a valid forged signature (M, ID, R, ω) satisfying

$$e(\omega, g) \;=\; e\left(H_2(M, ID), R \cdot Y^{H_1(ID,R)}\right).$$

Note that the probability that $\mathcal{F}$ does not halt when asking extraction and signature queries is $\mu^{Q_e} \cdot (1-(1-\mu)(1-\delta))^{Q_s}$. For simplicity, we consider a more simple lower bound: this probability is greater than $\mu^{Q_e}\delta^{Q_s}$.

We need that the forged signature (M, ID, R, ω) satisfies that, in the corresponding entry (c, ID, R, h_1, σ) in the table $\mathtt{TAB_{SK}}$, the bit c is equal to 1. This happens with probability $1-\mu$, and in this case we can be sure that the value $h_1 = H_1(ID, R)$ has been chosen after the value R (note that this is not true when the bit c is 0).

We denote by χ the whole set of random tapes that take part in an attack by $\mathcal{A}$, with the environment simulated by $\mathcal{F}$, but excluding the randomness related to the oracle H_1. The success probability of $\mathcal{A}$ in forging a valid signature scheme is then taken over the randomness (χ, H_1).

In an execution of the attacker $\mathcal{A}$, we use the notation $\mathcal{Q}_{1,1}, \mathcal{Q}_{1,2}, \ldots, \mathcal{Q}_{1,Q_1}$ for the different queries that $\mathcal{A}$ makes to the random oracle H_1, and we denote by $\rho = (\rho_1, \ldots, \rho_{Q_1})$ the list of the Q_1 answers of the random oracle H_1. So we can see an instantiation of the random oracle H_1 as a random choice of such a vector ρ.

If $\mathcal{A}$ produces a valid forged signature (M, ID, R, ω), by the ideal randomness of the oracle H_1, the probability that $\mathcal{A}$ has not asked to this oracle for the corresponding tuple (ID, R), and so $\mathcal{A}$ must have guessed the corresponding output, is less than $\frac{1}{q}$. We define $\beta = \infty$ in this case; otherwise, β denotes the index of the query where (ID, R) was asked. That is, $\mathcal{Q}_{1,\beta} = (ID, R)$.

We denote by $\mathcal{S}$ the set of successful executions of $\mathcal{A}$, with $\mathcal{F}$ simulating its environment, and such that $c = 1$ and $\beta \neq \infty$. We also define the following subsets of $\mathcal{S}$: for every $i = 1, 2, \ldots, Q_1$, the set $\mathcal{S}_i$ contains the successful executions such that $c = 1$ and $\beta = i$. This gives us a partition $\{\mathcal{S}_i\}_{i=1,\ldots,Q_1}$ of $\mathcal{S}$ in exactly Q_1 classes.

Summing up, the probability $\tilde{\varepsilon}$ that an execution (χ, H_1) of $\mathcal{A}$ with the environment simulated by $\mathcal{F}$ results in a valid forgery with $\beta \neq \infty$ and where the bit c is equal to 1, is

$$\tilde{\varepsilon} = \Pr[(\chi, H_2) \in \mathcal{S}] \geq \mu^{Q_e} \delta^{Q_s} \varepsilon (1 - \mu) \left(1 - \frac{1}{q}\right).$$

The oracle-replay technique. At this moment, we use a well-known technique (see [12], for example) which consists in repeating the attack $\mathcal{A}$, simulated by $\mathcal{F}$, with the same random tapes but with a different instantiation of the random oracle for H_1, from the query $\mathcal{Q}_{1,\beta} = (ID, R)$ on.

We define the set of indexes which are more likely to appear as

$$I = \{i \text{ such that } \Pr[(\chi, H_1) \in \mathcal{S}_i \mid (\chi, H_1) \in \mathcal{S}] \geq \frac{1}{2Q_1}\}.$$

And the corresponding subset of successful executions as $\mathcal{S}_I = \{(\chi, H_1) \in \mathcal{S}_i$ such that $i \in I\}$. For a specific index $i \in I$, the following inequality holds:

$$\Pr[(\chi, H_1) \in \mathcal{S}_i] = \Pr[(\chi, H_1) \in \mathcal{S}] \cdot \Pr[(\chi, H_1) \in \mathcal{S}_i \mid (\chi, H_1) \in \mathcal{S}] \geq \tilde{\varepsilon} \cdot \frac{1}{2Q_1}.$$

Lemma 2. *It holds that* $\Pr[(\chi, H_1) \in \mathcal{S}_I \mid (\chi, H_2) \in \mathcal{S}] \geq 1/2$.

Proof. Since the sets $\mathcal{S}_i$ are disjoint, we can write

$$\Pr[(\chi, H_1) \in \mathcal{S}_I \mid (\chi, H_1) \in \mathcal{S}] = \sum_{i \in I} \Pr[(\chi, H_1) \in \mathcal{S}_i \mid (\chi, H_1) \in \mathcal{S}] =$$
$$1 - \sum_{i \notin I} \Pr[(\chi, H_1) \in \mathcal{S}_i \mid (\chi, H_1) \in \mathcal{S}].$$

Since the complement of I contains at most Q_1 indexes, we have that this probability is greater than $1 - Q_1 \cdot \frac{1}{2Q_1} = 1/2$. $\square$

We come back to the first execution of $\mathcal{A}$ with the environment simulated by $\mathcal{F}$. With probability at least $\tilde{\varepsilon}$, such an execution (χ, H_1) results in a valid forgery with $\beta \neq \infty$ and $c = 1$. In this case, applying Lemma 2, we know that this successful execution belongs to $\mathcal{S}_I$ with probability at least 1/2. If this happens, then $\beta \in I$ and so $\Pr[\mathcal{S}_\beta] \geq \tilde{\varepsilon}/2Q_1$.

Now we split H_1 as $(H_{1\beta^-}, H_{1\beta^+})$, where $H_{1\beta^-} = (\rho_1, \ldots, \rho_{\beta-1})$ corresponds to the answers of all the queries to H_1 that happen before the query $\mathcal{Q}_{1,\beta}$, and $H_{1\beta^+} = (\rho_\beta, \ldots, \rho_{Q_1})$ corresponds to the rest of answers.

We apply the Splitting Lemma (Lemma 1), taking $X = (\chi, H_{1\beta^-})$, $Y = H_{1\beta^+}$, $A = \mathcal{S}_\beta$, $\gamma = \frac{\tilde{\varepsilon}}{2Q_1}$ and $\alpha = \frac{\tilde{\varepsilon}}{4Q_1}$. The lemma says that there exists a subset of executions Ω_β such that

$$\Pr[(\chi, H_1) \in \Omega_\beta \mid (\chi, H_1) \in \mathcal{S}_\beta] \geq \frac{\alpha}{\gamma} = \frac{1}{2}$$

and such that, for any $(\chi, H_1) \in \Omega_\beta$:

$$\Pr_{\tilde{H}_{1\beta^+}} [(\chi, H_{1\beta^-}, \tilde{H}_{1\beta^+}) \in \mathcal{S}_\beta] \geq \gamma - \alpha = \frac{\tilde{\varepsilon}}{4Q_1}.$$

Running $\mathcal{A}$ again. With probability at least $\frac{\tilde{\varepsilon}}{2}$, the first execution $(\chi, H_{1\beta^-}, H_{1\beta^+})$ of $\mathcal{A}$ simulated by $\mathcal{F}$ is successful (with bit $c = 1$) and the index β belongs to the set I. Furthermore, in this case we have that $(\chi, H_{1\beta^-}, H_{1\beta^+}) \in \Omega_\beta$ with probability at least $1/2$.

If $\mathcal{F}$ repeats this simulated execution of $\mathcal{A}$ with fixed $(\chi, H_{1\beta^-})$ and randomly chosen $\tilde{H}_{1\beta^+} = (\tilde{\rho}_\beta, \ldots, \tilde{\rho}_{Q_1}) \in (\mathbb{Z}_q)^{Q_1-\beta+1}$, then we know that $(\chi, H_{1\beta^-}, \tilde{H}_{1\beta^+}) \in \mathcal{S}_\beta$ and furthermore $\tilde{\rho}_\beta \neq \rho_\beta$ with probability at least $\frac{\tilde{\varepsilon}}{4Q_1}\left(1 - \frac{1}{q}\right)$. Here ρ_β denotes the answer to the query $\mathcal{Q}_{1,\beta}$ in the first execution of the attack.

This means that the second execution of $\mathcal{A}$, with a different instantiation $\tilde{H}_1$ of the hash function H_1, provides a new valid forged signature $(\tilde{M}, \tilde{ID}, \tilde{R}, \tilde{\omega})$ such that $\mathcal{Q}_{1,\beta} = (\tilde{ID}, \tilde{R})$. Recall that the forged signature obtained in the first execution is denoted as (M, ID, R, ω). Since the attacks are exactly equal until the query $\mathcal{Q}_{1,\beta} = (ID, R)$, we have that $\tilde{ID} = ID$ and $\tilde{R} = R$. Furthermore, the answers of the oracle to this query are different, so $h_1 = H_1(ID, R) \neq \tilde{H}_1(ID, R) = \tilde{h}_1$.

Now let us consider the corresponding entries (d, M, ID, h_2, t) and $(\tilde{d}, \tilde{M}, ID, \tilde{h}_2, \tilde{t})$ in the table $\mathtt{TAB}_2$ corresponding to the two forged signatures in the two executions of $\mathcal{A}$ simulated by $\mathcal{F}$. With probability $(1-\delta)^2$ we have that $d = \tilde{d} = 1$. This means that $h_2 = H_2(M, ID) = (g^b)^t$ and $\tilde{h}_2 = H_2(\tilde{M}, ID) = (g^b)^{\tilde{t}}$. In this case, considering the two verification equations satisfied by the two forged signatures (M, ID, R, ω) and $(\tilde{M}, ID, R, \tilde{\omega})$, and taking into account that $Y = g^a$, we have

$$e(\omega, g) \;=\; e\left((g^b)^t \;,\; R \cdot (g^a)^{h_1}\right),$$

$$e(\tilde{\omega}, g) \;=\; e\left((g^b)^{\tilde{t}} \;,\; R \cdot (g^a)^{\tilde{h}_1}\right).$$

Raising the second equation to $t/\tilde{t}$ and dividing then the two equations, we obtain

$$e(\omega/\tilde{\omega}^{t/\tilde{t}}, g) \;=\; e\left((g^b)^t \;, (g^a)^{h_1 - \tilde{h}_1}\right).$$

By the non-degeneration property of the bilinear pairing e, this equality implies that $\omega/\tilde{\omega}^{t/\tilde{t}} = (g^{ab})^{t(h_1 - \tilde{h}_1)}$, so the solution of the given instance of the Computational co-Diffie-Hellman problem is finally

$$g^{ab} \;=\; \left(\frac{\omega}{\tilde{\omega}^{t/\tilde{t}}}\right)^{\frac{1}{t(h_1-\tilde{h}_1)}}.$$

Revisiting all the intermediate probabilities, we have that the total probability ε' of solving the co-CDH problem has been

$$\varepsilon' \geq \frac{\tilde{\varepsilon}}{2} \cdot \frac{1}{2} \cdot \frac{\tilde{\varepsilon}}{4Q_1} \cdot \frac{q-1}{q} \cdot (1-\delta)^2 \geq \frac{\tilde{\varepsilon}^2}{16Q_1} \cdot (1-\delta)^2 \cdot \frac{q-1}{q} \geq$$

$$\geq \mu^{2Q_e}\delta^{2Q_s}(1-\mu)^2(1-\delta)^2\left(\frac{q-1}{q}\right)^3\frac{\varepsilon^2}{16Q_1}.$$

The values of the parameters μ and δ which maximize this expression are $\mu = \frac{Q_e}{Q_e+1}$ and $\delta = \frac{Q_s}{Q_s+1}$. With this choice, the final expression is

$$\varepsilon' \geq \frac{1}{(Q_e+1)^2}\frac{1}{(Q_s+1)^2}\left(\frac{q-1}{q}\right)^3\frac{\varepsilon^2}{16\hat{e}^4 Q_1},$$

where $\hat{e}$ is the base of natural logarithms.

With respect to the execution time T' of the solver $\mathcal{F}$ of the co-CDH problem, it is easy to see that the bound $T' \leq 2T + T_{exp}\,(2Q_e + 4Q_s + 2Q_1 + 3Q_2)$ is satisfied, where T_{exp} is the time needed to compute a modular exponentiation in $\mathbb{G}$.

□

4 Comparisons and Applications

We can compare our scheme with previously proposed ID-based signature schemes, such as the original RSA-based one in [15] or the ones in [13, 8, 6], with respect to efficiency, security and applications.

4.1 Efficiency

The reductions in the security proof of our scheme is far from being tight, as it happens in the case of the schemes in [15, 8, 6]. This is due to the fact that the security proofs of all these schemes employ the oracle-replay techniques (following the ideas of [12]). This leads to results of the following type, for the schemes in [15, 8, 6]: if there exists a $(T, \varepsilon, Q_i, Q_e, Q_s)$-forger against the corresponding ID-based signature scheme, then some hard computational problem (in this case, either the RSA or the CDH problem) can be solved in time $T' = \mathcal{O}(T + Q_i + Q_e + Q_s)$ and with probability $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon^2}{Q_e Q_i}\right)$. In the case of our scheme, the reduction that we have have proved is even less tight, since the relation is $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon^2}{Q_e^2 Q_s^2 Q_i}\right)$. Note, however, that the factor Q_s^2 can be removed from this relation if we apply to our scheme the techniques introduced by Katz and Wang in [9], at the cost of increasing by one bit the length of the resulting signatures.

For the scheme designed in [13], tight security reductions are possible as shown in [10]. See also [1] for a complete work about the security of identity-based signature schemes.

In all the previously cited schemes [15, 13, 8, 6], the basic idea is the opposed as in our scheme: the deterministic process to generate secret keys from identities consists in computing a FDH-RSA [3] or BLS [5] signature on the identity, whereas the signature phase is probabilistic, consisting in the application of some *generic* (as defined in [12]) signature scheme. As a result, the final signature on a message M has the form (R, ω), where R is a new value chosen at random for each new signature.

In the case of our scheme, the probabilistic extraction phase is a bit more costly, but the deterministic signature phase can be run more efficiently, since it only consists in computing a hash value and a modular exponentiation. This makes sense in many real situations, where the signature phase is run more often than the extraction phase. Furthermore, recall that in our scheme the final signature on a message M by some user with identity ID has the form (R, ω), where the value R is fixed for this user, independently of the signed message. This means that in situations where ID has to sign a lot of messages for the same(s) receiver(s), he can send (R, ω_1) for the first message M_1, but later he can send only the value ω_i for the following messages M_i, if the receivers are supposed to store the value R.

With respect to the verification of signatures, the efficiency of the new scheme is the same as in the most efficient ID-based schemes previously proposed: the most costly operation is the evaluation of two bilinear pairings.

4.2 Updating Secret Keys

The new approach to ID-based signatures that we propose in this work has some advantages. The main one, which was the initial motivation for this work and which is explained in the next section, is the computation of partially aggregated signatures.

A second advantage is the flexibility that has the master entity to update secret keys of the users in the system. Let us first consider the typical approach to ID-based signatures: the secret key $SK_{ID} = xH_1(ID)$ for a user with identity ID is a BLS signature on the message ID, computed by the master entity by using his secret key $x \in \mathbb{Z}_q$ and the public hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$. What happens if the secret key SK_{ID} is compromised before its expiry date, for example because of an attack or an accidental exposure? If the master entity wants this user to stay active in the system, then he must provide a new secret key to him. But to do this, the only solution consists in changing either his secret key x or the public hash function H_1. Both solutions imply that the secret keys of all the users in the system must be computed and distributed again.

This problem disappears with our approach, because the `Extract` phase of our ID-based signature scheme is probabilistic: the secret key for an identity ID is a Schnorr (probabilistic) signature $SK_{ID} = (R, \sigma)$ obtained by the master entity by choosing at random $a \in \mathbb{Z}_q$ and then by computing $R = g^a$ and $\sigma = a + xH_1(ID, R)$. If this secret key is compromised, the master entity can compute and distribute a new secret pair (R', σ') for ID just by choosing a different random value a'. In this way, the rest of users can keep their secret keys, because the parameters of the master entity remain unchanged.

For the same reason, the global security of our approach is higher than in the typical one, provided the master entity deletes from his memory (just after sending $SK_{ID} = (R, \sigma)$ to the user ID) the values a and σ that he obtains during the generation of SK_{ID}. In this case, even if the secret key x of the master entity is compromised, the secret key value σ of ID remains secure, because an attacker cannot obtain it from the knowledge of the values x, R, ID. Therefore, the user

ID could still sign messages in a secure way, despite the master entity being out of service for a moment.

4.3 Signatures with Partial Aggregation

The concept of aggregate signature schemes, introduced in [4], is very useful in network applications requiring authentication, for example e-commerce or routing protocols in wireless networks. The goal is to store many signatures without employing too much memory space. To do this, aggregate signatures allow to combine n different signatures θ_i on n different messages M_i by n (possibly different) users, and produce a single aggregate signature θ. The idea is that the length of θ should be constant, independent of the number n of aggregated signatures. Note, however, that the stored information will be always linear on the number of messages, since the identities (or public keys) of the signers and the messages themselves must be stored.

In [4] the authors propose a specific scheme based on the BLS signature scheme, for PKI-based scenarios. The common parameters of the scheme are the groups $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ of order q, the bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $H : \{0,1\} \to \mathbb{G}$. Each user U_i chooses his secret key as a random value $x_i \in \mathbb{Z}_q^*$; the matching public key is $Y_i = g^{x_i}$. To sign a message M_i, the user U_i computes $\theta_i = H(M_i)^{x_i}$. To verify the correctness of such a signature, the recipient of the message checks if the equality $e(\theta_i, g) = e(H(M_i), Y_i)$ holds or not.

To aggregate a set $\{\theta_1, \theta_2, \dots, \theta_n\}$ of n valid signatures on messages $\{M_1, M_2, \dots, M_n\}$ computed respectively by users with public keys $\{Y_1, Y_2, \dots, Y_n\}$, one computes the value

$$\theta = \prod_{i=1}^{n} \theta_i.$$

In effect, it is possible to verify if θ contains the n valid signatures, by checking the equation

$$e(\theta, g) = \prod_{i=1}^{n} e(H(M_i), Y_i).$$

The same idea can be applied to the Full Domain Hash RSA signature scheme [3], but in this case only signatures coming from the same signer can be aggregated (what we will call partial aggregation). In [11], a general construction of sequential aggregate signature schemes is proposed, where the order in which signatures are aggregated is important.

If we try to extend the idea of aggregate signatures to identity-based scenarios, we find some problems. Considering previous ID-based signature schemes (like [15, 13, 8, 6]), we have that the signature on a message M_i has the form $\theta_i = (R_i, \omega_i)$. Let us assume that we have n different signatures $\{\theta_i\}_{1 \le i \le n}$ on n messages $\{M_i\}_{1 \le i \le n}$. It is not difficult to see that the values $\{\omega_i\}_{1 \le i \le n}$ can be aggregated into a single value $\omega = \prod_{1 \le i \le n} \omega_i$ as before. However, the values $\{R_i\}_{1 \le i \le n}$ can not be aggregated, because they all must be explicitly known to verify the correctness of the aggregate signature.

Therefore, the length of the resulting aggregate signature $(\omega, R_1, R_2, \ldots, R_n)$ would be linear on the number of aggregated signatures.

The design of our new ID-based signature scheme does not allow to totally solve this problem, but it allows to provide an intermediate solution, that we call *partial* aggregation: all the signatures coming from the same signer can be aggregated into a constant-length signature. In this way, the length of an aggregate signature will be linear on the number of signers, and not on the number of signed messages. This solution can represent an important improvement in situations where some device must store many signatures coming from a small set of signers, for example in small networks, restricted e-mail applications, transactions between big companies, etc.

Specifically, let us assume that someone wants to aggregate $n = \sum_{1\leq i\leq m} n_i$ signatures on n messages from m users with identities $\{ID_i\}_{1\leq i\leq m}$, where the n_i messages and signatures coming from ID_i are denoted as $\{M_{ij}\}_{1\leq j\leq n_i}$ and $\{\theta_{ij}\}_{1\leq j\leq n_i}$. Recall that the signatures have the form $\theta_{ij} = (R_i, \omega_{ij})$. Then one computes the value

$$\omega = \prod_{i=1}^{m} \prod_{j=1}^{n_1} \omega_{ij},$$

and defines the aggregate signature to be the tuple $\theta = (\omega, R_1, R_2, \ldots, R_m)$. To verify the correctness of such a signature, one must check if the following equality holds:

$$e(\theta, g) \;=\; \prod_{i=1}^{m} e\Big(\prod_{j=1}^{n_1} H_2(ID_i, M_{ij}) \;,\; R_i \cdot Y^{H_1(ID_i, R_i)}\Big).$$

Security. Extending the model in [4], we can define the security of an ID-based aggregate signature scheme by considering the following game played by an adversary against a challenger: the challenger executes `Setup` phase of the ID-based scheme, then he gives `params` to the adversary and keeps secret the `master-key`. After that, the adversary can make hash queries (if the security proof is done in the random oracle model), extract queries $< ID >$, standard signature queries $< ID, M >$ and aggregate signature queries $< (ID_{i_1}, M_{i_1}), \ldots, (ID_{i_s}, M_{i_s}) >$, for inputs that it adaptively chooses. The challenger uses his knowledge of the `master-key` to properly answer all these queries.

Finally, the adversary outputs an aggregate signature θ on a set of n pairs of identities and messages $W = \{(ID_1, M_1), (ID_2, M_2), \ldots, (ID_n, M_n)\}$. The adversary succeeds if:

(i) the signature θ is valid; and

(ii) there is at least one pair $(ID_i, M_i) \in W$ such that: $< ID_i >$ has not been queried to the extract oracle and (ID_i, M_i) is not included in any query made to the standard or aggregate signature oracles.

An aggregate signature scheme is secure if any such adversary running against the scheme in polynomial time has a negligible probability of success (in the security parameter of the scheme).

In the case of our scheme, by using very similar techniques to those in [4] and those in the proof of Theorem 1 above, it is possible to reduce the security of the resulting aggregate signature scheme to the hardness of the Computational co-Diffie-Hellman problem.

5 Conclusion

We propose in this paper a new identity-based signature scheme which is deterministic. This fact makes it different from the previously proposed ID-based signature schemes [15, 13, 8, 6], and provides it with some good properties with respect to efficiency and length of the signatures. Furthermore, the new scheme allows partial aggregation: all the signatures coming from the same signer can be aggregated into a single one which has the same length as the original signatures. The security of the new scheme is formally proved by reduction to the difficulty of the Computational co-Diffie-Hellman problem.

Two problems related to this work remain open: on the one hand, to find a tighter security reduction for our deterministic ID-based signature scheme, if possible; on the other hand, to design aggregate signature schemes for identity-based scenarios where the length of the aggregate signature is totally constant, depending of neither the number of signed messages nor the number of signers.

References

[1] M. Bellare, C. Namprempre and G. Neven. Security proofs for identity-based identification and signature schemes. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 268–286 (2004).

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of 1st Conference on Computer and Communications Security*, ACM, pp. 62–73 (1993).

[3] M. Bellare and P. Rogaway. The exact security of digital signatures - How to sign with RSA and Rabin. *Proceedings of Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 399–416 (1996).

[4] D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Proceedings of Eurocrypt'03*, LNCS **2656**, Springer-Verlag, pp. 416–432 (2003).

[5] D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, Vol. **17** (4), Springer-Verlag, pp. 297–319 (2004). An extended abstract had appeared in the *Proceedings of Asiacrypt'01.*

[6] J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. *Proceedings of PKC'03*, LNCS **2567**, Springer-Verlag, pp. 18–30 (2002).

[7] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. *SIAM Journal of Computing*, **17** (2), pp. 281–308 (1988).

[8] F. Hess. Efficient identity based signature schemes based on pairings. *Proceedings of Selected Areas in Cryptography'02*, LNCS **2595**, Springer-Verlag, pp. 310-324 (2003).

[9] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. *Proceedings of the 10th Conference on Computer and Communication Security*, ACM, pp. 155–164 (2003).

[10] B. Libert and J.J. Quisquater. The exact security of an identity based signature and its applications. Preprint available at `http://  b  xug   b2emmv4. ollibeefood.rest/2004/10` (2004).

[11] A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham. Sequential aggregate signatures from trapdoor permutations. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 74–90 (2004).

[12] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, Vol. **13** (3), Springer-Verlag, pp. 361–396 (2000).

[13] R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. *Proceedings of the Symposium on Cryptography and Information Security, SCIS'00*, Okinawa, Japan, pp. 26–28 (2000).

[14] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, Vol. **4**, Springer-Verlag, pp. 161–174 (1991).

[15] A. Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of Crypto'84*, LNCS **196**, Springer-Verlag, pp. 47–53 (1984).

[16] The Pairing-Based Crypto Lounge, Web page maintained by Paulo Barreto:
`http://2zh 1vuga km0. ollibeefood.rest.br/informatica/paulobarreto/pblounge.html`