Efficient Certificateless Signatures Suitable for Aggregation

Rafael Castro* Ricardo Dahab

Abstract

This technical report describes a novel certificateless signature scheme suitable for aggregation that requires no pairing computations for signing and only 3 pairing computations for signature verification. We provide proofs for the security of single and aggregate signatures.

1 Introduction

Public-key cryptography represented a giant step forward in the security of computer-aided communications. However, the complexity of managing PKIs imposes a huge cost in the widespread adoption of cryptography, and good solutions to simplify this task have been for years a research goal for the cryptography community. One of the early proposals to solve this was Identity-Based Cryptography [Sha85]. In ID-Based cryptography, no public keys are needed: merely knowing some specific identity information about the person with whom you wish to communicate (say, her e-mail address) is enough to encrypt messages and verify signatures. Some central authority is responsible for managing the system, distributing the correct secret keys only upon verification of the user's identity. This way, no certification is required: if the authority is trusted, only the correct person will have access to each secret key.

The problem is that this trusted authority *(TA)* knows the secret keys of each and every user of the system. So, while a certification authority *(CA)* in a PKI needs only be trusted with verifying users' identities, here the TA must also be trusted not to misuse users' secret keys. This may pose no problem for certain environments (say, for use within a company or in the military), but is certainly not fit for widespread adoption. Certificateless cryptography [ARP03] is a novel approach that aims to find a good compromise between traditional certificate-based cryptography and identity-based cryptography. In the certificateless approach, a user's private key is comprised of two parts: one generated by a Key Generation Center *(KGC)* and associated with her identity; another generated by the user herself, and unknown to any other parties. This way key escrow is avoided (because the KGC does not know the whole private key) while no explicit certification of public keys is needed (because part of the private key is generated by the KGC and trusted to be known only by the legitimate user).

*Research supported in part by FAPESP grant #2006/06146-3.

Signature aggregation, as defined in [BGSL03], is a very useful generalization of multisignatures allowing multiple signatures on arbitrary messages to be combined in a single-length signature. In this paper we present a novel efficient certificateless signature scheme that is suitable for signature aggregation. This is, to our knowledge, the first treatment of signature aggregation in the certificateless scenario so, before the actual scheme, we spend some time discussing the security implications of signature aggregation in the certificateless scenario and build a security model upon which we analyze our proposed scheme.

Organization. The rest of the paper is organized as follows: in Section 2 we review a few important concepts used throughout our presentation; in Section 3 we review the idea of certificateless signature schemes, it's main properties and definitions; in Sections 4 and 5 we present, and analyze the security of, our proposed scheme; Section 6 brings some concluding remarks.

2 Preliminaries

2.1 Bilinear maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that satisfies the following properties.

1. **Bilinearity.** For all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(aP, bQ) = e(P, Q)^{ab}$
2. **Non-degeneracy.** Let Q be a generator of $\mathbb{G}_2$ and $\psi(.)$ an homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. Then $e(\psi(Q), Q) \neq 1$.

Additionally, we want the map e to be efficiently computable. Such a bilinear map is called *admissible*. In the particular case where $\mathbb{G}_1 = \mathbb{G}_2$, the map is called *symmetric*. Examples of bilinear maps widely used in cryptography are the Weil pairing (as in [BF01]) and the Tate pairing [Sco05].

2.2 Security assumptions and hard problems

In this paper we base our security reductions on the Computational Diffie-Hellman Problem, as defined below:

Definition 1. ***Computational Diffie-Hellman Problem*** **(CDHP).** *Given a multiplicative group* $(\mathbb{G}, .)$*, and* $\alpha, \alpha^a, \alpha^b \in \mathbb{G}$*, compute* α^{ab}*.*

The CDHP is widely regarded as a hard problem and is often used as the basis of cryptographic schemes.

2.3 Signature aggregation

The idea of signature aggregation was first introduced by Boneh et al. in [BGSL03]. It can be seen as a generalization of multisignatures, a concept some 20 years old [IN83].

A multisignature on a message M is a bitstring σ that proves that a set of users $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$ signed M. The main goal is to achieve a signature of constant size and a verification time that is significantly shorter than that of checking n signatures on M. In aggregate signatures, the unicity of the message is no longer a requirement. So, informally, an aggregate signature is comprised of a list of users $\mathbb{U} = (U_1, U_2, \ldots, U_n)$ and a list of messages $\mathbb{M} = (M_1, M_2, \ldots, M_n)$, and a bitstring that proves user U_i signed message M_i. Once again, we would like this object to be time- and space-efficient when compared to using n signatures.
We consider three types of signature aggregation:

- **Type 1 -** *Signatures on a single message by different signers.*
- **Type 2 -** *Signatures on different messages by a single signer.*
- **Type 3 -** *Signatures on multiple messages by multiple signers.*

Ideally a scheme should allow type-3 aggregation with no significant overhead, and will pose no restrictions on the list of signed messages (such as forbidding repetition[1]). Our proposed scheme is most efficient for type-2 aggregation, incurring in some (processing) overhead for type-3 aggregation.

3 Certificateless signature schemes

The original definition of a CL-PKS scheme was given by Al-Riyami and Paterson in [ARP03]. Since then, alternative formulations of that definition have been suggested [Den06, HWZD06]. We use the definition below:

Definition 2. *A Certificateless Public-Key Signature (CL-PKS) scheme consists of the following five polynomial-time algorithms:*

- **Setup.** Run by the KGC to initialize the system. Receives a security parameter 1^k and returns a list of system parameters `params` and the master key s.
- **Extract Partial Private Key.** Takes as input `params`, s, and the identity $ID \in \{0,1\}^*$ and outputs the partial private key D_{ID}, which is assumed to be sent to the correct user through a secure channel.
- **Generate Key Pair.** Takes as input `params`, D_{ID} and generates the user's public key P_{ID} and corresponding private key, S_{ID}.
- **CL-Sign.** Takes as input `params`, the user's identity ID, the pair of keys (D_{ID}, S_{ID}) and a message M. Outputs a correct signature σ on M.
- **CL-Verify.** This is the verification algorithm, that takes as input `params`, ID, P_{ID}, M and the signature σ, and outputs `ACCEPT` if and only if σ is a valid signature by user U_{ID} on M.

[1] In [BGSL03] Boneh et al. outline a simple attack that breaks their scheme if the signing of the same message by multiple users is allowed; thus, they forbid this. Later on, they claim that if the user concatenates their public key with the message prior to signing that restriction can be lifted. This is further discussed in [BNN06], and the *unrestricted* BLS aggregate scheme (signing the concatenation of message and the user's public key with no restrictions on repetition) is proven to be secure.

This is a more concise definition that still captures all the features of the original model, as shown in [HWZD06].

It is interesting to point out that, even though the certificateless scenario provides a new way to look at signatures, no new capabilities are gained by using certificateless signatures when compared to the use of traditional digital signatures. That is the case because the user needs to know all the private information before generating signatures, unlike the encryption case in which the user can publish a public key without knowing his partial key and anybody can encrypt messages for him (whose decryption will be possible after obtaining the partial key).

One direct consequence of this fact, in the signature case, is that it does not make any difference if the user obtains his partial key before or after publishing his public key: either way the partial key must be obtained before generating signatures. So the assumption in our scheme that partial keys are obtained before the full public key can be generated is not problematic and poses no restriction to the use of the scheme.

3.1 Certificateless aggregate signatures

Definition 3. *An Aggregate Certificateless Public-Key Signature (ACL-PKS) scheme consists of the five polynomial-time algorithms listed in Definition 2, and the two extra algorithms below:*

- **Aggregate.** Can be run by anyone. Takes as input a list $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of signatures, where σ_i is user's U_i signature on message M_i. Outputs the aggregate γ.
- **Aggregate verification.** Receives as input an aggregate γ and the lists $\mathbb{U}$, of users, and $\mathbb{M}$, of messages. Accepts if and only if γ was generated from the correct set σ of individual signatures.

3.2 Security of (Aggregate) Certificateless Signatures

The standard definition of security, as introduced by [GMR88], is *existential unforgeability under adaptively chosen message attack*. Since there is no certification of public keys, we have to assume that attackers can always replace an user's public key at will. On the other hand, we don't want to place unconditional trust on the KGC, so we must always take two types of adversaries into consideration:

- **Type I.** An adversary $\mathcal{A}_I$ that can replace public keys at will, but has no access to the master key s.
- **Type II.** An adversary $\mathcal{A}_{II}$ that knows the master key s but is not allowed to replace the public key of the user being attacked.

It is clear that if we were to let the KGC replace public keys the scheme would be trivially broken. Alternatives (e.g. key binding, as in [ARP03]) have been proposed to relax that assumption about the KGC. The security of certificateless signatures is thus expressed by two similar games, respectively against $\mathcal{A}_I$ and $\mathcal{A}_{II}$. All our security analysis use the Random Oracle Model and the adversaries have access to the following operations:

- `CreateUser`, `RevealPartialKey`, `RevealSecretValue`, `RevealPublicKey`, `Sign`, `QueryHash`, `ReplacePublicKey`.

If the oracle is required to generate signatures under public keys that were replaced by the adversary (as in [ARP03]), it is called a `StrongSign` oracle. If this requirement is dropped, then we have a `WeakSign` oracle. It is our opinion that, even though a proof using a `StrongSign` oracle may be desirable, as it gives the adversary more power, efficiency should not be sacrificed in order to achieve it. The power an adversary gets from a `StrongSign` oracle has no analogous in a real-world situation. On the other hand, the fact that our scheme is secure even against adversaries who are given access to a `StrongSign` oracle will be crucial to our proof of security for signature aggregation. Interestingly enough, this seems to be the first real-life justification for using such a model.

In the setting of aggregate signatures, a detail that seemed previously unimportant is a lot more relevant: namely, *are Type-II adversaries allowed to replace public keys other than that of the target identity*[2]? On the single signature case this is irrelevant, because this would give no real extra power to an adversary. On the aggregate case this is a lot more problematic, since the resulting aggregate forgery could include a signature under this replaced public key. So, in our security model we allow replacement of public keys *other than that of (one of) the user(s) under attack.*

We can now define the two following games, respectively, for $\mathcal{A}_I$ and $\mathcal{A}_{II}$;

Definition 4 (Game I: Single Signatures.)**.** *Let $\mathcal{C}_I$ be the challenger algorithm and k be a security parameter:*

1. *$\mathcal{C}_I$ generates (mpk, msk) with the same distribution of $(mpk, msk) \leftarrow$ `Setup`(1^k);*
2. *$\mathcal{C}_I$ runs $\mathcal{A}_I$ on 1^k and mpk. During its run, $\mathcal{A}_I$ has access to the following oracles: `RevealPublicKey, RevealPartialKey, RevealSecretValue, ReplacePublicKey, QueryHash, Sign`;*
3. *$\mathcal{A}_I$ outputs (ID^*, M^*, σ^*).*

$\mathcal{A}_I$ wins the game if `CL-Verify`(`params`,ID^*,P_{ID^*},M^*,σ^*)=`ACCEPT` and both conditions below hold:

- `Sign`(ID^*, M^*) was never queried;
- `RevealPartialKey`(ID^*) was also never queried.

Definition 5 (Game II: Single Signatures)**.** *Let $\mathcal{C}_{II}$ be the challenger algorithm and k be a security parameter:*

1. *$\mathcal{C}_{II}$ generates (mpk, msk) with the same distribution of $(mpk, msk) \leftarrow$ `Setup`(1^k);*
2. *$\mathcal{C}_{II}$ runs $\mathcal{A}_{II}$ on 1^k and (mpk, msk). During its run, $\mathcal{A}_{II}$ has access to the following oracles: `RevealPublicKey, RevealPartialKey, RevealSecretValue, ReplacePublicKey, QueryHash, Sign`;*
3. *$\mathcal{A}_{II}$ outputs (ID^*, M^*, σ^*).*

[2]The target identity is the one under which a forgery will be made.

$\mathcal{A}_{II}$ wins the game if `CL-Verify`(`params`,ID^*,P_{ID^*},M^*,σ^*)=`ACCEPT` and all conditions below hold:

- `Sign`(ID^*,M^*) was never queried;
- `RevealSecretValue`(ID^*) was never queried;
- `ReplacePublicKey`(ID^*, .) was never queried.

The two games used to prove the security of aggregation are very similar to these, except for the output of the adversary (an *aggregate* forgery) and that the procedure `CL-Agg-Verify` is used to check its validity. These will be defined in more detail prior to the proof of security of aggregation. For a longer discussion on alternative CLS security models, we refer the reader to [HWZD07].

4 Our scheme

In this section we present the main contribution of this paper, a novel certificateless signature scheme that is suitable for aggregation.

Setup. Let k be a security parameter;
let $\mathbb{G}$ and $\mathbb{G}_T$ be groups such that:

- the CDH assumption holds in $\mathbb{G}$;
- $p \leftarrow |\mathbb{G}| = |\mathbb{G}_T|$;
- $\exists e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ an admissible bilinear pairing;

let $P \in \mathbb{G}$ be an arbitrary generator; choose the following hash functions:

- $H_1 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_p^*$
- $H_2, H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$

choose $s \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$; let $P_0 = sP$;
output `params` $= \langle \mathbb{G}, \mathbb{G}_T, e, P, P_0 \rangle$ and `msk` $= s$.

PartialKeyExt(ID_i).
$r_i \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$; $R_i = r_i P$;
$d_{ID_i} = (r_i + sH_1(ID_i, R_i)) \bmod p$;
output $\langle d_{ID_i}, R_i \rangle$.

UserKeysGen(ID_i).
$x_{ID_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$;
$P_{ID_i} = x_{ID_i} P$;
output $\langle P_{ID_i}, x_{ID_i} \rangle$.

CL-Sign(M_i,ID_i,d_{ID_i},x_{ID_i}).
Output $\sigma = d_{ID_i} H_2(M_i, ID_i, P_{ID_i}, R_i) + x_{ID_i} H_3(M_i, ID_i, P_{ID_i}, R_i)$.

CL-Verify(σ,M_i,ID_i,P_{ID_i},R_i).
Let $h_1 = H_1(ID_i, R_i)$;

let $H_2 = H_2(M_i, ID_i, P_{ID_i}, R_i)$;
let $H_3 = H_3(M_i, ID_i, P_{ID_i}, R_i)$;
output `ACCEPT` if and only if

$$e(P, \sigma) \stackrel{?}{=} e(H_2, R_i + h_1 P_0) e(H_3, P_{ID_i}).$$

Note that public keys in this scheme are comprised of two components: the R generated by the KGC and the P_{ID} generated by the user. It is important to note that we do not assume either of them to be authenticated and when we deal with key replacement attacks, the adversary is allowed to replace either (or both) components of the public key.

4.1 Signature Aggregation

The following procedures are responsible for aggregating multiple signatures into one and for verifying aggregates.
Aggregate. Given the set $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ to aggregate, output

$$\gamma = \sum_{\forall i} \sigma_i.$$

Aggregate Verification. Given the aggregate γ, the lists of signers $\mathbb{U} = \{u_1, u_2, \ldots, u_n\}$ and messages $\mathbb{M} = \{M_1, M_2, \ldots, M_n\}$.
There may be repetitions both in the user list and in the message list.
For each user u_i in $\mathbb{U}$, define the set of messages $\mathbb{M}_{u_i} = \{M_j, u_j = u_i\}$ signed by that user. Let $\mathbb{U}^*$ be the set of users, i.e. eliminating repetitions from $\mathbb{U}$. Let

$$\begin{aligned} h_{u_i} &= H_1(ID_{u_i}, R_{u_i}) \\ \gamma_1 &= \prod_{\forall u_i \in \mathbb{U}^*} e\Big(\sum_{\forall M_i \in \mathbb{M}_{u_i}} H_2(M_{u_i}, ID_{u_i}, P_{ID_{u_i}}, R_{u_i}), R_{u_i} + h_{u_i} P_{pub} \Big) \\ \gamma_2 &= \prod_{\forall u_i \in \mathbb{U}^*} e\Big(\sum_{\forall M_i \in \mathbb{M}_{u_i}} H_3(M_{u_i}, ID_{u_i}, P_{ID_{u_i}}, R_{u_i}), P_{ID_{u_i}} \Big) \end{aligned}$$

Accept the aggregate signature if and only if:

$$e(P, \gamma) \stackrel{?}{=} \gamma_1 \gamma_2 \tag{1}$$

5 Security of the scheme

5.1 Type-I Adversaries

We want to prove that our scheme is secure against Type-I adversaries, as defined in §3.2. We achieve that by proving that if a Type-I adversary exists we can build an algorithm, which we refer to as $\mathcal{C}_I$, capable of breaking the CDHP with related probability. Since we assume the CDHP is *hard* in $\mathbb{G}$, no efficient Type-I adversary exists. So, here is the theorem we want to prove.

Theorem 1. *If there exists a Type-I adversary $\mathcal{A}_I$ that can break the EU-CMA security of our scheme with non-negligible probability $\lambda(k)$, then the CDHP can be solved in $\mathbb{G}$ with non-negligible probability approximately*

$$\Pr[\mathcal{W}] \approx \left(\frac{\lambda(k)^2}{q_u^2 q_m^2 q_{h_1}} \right)$$

Proof. Let $\mathcal{A}_I$ be a Type-I adversary against our scheme. We build a $\mathcal{C}_I$ that simulates, in the Random Oracle Model, an attack environment for $\mathcal{A}_I$. In an adaptive attack, $\mathcal{A}_I$ has access to the following oracles:

- $H_1(ID, R)$; $H_2(M, ID, P_{ID_i}, R_i)$; $H_3(M, ID, P_{ID_i}, R_i)$;
- `PartialKeyExtract`(ID);
- `PublicKeyExtract`(ID);
- `PublicKeyReplacement`(ID,R',P'_{ID});
- `SecretValueExtract`(ID);
- `Sign`(M,ID).

These must be simulated by $\mathcal{C}_I$.

Game Outline. We use games to simulate the attack environment for $\mathcal{A}_I$. This technique is somewhat more limited than pure simulation, because something is assumed about the adversary's behavior. But even considering these limitations, game-based proofs are far more common because they are much easier to understand and verify. The game we are considering has the following standard structure:

1. Let k be a security parameter; $\mathcal{C}_I$ generates suitable public parameters (`params`) and a master secret (`msk`);
2. $\mathcal{A}_I$ is run with `params` as input;
3. $\mathcal{A}_I$ makes q_{H_1}, q_{H_2}, and q_{H_3} queries respectively to the $H_1(.)$, $H_2(.)$, and $H_3(.)$ oracles, and q_S queries to the signing oracle;
4. after a polynomial number of steps $T = poly(k)$, $\mathcal{A}_I$ outputs
$$\gamma = \langle S, (\sigma^*, M^*, ID^*, P^*_{ID}, R^*) \rangle.$$
5. The success probability of $\mathcal{A}_I$ is $\Pr[S = 1] = \lambda(k)$.

The attack is considered successful if $\lambda(k)$ is non-negligible in k. We will use a technique called *game-hopping*: we will design a series of games, each a little different from the previous one, but with related success probabilities. The last of this series of games describes an $\mathcal{C}_I$ capable of solving the CDHP with probability related to $\lambda(k)$. This makes our proof more readable and less error-prone.

Game 0. This initial game is a straight simulation of the outline above.
$\mathcal{C}_I^0$ receives as input for the CDHP a group $\mathbb{G}$ and the tuple $(P, aP, bP) \in \mathbb{G}^3$.
$\mathcal{C}_I^0$ chooses $\langle \mathbb{G}_T, e \rangle$ such that:

- $|\mathbb{G}| = |\mathbb{G}_T|$
- $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an admissible pairing.

Let $p = |\mathbb{G}| = |\mathbb{G}_T|$.
$\mathcal{C}_I^0$ chooses $s \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and sets $P_0 = sP$.
$\mathcal{C}_I^0$ runs $\mathcal{A}_I$ with input $\langle \mathbb{G}, \mathbb{G}_T, e, P, P_0, \rangle$.
$\mathcal{C}_I^0$ simulates oracle queries as follows:

- $H_1(ID_i, R_i)$.
  If undefined, choose $h_{1_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_1(ID_i, R_i) = h_{1_i}$.
  Return $H_1(ID_i, R_i)$.
- $H_2(M_i, ID_i, P_{ID_i}, R_i)$.
  If undefined, choose $h_{2_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_2(M_i, ID_i, P_{ID_i}, R_i) = h_{2_i}P$.
  Return $H_2(M_i, ID_i, P_{ID_i}, R_i)$.
- $H_3(M_i, ID_i, P_{ID_i}, R_i)$.
  If undefined, choose $h_{3_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_3(M_i, ID_i, P_{ID_i}, R_i) = h_{3_i}P$.
  Return $H_3(M_i, ID_i, P_{ID_i}, R_i)$.
- `SecretValueExtract(`ID_i`)`.
  If undefined, choose $x_{ID_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $P_{ID_i} = x_{ID_i}P$.
  Return x_{ID_i}.
- `PartialKeyExtract(`ID_i`)`.
  If undefined, choose $r_i \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $R_i = r_iP$.
  Compute $d_{ID_i} = r_i + sH_1(ID_i, R_i) \bmod p$.
  Return $\langle d_{ID_i}, R_i \rangle$
- `PublicKeyExtract(`ID_i`)`
  If P_{ID_i} is undefined, call `SecretValueExtract(`ID_i`)`.
  If R_i is undefined, call `PartialKeyExtract(`ID_i`)`.
  Return $\langle P_{ID_i}, R_i \rangle$.
- `PublicKeyReplace(`ID_i, P'_{ID_i}, R'_i`)`.
  Set $P_{ID_i} = P'_{ID_i}$ and $R_i = R'_i$.
- `Sign(`M_i, ID_i`)`.
  Let $h_{1_i} = H_1(ID_i, R_i)$
  Let $h_{2_i}P = H_2(M_i, ID_i, P_{ID_i}, R_i)$.
  Let $h_{3_i}P = H_3(M_i, ID_i, P_{ID_i}, R_i)$.
  Output $\sigma_i = h_{2_i}R + h_{2_i}h_{1_i}P_0 + h_{3_i}P_{ID_i}$.

We'd like to point out that this is a so-called *Strong-Sign* oracle, because it is capable of signing messages even under replaced public-keys whose respective secret key is unknown. This will be very important for the security of signature aggregation. Simulated signatures are correct because:

$$\begin{aligned} e(P, \sigma_i) &= e(P, h_{2_i}R + h_{2_i}h_{1_i}P_0 + h_{3_i}P_{ID_i}) \\ &= e(P, h_{2_i}(R + h_{1_i}P_0))e(P, h_{3_i}P_{ID_i}) \\ &= e(h_{2_i}P, R + h_{1_i}P_0))e(h_{3_i}P, P_{ID_i}) \\ &= e(H_2(M_i, ID_i, P_{ID_i}, R_i), R + H_1(ID_i, R_i)P_0))e(H_3(M_i, ID_i, P_{ID_i}, R_i), P_{ID_i}). \end{aligned}$$

$\mathcal{A}_I$ should, after interacting with the environment simulated by $\mathcal{C}_I^0$, output a tuple $\langle S, (V^*, M^*, ID^*, P_{ID^*}, R^*)\rangle$. In this initial game, $\mathcal{C}_I^0$ is successful when $\mathcal{A}_I$ is. Therefore,

$$\Pr[\mathcal{C}_I^0] = \Pr[\mathcal{A}_I] = \Pr[S = 1] = \lambda(k)$$

Game 1. Now, $\mathcal{C}_I^1$ uses the second input for the CDHP in the game. Instead of computing s and then $P_0 = sP$, $\mathcal{C}$ makes $P_0 = aP$. $\mathcal{A}_I$ is then run as before. P_0's distribution does not change, so $\mathcal{A}_I$'s behavior also does not change. The only problem is that now $\mathcal{C}_I^1$ is unable to compute partial keys as before. Therefore we have to take special measures when computing $H_1(.)$ queries to assure that we are able to compute partial keys afterwards. Here are the new $H_1(.)$ and `PartialKeyExtraction` oracle:

- $H_1(ID_i, R_i)$
  If the partial key of ID_i has not been computed yet

  call `PartialKeyExtract(`ID_i`).`

  If $H_1(ID_i, R_i)$ is undefined

  choose $h_{1_i} \xleftarrow{r} \mathbb{Z}_p^*$;
  set $H_1(ID_i, R_i) = h_{1_i}$.

  Return $H_1(ID_i, R_i)$
- `PartialKeyExtract(`ID_i`).`
  If undefined

  choose $d_{ID_i}, h_{1_i} \xleftarrow{r} \mathbb{Z}_p^*$;
  compute $R_i = d_{ID_i}P - h_{1_i}P_0$;

  Return $\langle d_{ID_i}, R_i\rangle$

This change is unnoticeable to $\mathcal{A}_I$: the partial keys and hash values generated are valid and uniformly distributed in $\mathbb{Z}_p^*$. Therefore, $\mathcal{A}_I$'s output and success probability will remain unchanged.

$$\Pr[\mathcal{C}_I^1] = \Pr[\mathcal{C}_I^0] = \lambda(k).$$

Game 2. Game 2 is the same as Game 1, except that $\mathcal{C}_I^2$ randomly picks a target (ID, R)-pair and only outputs success if the forgery is under this (ID, R). An important detail is that $\mathcal{C}_I^2$ cannot know a valid partial key for the target (ID, R). If coincidently the partial keys for (ID, R) are known, $\mathcal{C}_I^2$ must abort and output `FAIL`.

Let q_u be the maximum number of distinct (ID, R) queried throughout the simulation.

A very conservative upper limit for q_u is $q_u \leq q_{H_1} + q_{H_2} + q_{H_3} + q_s + q_{pk}$.

Before the game starts, $\mathcal{C}_I^2$ chooses $t \xleftarrow{r} \{1, \ldots, q_u\}$.

The $H_1(ID, R)$ oracle is changed to:

- $H_1(ID_i, R_i)$
  If the partial key of ID_i has not been computed yet

  call `PartialKeyExtract(`ID_i`).`

If this is the t^{th} distinct query

set $ID_t = ID_i$ and $R_t = R_i$
if the known partial key for ID_t includes R_t,
abort and `FAIL`.

If $H_1(ID_i, R_i)$ is undefined

choose $h_{1_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$;
set $H_1(ID_i, R_i) = h_{1_i}$.

Return $H_1(ID_i, R_i)$

This change only affects $\mathcal{A}_I$ if $\mathcal{C}_I^2$ aborts. That happens with very low probability, only if the randomly selected R_i in ID_t's partial key equals R_t. Let F_1 be the event in which $\mathcal{C}_I^2$ aborts at this point. Then $\Pr[F_1] = \frac{1}{p}$.
Let F_2 be the event in which $\mathcal{A}_I$ does not query $H_1(ID^*, R^*)$. Since $H_1(.)$ is a random oracle, the probability of a valid forgery being generated without this value being queried is $\Pr[F_2] = \frac{1}{p}$. Additionally, we redefine $\mathcal{C}_I^2$'s output to

$$\text{output}_{\mathcal{C}_I^2} = \begin{cases} \langle 0, \perp \rangle, & \text{if } S = 1 \wedge (ID^* \neq ID_t \vee R^* \neq R_t) \\ \text{output}_{\mathcal{A}_I}, & \text{otherwise} \end{cases}$$

Now, $\mathcal{C}_I^2$'s probability of success is smaller than $\mathcal{A}_I$'s: the target (ID_t, R_t)-pair must also be correct. So we have

$$\Pr[\mathcal{C}_I^2] = \Pr[\mathcal{C}_I^1 \wedge \overline{F_1} \wedge \overline{F_2} \wedge (ID^* = ID_t \wedge R^* = R_t)] = \lambda(k) \left(1 - \frac{1}{p}\right)^2 \frac{1}{q_u}.$$

Still, if $\lambda(k)$ is non-negligible so is $\Pr[Y_I^2]$.
Game 3. In Game 3, $\mathcal{C}_I^3$ chooses a target $(M_i, ID_i, P_{ID_i}, R_i)$-tuple.

Let q_m be the maximum number of distinct $H_2(M_i, ID_i, P_{ID_i}, R_i)$ queried throughout the simulation.
A very conservative upper limit for q_m is $q_m \leq q_{H_2} + q_s$.
Before the game starts, $\mathcal{C}_I^3$ chooses $u \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$.

The $H_2(M_i, ID_i, P_{ID_i}, R_i)$ oracle is now changed to

- $H_2(M_i, ID_i, P_{ID_i}, R_i)$.
  If this is the u^{th} distinct query

  set $M_u = M_i$, $ID_u = ID_i$, $P_{ID_u} = P_{ID_i}$, $R_u = R_i$.

  If undefined, choose $h_{2_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_2(M_i, ID_i, P_{ID_i}, R_i) = h_{2_i} P$.
  Return $H_2(M_i, ID_i, P_{ID_i}, R_i)$.

Once again, this change has no consequences for the adversary, and its behavior will remain the same. We redefine $\mathcal{C}_I^3$'s output to be:

$$\text{output}_{\mathcal{C}_I^3} = \begin{cases} \langle 0, \perp \rangle, & \text{if } S = 1 \wedge (ID^* \neq ID_t \vee R^* \neq R_t) \\ \langle 0, \perp \rangle, & \text{if } S = 1 \wedge (M^* \neq M_u \vee ID^* \neq ID_u \vee P_{ID^*} \neq P_{ID_u} \vee R^* \neq R_u) \\ \text{output}_{\mathcal{A}_I}, & \text{otherwise} \end{cases}$$

$\mathcal{C}_I^3$'s probability of success is smaller than $\mathcal{C}_I^2$'s: the target $(M_i, ID_i, P_{ID_i}, R_i)$-tuple must also be correct.
Let

S_2^* be the event where $(ID^* = ID_t \wedge R^* = R_t)$;
S_3^* be the event where $(M^* = M_u \wedge ID^* = ID_u \wedge P_{ID^*} = P_{ID_u} \wedge R^* = R_u)$.

We have

$$\Pr[S_2^*] = 1/q_u \left(1 - \tfrac{1}{p}\right);$$
$$\Pr[S_3^*] = 1/q_m \left(1 - \tfrac{1}{p}\right);$$
$$\Pr[\mathcal{C}_I^3] = \Pr[Y_I^1 \wedge S_2^* \wedge S_3^*] = \Pr[\mathcal{C}_I^2 \wedge S_3^*] = \lambda(k) \left(\tfrac{1}{q_u}\right) \left(\tfrac{1}{q_m}\right) \left(1 - \tfrac{1}{p}\right)^3.$$

So, if $\lambda(k)$ is non-negligible so is $\Pr[\mathcal{C}_I^3]$.
Game 4. Now $\mathcal{C}_I^4$ uses the third input from the CDHP. $\mathcal{C}_I^4$ sets the value of $H_2(M_u, ID_u, P_{ID_u}, R_u) = bP$. This does not change $H_2(.)$'s distribution and is unnoticeable to $\mathcal{A}_I$. The only problem is that, now, $\mathcal{C}_I^4$ is unable to forge signatures on that exact input: if the `Sign` oracle receives input $(M_u, ID_u, P_{ID_u}, R_u)$ it must fail. But this can only happen if the forgery output by $\mathcal{A}_I$ is not of $(M_u, ID_u, P_{ID_u}, R_u)$, in which case $\mathcal{C}_I^3$ would already fail anyway. So this, in fact, does not change $\mathcal{C}_I^4$'s success probability

$$\Pr[\mathcal{C}_I^4] = \Pr[\mathcal{C}_I^3] = \frac{\lambda(k)}{q_u q_m} \left(1 - \frac{1}{p}\right)^3$$

Now, let's analyze $\mathcal{C}_I^4$'s output in case of success:

$$e(P, \sigma) = e(H_2(.), R^* + H_1(.)P_0) e(H_3(.), P_{ID^*}).$$

Let $W = \sigma - h_{3_i} P_{ID^*}$. We have

$$e(P, W) = e(H_2(.), R^* + H_1(.)P_0) = e(bP, R^* + H_1(.)aP).$$

Since we know the value of $H_1(ID^*, R^*)$, W almost gives the answer to our CDHP instance, were it not for the R^* factor: since we cannot assume the discrete log of $R^*(r, R^* = rP)$ is known to the adversary we cannot get our answer directly from W. We can however use the Oracle Replay Technique [PS00] and obtain another related forgery σ' that helps us solve the CDHP. In fact, we state the following claim, delaying its proof until §5.2:

Claim 1. *Let $\tilde{\lambda}(k) = \lambda(k) - \frac{1}{p}$, where $\lambda(k)$ is the success probability of $\mathcal{A}_I$ in the Game 4 above. We can use the Oracle Replay Technique to construct a $\mathcal{W}$ such that with probability*

$$\Pr[\mathcal{W}] = \left(\frac{1}{q_u q_m} \left(1 - \frac{1}{p}\right)^3\right)^2 \left(\frac{\tilde{\lambda}(k)^2}{8 q_{h_1}}\right) \left(1 - \frac{1}{q_{h_1}}\right) \approx \left(\frac{\tilde{\lambda}(k)^2}{8 q_{h_1} q_u^2 q_m^2}\right)$$

a pair of forgeries $\varsigma_i = (\sigma_i, M_i, ID_i, P_{ID_i}, R_i)$ for $i \in \{1, 2\}$ are obtained such that:

- $ID_1 = ID_2 = ID^*$;
- $R_1 = R_2 = R^*$;

- $H_1'(ID_1, R_1) \neq H_1''(ID_2, R_2)$

We will give a formal proof of this claim on the next subsection, but assume for now it is true. Now, since both forgeries are valid, we have that

$$\begin{aligned} e(P, \sigma_1) &= e(H_2'(.), R^* + H_1'(.)P_0)e(H_3'(.), P'_{ID_1}), \\ e(P, \sigma_2) &= e(H_2''(.), R^* + H_1''(.)P_0)e(H_3''(.), P'_{ID_2}). \end{aligned}$$

letting $W' = \sigma' - h'_{3_i} P_{ID_1}$ and $W'' = \sigma_2 - h''_{3_i} P_{ID_2}$, we have

$$\begin{aligned} &e(P, W'') = e(H_2''(.), R^* + H_1''(.)P_0) = e(bP, R^*)e(bP, H_1''(.)aP). \\ &e(P, W'' - W') = e(bP, R^*)e(bP, H_1''(.)aP)/[e(bP, R^*)e(bP, H_1'(.)aP)] \\ &e(P, W'' - W') = e(bP, (H_1'' - H_1')aP). \end{aligned}$$

So, $Y = (W'' - W')/(H_1'' - H_1') = abP$ is the answer to our CDHP instance. □

5.2 The Oracle Replay technique

Now we have to prove claim 1 and the proof of theorem 1 will be complete. Take an execution of the game 4 above. Let ϵ be the success probability of $\mathcal{A}_I^4$, i.e. the probability that $\mathcal{A}_I$ outputs $S = 1$ in the Game 4 above. The probability that $S = 1$ but the hash $H_1(ID^*, R^*)$ was never queried is $\frac{1}{p}$, due to the randomness of $H_1(.)$. Let β be the index of the $H_1(ID^*, R^*)$ query ($\beta = \infty$ if the query was never made).

Corollary 1. $\Pr[S = 1 \wedge \beta \neq \infty] \geq \epsilon - \frac{1}{p}$

We will define the machine $\mathcal{W}$, which acts exactly as $\mathcal{C}_I^4$ up until the point where a forgery is output by $\mathcal{A}_I$ and then "forks" the execution. Let χ be the set of all possible random tapes appearing in a simulation by $\mathcal{C}_I^4$ (and thus by $\mathcal{W}$).
Let $Q_{1,1}, Q_{1,2}, \ldots, Q_{1,q_{h_1}}$ be the different queries made by $\mathcal{A}_I$ to $H_1(.)$ throughout the simulation. Let $\rho = (\rho_1, \rho_2, \ldots, \rho_{q_{h_1}})$ be the list of answers computed by $\mathcal{C}_I^4$.
Fact. We know that $Q_{1,\beta} = (ID^*, R^*)$.
Let ψ be the set of executions of $\mathcal{A}_I$ such that $S = 1$ and $\beta \neq \infty$.
Let ψ_i be the subset of ψ such that $\beta = i$.
Fact. $\psi = \bigcup_{\forall i} \psi_i$, and $\Pr[(\chi, H_1(.)) \in \psi] = \epsilon - \frac{1}{p}$,
where the probability is computed over the possible executions χ and oracle instantiations of H_1. Let I be the set of indexes of the more likely ψ_i, i.e.

$$I = \left\{ i, \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi] \geq \frac{1}{2q_{h_1}} \right\} \tag{2}$$

Now let $\psi_I = \{(\chi, H_1) \in \psi_i, i \in I\}$. Then $\forall i \in I$,

$$\Pr[(\chi, H_1) \in \psi_i] = \Pr[(\chi, H_1) \in \psi]. \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi] \geq \left(\epsilon - \frac{1}{p}\right)\left(\frac{1}{2q_{h_1}}\right)$$

Lemma 1. $\Pr[(\chi, H_1) \in \psi_I | (\chi, H_1) \in \psi] \geq \frac{1}{2}$.

Proof. Since the sets are disjoint,

$$\begin{aligned}\Pr[(\chi, H_1) \in \psi_I | (\chi, H_1) \in \psi] &= \sum_{i \in I} \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi] \\ &= 1 - \sum_{i \notin I} \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi]\end{aligned}$$

From the definition of I (eq. (2)),

$$\forall i \notin I, \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi] < \frac{1}{2q_{h_1}}$$

and since there are at most q_{h_1} i-indexes we have that

$$\begin{aligned}\Pr[(\chi, H_1) \in \psi_I | (\chi, H_1) \in \psi] &= 1 - \sum_{i \notin I} \Pr[(\chi, H_1) \in \psi_i | (\chi, H_1) \in \psi] \\ &\geq 1 - q_{h_1} \left(\frac{1}{2q_{h_1}} \right) \\ &\geq \frac{1}{2}\end{aligned}$$

□

That proves the fact that the probability of a successful execution being in the most likely set is at least $\frac{1}{2}$. We proceed by using the following Splitting Lemma.

Lemma 2. (The Splitting Lema.) *Let X and Y be two finite sets where two probabilities 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*

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

and $\overline{B} = X \times Y - B$, 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. $\Pr[B|A] \geq \alpha/\gamma$.

This lemma allows us to "split" the execution in two: the queries made before and after $Q_{1,\beta}$. By doing this we can apply the splitting lemma to prove that there are enough valid executions starting with the same queries so that we can expect two successful attacks coinciding up to the point the query $Q_{1,\beta}$ is made to exist.

More specifically, we can use the Splitting Lemma with the following values:

$$\begin{aligned}X &= (\chi, H_{1\beta^-}) \\ Y &= H_{1\beta^+} \\ \gamma &= \tilde{\epsilon}/2q_{h_1} \\ \alpha &= \tilde{\epsilon}/4q_{h_1} = \gamma/2\end{aligned}$$

where:

- H_{1i^-} denotes the hash queries $\{H_{1,1}, H_{1,2}, \ldots, H_{1,i-1}\}$;
- H_{1i^+} denotes the hash queries $\{H_{1,i}, H_{1,i+1}, \ldots, H_{1,q_{h_1}}\}$;
- $\tilde{\epsilon} = \Pr[S = 1 \wedge \beta \neq \infty] = \epsilon\left(1 - \frac{1}{p}\right)$.

Thus, if we let $A = \psi_\beta$, there exists a $\Omega_\beta \subset \psi_\beta$ (the B in the Splitting Lemma definition above) such that:

$$\Pr[(\chi, H_1) \in \Omega_\beta | (\chi, H_1) \in \psi_\beta] = \frac{\alpha}{\gamma} = \frac{1}{2}$$

and $\forall (\chi, H_1) \in \Omega_\beta$

$$\Pr[(\chi, H_{1\beta^-} || H_{1\beta^+}) \in \psi_\beta] = \gamma - \alpha = \frac{\tilde{\epsilon}}{4q_{h_1}}$$

So, let $\mathcal{W}$ be the algorithm that, on a successful execution of $\mathcal{C}_I^4$, repeats the simulation with a fixed $(\chi, H_{1\beta^-})$ and random $\tilde{H_{1\beta^+}}$. We know that:

$$\Pr\left[\left((\chi, H_{1\beta^-} || \tilde{H_{1\beta^+}}) \in \psi_\beta\right) \wedge (\tilde{\rho_\beta} \neq \rho_\beta)\right] = \frac{\tilde{\epsilon}}{4q_{h_1}}\left(1 - \frac{1}{q_{h_1}}\right)$$

Let $\mathcal{W}$ output $\langle S, (V_1, M_1, ID_1, P_{ID_1}, R_1), (V_2, M_2, ID_2, P_{ID_2}, R_2)\rangle$, where $\varsigma_i = (V_i, M_i, ID_i, P_{ID_i}, R_i)$ is the forgery obtained on the i^{th} execution of the attack, and $S = 1$ if and only if $(ID_1 == ID_2 \wedge R_1 == R_2)$ and both forgeries are valid. For $\mathcal{W}$ to be successful four things need to happen:

1. The first execution of $\mathcal{A}_I$ has to belong to the set of most likely successful executions, I. Let this event be called W_1.

$$\Pr[W_1] = \Pr[(\chi, H_1) \in \psi_I] = \frac{1}{2}\left(\epsilon - \frac{1}{p}\right) = \frac{\tilde{\epsilon}}{2}$$

2. The execution must also be in the B set of the Splitting Lemma definition. Let this event be called W_2.

$$\Pr[W_2] = \Pr[(\chi, H_1) \in \Omega_\beta | (\chi, H_1) \in \psi_\beta] = \frac{1}{2}$$

3. The replay of the oracle must also be successful. Let this event be called W_3.

$$\Pr[W_3] = \Pr[((\chi, H_{1\beta^-} || \tilde{H_{1\beta^+}}) \in \psi_\beta) \wedge (\tilde{\rho_\beta} \neq \rho_\beta)] = \left(\frac{\tilde{\epsilon}}{4q_{h_1}}\right)\left(1 - \frac{1}{q_{h_1}}\right)$$

4. Finally, the conditions that make $\mathcal{C}_I^4$ successful must also hold in both executions. Let this event be called W_4:

$$\Pr[W_4 | W_1 \wedge W_2 \wedge W_3] = \left(\frac{1}{q_u q_m}\left(1 - \frac{1}{p}\right)^3\right)^2$$

That gives us the overall success probability of $\mathcal{W}$ and concludes the proof of claim 1:

$$\Pr[\mathcal{W}] = \Pr[W_1]\Pr[W_2]\Pr[W_3]\Pr[W_4] = \left(\frac{\tilde{\epsilon}^2}{8q_{h_1}}\right)\left(1 - \frac{1}{q_{h_1}}\right)\left(\left(\frac{1}{q_u q_m}\right)\left(1 - \frac{1}{p}\right)^3\right)^2$$

□

5.3 Type-II Adversaries

The security proof against Type-II adversaries follows the same general ideas as the one against Type-I adversaries.

Theorem 2. *If there exists a Type-II adversary $\mathcal{A}_{II}$ that can break the EU-CMA security of our scheme with non-negligible probability $\lambda(k)$, then the CDHP can be solved in $\mathbb{G}$ with non-negligible probability*

$$\Pr[\mathcal{C}_{II}^5] = \frac{\lambda(k)}{q_{ID}q_m}(1 - \frac{1}{p})^2.$$

Proof. Let $\mathcal{A}_{II}$ be a Type-II adversary against our scheme. During the attack simulation, $\mathcal{A}_{II}$ may query the following oracles:

- $H_1(ID, R)$; $H_2(M, ID, P_{ID}, R)$; $H_3(M, ID, P_{ID}, R)$;
- `PartialKeyExtract`(ID);
- `PublicKeyExtract`(ID);
- `PublicKeyReplacement`(ID,P'_{ID},R');
- `SecretValueExtract`(ID);
- `Sign`(M,ID).

The game goes roughly as follows:

1. Let k be a security parameter; $\mathcal{C}_{II}$ creates suiting (mpk, msk);
2. $\mathcal{A}_{II}$ is run with input (mpk, msk);
3. $\mathcal{A}_{II}$ makes q_{H_1}, q_{H_2}, and q_{H_3} queries respectively to the $H_1(.)$, $H_2(.)$, and $H_3(.)$ oracles, and q_S queries to the signing oracle;
   - remember that all $q_{H_1}, q_{H_2}, q_{H_3}, q_s \in O(poly(k))$.
4. after a polynomial number of steps $T = O(poly(k))$, $\mathcal{A}_{II}$ outputs

$$\gamma = \langle S, (\sigma^*, M^*, ID^*, P^*_{ID}, R^*)\rangle.$$

5. The attack is deemed successful ($S = 1$) if:
   - A valid forgery is generated;
   - $\mathcal{A}_{II}$ has not replaced ID^*'s public key;
   - $\mathcal{A}_{II}$ has not queried ID^*'s secret value.
6. The success probability of $\mathcal{A}_{II}$ is $\Pr[\mathcal{A}_{II}] = \Pr[S = 1] = \lambda(k)$

Game 0. $\mathcal{C}_{II}$ receives as input to the CDHP the group $\mathbb{G}$ and $(P, aP, bP) \in \mathbb{G}^3$.
$\mathcal{C}_{II}^0$ chooses $\langle e, \mathbb{G}_T\rangle$, such that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible pairing.
Let $p = |\mathbb{G}| = |\mathbb{G}_T|$.
$\mathcal{C}_{II}^0$ chooses $s \xleftarrow{r} \mathbb{Z}_p^*$. Let $P_0 = sP$.
$\mathcal{C}_{II}^0$ runs $\mathcal{A}_{II}$ with input `params` $= \langle \mathbb{G}, \mathbb{G}_T, e, P, P_0\rangle$, and `msk` $= s$.
$\mathcal{C}_{II}^0$ simulates oracle queries as follows:

- $H_1(ID_i, R_i)$.
  If undefined, choose $h_{1_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_1(ID_i, R_i) = h_{1_i}$.
  Return $H_1(ID_i, R_i)$.
- $H_2(M_i, ID_i, P_{ID_i}, R_i)$.
  If undefined, choose $h_{2_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_2(M_i, ID_i, P_{ID_i}, R_i) = h_{2_i}P$.
  Return $H_2(M_i, ID_i, P_{ID_i}, R_i)$.
- $H_3(M_i, ID_i, P_{ID_i}, R_i)$.
  If undefined, choose $h_{3_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $H_3(M_i, ID_i, P_{ID_i}, R_i) = h_{3_i}P$.
  Return $H_3(M_i, ID_i, P_{ID_i}, R_i)$.
- `SecretValueExtract(`ID_i`)`.
  If undefined, choose $x_{ID_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $P_{ID_i} = x_{ID_i}P$.
  Return x_{ID_i}.
- `PartialKeyExtract(`ID_i`)`.
  If undefined, choose $r_i \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $R_i = r_iP$.
  Compute $d_{ID_i} = r_i + sH_1(ID_i, R_i)$.
  Return $\langle d_{ID_i}, R_i \rangle$
- `PublicKeyExtract(`ID_i`)`
  If P_{ID_i} is undefined, call `SecretValueExtract(`ID_i`)`.
  If R_i is undefined, call `PartialKeyExtract(`ID_i`)`.
  Return $\langle P_{ID_i}, R_i \rangle$.
- `PublicKeyReplace(`ID_i, P'_{ID_i}, R'_i`)`.
  Set $P_{ID_i} = P'_{ID_i}$ and $R_i = R'_i$.
- `Sign(`M_i, ID_i`)`.
  Let $h_{1_i} = H_1(ID_i, R_i)$
  Let $h_{2_i}P = H_2(M_i, ID_i, P_{ID_i}, R_i)$.
  Let $h_{3_i}P = H_3(M_i, ID_i, P_{ID_i}, R_i)$.
  Output $\sigma_i = h_{2_i}R + h_{2_i}h_{1_i}P_0 + h_{3_i}P_{ID_i}$.

The simulated signature is correct because

$$\begin{aligned} e(P, \sigma_i) &= e(P, h_{2_i}R + h_{2_i}h_{1_i}P_0 + h_{3_i}P_{ID_i}) \\ &= e(P, h_{2_i}(R + h_{1_i}P_0))e(P, h_{3_i}P_{ID_i}) \\ &= e(h_{2_i}P, R + h_{1_i}P_0))e(h_{3_i}P, P_{ID_i}) \\ &= e(H_2(M_i, ID_i, P_{ID_i}, R_i), R + H_1(ID_i, R_i)P_0))e(H_3(M_i, ID_i, P_{ID_i}, R_i), P_{ID_i}). \end{aligned}$$

$\mathcal{A}_{II}$ should, after interacting with the environment simulated by $\mathcal{C}_{II}^0$, output a tuple $\langle S, (V^*, M^*, ID^*, P_{ID^*}, R^*) \rangle$. In this initial game, $\mathcal{C}_{II}^0$ outputs exactly the same as $\mathcal{A}_{II}$. Therefore, they have the same success probability

$$\Pr[\mathcal{C}_{II}^0] = \Pr[S = 1] = \lambda(k)$$

Game 1. Game 1 is the same as Game 0, except that $\mathcal{C}_{II}^1$ randomly picks a target identity and only outputs success if the forgery is under this target identity.

Let q_{ID} be the total number of different identities queried throughout the simulation; before the game starts, $\mathcal{C}_{II}^1$ chooses $t \stackrel{r}{\leftarrow} \{1, \ldots, q_{ID}\}$.

To keep track of how many different identities have been queried we can define a procedure called at the beginning of every oracle call, `CreateUser`ID_i, that simply checks if any query has been made on ID_i and increments a global counter in case this is the first.
The `CreateUser` oracle is defined as:

- `CreateUser`(ID_i).
  If ID_i has not been queried before

  $id\_count = id\_count+1$;

  If this is the t^{th} distinct ID queried (i.e., $id\_count == t$):

  set $ID_t = ID_i$.

Since this change has no consequences for the adversary, its behavior will remain the same and so will its probability of outputting a forgery. Now we redefine $\mathcal{C}_{II}^1$'s output to be:

$$\text{output}_{\mathcal{C}_{II}} = \begin{cases} \langle 0, \perp \rangle, & \text{if } S = 1 \wedge (ID^* \neq ID_t) \\ \text{output}_{\mathcal{A}_{II}}, & \text{otherwise} \end{cases}$$

Now, $\mathcal{C}_{II}^1$'s probability of success is smaller than $\mathcal{A}_{II}$'s: the target ID must be correctly guessed. So we have

$$\Pr[\mathcal{C}_{II}^1] = \lambda(k) \left(\frac{1}{q_{ID}}\right) \left(1 - \frac{1}{p}\right)$$

So, if $\lambda(k)$ is non-negligible (in k) so is $\Pr[\mathcal{C}_{II}^1]$.

Game 2. Now the game is altered so that two queries, extraction of the secret value and the replacing of the public key, cannot be made on ID_t:

- `SecretValueExtract(`ID_i`)`.
  Call `CreateUser`(ID_i).
  If $ID_i = ID_t$, `FAIL`.
  If x_{ID_i} and P_{ID_i} are undefined,

  choose $x_{ID_i} \stackrel{r}{\leftarrow} \mathbb{Z}_p^*$ and set $P_{ID_i} = x_{ID_i} P$.

  Return x_{ID_i}.
- `PublicKeyReplace(`ID_i, P'_{ID_i}, R'_i`)`.
  Call `CreateUser`(ID_i).
  If $ID_i = ID_t$, `FAIL`.
  Set $P_{ID_i} = P'_{ID_i}$ and $R_i = R'_i$.

The probability of success remains the same, since by definition in every execution where $\mathcal{C}_{II}^2$ was successful, i.e. $ID^* = ID_t$, the two oracles above couldn't have been queried on ID^*.

$$\Pr[\mathcal{C}_{II}^2] = \Pr[\mathcal{C}_{II}^1] = \lambda(k)\left(\frac{1}{q_{ID}}\right)\left(1 - \frac{1}{p}\right)$$

Game 3. In game 3 the third input to the CDHP is used as the target ID's public-key.

- `PublicKeyExtract(`ID_i`)`.
  Call `CreateUser(`ID_i`)`.
  If $ID_i = ID_t$ set $P_{ID_i} = bP$ and $x_{ID_i} = \perp$.
  If P_{ID_i} is undefined, call `SecretValueExtract(`ID_i`)`.
  If R_i is undefined, call `PartialKeyExtract(`ID_i`)`.
  Return $\langle P_{ID_i}, R_i \rangle$.

Since bP is uniformly chosen, the distribution of P_{ID_i} is unchanged, yielding

$$\Pr[\mathcal{C}_{II}^3] = \Pr[\mathcal{C}_{II}^2] = \lambda(k)\left(\frac{1}{q_{ID}}\right)\left(1 - \frac{1}{p}\right)$$

Game 4. In Game 4 $\mathcal{C}_{II}^4$ chooses a target $(M_i, ID_i, P_{ID_i}, R_i)$-tuple.

Let q_m be the maximum number of distinct $H_3(M_i, ID_i, P_{ID_i}, R_i)$ queried throughout the simulation.

A conservative upper limit for q_m is $q_m \leq q_{H_3} + q_s$.

Before the game starts, $\mathcal{C}$ chooses $u \xleftarrow{r} \{1, \ldots, q_m\}$.

The $H_3(M_i, ID_i, P_{ID_i}, R_i)$ oracle is now changed to

- $H_3(M_i, ID_i, P_{ID_i}, R_i)$.
  Call `CreateUser(`ID_i`)`;
  if this is the u^{th} distinct query;

  set $M_u = M_i$, $ID_u = ID_i$, $P_{ID_u} = P_{ID_i}$, $R_u = R_i$.

  If undefined, choose $h_{3_i} \xleftarrow{r} \mathbb{Z}_p^*$ and set $H_3(M_i, ID_i, P_{ID_i}, R_i) = h_{3_i}P$.
  Return $H_3(M_i, ID_i, P_{ID_i}, R_i)$.

Once again, this change is unnoticeable for the adversary, and its behavior will remain the same. We redefine $\mathcal{C}_{II}^4$'s output to be:

$$\text{output}_{\mathcal{C}_{II}^4} = \begin{cases} \langle 0, \perp \rangle, & \text{if } S = 1 \wedge ID^* \neq ID_t \\ \langle 0, \perp \rangle, & \text{if } S = 1 \wedge (M^* \neq M_u \vee ID^* \neq ID_u \vee P_{ID^*} \neq P_{ID_u} \vee R^* \neq R_u) \\ \text{output}_{\mathcal{A}_{II}}, & \text{otherwise} \end{cases}$$

$\mathcal{C}_{II}^4$'s probability of success is even smaller than $\mathcal{C}_{II}^4$'s: the target $(M_i, ID_i, P_{ID_i}, R_i)$-tuple must also be correct. Let S_2^* be the event where $(ID^* = ID_t)$ and S_3^* be the event where

$(M^* = M_u \wedge ID^* = ID_u \wedge P_{ID^*} = P_{ID_u} \wedge R^* = R_u)$.
So we have

$$\Pr[S_2^*] = 1/q_u(1 - \tfrac{1}{p});$$
$$\Pr[S_3^*] = 1/q_m(1 - \tfrac{1}{p});$$
$$\Pr[\mathcal{C}_{II}^4] = \Pr[\mathcal{C}_{II}^3 \wedge S_3^*] = \Pr[\mathcal{C}_{II}^0 \wedge S_2^* \wedge S_3^*] = \tfrac{\lambda(k)}{q_{ID}q_m}(1 - \tfrac{1}{p})^2.$$

So, if $\lambda(k)$ is non-negligible so is $\Pr[\mathcal{C}_{II}^4]$.

Game 5. Now, the last input of the CDHP is used, and $H_3(M_u, ID_u, P_{ID_u}, R_u) \leftarrow bP$. Once again this change is unnoticeable for $\mathcal{A}_{II}$, so

$$\Pr[\mathcal{C}_{II}^5] = \Pr[\mathcal{C}_{II}^4] = \frac{\lambda(k)}{q_{ID}q_m}(1 - \frac{1}{p})^2.$$

Solving the CDHP. If $\mathcal{C}_{II}^5$ outputs $\langle S, (\sigma^*, M^*, ID^*, P_{ID^*}, R^*)\rangle$ such that $S = 1$, we know that:

$$e(P, \sigma^*) = e(H_2(M^*, ID^*, P_{ID^*}, R_i), R^* + H_1(ID^*, R^*)P_0)e(H_3(M^*, ID^*, P_{ID^*}, R_i), P_{ID^*}).$$

where

- $P_{ID^*} = aP$
- $H_3(M^*, ID^*, P_{ID^*}, R_i) = bP$

So we can compute $W = \sigma^* - h_{2^*}R^* - h_{2^*}H_1(ID^*, R^*)P_0$, which gives us:

$$\begin{aligned} e(P, W) &= e(H_3(M^*, ID^*, P_{ID^*}, R_i), P_{ID^*}) \\ &= e(bP, aP). \end{aligned}$$

So, W is the answer to our CDHP instance.

□

5.4 Security of aggregation

To prove the security of aggregation we will use the technique from [BNN06]. Basically we will reduce forging a single signature to the forgery of an aggregate, thus proving that if the plain signature scheme is secure, so is its aggregation. An important point of this proof is that we need our scheme to be secure even if the adversary is provided with a `StrongSign` oracle, which is the case of the proof from the previous section. This fact will be crucial for the proof of security of aggregation.

Let's assume there is an adversary $\mathcal{A}$ capable of forging aggregate signatures. We create an algorithm $\mathcal{B}$, that uses $\mathcal{A}$ to forge a single signature, thus breaking the security of the signature scheme. $\mathcal{B}$, being an adversary itself, is provided with an attack environment just like the adversaries in previous proofs of security. It then simulates an attack environment for $\mathcal{A}$ and makes sure that if $\mathcal{A}$ is capable of generating an aggregate forgery, $\mathcal{B}$ generates a single forgery. The basic execution of $\mathcal{B}$ looks like this:

1. $\mathcal{B}$ runs and receives the system parameters $\text{params}_{\mathcal{B}}$;
2. $\mathcal{B}$ computes $\text{params}_{\mathcal{A}}$ (possibly equal to $\text{params}_{\mathcal{B}}$);
3. $\mathcal{B}$ runs $\mathcal{A}$ with $\text{params}_{\mathcal{A}}$ as parameter;
4. $\mathcal{A}$ runs for a polynomial number of steps
   - $\mathcal{B}$ has to simulate the usual oracles for $\mathcal{A}$, probably using the oracles it ($\mathcal{B}$) was given as part of the attack.
5. $\mathcal{A}$ outputs an aggregate forgery γ^* with associated lists of users ($\mathbb{U}^*$) and messages ($\mathbb{M}^*$).
   - For the attack to be successful, there must be at least one i such that the signature of (U_i, M_i) has not been queried by $\mathcal{A}$ to the signing oracle and, relative to which, the conditions of the specific type of attack (Type-I or Type-II adversary) are respected.
6. $\mathcal{B}$ uses γ^* to create a forgery σ by U^* on message M^*, such that $\mathcal{B}$ has not queried (U^*, M^*) to its signing oracle.

If we can construct such a $\mathcal{B}$, we prove that aggregate signatures are as secure as single signatures in our scheme.

Theorem 3. *If the standard version of our scheme is secure then so is its aggregate version*

Proof. $\mathcal{B}$ "simulates" oracle queries simply by forwarding them to the oracles it ($\mathcal{B}$) has access to. The oracles defined for the single-signature games and for the aggregate-signature games are exactly the same, and the restrictions on the operation of an aggregate adversary of each type are the same of those of a single-signature adversary. So, $\mathcal{B}$ does not have to take any special precautions throughout this simulation.

After a valid simulation, $\mathcal{A}$ outputs $\langle \gamma^*, \mathbb{U}, \mathbb{M} \rangle$ as specified above. Let (U^*, M^*) be an arbitrary pair that has not been queried to the signing oracle by $\mathcal{A}$. We know (eq. 1) that, if γ^* is a valid aggregate forgery,

$$e(P, \gamma^*) = \prod_{u_i} \left(e(\sum_{M_i \in \mathbb{M}_{u_i}} H_2(.), R_{u_i} + h_{u_i} P_{\text{pub}}) e(\sum_{M_i \in \mathbb{M}_{u_i}} H_3(.), P_{ID_{u_i}})) \right)$$

We will now divide the (u_i, m_i) pairs in three types:

1. If $u_i \neq U^*$, then $i \in C_1$;
2. If $u_i = U^*$ but $m_i \neq M^*$, then $i \in C_2$;
3. If $u_i = U^*$ and $m_i = M^*$, then $i \in C_3$;

What $\mathcal{B}$ does next is "remove" each σ_i corresponding to $i \in (C_1 \cup C_2)$ from γ^* by querying its oracle for signatures σ_i on (u_i, m_i) and subtracting them from γ^*. Two features of our signature scheme guarantee that this procedure works as expected:

1. Signatures are not randomized; when $\mathcal{B}$ queries its oracle for (u_i, m_i) it will receive the *only* valid σ_i, making sure that if γ^* is an aggregate signature on $\langle \mathbb{U}, \mathbb{M} \rangle$, $\gamma^* - \sigma_i$ is an aggregate signature on $\langle \mathbb{U} - u_i, \mathbb{M} - m_i \rangle$.

2. The oracles $\mathcal{B}$ has access to are `StrongSign` oracles; so even if $\mathcal{A}$ replaces public keys for some $u_j \in \mathbb{U}$ and outputs the forgery under these public keys, $\mathcal{B}$ can still obtain valid signatures under these replaced public keys from the `StrongSign` oracle.

After computing $\gamma' = \gamma^* - \sum_{i \in (C_1 \cup C_2)} \sigma_i$, $\mathcal{B}$ is basically left with (possibly) multiple copies of the same signature[3], because $\gamma' = k\sigma^*$, where $k = |C_3|$. So, $\mathcal{B}$ can easily compute $\sigma^* = k^{-1}\gamma'$.

This proves that signature aggregation is secure if the underlying signature scheme is. □

5.5 Efficiency

Most known secure certificateless signature schemes require at least 4 pairing computations for signature verification [LCS05, ZWXF06]. Recently, 2 similar schemes requiring only 1 pairing have proposed in [CPHL07] and [DW07], but doubts have been raised about their security claims in [CD07]. If these schemes are in fact secure, the biggest atraction of our scheme would the possibility of aggregating signatures. If they are proven to be insecure, our scheme would be the most efficient available also on the single-signature case.

Our scheme becomes specially attractive in situations where Type-2 aggregation is desirable. Recall that in Type-2 aggregation, an user U wants to sign a set $\mathbb{M} = \{M_1, M_2, \ldots, M_n\}$ of messages. We achieve this for the verification-cost of 3 pairings, regardless of how many messages are aggregated. For Type-3 aggregation (the most general case of all: distinct signers, multiple messages for each), we still incur in some (processing) overhead, needing an extra 2 pairings for each distinct signer. Even though the processing overhead highly depends on the type of aggregation being performed, the scheme is very space efficient: regardless of the aggregation type, aggregates are constant-size, requiring only one elliptic curve point.

6 Conclusion

In this paper we presented the first treatment of signature aggregation in certificateless cryptography. Our scheme uses many ideas from the scheme of [Her06]. In that paper, however, the author does not deal with the possibility of replacing the public component R on the proof of security; it is not clear whether R is supposed to be authenticated or not. Many other certificateless signature schemes have been proposed lately. The first schemes requiring less than 4 pairing for signature verification have just been proposed in [CPHL07] and [DW07] but have their security proofs questioned in [CD07]. The security mediated

[3]Remember that since signatures are deterministic and all elements of C_3 are of the form (U^*, M^*), this must be true.

scheme from [YCHG07] looks very promising as the authors claim it can be easily turned into a certificateless scheme and it requires no pairing computation whatsoever.

To our knowledge, no signature aggregation scheme for certificateless cryptography has been proposed prior to this work. In this technical report, we developed a suitable security model for aggregate signatures which is a natural extension of standard models for certificateless signatures. Then we proposed a scheme suitable for aggregation and proved its security using the aforementioned model. Our scheme is efficient, requiring 3 pairings for signature verification and becomes specially attractive for Type-2 aggregation, in which regardless of the number of messages signed, only 3 pairings are needed for verification. This scheme still leaves however certain questions open, specially whether more efficient schemes (similar overhead for Type-3 aggregation to what we could get for Type-2 aggregation or less than 3 pairings for signature verification) can be developed.

References

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BGSL03] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps, 2003.

[BNN06] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. Cryptology ePrint Archive, Report 2006/285, 2006. `http://eprint.iacr.org/`.

[CD07] Rafael Castro and Ricardo Dahab. Two notes on the security of certificateless signatures. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *The 1st International Conference on Provable Security (ProvSec) 2007*, volume 4784 of *Lecture Notes in Computer Science*. Springer, 2007.

[CPHL07] Kyu Young Choi, Jong Hwan Park, Jung Yeon Hwang, and Dong Hoon Lee. Efficient certificateless signature schemes. In Katz and Yung [KY07], pages 443–458.

[Den06] Alexander W. Dent. A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211, 2006.

[DW07] Hongzhen Du and Qiaoyan Wen. Efficient and provably-secure certificateless short signature scheme from bilinear pairings. Cryptology ePrint Archive, Report 2007/250, 2007. `http://eprint.iacr.org/`.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[Her06] Javier Herranz. Deterministic identity-based signatures for partial aggregation. *Comput. J.*, 49(3):322–330, 2006.

[HWZD06] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Key replacement attack against a generic construction of certificateless signature. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2006.

[HWZD07] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Certificateless signature: a new security model and an improved generic construction. *Des. Codes Cryptography*, 42(2):109–126, 2007.

[IN83] K. Itakura and K. Nakamura. A public key cryptosystem suitable for digital multisignatures. In *NEC Research & Development*, volume 71, pages 1–8, 1983.

[KY07] Jonathan Katz and Moti Yung, editors. *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, volume 4521 of *Lecture Notes in Computer Science*. Springer, 2007.

[LCS05] X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76–83, 2005.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.

[Sco05] Michael Scott. Computing the tate pairing. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[YCHG07] Wun-She Yap, Sherman S. M. Chow, Swee-Huay Heng, and Bok-Min Goi. Security mediated certificateless signatures. In Katz and Yung [KY07], pages 459–477.

[ZWXF06] Zhenfeng Zhang, Duncan S. Wong, Jing Xu, and Dengguo Feng. Certificateless public-key signature: Security model and efficient construction. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 293–308, 2006.