Traceable Signature with Stepping Capabilities

Olivier Blazy and David Pointcheval

ENS/CNRS/INRIA

Abstract Traceable signatures schemes were introduced by Kiayias, Tsiounis and Yung in order to solve traceability issues in group signature schemes. They wanted to enable authorities to delegate some of their detection capabilities to tracing sub-authorities. Instead of opening every single signatures and then threatening privacy, tracing sub-authorities are able to know if a signature was emitted by specific users only.

In 2008, Libert and Yung proposed the first traceable signature schemes proven secure in the standard model. We design another scheme in the standard model, with two instantiations based either on the SXDH or the DLin assumptions. Our construction is far more efficient, both in term of group elements for the signature, and pairing computation for the verification. Besides the "step-in" (confirmation) feature that allows a user to prove he was indeed the signer, our construction provides the "step-out" (disavowal) procedure that allows a user to prove he was not the signer.

Since list signature schemes are closely related to this primitive, we consider them, and answer an open problem: list signature schemes are possible without random oracles.

Keywords: Traceable Signature, List Signature, Standard Model

1 Introduction

Traceable signatures were introduced by Kiayias, Tsiounis and Yung in [KTY04] as an improvement of group signatures (defined in [Cv91]). In addition to the classical properties of a group signature scheme, that allows users to sign in the name of the group, while the opener only is able to trace back the actual signer, traceable signatures allow the opener to delegate the tracing decision for a specific user without revoking the anonymity of the other users: the opener can delegate its tracing capability to sub-openers, but against specific signers without letting them trace other users. This gives two crucial advantages: on the one hand tracing agents (sub-openers) can run in parallel; on the other hand, honest users do not have to fear for their anonymity if authorities are looking for signatures produced by misbehaving users only. This is in the same vein as searchable encryption [ABC$^+$05], where a trapdoor, specific to a keyword, allows to decide whether a ciphertext contains this keyword or not, and provides no information about ciphertexts related to other keywords.

The first efficient traceable signature scheme, provably secure in the standard model, has been introduced by Libert and Yung in [LY09]. We present another approach to provide such a scheme. In addition, our construction is more efficient and provides some extra features.

Our construction can also be used to solve an open problem on list signatures. List signatures were introduced by Canard et al. in [CSST06]. They let users sign anonymously, in an irrevocable way, but grant linkability in a specific time-frame: no one can trace back the actual signer, but if a user signs two messages within a specific time-frame, the signatures will be linkable. Since then, it has been an open problem to know if there was any way to construct such a list signature scheme in the standard model.

Contribution. We present simple and efficient constructions of both traceable signatures and list signatures. They can be proven under reasonable assumptions (variations of the $q - $ SDH, and DDHI assumptions). We prove the security of both schemes in the standard model. In this paper we combine the use of the Dodis-Yampolskiy pseudo-random function [DY05], a Delerablée-Pointcheval [DP06] kind of certificate, Waters' signature [Wat05] and the Groth-Sahai [GS08] methodology.

First, we present our traceable signature scheme: we extend the initial security model, with additional features. The Delerablée-Pointcheval [DP06]-like certificate will allow delegation of tracing, since a trapdoor, not enough for signing, enables tracing decision between a signature and an alleged user. Users will also be able to confirm (step-in) or deny (step-out) being the actual signer, using their signing key only, in a convincing way, which is a new attractive property. To achieve this, we define

the notion of unique identifier, related to each signature, and specific to the user and an additional input.

Granted this technique of unique identifier, we can give a positive answer to the open problem of list signatures in the standard model: if we make the unique identifier specific to the user and the time-frame, in a deterministic way, then two signatures by the same user within the same time-frame will have the same identifier, which provides linkability. However, in this second setting the identifier does not allow to get back to the actual signer.

Organization. In the next section, we present the primitive of traceable signature and the security model, in the same vein as the BSZ model [BSZ05]. Then, we present the basic tools on which our instantiations will rely. Eventually, we describe our schemes, in the SXDH setting, with the corresponding assumptions for the security analysis that is provided. For the sake of consistency, in the appendix, we then explain the results with the (intuitive) DLin instantiations of this scheme as it requires roughly the same number of group elements and, based on the chosen elliptic curve and the way one wants to verify the signatures, one may prefer one instantiation to the other. It also allows us to compare our signature with the previous one, and show that we are at least twice as more efficient. A recent paper [BFI+10] has shown that DLin signatures can be batch verified more efficiently than the SXDH ones, which can also help our sub-openers, if they want to trace a user on several signatures.

2 Preliminaries

2.1 Definition

We use similar notations as [BSZ05], for the BSZ model for group signatures, since traceable signatures are a natural extension to the latter. We follow the original model from traceable signatures with some improvements, but with similar notations and terminology. In a traceable signature scheme, there are several users, which are all registered in a PKI. We thus assume that any user $\mathcal{U}_i$ owns a pair $(\mathsf{usk}[i], \mathsf{upk}[i])$ of secret and public keys, certified by the PKI. There are several authorities:

- the group manager, also known as *Issuer*: it issues certificates for users to grant access to the group.
- the *Opener*, (which is the same party as the group manager in [LY09], but we prefer to separate the roles as in [BSZ05], since this is a stronger model): it is able to *open* or *trace* any signature. The former means that it can learn who is the actual signer of a given signature while the latter decides, on a given signature and an alleged signer, whether the signature has really been generated by this signer or not. It is also able to delegate the latter tracing capability but for specific users only. To this aim, it *reveals* a trapdoor to a *Sub-Opener*. The latter gets the ability to trace a specific user only (decide whether the signer associated to the trapdoor is the actual signature of a signature) without learning anything about the other users.

In the initial model, users also have the capability to *Claim* a signature, *i.e.* they are able to publicly confirm they are the author of a given signature. We enhance the functionalities with a *Deny* algorithm, that allows a user to prove that he is not the actual author of a given signature. Both are combined in a Step algorithm, with Step-in and Step-out procedures to confirm and deny a signature respectively, using the signing key only.

A traceable signature scheme is defined by a sequence of (interactive) protocols, TS = (Setup, Join, Sig, Verif, Open, Reveal, Trace, Step):

In some security models, there is an additional party called "Judge" that verifies all the claims. But since all the following proofs are publicly verifiable, this player is not required here.

- Setup(1^k): It generates the global parameters, the public key pk and the private keys: the master secret key msk for the group manager, and the opening key skO for the Opener.
- Join($\mathcal{U}_i$): This is an interactive protocol between a user $\mathcal{U}_i$ (using his secret key $\mathsf{usk}[i]$) and the group manager (using his private key msk). At the end of the protocol, the user obtains a signing key $\mathsf{sk}[i]$, and the group manager adds the user to the registration list, storing some information in Reg. We note I the set of registered users.

- $\mathsf{Sig}(\mathsf{pk}, i, m, \mathsf{sk}[i])$: This is a protocol expected to be made by a registered user $i \in I$, using his own signing key $\mathsf{sk}[i]$. It produces a signature σ on the message m.
- $\mathsf{Verif}(\mathsf{pk}, m, \sigma)$: Anybody should be able to verify the validity of the signature, with respect to the public key pk. This algorithm thus outputs 1 iff the signature is valid.
- $\mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO})$: If σ is valid, the opener, using skO, outputs a user i assumed to be the author of the signature with a publicly verifiable proof Π_O of this accusation.
- $\mathsf{Reveal}(\mathsf{pk}, i, \mathsf{skO})$: This algorithm, with input skO and a target user i, outputs a tracing key $\mathsf{tk}[i]$ specific to the user i, together with a proof $\Pi_{\mathcal{E}}$ confirming this $\mathsf{tk}[i]$ is indeed a tracing key of the user i.
- $\mathsf{Trace}(\mathsf{pk}, m, \sigma, \mathsf{tk}[i])$: Using the sup-opener key $\mathsf{tk}[i]$ for user i, this algorithm outputs 1 iff σ is a valid signature produced by i, together with a proof Π_{sO} confirming the decision.
- $\mathsf{Step}(\mathsf{pk}, m, \sigma, \mathsf{sk}[i])$: Using the user' secret key $\mathsf{sk}[i]$, this algorithm outputs 1 iff σ if a valid signature produced by i, together with a proof Π_c confirming the claim.
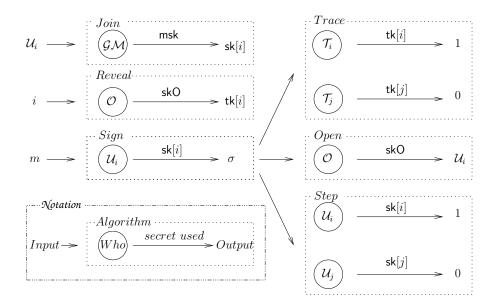

Figure 1. An Improved Traceable Signature Scheme

2.2 Security Notions

Before being secure, the scheme must be correct. We thus first explain how it works, and then we define the security model.

Correctness. The *correctness* notion guarantees that honest users should be able to generate valid signatures, those notions are direct extensions of the classic ones with an additional authority and also consistency between open, trace and step algorithms. More precisely the correctness guarantees that

- a message signed by an honest user i should
  - successfully pass the verification process;
  - open to i;
  - lead to a positive answer for the trace and step procedures under user i's related keys;
- as the traceability property of group signatures, for any valid signature σ, the opening algorithm should designate some user. And the latter should be accepted by the trace and step procedures;

In the following experiments that formalize the security notions, the adversary can run the Join protocol, either passively (receives only public values, as seen by an eavesdropper) or actively (receives all the values, as the legitimate user):

   – either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the secret keys: the index i is added to the HU (Honest Users) list;

   – or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user it will control: the index i is added to the CU (Corrupted Users) list.

Note that when the adversary is given the master key (the group manager is corrupted) then the adversary does not need access to the joinA oracle since it can simulate it by itself, to create corrupted users (that are not necessarily in CU). After a user is created, the adversary plays the role of corrupted users, and can interact with honest users, granted some oracles:

   – corrupt(i), if $i \in$ HU, provides the specific secret key of this user. The adversary can now control it during the whole simulation. Therefore i is moved from HU to CU;

   – sig(i, m), if $i \in$ HU, plays as the honest user i would do in the signature process to generate a signature on message m. Then (i, m, σ) is appended to the list $\mathcal{S}$ of generated signatures;

   – open(m, σ), if (m, σ) is valid, returns the identity i of the signer. Then (i, m, σ) is appended to the list $\mathcal{O}$ of opened signatures;

   – reveal(i), if $i \in$ HU, returns the tracing key tk[i] for the user i. Then i is appended to the list $\mathcal{R}$ of the revealed users;

   – tr(i, m, σ), if $i \in$ HU and (m, σ) is valid, returns 1 iff i is the signer who made σ on m is i. Then (i, m, σ) is appended to the list $\mathcal{T}$ of traced signatures;

   – step(i, m, σ), if $i \in$ HU, plays as the honest user i would do to step in/out of the signature σ on message m.

Note that for a corrupted user i, with the secret key sk[i], the adversary can run itself the Step and Trace procedures, and or course sign too. We thus have the following sets:

   – I, the set of registered users, and HU, CU the honest and dishonest users respectively;

   – $\mathcal{S}$, the list of generated signatures (i, m, σ), and $\mathcal{S}[m] = \{i | (i, m, \sigma) \in \mathcal{S}\}$;

   – $\mathcal{O}$, the list of opened signatures (i, m, σ), and $\mathcal{O}[m] = \{i | (i, m, \sigma) \in \mathcal{O}\}$;

   – $\mathcal{R}$, the list of revealed users i;

   – $\mathcal{T}$, the list of traced signatures (i, m, σ), and $\mathcal{T}[m] = \{i | (i, m, \sigma) \in \mathcal{T}\}$.

    A signature is identified by a user-message pair and not σ itself in those sets because we do not expect for strong unforgeability, also known as non-malleability [SPMLS02]. In our instantiations, signatures will be malleable, and even re-randomizable: it is easy for anyone to produce a new valid signature σ' from a previous one σ, but on the same message. To this end we note $\sigma \overset{(i,m)}{\equiv} \sigma'$. The subsequent relaxation on the security has already been used in [LY09] for traceable signatures.

Soundness. This is the main security notion that defines two unforgeability properties. The security games are shortened thanks to the correctness which implies that the opening, the tracing and the stepping processes are consistent:

   – Misidentification, which means that the adversary should not be able to produce a non-trivial valid signature that could not be opened to a user under its control. The adversary wins if Open either accuses an unknown user or has an invalid proof, so returning $\perp$. (see Figure 2 (a));

   – Non-Frameability, which means that the adversary should not be able to produce a non-trivial valid signature corresponding (that opens) to an honest user even if the authorities are corrupted (see Figure 2 (b));

TS is *Sound* if, for any polynomial adversary $\mathcal{A}$, both advantages $\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{Misl}}(k)$ and $\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{nf}}(k)$ are negligible. The first notion (Misidentification) guarantees traceability (the Open algorithm always succeeds on valid signatures) but also honest users cannot be framed when the group manager is honest. The second one (non-frameability) is somewhat stronger since it allows the group manager to be corrupted, but would not guarantee by itself traceability.

(a) Experiment $\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{Misl}}(k)$
  $(\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^k)$
  $(m, \sigma) \leftarrow \mathcal{A}(\mathsf{pk} : \mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}, \mathsf{sig}, \mathsf{reveal})$
  IF $\mathsf{Verif}(\mathsf{pk}, m, \sigma) = 0$, RETURN 0
  IF $\mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO}) = \bot$, RETURN 1
  IF $\exists j \notin \mathsf{CU} \cup \mathcal{S}[m]$,
      $\mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO}) = (j, \Pi)$
    RETURN 1
  ELSE RETURN 0

$$\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{Misl}}(k) = \Pr[\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{Misl}}(k) = 1]$$

(b) Experiment $\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{nf}}(k)$
  $(\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^k)$
  $(m, \sigma) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{msk}, \mathsf{skO} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig})$
  IF $\mathsf{Verif}(\mathsf{pk}, m, \sigma) = 0$ RETURN 0
  IF $\exists i \in \mathsf{HU} \setminus \mathcal{S}[m]$,
      $\mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO}) = (i, \Pi)$
    RETURN 1
  ELSE RETURN 0

$$\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{nf}}(k) = \Pr[\mathsf{Exp}_{\mathsf{nf},\mathcal{A}}^{\mathsf{nf}}(k) = 1]$$

Figure 2. Security Notions: Soundness

Anonymity. We now address the privacy concerns. For two distinct signers i_0, i_1, chosen by the adversary, the latter should not have any significant advantage in guessing if the issued signature comes from i_0 or i_1. We can consider either a quite strong anonymity notion (usually named full-anonymity) where the adversary is allowed to query the opening oracle (resp. tracing, stepping) on any signatures, excepted signatures that are equivalent to the challenge signature with respect to the signers i_0 or i_1; or the classical anonymity notion, where open, tr and of course reveal are not available to the adversary. TS is *anonymous* if, for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{anon}}(k)$ is negligible (see Figure 3).

Experiment $\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{anon}_b}(k)$
  $(\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^k)$
  $(m, i_0, i_1) \leftarrow \mathcal{A}(\texttt{FIND}, \mathsf{pk}, \mathsf{msk} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig}, \mathsf{open}^*, \mathsf{tr}^*, \mathsf{reveal}^*, \mathsf{step}^*)$
  $\sigma \leftarrow \mathsf{Sig}(\mathsf{pk}, i, m, \mathsf{sk}[i_b])$
  $b' \leftarrow \mathcal{A}(\texttt{GUESS}, \sigma : \mathsf{joinA}, \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig}, \mathsf{open}^*, \mathsf{tr}^*, \mathsf{reveal}^*, \mathsf{step}^*)$
  IF $i_0 \notin \mathsf{HU} \setminus (\mathcal{R} \cup \mathcal{T}[m] \cup \mathcal{O}[m] \cup \mathcal{S}[m])$ OR $i_1 \notin \mathsf{HU} \setminus (\mathcal{R} \cup \mathcal{T}[m] \cup \mathcal{O}[m] \cup \mathcal{S}[m])$ RETURN 0
  ELSE RETURN b'

$$\mathsf{Adv}_{\mathsf{TS},\mathcal{A}}^{\mathsf{anon}}(k) = \Pr[\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{anon}-1}(k) = 1] - \Pr[\mathsf{Exp}_{\mathsf{TS},\mathcal{A}}^{\mathsf{anon}-0}(k) = 1]$$

Figure 3. Security Notions: Anonymity ($\mathsf{open}^* = \mathsf{tr}^* = \mathsf{reveal}^* = \mathsf{step}^* = \emptyset$) and Full-Anonymity ($\mathsf{open}^* = \mathsf{open}, \mathsf{tr}^* = \mathsf{tr}, \mathsf{reveal}^* = \mathsf{reveal}, \mathsf{step}^* = \mathsf{step}$)

2.3 Cryptographic Tools

Pseudo Random Function. We will use a variation of the Dodis-Yampolskiy VRF [DY05], introduced in [CHL05]. It basically states that for a polynomial number of scalars z_i, and a pair $(g, g^x) \in \mathbb{G}_1^2$, the values $g^{1/(x+z_i)}$ look random and independent. We will use this property to build our identifiers. In the proof of anonymity, the simulator will be able to choose the z_i prior to any interaction with the adversary so we rely in the framework where the VRF is secure under the $q - \mathsf{DDHI}$ assumption.

Certification. Since our new primitive is quite related to group signatures, we now introduce the BBS-like certification [BBS04] proposed by Delerablée and Pointcheval [DP06], in order to achieve non-frameability. The system needs a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, and a generator k of $\mathbb{G}_1$. During the setup, the group manager chooses two additional independent generators g_1 and g_2, of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and a master secret key $\mathsf{msk} = \gamma \in \mathbb{Z}_p$. It sets $\mathsf{pk} = (g_1, k, g_2, \Omega = g_2^\gamma)$. During the join procedure, the authority chooses an x_i for the user, and they choose together y_i (so that it is unknown to the group manager, but known to the user: his secret key). After an interactive process, the user gets his certificate, $(g_1^{x_i}, g_2^{x_i}, y_i, A_i = (kg_1^{y_i})^{1/(\gamma+x_i)})$, where the verification consists in

$$e(A_i, \Omega g_2^{x_i}) = e(k, g_2) \times e(g_1, g_2)^{y_i} \qquad e(g_1^{x_i}, g_2) = e(g_1, g_2^{x_i}).$$

Signature. We use a slight variant of Waters signature [Wat05,BFPV11], in the SXDH setting: Given three generators $(g_1, k_1, g_2) \in \mathbb{G}_1^2 \times \mathbb{G}_2$, a public key $\mathsf{pk} = (g_1^z, g_2^z)$, the secret key z, to sign a message

m, a user simply needs to pick a random scalar r and compute $(k_1^z \cdot \mathcal{F}(m)^r, g_2^r)$. Here, $\mathcal{F}$ is the Waters function defined as $\mathcal{F}(m) = u_0 \Pi u_i^{m_i}$, where (u_i) are independent generators of $\mathbb{G}_1$. The verification simply consists in checking if $e(\sigma_1, g_2) = e(k_1, \mathsf{pk}_2) \cdot e(\mathcal{F}(m), \sigma_2)$.

Groth-Sahai Commitments. We will follow the Groth-Sahai methodology for SXDH-based commitment in the SXDH setting. The commitment key consists of $\mathbf{u} \in \mathbb{G}_1^{2 \times 2}$ and $\mathbf{v} \in \mathbb{G}_2^{2 \times 2}$. There exist two initializations of the parameters either in the perfectly binding setting, or in the perfectly hiding one. Those initializations are indistinguishable under the SXDH assumption which will be used in the simulation. We note $\mathcal{C}(X)$ a commitment of a group element X. An element is always committed in the group ($\mathbb{G}_1$ or $\mathbb{G}_2$) it belongs to. If one knows the commitment key in the perfectly binding setting, one can extract the value of X, else it is perfectly hidden. We note $\mathcal{C}^{(1)}(x)$ a commitment of a scalar x embedded in $\mathbb{G}_1$ as g_1^x. If one knows the commitment key in the perfectly binding setting, on can extract the value of g_1^x else x is perfectly hidden. The same things can be done in $\mathbb{G}_2$, if we want to commit a scalar, embedding it in $\mathbb{G}_2$.

Proofs. Under the SXDH assumption, the two initializations of the commitment key (perfectly binding or perfectly hiding) are indistinguishable. The former provides perfectly sound proofs, whereas the latter provides perfectly witness hiding proofs. A Groth-Sahai proof, is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2 \times 2} \times \mathbb{G}_2^{2 \times 2}$. These elements are constructed to help verifying pairing relations on committed values. Being able to produce a valid pair implies knowing plaintexts verifying the appropriate relation.

We will use three kinds of relations:

- pairing products equation which require 4 extra elements in each group;
- multi-scalar multiplication which require 2 elements in one group and 4 in the other;
- quadratic equations which only require 2 elements in each group.

If some of these equations are linear, some of the extra group elements are not needed, which leads to further optimizations.

3 Traceable Signature

3.1 Our Scheme

We first describe our construction, without anonymity. The latter security property will be achieved granted commitments and proofs of validity, that will be easy and efficient since everything fits withing the Groth-Sahai methodology.

$\mathsf{Setup}(1^k)$*:* The system generates a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. One chooses independent generators (g_1, k_1, g_2) of $\mathbb{G}_1^2 \times \mathbb{G}_2$. One also chooses independent generators in $\mathbb{G}_1$ to be able to define the Waters function $\mathcal{F}$, as described in the previous section. We furthermore define commitment parameters $(\mathbf{u}, \mathbf{v})$ using the perfectly binding procedure. These are the global parameters.

The group manager chooses a scalar $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p$ for the master key $\mathsf{msk} = \gamma$, and computes $\Omega = g_2^\gamma$. The opener produces a (perfectly binding) setup of Groth-Sahai commitment keys: he knows (α_1, α_2) for a binding setup with $\mathbf{u}' = (\mathbf{u}_1', \mathbf{u}_1'^{\alpha_1}), \mathbf{v}' = (\mathbf{v}_1', \mathbf{v}_1'^{\alpha_2})$. We actually note $\mathcal{E}$ this instantiation as it will be used as an encryption that fits into Groth-Sahai methodology. Intuitively group elements committed/encrypted under this instantiation can be decrypted by the opener, which will be required for A_i and $\mathsf{tk}[i]$. We will note $\mathcal{C}$ when we commit with the system commitment base $(\mathbf{u}, \mathbf{v})$ defined in the global parameters, (which will be alternatively binding or hiding in the security proof, but binding only in the real-life). In order to use Groth-Sahai methodology without any extra concern, we may pay attention to always have in a same equation, elements in the same group committed within the same base. (*i.e.* , an equation where all elements in $\mathbb{G}_1$ are committed thanks to $\mathbf{u}'$, and all in $\mathbb{G}_2$ thanks to $\mathbf{v}$ perfectly fits into the methodology). The public key (with the global parameters) is then $\mathsf{pk} = (g_1, g_2, k_1, \Omega, \mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}')$.

$\mathsf{Join}(\mathcal{U}_i)$: In order to join the system, a user $\mathcal{U}_i$, with a pair of keys $(\mathsf{usk}[i], \mathsf{upk}[i])$ in the PKI, interacts with the group manager (similarly to [DP06]):

- $\mathcal{U}_i$ chooses a random $y_i' \in \mathbb{Z}_p$, computes and sends $Y_i' = g_1^{y_i'}$, an extractable commitment of y_i' with a proof of consistency. Actually the trapdoor of the commitment will not be known to anybody, except to our simulator in the security proof to be able to extract y_i'.
- The group manager chooses a new $x_i \in \mathbb{Z}_p$ and a random $y_i'' \xleftarrow{R} \mathbb{Z}_p$, computes and sends y_i'', $A_i = (k_1 Y_i' Y_i'')^{1/(\gamma + x_i)}$ and $X_{i,2} = g_2^{x_i}$ where $Y_i'' = g_1^{y_i''}$;
- $\mathcal{U}_i$ checks whether $e(A_i, \Omega g_2^{x_i}) = e(k_1, g_2) \times e(g_1, g_2)^{y_i' + y_i''}$. He can then compute $y_i = y_i' + y_i''$. And so we have $e(A_i, \Omega X_{i,1}) = e(k_1, g_2) \times e(g_1, g_2)^{y_i}$. He produces a commitment of $\mathsf{tk}[i] = g_2^{y_i}$: $e_i = \mathcal{E}(g_2^{y_i})$ and a proof of consistency $\pi_J[i]$, and signs $(g_1^{y_i}, A_i, X_{i,2}, e_i, \pi_J[i])$ under $\mathsf{usk}[i]$ into s_i.
- The group manager verifies s_i under $\mathsf{upk}[i]$ and the given proof, and appends the tuple $(i, \mathsf{upk}[i], X_i, g_1^{y_i}, A_i, e_i, \pi_J[i], s_i)$ to Reg. He can then send the last part of the certificate $X_{i,1} = g_1^{x_i}$.
- $\mathcal{U}_i$ thus checks, if he did receive $g_2^{x_i}$ before (*i.e.* if $e(X_{i,1}, g_2) = e(g_1, X_{i,2})$), and then owns a valid certificate (A_i, X_i, y_i), where $\mathsf{sk}[i] = y_i$ is known to him only. The secrecy of y_i will be enough for the overall security. Note that if $X_{i,1}$ is invalid, one can ask for it again. In any case, the group manager cannot frame the user, but just do a denial of service attack, which is unavoidable. We expect the Reg array to be constantly certified, *i.e.* , we expect the Group Manager to sign every rows. (This will only be required in our step in/out process)

At this stage, $\mathsf{Reg} = \{(i, \mathsf{upk}[i], X_i, g_1^{y_i}, A_i, e_i, \pi_J[i], s_i)\}$, and $\mathsf{sk}[i] = y_i$.

$\mathsf{Sig}(\mathsf{pk}, m, \mathsf{sk}[i])$: When a user i wants to sign a message m, he computes the signature of m under his private key $\mathsf{sk}[i]$. First, he creates an ephemeral ID, $\mathsf{ID}(y_i, z) = g_1^{1/(z+y_i)}$, and publishes σ :

$$(\sigma_0 = \mathsf{ID}(y_i, z), \sigma_1 = X_i, \sigma_2 = y_i, \sigma_3 = A_i, \sigma_4 = (g_1^z, g_2^z), \sigma_5 = k_1^z \mathcal{F}(m)^s, \sigma_6 = g_2^s)$$

that satisfy the relations:

$$e(\sigma_0, \sigma_{4,2} g_2^{\sigma_2}) = e(g_1, g_2) \qquad\qquad e(\sigma_{1,1}, g_2) = e(g_1, \sigma_{1,2})$$
$$e(\sigma_3, \Omega\sigma_{1,2}) = e(k_1, g_2) \times e(g_1, g_2^{\sigma_2}) \qquad\qquad e(g_1^{\sigma_2}, g_2) = e(g_1, g_2^{\sigma_2})$$
$$e(\sigma_5, g_2) = e(k_1, \sigma_{4,2}) \times e(\mathcal{F}(m), \sigma_6) \qquad\qquad e(\sigma_{4,1}, g_2) = e(g_1, \sigma_{4,2})$$

Basically, σ_0 is a certificate of the public key σ_4, and (σ_5, σ_6) is a Waters' signature of m under this key. As explained above, for the sake of clarity, we started with a non-anonymous scheme. To achieve anonymity, some of these tuples are thereafter committed/encrypted, but only those that can be linked to a user. As shown in the equations σ_2 is a scalar, but needs to be committed in both groups, which will be perfect for the following proofs as it will be enough to extract both g_1^y and g_2^y (see the computational assumption in the next section):

$$\sigma = (\sigma_0, \mathcal{C}(\sigma_1), \mathcal{C}(\sigma_2), \mathcal{E}(\sigma_3), \sigma_4, \sigma_5, \sigma_6)$$

We add the corresponding Groth-Sahai proofs to prove the validity of the previous pairing equations. The second (X_i is well-formed) and third (A_i is well-formed) ones are pairing products, so need 4 elements in each group, the first one (ID is well formed) is a Linear Pairing Product, so needs only 2 extra elements in $\mathbb{G}_1$, the fourth one (the same y_i is committed in two bases) is a quadratic equation and so can be proven with 2 elements in each group. The two last ones do not use any committed data and so can be directly checked. Overall we will need 21 group elements in $\mathbb{G}_1$ and 16 in $\mathbb{G}_2$, which is far under the 83 required in the Libert-Yung construction. Especially if we consider elements in $\mathbb{G}_1$ to be half the size of those in $\mathbb{G}_2$ (as often done in standard implementations).

$\mathsf{Verif}(\mathsf{pk}, m, \sigma)$: One simply has to verify if all the pairing equations hold.

$\mathsf{Open}(\mathsf{pk}, m, \sigma, \alpha)$: The Opener just opens the commitment of A_i in σ_3, and then outputs a Groth-Sahai proof of knowledge of an α such that $e(\sigma_{3,1}, g_2) = e(A_i, g_2).e(\sigma_{3,2}, g_2^\alpha)$. He checks s_i, and depending on its consistency blames the user U_i or the Group Manager or $\bot$. This is a linear multi-scalar multiplication in $\mathbb{G}_1$ and so the proof Π is composed of only 1 group element in $\mathbb{G}_1$ and is publicly verifiable.

Reveal(pk, i, α): The Opener verifies $\pi_J[i]$, and s_i in Reg and uses α to decrypt e_i and extracts the tracing key: $\mathsf{tk}[i] = g_2^{y_i}$. He then send it to the sub-opener together with a publicly verifiable proof showing that $\mathsf{tk}[i]$ is a valid decryption of e_i. (Again a linear MS but in $\mathbb{G}_2$ this time).

Trace$(\mathsf{pk}, m, \sigma, \mathsf{tk}[i])$: The Sub-Opener picks $\delta \xleftarrow{R} \mathbb{Z}_p$ and outputs a blinded tuple $(c_1 = \mathsf{tk}[i]^\delta, c_2 = \sigma_{4,2}^\delta, c_3 = g_2^\delta)$ and the target user i. Anyone can then check the validity of the tuple that should satisfy:

$$e(g_1^{y_i}, c_3) = e(g_1, c_1) \quad \text{and} \quad e(\sigma_{4,1}, c_3) = e(g_1, c_2)$$

and then know the result of the trace process from the test $e(\sigma_0, c_2 c_1) = e(g_1, c_3)$. We recall that $g_1^{y_i}$ is included in Reg$[i]$ and is thus considered public.

Step$(\mathsf{pk}, m, \sigma, \mathsf{sk}[i])$: To step in or out, a user picks a random δ, and publishes a similar blinded tuple $(c_1 = g_2^{\delta y_i}, c_2 = \sigma_{4,2}^\delta, c_3 = g_2^\delta)$ and i. Anyone can then check the validity of this tuple as above:

$$e(g_1^{y_i}, c_3) = e(g_1, c_1) \quad \text{and} \quad e(\sigma_{4,1}, c_3) = e(g_1, c_2)$$

and then if the step is in or out with: $e(\sigma_0, c_2 c_1) \stackrel{?}{=} e(g_1, c_3)$.

Another way to step in or out of a given signature, less efficient but which induces the knowledge of y_i: a user just does the same thing as a sub-opener, together with either an extra signature involving his private key or a bit-per-bit proof of knowledge of y_i. This adds an extra-property outside the scope of our model, which proves that the step in/out has really been initiated by the user itself (and not a tracing authority).

3.2 Security

Computational Assumptions. Our protocol will work with a pairing-friendly elliptic curve, of prime order:

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of finite prime order p, and g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$;
- e is a map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, that is bilinear and non-degenerated, such that $e(g_1, g_2)$ is generator of $\mathbb{G}_T$.

Definition 1 (Advanced Computational Diffie-Hellman [BFPV11]).
Let us be given two cyclic groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order p with (g_1, g_2) as respective generators and e an admissible bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

Definition 2 (Symmetric external Diffie-Hellman [BBS04]). Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. The SXDH assumption states that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 3 (q-Decisional Diffie-Hellman Inverse in $\mathbb{G}_1$). Let $\mathbb{G}_1$ be a cyclic group of order p generated by g_1. The q-DDHI problem consists, given $(g_1, g_1^\gamma, ..., g_1^{\gamma^q}, D)$, in deciding whether $D = g_1^{1/\gamma}$ or not.

Definition 4 (q-Hybrid Hidden Strong Diffie-Hellman in $\mathbb{G}_1, \mathbb{G}_2$). Let $\mathbb{G}_1, \mathbb{G}_2$ be multiplicative cyclic groups of order p generated by g_1, g_2 respectively. The q-HHSDH problem consists, given $(g_1, k, g_2, g_2^\gamma)$ and several partly hidden tuples $(g_1^{x_i}, g_2^{x_i}, y_i, (k g_1^{y_i})^{1/(\gamma+x_i)})_{i \in [1,q]}$, in computing $(g_1^x, g_2^x, g_1^y, g_2^y, (k g_1^y)^{1/\gamma+x})$ for a new pair (x, y).

About that last assumption: intuitively, under KEA, it can be reduced to a standard $q - \mathsf{SDH}$ (Under KEA, the reduction to $q - \mathsf{SDH}$ is similar to the one in [DP06]). It follows the idea of the BB-SDH introduced in [BCC+08]. However, in our construction neither the scalar given is the one involved directly in the SDH part, nor we give a second group element raised to the power γ. Therefore this new assumption seems to remain reasonable.

Correctness. Correctness of our scheme is guaranteed by the perfect soundness of the Groth-Sahai proofs in the signature and in the output of procedures Open, Trace and Step.

Anonymity. We now study the anonymity property, but do not address the full-anonymity.

Theorem 5. *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme, then there exists an adversary $\mathcal{B}$ that can break the $\ell - \mathsf{DDHI}$ problem in $\mathbb{G}_1$ or the SXDH assumption, where ℓ is the maximal number of signing queries for a user.*

Proof. Let us assume that an adversary is able to break the anonymity property of our scheme. It means than in the anonymity game, he has a non-negligible advantage $\epsilon > 0$ to distinguish $G^{(0)}$ where $b = 0$ from $G^{(1)}$ where $b = 1$. We start our sequence of games from $G^{(0)}$, denoted G_0. We first replace this game by G_1, where we make some simulations that are perfectly hiding: The simulator $\mathcal{B}$ is given a challenge $A = (g, g^y, \ldots, g^{y^\ell}) \in \mathbb{G}_1^{\ell+1}$ and $D = g^{1/y}$, for an unknown y. $\mathcal{B}$ first chooses different random values $z^*, z_1, \ldots, z_{\ell-1} \in \mathbb{Z}_p^\ell$, which completely define the polynomial $P = \prod_{i=1}^{\ell-1}(X + z_i)$, of degree $\ell - 1$. Granted the above challenge, $\mathcal{B}$ can compute $g_1 = g^{P(y)}$, and the rest of the public key is computed as in a normal scheme. In particular, one knows the master secret key γ.

The future challenge user i_0 will virtually have $y_{i_0} = \mathsf{sk}[i_0] = y - z^*$. $\mathcal{B}$ can compute $g^{y_{i_0}} = g_1^y/g_1^{z^*}$ (from the input A without using y, even if we know it here). The public certificate (in the Reg list) for the challenge user is $(g_1^{x_{i_0}}, g_2^{x_{i_0}}, g_1^y/g_1^{z^*}, (k_1 g_1^y/g_1^{z^*})^{1/(\gamma+x)})$, plus some proofs and signatures. $\mathcal{B}$ is also given Groth-Sahai commitment keys and encryption keys (decryption keys are unknown, hence the classical notion of anonymity and not full-anonymity). This setup is indistinguishable from the real one since all the keys are generated as in the real game. Because of the knowledge of the master secret key, $\mathcal{B}$ simulator easily answers any join queries, both active and passive. But has to guess i_0, which is correct with probability $1/n$, where n is the total number of passive join queries. It can also answer any corruption, that should not happen for the challenge user, even if we know y in this game.

As he is able to know all $x_i's$ and all $y_i's$ for registered users (except the challenge one), he will be able to answer signing queries as users would do. For the challenge user, on the j-th signing query, he computes $\sigma_0 = g_1^{1/(\mathsf{sk}[i]+z_j)} = g^{\prod_{i \neq j}(y+z_i)}$, that can be done from the challenge input A, the rest is done as in the real game using y and z_j as ephemeral random. For the challenge signing query, he does the same has above with the ephemeral value z^*, and the expected ID is $g_1^{1/(\mathsf{sk}[i]+z^*)} = g^{P(y)/y} = g^{Q(y)}g^{\prod(z_i)/y}$, where $Q = (\prod_{i=1}^{\ell}(X + z_i) - \prod(z_i))/X$ is a polynomial of degree $\ell - 1$ and thus $g^{Q(y)}$ can be computed from the instance. He thus outputs $\sigma_0 = g^{Q(y)} \cdot D^{\prod(z_i)}$. Since $D = g^{1/y}$, the signature is similar to the above one, and so is indistinguishable from a real signature.

We then modify the game, into G_2, where we initialize Groth-Sahai commitment keys in a perfectly hiding setting with the trapdoor, to allow the simulator to cheat in the proofs. Then all the proofs are simulated. This game is indistinguishable from the previous one under the SXDH.

In G_3, for the challenge user signing queries, we use random commitments and ciphertext for $\sigma_1, \sigma_2, \sigma_3$. The commitments were already random, because of the perfectly hiding setting, but a random ciphertext is indistinguishable from the real one under the semantic security of the encryption scheme, the SXDH assumption.

In G_4, we do not know anymore y, that we did not use anymore anyway, and thus this game is perfectly indistinguishable from the previous one. In G_5, D is a random value, which is indistinguishable from the real one under the $\ell - \mathsf{DDHI}$ assumption as we only have a polynomial number of z_i in input like in the Dodis-Yampolskiy PRF: the challenge signature does not depend anymore on the challenge user.

To complete the proof, we should make the same sequence again, starting from G_0' that is $G^{(1)}$, up to G_5', that is perfectly indistinguishable from G_5, hence the computational indistinguishability between $G_0' = G^{(1)}$ and $G_0 = G^{(0)}$. $\qquad\square$

Soundness. Within the soundness analysis, we prove traceability (misidentification) and non-frameability.

Theorem 6. *If there exists an adversary $\mathcal{A}$ against the soundness of the scheme, then we can build an adversary $\mathcal{B}$ that can either break the $Q - \mathsf{HHSDH}$, the $Q' - \mathsf{HSDH}$ or the CDH^+ computational problems, or the 1-DDHI or the SXDH decisional problems, where Q is maximal number of users, and Q' is the maximal number of signing queries for a user.*

Note that the 1-DDHI is equivalent to the Decisional Square Diffie-Hellman, since we have a sequence $(g_1, g_1^\gamma, g_1^\delta)$ where one has to decide whether $\delta = 1/\gamma$, which can be written $(G = g_1^\gamma, G^\alpha = g_1, G^\beta = g_1^\delta)$, where $\alpha = 1/\gamma$, and $\beta = \delta/\gamma$, and one has to decide whether $\delta = 1/\gamma$, and thus whether $\beta = \alpha^2$.

Proof. Non-frameability and misidentification are very closely related, we will treat both simultaneously, there are three ways to cheat the soundness of our scheme : either by creating a new certificate (G_1) which induces a misidentification attack, or by using an existing certificate but on a new message ($G_{2a,2b}$) which breaks the non-frameability.

We study the security of the unencrypted version of our scheme (because of the perfect soundness of the Groth-Sahai proofs in the perfectly binding setting, and extractability of the commitments). We will construct three different games, in the first one, we assume the adversary is able to forge a signature by generating a new certificate (a new tuple $(\sigma_1, \sigma_2, \sigma_3)$) in ($G_1$), in the second one ($G_{2a}$) the adversary is able to forge a new σ_0 and so break the tracing or step procedure, in the last game (G_{2b}) the adversary forges a new Waters signature (a new tuple $(\sigma_4, \sigma_5, \sigma_6)$).

(G_1): Let us be given a $Q-\mathsf{HHSDH}$ challenge (g_1, k, g_2, Ω), $(g_1^{x_i}, g_2^{x_i}, y_i, A_i)_{i \in [\![1,Q]\!]}$. We build an adversary $\mathcal{B}$ able to solve this challenge, from $\mathcal{A}$ that breaks the soundness of our scheme by generating a new tuple $(\sigma_1, \sigma_2, \sigma_3)$. $\mathcal{B}$ generates the commitment keys and the encryption keys, so that he knows the trapdoor, and publishes the group public key $(g_1, g_2, k, \Omega, \mathcal{E})$. To answer the i-th join queries, if this is an active join, $\mathcal{B}$ extracts y_i' and adapts his y_i'' so that $y_i' + y_i'' = y_i$, if it is a passive join, $\mathcal{B}$ directly chooses y_i. As he knows the decryption key, he can give the opening key skO to the adversary.

After at most Q join queries, $\mathcal{A}$ is able to output a new signature with a new certificate tuple with non-negligible probability. As mB knows the trapdoor of the commitment scheme, he can obtain $(g_1^x, g_2^x, g_1^y, g_2^y, A = (kg_1^y)^{1/(\gamma+x)})$ and so he is able to answer the challenge $Q - \mathsf{HHSDH}$ instance.

(G_{2a}): Let us be given a $Q - \mathsf{HSDH}$ challenge (g_1, g_2, g_2^y) and $(g_1^{t_i}, g_2^{t_i}, ID(y, t_i) = g_1^{1/(y+t_i)})_{i \in [\![1,Q]\!]}$. We build an adversary $\mathcal{B}$ answering this challenge, from an adversary $\mathcal{A}$ breaking the soundness of our scheme by forging a new ID.

$\mathcal{B}$ generates a new γ, skO, he then gives $\mathsf{msk} = \gamma, \mathsf{skO}$ to $\mathcal{A}$, together with the public key $(g, g_1, g_2, \Omega = g_2^\gamma, \mathcal{E})$. $\mathcal{B}$ can answer any joinP queries as he knows msk, the user on which we expect the attack (the challenge user) will have a certificate corresponding to one with y as a secret key. (Specifically $\mathsf{tk}[i] = g_2^y$). $\mathcal{A}$ can corrupt any user, if he tries to corrupt the challenge user, the simulation fails. As all uncorrupted user looks the same, with non-negligible probably the simulation continues. Thanks to the challenge tuple, $\mathcal{B}$ can answer to at most Q signing queries for challenge user (each time using a new ID).

After at most Q signing queries, $\mathcal{A}$ succeeds in breaking the non-frameability with non-negligible probability by generating a new ID, on an uncorrupted user. As uncorrupted users are indistinguishable, with non negligible probability this user is the challenge one, and so $\mathcal{B}$ is able to produce a new tuple $(g_1^t, g_2^t, g_1^{1/(t+y)})$, which breaks the $Q - \mathsf{HSDH}$ assumption.

(G_{2b}): Let us be given an asymmetric Waters public key ($\mathsf{pk} = (g_1^t, g_2^t)$ for the global parameters $(g_1, g_2, k_1, \mathcal{F})$. We build an algorithm $\mathcal{B}$ that break this signature, and thus the CDH^+ problem, from an adversary $\mathcal{A}$ breaking the non-frameability property of our scheme by reusing an existing ID with the corresponding certificate, but on a new message.

In the first game, $\mathcal{B}$ knows the discrete logarithm value t, generates a new γ, skO, he then gives $\mathsf{msk} = \gamma, \mathsf{skO}$ to $\mathcal{A}$, together with the public key $(g_1, g_2, k_1, \Omega = g_2^\gamma, \mathcal{E})$. $\mathcal{B}$ can answer any joinP queries as he knows msk, and extract the secret keys from the extraction keys of the commitment scheme, one of those uncorrupted user is expected to be our challenge user, with secret key y, the one $\mathcal{A}$ has to frame.

$\mathcal{B}$ can answer any signing queries. On one of them for our challenge user, say on m, he will use the above t as ephemeral Waters public key (for the z), and thus computes a $\sigma_0 = ID(y, t)$ with the corresponding Groth-Sahai proof. This way $\mathcal{A}$ now possesses a valid signature on m, with $\sigma_4 = (g_1^t, g_2^t), \sigma_5 = k_1^t \mathcal{F}(m)^s, \sigma_6 = g_2^s$. With non-negligible probably $\mathcal{A}$ breaks the non-frameability of our scheme, by hypothesis $\mathcal{A}$ does it by reusing an existing $\sigma_0, \ldots, \sigma_4$, as uncorrupted users are indistinguishable, $\mathcal{A}$ frames our challenge user with non-negligible probability, and as he makes a finite number of signing queries, he will use with non-negligible probability $\sigma_4 = (g_1^t, g_2^t)$.

Therefore, with non-negligible probability $\mathcal{A}$ outputs a new valid signature on m' with $\sigma_4 = (g_1^t, g_2^t)$, this means we have $(\sigma_4, \sigma_5, \sigma_6)$ such that

$$e(\sigma_{4,1}, g_2) = e(g_1, \sigma_{4,2}) \quad e(\sigma_5, g_2) = e(k_1, \sigma_{4,2}) \cdot e(\mathcal{F}(m'), \sigma_6),$$

and so $\mathcal{B}$ can outputs a valid forgery on the Waters challenge for the public key (g_1^t, g_2^t). But in this game, we know t.

In a second game, we replace the Groth-Sahai setup into the hiding one, so that the proofs can be simulated, and namely without using t when proving the validity of σ_0. This is indistinguishable from the previous game under the SXDH assumption. In a third game, we replace σ_0 by a random value, still simulating the proofs. As explained in the anonymous proof, a random ID is indistinguishable from the real one under the DDHI problem. Furthermore, here there is only one element, hence the 1-DDHI assumption. In this last game, one does not need to know t anymore, and thus the signature forgery reduces to breaking the asymmetric CDH$^+$.

Now let $\mathcal{A}$ be an adversary against the soundness of our scheme with an advantage ϵ. If with probability greater than $\epsilon/3$, $\mathcal{A}$ breaks the misidentification property of the scheme, then we can run the game G_1, else if with probability greater than $\epsilon/3$, $\mathcal{A}$ breaks the non-frameability property with a new ID, then we can run the game G_{2a}, else $\mathcal{A}$ breaks the non-frameability property with a new Waters component and so we run the game G_{2b}. So if there exists an adversary against the soundness of our scheme, we can break with non-negligible probability one of the previous problems. $\qquad \square$

In the DLin Setting. The description and the proofs in the DLin setting are deferred to the Appendix A. But let us compare the number of elements required in the signature:

	SXDH	DLin
Uncommitted elements	(3,2)	5
Committed elements	(6,4)	15
ID proof	(2,0)	3 (LPP)
X_i proof	(4,4)	3
A_i proof	(4,4)	9
Y_i proof	(2,2)	3 (LPP)

So we end up with 35 group elements in $\mathbb{G}$ instead of the 21 in $\mathbb{G}_1$ and 16 in $\mathbb{G}_2$ with SXDH in the asymmetric setting. As explained before the result with SXDH is equivalent to approximately 29 elements in DLin only, because of the different lengths of the group elements.

4 List Signature

4.1 Definition

We will once again use similar notations as [BSZ05]. In a list signature scheme, there are several users, which are all registered in a PKI. We thus assume that each user $\mathcal{U}_i$ owns a pair $(\mathsf{usk}[i], \mathsf{upk}[i])$ certified by the PKI. In standard implementation there is only one authority: The group manager, also known as *Issuer*: it issues certificates for users to grant access to the group. (Technically, we can still use an Opener, it will work exactly as before, however for the sake of clarity we will skip this part to lighten our construction.)

A List Signature scheme is thus defined by a sequence of (interactive) publicly verifiable protocols, $\mathsf{LS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sig}, \mathsf{Verif}, \mathsf{Match})$:

- $\mathsf{Setup}(1^k)$, where k is the security parameter. This algorithm generates the global parameters of the system, the public key pk and the private keys: the master secret key msk given to the group manager.
- $\mathsf{Join}(\mathcal{U}_i)$: this is an interactive protocol between a user $\mathcal{U}_i$ (using his secret key $\mathsf{usk}[i]$) and the group manager (using his private key msk). At the end of the protocol, the user obtains a signing key $\mathsf{sk}[i]$, and the group manager adds the user to the registration list, storing some information in $\mathsf{Reg}[i]$.

(a) Experiment $\mathsf{Exp}^{uf}_{LS,\mathcal{A}}(k)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^k)$
$(t, (m_i, \sigma_i)_{i \in [\![1,n]\!]}) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{msk} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig})$
IF $\exists i \mathsf{Verif}(\mathsf{pk}, m_t, t, \sigma_t) = 0$, RETURN 0
IF $\exists i \neq j, \mathsf{Match}(\mathsf{pk}, m_i, t, m_j, t, \sigma_i, \sigma_j) = 1$
  RETURN 0
IF $n > \#\mathsf{CU} + \mathcal{S}(t)$, RETURN 1
ELSE RETURN 0
$$\mathsf{Adv}^{uf}_{LS,\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{uf}_{LS,\mathcal{A}}(k) = 1]$$

(b) Experiment $\mathsf{Exp}^{anon-b}_{LS,\mathcal{A}}(k)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^k)$
$(m, t, i_0, i_1) \leftarrow \mathcal{A}(\mathtt{FIND}, \mathsf{pk} : \mathsf{joinA}, \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig})$
$\sigma \leftarrow \mathsf{Sig}(\mathsf{pk}, i_b, m, t, \{sk[i_b]\})$
$b' \leftarrow \mathcal{A}(\mathtt{GUESS}, \sigma : \mathsf{joinA}, \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sig})$
IF $i_0 \in \mathsf{CU}$ OR $i_1 \in \mathsf{CU}$ RETURN $\perp$
IF $(i_0, *) \in \mathcal{S}(t)$ OR $(i_1, *) \in \mathcal{S}(t)$ RETURN $\perp$
ELSE RETURN b'

Figure 4. Security Notions for List Signatures

- $\mathsf{Sig}(\mathsf{pk}, i, m, t, sk[i]\})$: this is a (possibly interactive) protocol expected to be made by a registered user i, using his own key $\mathsf{sk}[i]$. It produces a signature σ on the message m at the timeframe t .
- $\mathsf{Verif}(\mathsf{pk}, m, t, \sigma)$: anybody should be able to verify the validity of the signature, with respect to the public key pk. This algorithm thus outputs 1 if the signature is valid, and 0 otherwise.
- $\mathsf{Match}(\mathsf{pk}, m_1, t_1, m_2, t_2, \sigma_1, \sigma_2)$: This outputs outputs 1 iff $t_1 = t_2$ and σ_1 and σ_2 were produced by the same user.

4.2 Security Notions

Before being secure, the scheme must be correct. The *correctness* notion guarantees that honest users should be able to generate valid signatures.

In the following experiments that formalize the security notions, the adversary can run the Join protocol,

- either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the secret keys: the index i is added to the HU (Honest Users) list;
- or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user it will control: the index i is added to the CU (Corrupted Users) list.

After a user is created, the adversary can interact with honest users, granted some oracles:

- $\mathsf{corrupt}(i)$, if $i \in \mathsf{HU}$, provides the specific secret key of this user. The adversary can now control it during the whole simulation. Therefore i is added to CU;
- $\mathsf{sig}(\mathsf{pk}, i, m, t)$, if $i \in \mathsf{HU}$, plays as the honest user i would do in the signature process to generate a signature on message m during the time-frame t. Then (i, m, t) is appended to the list $\mathcal{S}$ (generated signatures).

Soundness. This is the main security notion, see Figure 4 (a): An adversary can produce at most one valid signature per time-frame per corrupted player. LS is *Sound* if for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{uf}_{LS,\mathcal{A}}(k)$ is negligible.

Anonymity We now address the privacy concerns, see Figure 4 (b). Given two honest users i_0 and i_1, the adversary should not have any significant advantage in guessing which one of them has issued a valid signature. LS is *anonymous* if, for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{anon}_{LS,\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{anon-1}_{LS,\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{anon-0}_{LS,\mathcal{A}}(k) = 1]$ is negligible.

4.3 Our Instantiation

The protocol is quite the same as before except for two things, z is no longer chosen at random, but is simply a scalar corresponding to the time-frame t, and we can no longer use k_1^z as a private Waters key, but h_1^y is private. Once again, we will focus on the SXDH instantiation:

$\mathsf{Setup}(1^k)$: The system generates a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. One also chooses independent generators (g_1, g_2, k_1) of $\mathbb{G}_1^2 \times \mathbb{G}_2$. The group manager chooses a scalar $\gamma \xleftarrow{R} \mathbb{Z}_p$ for the master key $\mathsf{msk} = \gamma$, and computes $\Omega = g_2^\gamma$. The public key is then $\mathsf{pk} = (g_1, g_2, h_1, k_1, \Omega)$.

Join($\mathcal{U}_i$): In order to join the system, a user $\mathcal{U}_i$, with a pair of keys (usk[i], upk[i]) in the PKI, inter-
acts with the group manager, similarly to the previous scheme, so that at the end, the user owns a
certificate $\{i, \text{upk}[i], X_i = (g_1^{x_i}, g_2^{x_i}), g_1^{y_i}, A_i\}$, where x_i is chosen by the group manager but y_i is cho-
sen in common, but private to the user, while still extractable for our simulator in the proof. Then,
Reg[i] = $\{i, \text{upk}[i], X_i, g_1^{y_i}, A_i\}$, whereas sk[$i$] = y_i.

Sig(pk, $m, t, $sk[$i$]): When a user i wants to sign a message m during the time-frame t, he computes the
signature of m under his private key sk[i]: First, he will create his ephemeral ID, $\text{ID}(i,t) = g_1^{1/(t+y_i)}$,
and computes

$$\sigma = (\sigma_0 = \text{ID}(i,t), \sigma_1 = X_i, \sigma_2 = y_i, \sigma_3 = A_i, \sigma_4 = g_2^s, \sigma_5 = h_1^{y_i} \mathcal{F}(m)^s).$$

The relations could be verified by:

$$e(\sigma_0, g_2^t g_2^{\sigma_2}) = e(g_1, g_2) \qquad\qquad e(\sigma_{1,1}, g_2) = e(g_1, \sigma_{1,2})$$
$$e(\sigma_3, \Omega\sigma_{1,2}) = e(k_1, g_2) \times e(g_1, g_2^{\sigma_2}) \qquad\qquad e(\sigma_5, g_2) = e(h_1, g_2^{\sigma_2}) \times e(\mathcal{F}(m), \sigma_4)$$

In order to get anonymity, before publication, some of these tuples are thereafter committed, together
with the corresponding Groth-Sahai proofs, to prove the validity of the previous pairing equations

$$\sigma_i = (\sigma_0, \mathcal{C}(\sigma_1), \mathcal{C}(\sigma_2), \mathcal{C}(\sigma_3), \sigma_4, \sigma_5)$$

Match(pk, $m, t, m', t', \sigma, \sigma'$): This algorithm return 1 iff $t = t'$ and $\sigma_0 = \sigma_0'$.

4.4 Security Analysis

The security of this scheme can be proven in a similar way to the previous one. The main difference
between the two schemes comes from σ_5 where we cannot use t but a the private value y_i that appears
in some other equations. This hardens a little the security proof of the anonymity where we have to
change the last game so that we randomize, at the same time, both σ_0 and σ_5. But since the DDHI
assumption clearly implies the DDH one, and as the adversary can only make a limited number of
signature queries, he will only be able to work on a polynomial number of time-frames t and so we
can still use the Dodis-Yampolskiy VRF. The security analysis can be found in the Appendix B. The
proof of unforgeability remains quite the same.

5 Conclusion

We have exhibited a new way to build traceable signatures in the standard model. It requires around
40% of the elements used in the previous schemes. We also strengthen the security model of traceable
signatures: we have separated the opener from the group manager, which eases the non-frameability;
we also have extended the claim procedure, by creating a way to step-out (deny) in addition to the
step-in (confirmation). To the best of our knowledge, this is the first step-out group signature scheme
in the standard model.

Our identifier techniques also answers a problem opened 4 years ago, by creating the first List
Signature scheme in the Standard model: granted our new unique ID technique, we get a provably
secure list signature scheme.

Our solutions are quite efficient. If we consider the DLin implementation (to compare with the
previous one): 35 group elements, with 256-bit prime groups, we end up with a bit more than one
kilo-byte signature, which is rather small (especially in the standard model, whereas we are using
Groth-Sahai proofs.). At a first glance, many pairing computations may be involved in the verification
of n signatures. However, using batching techniques from [BFI$^+$10], it can be reduced to a quite
reasonable number. Namely, around 50 pairing computations only will be required for a single signature
verification.

References

[ABC+05]   Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, August 2005.

[BBS04]   Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.

[BCC+08]   Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Delegatable anonymous credentials. Cryptology ePrint Archive, Report 2008/428, 2008.

[BFI+10]   Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch groth-sahai. In *ACNS 2010*, volume 6123 of *LNCS*, pages 218–235, 2010.

[BFPV11]   Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *Proceedings of PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, 2011.

[BSZ05]   Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, February 2005.

[CHL05]   Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, May 2005.

[CSST06]   Sï¿½bastien Canard, Berry Schoenmakers, Martijn Stam, and Jacques Traorï¿½. List Signature Schemes. *Discrete Appl. Math.*, 154(2):189–201, 2006.

[Cv91]   David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, April 1991.

[DP06]   Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210. Springer, September 2006.

[DY05]   Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, January 2005.

[GS08]   Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.

[KTY04]   Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, May 2004.

[LY09]   Benoit Libert and Moti Yung. Efficient traceable signatures in the standard model. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography - Pairing 2009*, volume 5671 of *LNCS*, pages 187–205. Springer, 8 2009.

[SPMLS02]   Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, August 2002.

[Wat05]   Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

A   DLin Traceable Signature Scheme

A.1   Assumptions

Definition 7 (Decision Linear Problem in $\mathbb{G}$ [BBS04]). Let $\mathbb{G}$ be a cyclic group of order p generated by g. The *DLin assumption* states that the two distributions (u, v, w, u^a, v^b, w^c) and $(u, v, w, u^a, v^b, w^{a+b})$ are computationally indistinguishable for random group elements $u, v, w \in \mathbb{G}$ and random scalars $a, b, c \in \mathbb{Z}_p$.

Our scheme can easily be adapted in the DLin setting, instead of working in two different groups, we only need one, most of the previous assumptions can be adapted by stating that g_1, g_2 are two independent generators of a group $\mathbb{G}$.

A.2   Description

Setup(1^k)*:* The system generates a pairing-friendly system $(\mathbb{G}, \mathbb{G}_T, p, e)$. One also chooses independent generators (f, g, k) of $\mathbb{G}^3$.

The group manager chooses a scalar $\gamma \xleftarrow{R} \mathbb{Z}_p$ for the master key $\mathsf{msk} = \gamma$, and computes $\Omega = g^\gamma$. The opener produces a computationally binding Groth-Sahai environment with α as an opening key. *Commitments* in this setting will be noted $\mathcal{E}$. Technically they are double linear encryption and so fit well in the methodology. The public key is then $\mathsf{pk} = (f, g, k, \Omega, \mathcal{E})$.

Join($\mathcal{U}_i$): In order to join the system, a user $\mathcal{U}_i$, with a pair of keys $(\mathsf{usk}[i], \mathsf{upk}[i])$ in the PKI, interacts with the group manager (similarly to [DP06]):

- $\mathcal{U}_i$ chooses a random $y_i' \in \mathbb{Z}_p$, computes and sends $Y_i' = g^{y_i'}$, an extractable commitment of y_i' with a proof of consistence. Actually the trapdoor of the commitment will not be known to anybody, except to our simulator in the security proofs.
- The group manager chooses a new $x_i \in \mathbb{Z}_p$ and a random $y_i'' \xleftarrow{R} \mathbb{Z}_p$, computes and sends y_i'', $A_i = (kY_i'Y_i'')^{1/(\gamma + x_i)}$ and $X_{i,2} = g^{x_i}$ where $Y_i'' = g^{y_i''}$;
- $\mathcal{U}_i$ checks whether $e(A_i, \Omega g^{x_i}) = e(k, g) \times e(g, g)^{y_i' + y_i''}$. He can then compute $y_i = y_i' + y_i''$. He signs A_i under $\mathsf{usk}[i]$ into s_i for the group manager and produces, and a commitment of f^{y_i} under $\mathcal{E}$ with a proof showing it is well-formed. At this step, he also sign a part of $Reg[i]$ as in the SXDH version.
- The group manager verifies s_i under $\mathsf{upk}[i]$ on the message A_i, and appends the tuple $(i, \mathsf{upk}[i], X_i, A_i)$ to $\mathsf{Reg}[i]$. He can then send the last part of the certificate $X_{i,1} = f^{x_i}$.
- $\mathcal{U}_i$ thus checks, if he did receive f^{x_i} before (*i.e.* if $e(X_{i,1}, g) = e(f, X_{i,2})$), and then owns a valid certificate (A_i, X_i, y_i), where $\mathsf{sk}[i] = y_i$ is known to him only. Once again, the secrecy of y_i will be enough for the overall security. Note that if $X_{i,1}$ is invalid, one can ask for it again. In any case, the group manager cannot frame the user, but just do a denial of service attack, which is unavoidable.

    At this step $\mathsf{Reg}[i] = \{i, \mathsf{upk}[i], X_i, A_i, e_i, \pi_J[i], s_i\}$, whereas $\mathsf{sk}[i] = y_i$.

Sig($\mathsf{pk}, m, \mathsf{sk}[i]$): When a user i wants to sign a message m, he computes the signature of m under his private key $\mathsf{sk}[i]$:

First, he will create his ephemeral ID, $\mathsf{ID}(y_i, z) = f^{1/(z + y_i)}$, and publishes σ:

$$\sigma_0 = \mathsf{ID}(y_i, z), \sigma_1 = X_i, \sigma_2 = (f^{y_i}, g^{y_i}),$$
$$\sigma_3 = A_i, \sigma_4 = (f^z, g^z), \sigma_5 = k^z F(m)^s, \sigma_6 = g^s$$

Verifying the relations:

$$e(\sigma_0, \sigma_{4,2}\sigma_{2,2}) = e(f, g) \qquad\qquad e(\sigma_{1,1}, g) = e(f, \sigma_{1,2})$$
$$e(\sigma_3, \Omega\sigma_{1,2}) = e(f, g) \times e(f, \sigma_{2,2}) \qquad\qquad e(\sigma_{1,2}, g) = e(f, \sigma_{2,2})$$
$$e(\sigma_5, g) = e(k, \sigma_{4,2}) \times e(F(m), \sigma_6) \qquad\qquad e(\sigma_{4,1}, g) = e(f, \sigma_{4,2})$$

In order to get anonymity, some of these tuples are thereafter committed as group elements.

$$\sigma_i = (\sigma_0, \mathcal{C}(\sigma_1), \mathcal{C}(\sigma_2), \mathcal{E}(\sigma_3), \sigma_4, \sigma_5, \sigma_6)$$

With the corresponding Groth-Sahai proofs, to prove the validity of the previous pairing equations. The second one is a pairing product, so need 9 group elements in each group, the first, third and fourth are Linear Pairing Product, so needs only 3 extra elements in $\mathbb{G}$. The last ones do not use any committed data and so do not need any extra elements.

Verif(pk, m, σ): One simply has to verify if all the pairing equations hold.

Open(α, m, σ): The Opener just opens the commitment of A_i in σ_3, and then outputs a Groth-Sahai proof of knowledge of α.

Reveal(pk, i, α): The Opener verifies $\pi_J[i]$ and uses α to decrypt e_i and extracts the tracing key: $\mathsf{tk}[i] = f^{y_i}$. He then send it to the sub-opener together with a publicly verifiable proof showing that $\mathsf{tk}[i]$ is a valid decryption.

Trace($\mathsf{pk}, m, \sigma, \mathsf{tk}[i]$): The Sub-Opener picks $\beta \xleftarrow{R} \mathbb{Z}_p$ and outputs $(\mathsf{tk}[i]^\beta, \sigma_4^\beta, f^\beta)$, where anyone can use $Reg[i]$ to check if $e(g^{y_i}, f^\beta) = e(g, \mathsf{tk}[i]^\beta)$. Anyone can then check the validity of the output tuple and if $e(\sigma_0, \sigma_4^\beta \mathsf{tk}[i]^\beta) = e(f^\beta, g)$.

$\mathsf{Step}(\mathsf{pk}, m, \sigma, \mathsf{sk}[i])$: To step in or out of a given signature, a user just does the same thing as a sub-opener.

B List Signature Scheme

The security can be proven quite like before. There are a few changes, in the anonymity we have an extra term with some information on y_i, in the simulation we can not directly hide σ_5 as before, however we can put any random values in the certificate part, and program k_1 to still be able to conclude the reduction to the DDHI as long as we only work on a polynomial number of time-frames.

The soundness property is easier to achieve than the previous one, being able to sign twice in the same time-frame implies to be able either to generate two different ID for the same t, and so implies to work with two different y_i, and so to have two certificates, either to sign another message with the same user and to break the Waters unforgeability.

Theorem 8 (Anonymity). *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme, then there exists an adversary $\mathcal{B}$ that can break the $\ell - \mathsf{DDHI}$ problem in $\mathbb{G}_1$ or the SXDH assumption, where ℓ is the maximal number of signing queries for a user.*

Proof. Let us assume that an adversary is able to break the anonymity property of our scheme. It means than in the anonymity game, he has a non-negligible advantage $\epsilon > 0$ to distinguish $G^{(0)}$ where $b = 0$ from $G^{(1)}$ where $b = 1$. We start our sequence of games from $G^{(0)}$, denoted G_0. We first replace this game by G_1, where we make some simulations that are perfectly hiding: The simulator $\mathcal{B}$ is given a challenge $A = (g, g^y, \ldots, g^{y^\ell}) \in \mathbb{G}_1^{\ell+1}$ and $D = g^{1/y}$, for an unknown y. $\mathcal{B}$ first chooses different random values $z^*, z_1, \ldots, z_{\ell-1} \in \mathbb{Z}_p^\ell$, which completely define the polynomial $P = \prod_{i=1}^{\ell-1}(X + z_i)$, of degree $\ell - 1$. Granted the above challenge, $\mathcal{B}$ can compute $g_1 = g^{P(y)}$, and the rest of the public key is computed as in a normal scheme. In particular, one knows the master secret key γ. We also define $k_1 = D^\alpha, h_1 = k_1^\beta$ for a chosen α, β.

The future challenge user i_0 will virtually have $y_{i_0} = \mathsf{sk}[i_0] = y - z^*$. $\mathcal{B}$ can compute $g^{y_{i_0}} = g_1^y/g_1^{z^*}$ (from the input A without using y, even if we know it here). It will also be able to compute $\sigma_5 = h_1^{y_{i_0}} \mathcal{F}(m) = D^{\alpha\beta(y-z^*)}\mathcal{F}(m) = g^{\alpha\beta}\mathcal{F}(m)/D^{\alpha\beta z^*}$ – when $D = g^{1/y}$). The public certificate (in the Reg list) for the challenge user is $(g_1^{x_{i_0}}, g_2^{x_{i_0}}, g_1^y/g_1^{z^*}, (k_1 g_1^y/g_1^{z^*})^{1/(\gamma+x)})$, plus some proofs and signatures. $\mathcal{B}$ is also given Groth-Sahai commitment keys and encryption keys (decryption keys are unknown, hence the classical notion of anonymity and not full-anonymity). This setup is indistinguishable from the real one since all the keys are generated as in the real game. Because of the knowledge of the master secret key, $\mathcal{B}$ simulator easily answers any join queries, both active and passive. But has to guess i_0, which is correct with probability $1/n$, where n is the total number of passive join queries. It can also answer any corruption, that should not happen for the challenge user, even if we know y in this game.

As he is able to know all $x_i's$ and all $y_i's$ for registered users (except the challenge one), he will be able to answer signing queries as users would do. For the challenge user, on the j-th timeframe signing query, he computes $\sigma_0 = g_1^{1/(\mathsf{sk}[i]+z_j)} = g^{\prod_{i \neq j}(y+z_i)}$, that can be done from the challenge input A, the rest is done as in the real game using y and z_j as ephemeral random with each time an additional random s. For the challenge signing query, if it happens in an already used timeframe then he aborts, else he does the same has above with the ephemeral value z^*, and the expected ID is $g_1^{1/(\mathsf{sk}[i]+z^*)} = g^{P(y)/y} = g^{Q(y)}g^{\prod(z_i)/y}$, where $Q = (\prod_{i=1}^{\ell}(X + z_i) - \prod(z_i))/X$ is a polynomial of degree $\ell - 1$ and thus $g^{Q(y)}$ can be computed from the instance. He thus outputs $\sigma_0 = g^{Q(y)} \cdot D^{\prod(z_i)}$ and $\sigma_5 = g^{\alpha\beta}\mathcal{F}(m)/D^{\alpha\beta z^*}$. Since $D = g^{1/y}$, the signature is similar to the above one, and so is indistinguishable from a real signature.

We then modify the game, into G_2, where we initialize Groth-Sahai commitment keys in a perfectly hiding setting with the trapdoor, to allow the simulator to cheat in the proofs. Then all the proofs are simulated. This game is indistinguishable from the previous one under the SXDH.

In G_3, for the challenge user signing queries, we use random commitments, ciphertext for $\sigma_1, \sigma_2, \sigma_3$. The commitments were already random, because of the perfectly hiding setting, but a random ciphertext is indistinguishable from the real one under the semantic security of the encryption scheme, the SXDH assumption.

In G_4, we do not know anymore y, that we did not use anymore anyway, and thus this game is perfectly indistinguishable from the previous one.

In G_5, D is a random value, which is indistinguishable from the real one under the $\ell - \mathsf{DDHI}$ assumption as we only have a polynomial number of z_i in input like in the Dodis-Yampolskiy PRF: the challenge signature does not depend anymore on the challenge user, since σ_0 is random because of the random D, and σ_5 is random and independent because of the additional randomness α.

To complete the proof, we should make the same sequence again, starting from G_0' that is $G^{(1)}$, up to G_5', that is perfectly indistinguishable from G_5, hence the computational indistinguishability between $G_0' = G^{(1)}$ and $G_0 = G^{(0)}$. $\qquad\square$