Improving Revocation for Group Signature with Redactable Signature

Olivier Sanders(✉)

Applied Crypto Group, Orange Labs, Cesson-Sévigné, France
olivier.sanders@orange.com

Abstract. Group signature is a major cryptographic tool allowing anonymous access to a service. However, in practice, access to a service is usually granted for some periods of time, which implies that the signing rights must be deactivated the rest of the time. This requirement thus calls for complex forms of revocation, reminiscent of the concept of time-bound keys. However, schemes implementing this concept are rare and only allow revocation with limited granularity. That is, signing keys are associated with an expiry time and become definitively useless once the latter has passed.

In this paper, we revisit the notion of group signatures with time-bound keys with several contributions. Firstly, we extend this notion to allow high granularity revocation: a member's signing key can in particular be deactivated at some moments and then be automatically reinstated. Secondly, we show that this complex property is actually simple to achieve using redactable signature. In particular, we consider in this context a recent redactable signature scheme from PKC 20 that we improve by dramatically reducing the size of the public key. The resulting construction is of independent interest.

1 Introduction

Group signature, introduced by Chaum and van Heyst [10], enables anonymous, yet accountable, authentication to a service. In such a system, a so-called group manager has the responsibility of a group of users who can issue anonymous signatures on behalf of the group. More specifically, anyone can check that the resulting signatures were issued by a group member but it is impossible, except for the group manager, to identify the actual signer. This means for example that a service provider can check that the user has the right to access the service whereas the user has the assurance that this authentication leaks as little information as possible.

This ability to reconcile the interests of all parties makes it an ideal solution in many scenarios, which explains the countless papers on this topic. We in particular note that some simple variants such as DAA or EPID are today massively deployed [1,25]. Group signature has also been proposed in the context of public transport (*e.g.* [12,15]) to implement an anonymous version of a transport subscription pass such as, for example, the Navigo pass [19] in Paris,

J. A. Garay (Ed.): PKC 2021, LNCS 12710, pp. 301–330, 2021.
https://doi.org/10.1007/978-3-030-75245-3_12

that allows a passenger to take unlimited trips within some fixed periods of time. With group signature, this passenger could prove that he has the right to access the transport service without being identified by the transport operator.

This use-case thus highlights the benefits of group signatures at first glance but also reveals their limitations when we consider more thoroughly a real-world application. Indeed, access to a service is usually not granted for ever but only for some periods of time. For example, the user of a public transport system typically pays for a 1 month or year subscription starting from a date of his choice. We can also envision alternative cases where one would subscribe to a pass providing unlimited access but only during weekends.

Providing signing rights without the ability to limit them to some time periods is therefore extremely problematic in practice. We in particular note that this cannot be fixed by revealing these time periods in the signature as it would break anonymity. We here touch a weak point of group signatures. Although most schemes come with efficient enrolment procedures, the problem of limiting the signing rights is rarely considered in papers, and usually only through the concept of revocation that can be implemented in three different ways.

The first kind of revocation approach is the one where the group manager regularly changes the group public key, thus resetting the group. This is compatible with any scheme but is highly impractical in practice as it is becomes necessary to issue a new group signing key for each user at the beginning of each time period.

The second kind of revocation is the one where the group manager provides at each time period an updated information on the current set of non-revoked members. This information is then used by the group members during the generation of the signature to prove that they are still part of the group. In other words, a group member can no longer issue a valid group signature once he is revoked. This approach may offer nice asymptotic complexity (e.g. [16,17]) but increases both the computational cost and the size of a group signature while forcing the user to regularly update their group signing key.

The last kind of revocation is based on the notion of revocation lists formalized by Boneh and Shacham under the name of Verifier-Local Revocation (VLR) [6]. Here, a member is revoked by adding a specific information in a revocation list allowing the verifiers to trace all signatures issued by this member, at least for a certain time. This revocation technique is interesting because it does not require to update group members' signing keys and has no impact on the complexity of the group signature itself. Unfortunately, it makes the verification process linear in the number of revoked users and so can only be used in situations where this revocation list remains relatively short. In particular, we cannot use it directly in the context of public transport to deactivate users' signing keys when their subscription is over as it would quickly lead to revocation lists containing millions of elements. It can only be used for exceptional situations such as the theft or loss of some user's smartphone.

To address this problem, Chu *et al.* [11] proposed to improve VLR group signatures by associating signing keys with an expiry time beyond which the

group member loses his ability to sign, hence the name of group signature with time-bound keys. Such systems thus deal with two kinds of revocation, a *natural* revocation that automatically excludes users once their expiry time has passed and a *premature* revocation that works exactly as in a standard VLR scheme. This way, such systems dramatically limit the size of the revocation list and so the computational cost of the verification process.

Following [11], Emura *et al.* [13] recently proposed an improved security model for this primitive along with an efficient construction that blends the last two kinds of revocation we mentioned above. Concretely, the natural revocation is based on [17] by providing at each time period an information that enables non-revoked users to issue group signatures while premature revocation is still based on VLR. The resulting construction has nice asymptotic complexity but suffers from the limitations listed above, namely the need to prove in each signature that the signing key is still active and the need to update the latter at each time period.

Our Contribution. We propose in this paper to improve group signature with time-bound keys in several ways.

Firstly, we allow the group manager to associate a group signing key with any set of time periods and not just an expiry time as in previous works. Concretely, this means that a user may be able to sign at some time period t_1 and then be considered as revoked during the subsequent time periods before being automatically reinstated at a later time period t_2. This can for example be useful in the case mentioned above where a user would have access to a service only during weekends. The signing key will then be active only during the weekends and not during the other days of the week. This thus improves the granularity of the revocation but raises some privacy issues as it now means that revocation is not necessarily permanent: we must therefore ensure both backward and forward unlinkability for revoked users. We also allow opening queries in our anonymity experiment, contrarily to [13], and thus achieve a stronger notion of privacy.

Our second contribution is a construction of a very efficient scheme satisfying our new definition based on unlinkable redactable signatures (URS) [9,22]. An URS scheme enables to issue a signature σ on a set of messages $\{m_i\}_{i=1}^n$ and then to publicly derive from σ an unlinkable signature σ' that only proves authenticity of a subset of messages $\{m_i\}_{i\in\mathcal{I}}$, for some $\mathcal{I} \subset [1, n]$. Here, unlinkability means that σ' does not leak information on the set of *redacted* messages $\{m_i\}_{i\notin\mathcal{I}}$ and cannot be linked to σ beyond the fact that both signatures coincide on $\mathcal{I}$.

We use URS to construct group signature with time-bound keys as follows. During the enrolment process a group manager will issue a redacted signature σ on a set of messages $\{m_i\}_{i=1}^n$ where $m_i \neq 0$ if and only if the new member has the right to issue group signatures at the time period i. To generate a group signature at a time period t this member then only has to redact all the messages $\{m_i\}_{i=1}^n$ but m_t and then send the resulting derived signature σ' attesting that $m_t \neq 0$. Intuitively, thanks to unlinkability of URS schemes, we do not have to hide σ' or the redacted messages, which leads to a very efficient

and simple protocol. Similarly, signatures from an unregistered member or an illicit extension of signing rights (*i.e.* a member that manages to sign outside his periods of activity) imply a forgery against the URS scheme. It then essentially remains to add a premature revocation mechanism that still retains backward and forward unlinkability but this can be done using rather standard techniques. An interesting outcome of our approach based on URS is that our group members no longer need to update their signing keys at each time period. They only need to know their original signing key and the current time period to generate a group signature.

So far we have shown that URS schemes lead to simple constructions of group signature with time-bound keys. However, this result is of practical significance only if we can propose an efficient candidate for the URS scheme. An interesting candidate was proposed very recently at PKC 20 [22], with extremely short signatures containing only four elements and which can be verified with essentially one exponentiation by non-redacted message. This might seem perfect in our context (as each group signature only involves one non-redacted element) but unfortunately the construction in [22] suffers from a large public key, quadratic in n. In the context of public transport, where it seems reasonable to consider one-day time period and a public key valid for the next 3 years, this means that the public parameters would contain millions of elements, which can be cumbersome. We therefore propose an improved version of the construction of [22], which retains all the nice features of the latter but with a public key only *linear* in n. We believe that this contribution is of independent interest, although its security analysis is done in the generic group model and the random oracle model.

Organisation. We recall in Sect. 2 the notion of bilinear groups and present the computational assumptions that underlay the security of our protocols. Section 3 is dedicated to URS and contains in particular a new construction with shorter public keys. Section 4 presents an improved model for group signature with time-bound keys whereas Sect. 5 shows how to instantiate this primitive with URS. Finally, the last section compares the efficiency of our contributions with the most relevant schemes from the state-of-the-art.

2 Preliminaries

Bilinear Groups. Our construction requires bilinear groups whose definition is recalled below.

Definition 1. *Bilinear groups are a set of three groups* $\mathbb{G}_1$, $\mathbb{G}_2$, *and* $\mathbb{G}_T$ *of order* p *along with a map, called pairing,* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *that is*

1. *bilinear: for any* $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, *and* $a, b \in \mathbb{Z}_p$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$;
2. *non-degenerate: for any* $g \in \mathbb{G}_1^*$ *and* $\widetilde{g} \in \mathbb{G}_2^*$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$;
3. *efficient: for any* $g \in \mathbb{G}_1$ *and* $\widetilde{g} \in \mathbb{G}_2$, $e(g, \widetilde{g})$ *can be efficiently computed.*

As most recent cryptographic papers, we only consider bilinear groups of prime order with *type 3* pairings [14], meaning that no efficiently computable homomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Computational Assumptions. The security analysis of our protocols will make use of the following two assumptions.

- SDL assumption: Given $(g, g^a) \in \mathbb{G}_1^2$ and $(\widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_2^2$, this assumption states that it is hard to recover a. It thus essentially extends the standard discrete logarithm assumption to the case of bilinear groups.
- EDDH assumption: Given $(\{g^{a \cdot c^i}\}_{i=0}^{2n^2}, \{g^{b \cdot c^i}\}_{i=0}^{n-1}, \{g^{c^i}\}_{i=1}^{3n^2}, \{g^{d \cdot c^i}\}_{i=1}^{2n^2}, g^z) \in \mathbb{G}_1^{7n^2+n+2}$ and $(\{\widetilde{g}^{c^i}\}_{i=1}^{2n^2}, \widetilde{g}^d, \{\widetilde{g}^{a \cdot c^i}\}_{i \in [1,2n^2] \setminus]n^2-n, n^2+n[}) \in \mathbb{G}_2^{4n^2-2n+1}$, the EDDH assumption states that it is hard to decide whether $z = a \cdot b \cdot c^{n^2} + b \cdot d$ or z is random.

We note that our EDDH assumption is an instance of the generic BBG assumption [8]. The hardness of the underlying problem is studied in the generic group model in the full version [23] but it is intuitively based on the following rationale. A non-random z is the sum of two monomials, $a \cdot b \cdot c^{n^2}$ and $b \cdot d$, that are both multiple of b. As b is only provided in $\mathbb{G}_1$ with $\{g^{b \cdot c^i}\}_{i=0}^{n-1}$, any attempt to distinguish z from randomness will intuitively require to pair an element of this set with an element of $\mathbb{G}_2$. If the latter belongs to $\{\widetilde{g}^{a \cdot c^i}\}_{i \in [1,2n^2] \setminus]n^2-n, n^2+n[}$, then we get an element of $\mathbb{G}_T$ whose exponent is of the form $a \cdot b \cdot c^i$ for some $i \in [1, n^2 - 1] \cup [n^2 + n, 2n^2]$. This is not sufficient to distinguish the first monomial in z so we must pair g^z with some g^{c^i} for $i \geq n$, resulting (once we remove the first monomial) in an element $e(g, \widetilde{g})^{b \cdot d \cdot c^i}$ with $i \geq n$. The latter element cannot be computed from the EDDH instance as we only have $g^{b \cdot c^i}$, for $i < n$, in $\mathbb{G}_1$ and $\widetilde{g}^d$ in $\mathbb{G}_2$. The same reasoning applies if we start by trying to remove the second monomial.

3 Redactable Signatures with Linear Size Public Key

3.1 Unlinkable Redactable Signature

Before presenting our construction, we recall the notion of unlinkable redactable signature (URS) from [9], using the notations from [22]. The core idea of this primitive is that a signature σ issued on a set[1] of messages $\{m_i\}_{i=1}^n$ can be publicly redacted so as to be valid only on a subset $\{m_i\}_{i \in \mathcal{I}}$, for some $\mathcal{I} \subset [1, n]$. This feature is important both for efficiency and privacy reasons. The set of redacted messages is then $\{m_i\}_{i \in \overline{\mathcal{I}}}$, where $\overline{\mathcal{I}} = [1, n] \setminus \mathcal{I}$.

[1] We stress that the index of each message is important as the term "set" might lead to a confusion. Here, m_i means that this message has been signed under the i-th element of the public key and is only valid for this position. In particular, deriving from σ a signature on $\{m_{\pi(i)}\}_{i=1}^n$ for some permutation π would constitute a forgery.

Syntax. An URS scheme consists of the 4 following algorithms.

- $\texttt{Keygen}(1^\lambda, n)$: On input a security parameter 1^λ and an integer n, this algorithm returns a key pair $(\texttt{sk}, \texttt{pk})$ supporting signatures on sets of n messages $\{m_i\}_{i=1}^n$.
- $\texttt{Sign}(\texttt{sk}, \{m_i\}_{i=1}^n)$: On input n messages $\{m_i\}_{i=1}^n$ and the signing key $\texttt{sk}$, this algorithm outputs a signature σ.
- $\texttt{Derive}(\texttt{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$: On input a signature σ on $\{m_i\}_{i=1}^n$, the public key $\texttt{pk}$ and a subset $\mathcal{I} \subseteq [1, n]$, this algorithm returns a redacted (or derived) signature $\sigma_\mathcal{I}$ on the subset of messages $\{m_i\}_{i\in\mathcal{I}}$.
- $\texttt{Verify}(\texttt{pk}, \sigma, \{m_i\}_{i\in\mathcal{I}})$: On input the public key $\texttt{pk}$, a set of messages $\{m_i\}_{i\in\mathcal{I}}$ and a signature σ (generated by `Sign` or `Derive`), this algorithm outputs 1 (valid) or 0 (invalid).

Security Model. As any signature, a redactable signature must be unforgeable, meaning that it is impossible to output a valid signature on an unsigned set (or subset) of messages. However, a subtlety arises if we consider the generation of a new derived signature as an attack, even if the latter is only valid on an already signed subset of messages. Following the terminology of standard signature schemes, a construction preventing generation of new signatures is said to be strongly unforgeable. As its name suggests, strong unforgeability implies unforgeability. In [22], these two notions were defined as in Fig. 1. These experiments make use of the following oracles that define a counter c and three tables, Q_1, Q_2 and Q_3:

- $\mathcal{O}\texttt{Sign}^*(\{m_i\}_{i=1}^n)$: on input a set of n messages, this oracle returns $\texttt{Sign}(\texttt{sk}, \{m_i\}_{i=1}^n)$, stores $Q_1[c] = (\sigma, \{m_i^{(c)}\}_{i=1}^n)$ and increments $c \leftarrow c + 1$.
- $\mathcal{O}\texttt{Sign}(\{m_i\}_{i=1}^n)$: on input a set of n messages, this oracle computes $\sigma \leftarrow \texttt{Sign}(\texttt{sk}, \{m_i\}_{i=1}^n)$, stores $Q_1[c] = (\sigma, \{m_i^{(c)}\}_{i=1}^n)$ and increments $c \leftarrow c + 1$.
- $\mathcal{O}\texttt{Derive}(k, \mathcal{I})$: on input an index k and a set $\mathcal{I}$, this algorithm returns $\perp$ if $Q_1[k] = \emptyset$ or if $\mathcal{I} \nsubseteq [1, n]$. Else, it uses σ and $\{m_i\}_{i=1}^n$ stored in $Q_1[k]$ to return $\texttt{Derive}(\texttt{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$. The set $\{m_i\}_{i\in\mathcal{I}}$ is then added to Q_2.
- $\mathcal{O}\texttt{Reveal}(k)$: on input an index k, this algorithm returns $\perp$ if $Q_1[k] = \emptyset$ and $Q_1[k] = (\sigma, \{m_i^{(k)}\}_{i=1}^n)$ otherwise. The set $\{m_i^{(k)}\}_{i=1}^n$ is then added to Q_3.

The difference between $\mathcal{O}\texttt{Sign}^*$ and $\mathcal{O}\texttt{Sign}$ is that the latter does not return anything. $\mathcal{O}\texttt{Sign}$ indeed simply generates a signature that can be used as input of subsequent $\mathcal{O}\texttt{Derive}$ queries. Finally, we also recall in the same figure the notion of unlinkability that provides strong privacy guarantees as it ensures that no information leak on redacted messages and that it is impossible to link the input (σ) and the output (σ') of the `Derive` algorithm beyond the fact that they both coincide on the set $\{m_i\}_{i\in\mathcal{I}}$ of revealed messages.

Let $\mathcal{A}$ be a probabilistic polynomial adversary. An URS scheme is

- unforgeable if $\texttt{Adv}^{uf}(\mathcal{A}) = |\Pr[\texttt{Exp}_\mathcal{A}^{uf}(1^\lambda, n) = 1]|$ is negligible for any $\mathcal{A}$.

Unforgeability
$\mathtt{Exp}^{uf}_{\mathcal{A}}(1^\lambda, n)$
1. $c \leftarrow 0$; $Q_1 \leftarrow \emptyset$;
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
3. $(\sigma^*, \{m_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}^*}(\mathsf{pk})$
4. Return 1 if $\mathcal{I} \neq \emptyset$
   and $\mathtt{Verify}(\mathsf{pk}, \sigma^*, \{m_i\}_{i\in\mathcal{I}}) = 1$
   and $\forall j < c, \exists k_j \in \mathcal{I} : m_{k_j} \neq m^{(j)}_{k_j}$
5. Else, return 0

Strong Unforgeability
$\mathtt{Exp}^{suf}_{\mathcal{A}}(1^\lambda, n)$
1. $Q_1, Q_2, Q_3 \leftarrow \emptyset$;
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
3. $(\sigma^*, \{m_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign},\mathcal{O}\mathtt{Derive},\mathcal{O}\mathtt{Reveal}}(\mathsf{pk})$
4. Return 1 if $\mathcal{I} \neq \emptyset$
   and $\mathtt{Verify}(\mathsf{pk}, \sigma^*, \{m_i\}_{i\in\mathcal{I}}) = 1$
   and $\{m_i\}_{i\in\mathcal{I}} \notin Q_2$
   and $\forall \{m^\ell_i\}_{i=1}^n \in Q_3$:
   $\exists k_j \in \mathcal{I} : m_{k_j} \neq m^\ell_{k_j}$
5. Else, return 0

Unlinkability
$\mathtt{Exp}^{unl-b}_{\mathcal{A}}(1^\lambda, n)$
1. $(\mathsf{pk}, \mathcal{I}, \{m^{(0)}_i\}_{i=1}^n, \{m^{(1)}_i\}_{i=1}^n, \sigma^{(0)}, \sigma^{(1)}) \leftarrow \mathcal{A}()$
2. If $\exists b' \in \{0,1\} : \mathtt{Verify}(\mathsf{pk}, \sigma^{(b')}, \{m^{(b')}_i\}_{i=1}^n) = 0$, return 0
3. If $\exists j \in \mathcal{I} : m^{(0)}_j \neq m^{(1)}_j$, return 0
4. $\sigma^{(b)}_{\mathcal{I}} \leftarrow \mathtt{Derive}(\mathsf{pk}, \sigma^{(b)}, \{m^{(b)}_i\}_{i=1}^n, \mathcal{I})$
5. $b^* \leftarrow \mathcal{A}(\sigma^{(b)}_{\mathcal{I}})$
6. Return $(b^* = b)$.

Fig. 1. Security notions for redactable signatures

- strongly unforgeable if $\mathtt{Adv}^{suf}(\mathcal{A}) = |\Pr[\mathtt{Exp}^{suf}_{\mathcal{A}}(1^\lambda, n) = 1]|$ is negligible for any $\mathcal{A}$.
- unlinkable $\mathtt{Adv}^{unl} = |\Pr[\mathtt{Exp}^{unl-1}_{\mathcal{A}}(1^\lambda, n) = 1] - \Pr[\mathtt{Exp}^{unl-0}_{\mathcal{A}}(1^\lambda, n) = 1]|$ is negligible for any $\mathcal{A}$.

3.2 Our Construction

Intuition. The system of [22] is constructed upon the Pointcheval-Sanders (PS) signature scheme [20] for blocks of n messages, by aggregating the redacted messages in one element and then proving, thanks to an additional element, that the latter was honestly generated. Unfortunately, in [22], this is done by adding a quadratic number (in n) of elements in the public key, which quickly becomes inefficient.

Our approach shares some similarities with [22] but differs in several important ways. Our first difference is that we do not start from the original PS signature scheme but rather from a specific instantiation where the secret key only contains two scalars x and y and where a signature $(\sigma_1, \sigma_2) \in \mathbb{G}_1^2$ on a set of messages $\{m_i\}_{i=1}^n$ is $(h, h^{x+\sum_{i=1}^n y^i \cdot m_i})$, for some random element $h \in \mathbb{G}_1$. Concretely, this implicitly sets $y_i = y_1^i$, for $i \geq 2$, in the original PS signature scheme. The original proof of PS signatures in the generic group model readily adapts to this particular settings. Actually, this instantiation has been recently

studied by McDonald [18]. In any case, the validity of σ can be checked by simply testing whether the following equation holds:

$$e(\sigma_1, \widetilde{g}^x \cdot \prod_{i=1}^{n} \widetilde{g}^{y^i \cdot m_i}) = e(\sigma_2, \widetilde{g}),$$

where $\widetilde{g}^x$ and $\{\widetilde{g}^{y^i}\}_{i=1}^n$ are parts of the public key. As in [22], we can compute an element $\widetilde{\sigma} \leftarrow \prod_{i \in \overline{\mathcal{I}}} \widetilde{g}^{y^i \cdot m_i}$ that aggregates all the redacted messages but we must then ensure that $\widetilde{\sigma}$ will not be used by an adversary to cheat the verification procedure. Concretely, since the `Verify` algorithm now checks the following equation:

$$e(\sigma_1, \widetilde{g}^x \cdot \widetilde{\sigma} \cdot \prod_{i \in \mathcal{I}} \widetilde{g}^{y^i \cdot m_i}) = e(\sigma_2, \widetilde{g}),$$

one must intuitively ensure that $\widetilde{\sigma}$ has not been used to aggregate illicit elements of the form $\widetilde{g}^{-x}$ or $\widetilde{g}^{y^i \cdot m'_i}$, for some $i \in \mathcal{I}$, which would lead to trivial forgeries. Here, we can't use the solution from [22] anymore because our secret key is different, but primarily because it would entail a public key containing $O(n^2)$ elements.

The first step of our new strategy is to notice that the following pairing

$$G = e(\prod_{i \in \mathcal{I}} g^{y^{n+1-i}}, \widetilde{\sigma})$$

is of the form $e(g^{\sum_{i \in \mathcal{I}, j \in \overline{\mathcal{I}}} y^{n+1-i+j} \cdot m_j}, \widetilde{g})$ for an honestly generated $\widetilde{\sigma}$. Since $\mathcal{I} \cap \overline{\mathcal{I}} = \emptyset$, we note that the first input of the resulting pairing can be computed without the knowledge of $g^{y^{n+1}}$. In particular, it can be computed only from the $2n-1$ elements g^{y^i}, for $i \in [1, n] \cup [n+2, 2n]$, that we add to the public key.

Now, following [22], it might be tempting to conclude that there is an equivalence here, namely that an ill-formed $\widetilde{\sigma}$ will necessarily lead to a pairing G involving $g^{y^{n+1}}$ or an element of the form $g^{x \cdot y^u}$, for some $u > 0$, that are not provided in the public key.

Unfortunately, this is not true because, in the case where G is computed from an ill-formed $\widetilde{\sigma} \leftarrow \prod_{i=1}^{n} \widetilde{g}^{y^i \cdot m'_i}$ (for example, one such that $\exists i \in \mathcal{I}$ with $m'_i \neq 0$), we have:

$$G = e(g^{\sum_{u=1}^{2n} y^u \cdot a_u}, \widetilde{g})$$

with $a_{n+1} = \sum_{i \in \mathcal{I}} m'_i$. It is thus trivial for the adversary to select values m'_i, for $i \in \mathcal{I}$, that will cancel the coefficient a_{n+1} of y^{n+1}. In such a case, it can create a forgery using only the elements g^{y^i}, for $i \in [1, n] \cup [n+2, 2n]$, which are provided in the public key.

This solution therefore does not work as it is, but this does not mean that we should completely discard it either. Instead, we will keep the same approach but add some unpredictability in the coefficient a_{n+1} to thwart the previous attack.

To this end, we will generate the hash outputs $c_i \leftarrow \mathtt{H}(\sigma_1||\sigma_2||\widetilde{\sigma}||\mathcal{I}||i)$, for $i \in \mathcal{I}$, and use them to compute a different pairing

$$e(\prod_{i\in\mathcal{I}} g^{y^{n+1-i}\cdot c_i}, \widetilde{\sigma}) = e(g^{\sum\limits_{i\in\mathcal{I},j\in\overline{\mathcal{I}}} y^{n+1-i+j}\cdot c_i\cdot m_j}, \widetilde{g}).$$

Here, our previous remark is still valid: for a honestly generated $\widetilde{\sigma}$, there is no monomial in y^{n+1}. But now, in the case of an ill-formed $\widetilde{\sigma}$ as above, the coefficient a_{n+1} of y^{n+1} is $\sum_{i\in\mathcal{I}} c_i \cdot m'_i$. Since c_i depends on $\widetilde{\sigma}$ and so on m'_i, we see that any strategy to choose the scalars m'_i in such a way that $a_{n+1} = 0$ is unlikely to succeed as any change in a value of m'_i will lead to a completely new set of values $\{c_i\}_{i\in\mathcal{I}}$. This solution is reminiscent of the approach to prevent rogue key attacks in aggregate signature or multi-signature schemes (see *e.g.* [5]) but we use it here for another purpose and with a significant difference. We can't indeed derive c_i directly from the messages m_i as it would prevent efficient proofs of knowledge of m_i but rather from the first elements $(\sigma_1, \sigma_2, \widetilde{\sigma})$ of the signature. One of the difficulties of the security proof is to show that the adversary can't leverage this fact to create forgeries.

At this stage, we have an unforgeable redactable signature scheme. To achieve unlinkability, we use exactly the same trick as in [22], namely we aggregate a signature on a random message t under a dummy public key and then redact t to perfectly hide the set $\{m_i\}_{i\in\overline{\mathcal{I}}}$. Together with the re-randomizability of the PS signatures (that our variant inherits) this leads to an unlinkable redactable signature scheme, as defined in Sect. 3.1.

The Scheme

- $\mathtt{Keygen}(1^\lambda, n)$: this algorithm generates $(g, \widetilde{g}) \xleftarrow{\$} \mathbb{G}_1^* \times \mathbb{G}_2^*$ along with two random scalars $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$ and computes the following elements:
  - $\widetilde{X} \leftarrow \widetilde{g}^x$;
  - $\widetilde{Y}_i \leftarrow \widetilde{g}^{y^i}$, $\forall 1 \le i \le n$;
  - $Y_i \leftarrow g^{y^i}$, $\forall i \in [1, n] \cup [n+2, 2n]$.

  The secret key sk is then (x, y) whereas the public key pk is $(\mathtt{H}, g, \widetilde{g}, \{Y_i\}_{i=1}^n, \{Y_i\}_{i=n+2}^{2n}, \widetilde{X}, \{\widetilde{Y}_i\}_{i=1}^n)$, where $\mathtt{H} : \{0,1\}^* \to \mathbb{Z}_p^*$ is the description of a hash function.
- $\mathtt{Sign}(\mathsf{sk}, \{m_i\}_{i=1}^n)$: to sign n messages $m_1, \ldots, m_n$, the signer selects a random element $\sigma_1 \xleftarrow{\$} \mathbb{G}_1$, computes $\sigma_2 \leftarrow \sigma_1^{x+\sum_{i=1}^n y^i\cdot m_i}$ and then outputs the signature $\sigma = (\sigma_1, \sigma_2)$.
- $\mathtt{Derive}(\mathsf{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$: on input a signature $\sigma = (\sigma_1, \sigma_2)$ on $\{m_i\}_{i=1}^n$, the public key pk and a subset $\mathcal{I} \subset [1, n]$, this algorithm generates two random scalars $(r, t) \xleftarrow{\$} \mathbb{Z}_p^2$ and computes:
  - $\sigma'_1 \leftarrow \sigma_1^r$;
  - $\sigma'_2 \leftarrow \sigma_2^r \cdot (\sigma'_1)^t$;
  - $\widetilde{\sigma}' \leftarrow \widetilde{g}^t \cdot \prod_{j\in\overline{\mathcal{I}}} \widetilde{Y}_j^{m_j}$.

Then, for all $i \in \mathcal{I}$, it computes the scalar $c_i \leftarrow \mathtt{H}(\sigma'_1||\sigma'_2||\widetilde{\sigma}'||\mathcal{I}||i)$ that is used to generate:

$$\sigma'_3 \leftarrow \prod_{i\in\mathcal{I}}[Y^t_{n+1-i} \cdot \prod_{j\in\overline{\mathcal{I}}} Y^{m_j}_{n+1-i+j}]^{c_i}.$$

where $\overline{\mathcal{I}} = [1,n] \setminus \mathcal{I}$. Finally, the signer returns the derived signature $\sigma_{\mathcal{I}} = (\sigma'_1, \sigma'_2, \sigma'_3, \widetilde{\sigma}')$ on $\{m_i\}_{i\in\mathcal{I}}$.

– $\mathtt{Verify}(\mathsf{pk}, \sigma, \{m_i\}_{i\in\mathcal{I}})$: this algorithm parses σ as $(\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma}) \in \mathbb{G}_1^3 \times \mathbb{G}_2$, setting $\sigma_3 = 1_{\mathbb{G}_1}$ and $\widetilde{\sigma} = 1_{\mathbb{G}_2}$ if $\sigma \in \mathbb{G}_1^2$ (*i.e.* if σ has not been derived). If $\sigma_1 = 1_{\mathbb{G}_1}$, then it returns $\perp$. Else, the algorithm tests if the following equations hold, in which case it returns 1.
  1. $e(\sigma_1, \widetilde{X} \cdot \widetilde{\sigma} \cdot \prod_{i\in\mathcal{I}} \widetilde{Y}_i^{m_i}) = e(\sigma_2, \widetilde{g})$;
  2. $e(\sigma_3, \widetilde{g}) = e(\prod_{i\in\mathcal{I}} Y^{c_i}_{n+1-i}, \widetilde{\sigma})$;

  where $c_i \leftarrow \mathtt{H}(\sigma_1||\sigma_2||\widetilde{\sigma}||\mathcal{I}||i)$. If (at least) one of these equations is not satisfied, then the algorithm returns 0.

Remark 2. 1) We note that the elements provided in the public key are sufficient to compute derived signatures, and in particular the element σ'_3 since, for all $i \in \mathcal{I}$ and $j \in \overline{\mathcal{I}}$, we have $n+1-i \in [1,n]$ and $n+1-i+j \in [1,n] \cup [n+2,2n]$.

2) We have defined σ'_3 as $\prod_{i\in\mathcal{I}}[Y^t_{n+1-i} \cdot \prod_{j\in\overline{\mathcal{I}}} Y^{m_j}_{n+1-i+j}]^{c_i}$ for ease of exposition but we note that applying directly this formula to compute this element would be rather inefficient in most cases as it would entail $|\mathcal{I}|(n-|\mathcal{I}|+1)$ exponentiations. For all $u \in [1,n] \cup [n+2,2n]$, let us define $t_u = t$ if $u = n+1-i$ for some $i \in \mathcal{I}$ and $t_u = 0$ otherwise. Then, σ'_3 can also be written as follows:

$$\sigma'_3 = \prod_{u\in[1,n]\cup[n+2,2n]} Y_u^{t_u+s_u}$$

where $s_u = \sum_{\substack{i\in\mathcal{I}, j\in\overline{\mathcal{I}}:\\ j-i=u-n-1}} c_i \cdot m_j$, for all $u \in [1,n] \cup [n+2,2n]$. Computing σ'_3 this way requires at most 2n-1 exponentiations.

Correctness. We prove here that the `Verify` algorithm returns 1 for any signature σ returned by the `Sign` or the `Derive` algorithm.

First, in the case where σ has not been derived, we note that the second verification equation is trivially satisfied as σ_3 and $\widetilde{\sigma}$ are the neutral elements of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. Moreover, in this case, we have $e(\sigma_2, \widetilde{g}) = e(\sigma_1^{x+\sum_{i=1}^n y^i \cdot m_i}, \widetilde{g}) = e(\sigma_1, \widetilde{g}^{x+\sum_{i=1}^n y^i \cdot m_i}) = e(\sigma_1, \widetilde{X} \cdot \prod_{i=1}^n \widetilde{Y}_i^{m_i})$, which concludes the proof.

Let us now assume that σ is an output of the `Derive` algorithm for some subset $\mathcal{I}$. We have

$$\begin{aligned} e(\sigma'_1, \widetilde{X} \cdot \widetilde{\sigma}' \cdot \prod_{i \in \mathcal{I}} \widetilde{Y}_i^{m_i}) &= e(\sigma'_1, \widetilde{g}^{x+\sum_{i \in \mathcal{I}} y^i \cdot m_i} \cdot \widetilde{g}^t \cdot \prod_{j \in \overline{\mathcal{I}}} \widetilde{Y}_j^{m_j}) \\ &= e(\sigma'_1, \widetilde{g}^{t+x+\sum_{i=1}^n y^i \cdot m_i}) \\ &= e((\sigma'_1)^{x+\sum_{i=1}^n y^i \cdot m_i} \cdot (\sigma'_1)^t, \widetilde{g}) \\ &= e(\sigma'_2, \widetilde{g}) \end{aligned}$$

and

$$\begin{aligned} e(\prod_{i \in \mathcal{I}} Y_{n+1-i}^{c_i}, \widetilde{\sigma}') &= e(\prod_{i \in \mathcal{I}} Y_{n+1-i}^{c_i}, \widetilde{g}^{t+\sum_{j \in \overline{\mathcal{I}}} y^j \cdot m_j}) \\ &= e([\prod_{i \in \mathcal{I}} Y_{n+1-i}^{c_i}]^{t+\sum_{j \in \overline{\mathcal{I}}} y^j \cdot m_j}, \widetilde{g}) \\ &= e(\prod_{i \in \mathcal{I}} [Y_{n+1-i}^{t+\sum_{j \in \overline{\mathcal{I}}} y^j \cdot m_j}]^{c_i}, \widetilde{g}) \\ &= e(\prod_{i \in \mathcal{I}} [Y_{n+1-i}^t \cdot \prod_{j \in \overline{\mathcal{I}}} Y_{n+1-i+j}^{m_j}]^{c_i}, \widetilde{g}), \end{aligned}$$

which means that both equations are satisfied.

3.3 Security Analysis

By modelling `H` as a random oracle, we can prove the strong unforgeability (and hence the basic unforgeability) of our construction in the generic group model. We note that relying on a q-type assumption, as it was done in [21] for PS signatures, seems extremely difficult here as the use of several powers of the secret value y prevents to use the strategy from [21]. Fortunately, proving the unlinkability of our scheme does not require such kinds of assumption as we show that this property holds unconditionally. This is formally stated by the following theorem, proven below.

Theorem 3. – *In the random oracle and generic group models, our construction is strongly unforgeable.*
– *Our construction is unconditionally unlinkable.*

Proof of Strong Unforgeability

Lemma 4. *In the generic group model, no adversary can break the strong unforgeability of our scheme with probability greater than* $\frac{q_H+2+2n\cdot(3n+2+2q_R+4q_D+q_G)^2}{2p}$, *where* q_H *is a bound on the number of random oracle queries,* q_G *is a bound on the number of group oracle queries,* q_D *is a bound on the number of* $\mathcal{O}$`Derive` *queries and* q_R *is a bound on the number of* $\mathcal{O}$`Reveal` *queries.*

Proof. In the generic group model, the adversary $\mathcal{A}$ has only access to the elements from the public key and the ones resulting from queries to the different oracles. Each of these elements is associated with a polynomial whose formal variables are the scalars unknown to the adversary (either the values from the secret key or the random scalars r and t used to derive signatures). Concretely, the adversary has access to

- $\mathsf{pk} = (\{g^{y^i}\}_{i=0}^{n}, \{g^{y^i}\}_{i=n+2}^{2n}, \widetilde{g}^x, \{\widetilde{g}^{y^i}\}_{i=0}^{n})$;
- $(g^{r_k}, g^{r_k(x+\sum_{i=1}^{n} y^i \cdot m_{k,i})})$ obtained, for $k \in [1, q_R]$, via the $\mathcal{O}\texttt{Reveal}$ oracle on a set of messages $\{m_{k,i}\}_{i=1}^{n}$ adaptively chosen by the adversary;
- $(g^{r'_\ell}, g^{r'_\ell(x+t_\ell+\sum_{i=1}^{n} y^i \cdot m_{\ell,i})}, \widetilde{g}^{t_\ell} \cdot \prod_{j\in\overline{\mathcal{I}}_\ell} \widetilde{g}^{y^j \cdot m_{\ell,j}}, \prod_{i\in\mathcal{I}_\ell}[\prod_{j\in\overline{\mathcal{I}}_\ell}(g^{y^{n+1-i+j}\cdot m_j}) \cdot g^{y^{n+1-i}\cdot t_\ell}]^{c_{\ell,i}})$ obtained through $\mathcal{O}\texttt{Derive}$ queries, for $\ell \in [1, q_D]$, where $c_{\ell,i}$ is computed as described in Sect. 3.2, namely by hashing the first three elements of the signature concatenated with $\mathcal{I}_\ell$ and i.

From these elements, $\mathcal{A}$ must create a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ on a set of messages $\{m_i\}_{i\in\mathcal{I}}$ (we may have $\mathcal{I} = [1, n]$) that would be considered as a valid forgery by the strong unforgeability experiment. We note that we may have $(\sigma_3, \widetilde{\sigma}) = (1_{\mathbb{G}_1}, 1_{\mathbb{G}_2})$ so we can consider a four-elements forgery without loss of generality.

In this proof, we will use a register $\texttt{L}$ to handle random oracle queries. For each query x, we first check whether $\texttt{L}[x]$ already contains a scalar $y \in \mathbb{Z}_p^*$, in which case we return the latter. Else, we generate a random $y \xleftarrow{\$} \mathbb{Z}_p^*$ that is returned to $\mathcal{A}$ and then stored in $\texttt{L}[x]$.

In the generic group model, σ must have been created as a combination of the elements above. This means that there are known scalars $\{(\alpha_{1,i}, \beta_{1,i}, \gamma_{1,i})_{i\in[0,n]\cup[n+2,2n]}$, δ_2, $\{\delta_{3,i}\}_{i\in[0,n]}$, $\{(\alpha_{4,k,b}, \beta_{4,k,b}, \gamma_{4,k,b})\}_{k\in[1,q_R],b\in[1,2]}$, $\{(\alpha_{5,\ell,b}, \beta_{5,\ell,b}, \gamma_{5,\ell,b})\}_{\ell\in[1,q_R],b\in[1,3]}$ and $\{\delta_{5,\ell}\}_{\ell\in[1,q_R]}$ such that:

- $[\sigma_1] = \sum_{i\in[0,n]\cup[n+2,2n]} \alpha_{1,i} \cdot y^i + \sum_{k=1}^{q_R} \alpha_{4,k,1} \cdot r_k + \alpha_{4,k,2} \cdot r_k(x + \sum_{i=1}^{n} y^i \cdot m_{k,i}) + \sum_{\ell=1}^{q_D} (\alpha_{5,\ell,1} \cdot r'_\ell + \alpha_{5,\ell,2} \cdot r'_\ell(x + t_\ell + \sum_{i=1}^{n} y^i \cdot m_{\ell,i}) + \alpha_{5,\ell,3} \cdot \sum_{i\in\mathcal{I}_\ell} c_{\ell,i}[y^{n+1-i} \cdot t_\ell + \sum_{j\in\overline{\mathcal{I}}_\ell} y^{n+1-i+j} \cdot m_j]$
- $[\sigma_2] = \sum_{i\in[0,n]\cup[n+2,2n]} \beta_{1,i} \cdot y^i + \sum_{k=1}^{q_R} \beta_{4,k,1} \cdot r_k + \beta_{4,k,2} \cdot r_k(x + \sum_{i=1}^{n} y^i \cdot m_{k,i}) + \sum_{\ell=1}^{q_D} (\beta_{5,\ell,1} \cdot r'_\ell + \beta_{5,\ell,2} \cdot r'_\ell(x + t_\ell + \sum_{i=1}^{n} y^i \cdot m_{\ell,i}) + \beta_{5,\ell,3} \cdot \sum_{i\in\mathcal{I}_\ell} c_{\ell,i}[y^{n+1-i} \cdot t_\ell + \sum_{j\in\overline{\mathcal{I}}_\ell} y^{n+1-i+j} \cdot m_j]$

- $[\sigma_3] = \sum_{i\in[0,n]\cup[n+2,2n]} \gamma_{1,i}\cdot y^i + \sum_{k=1}^{q_R} \gamma_{4,k,1}\cdot r_k + \gamma_{4,k,2}\cdot r_k(x+\sum_{i=1}^{n} y^i\cdot m_{k,i}) + \sum_{\ell=1}^{q_D}(\gamma_{5,\ell,1}\cdot r'_\ell + \gamma_{5,\ell,2}\cdot r'_\ell(x+t_\ell+\sum_{i=1}^{n} y^i\cdot m_{\ell,i}) + \gamma_{5,\ell,3}\cdot \sum_{i\in\mathcal{I}_\ell} c_{\ell,i}[y^{n+1-i}\cdot t_\ell + \sum_{j\in\overline{\mathcal{I}}_\ell} y^{n+1-i+j}\cdot m_j]$
- $[\widetilde{\sigma}] = \delta_2\cdot x + \sum_{i=0}^{n}\delta_{3,i}\cdot y^i + \sum_{\ell=0}^{q_D}\delta_{5,\ell}(t_\ell + \sum_{j\in\overline{\mathcal{I}}_\ell} y^j\cdot m_{\ell,j})$

where $\sigma_i \leftarrow g^{[\sigma_i]}$ and $\widetilde{\sigma} \leftarrow \widetilde{g}^{[\widetilde{\sigma}]}$. As σ is expected to be a valid signature, the following two conditions must be satisfied:

1. $e(\sigma_1, \widetilde{X}\cdot\widetilde{\sigma}\cdot\prod_{i\in\mathcal{I}}\widetilde{Y}_i^{m_i}) = e(\sigma_2,\widetilde{g})$;
2. $e(\sigma_3,\widetilde{g}) = e(\prod_{i\in\mathcal{I}} Y_{n+1-i}^{c_i},\widetilde{\sigma})$;

but these are just necessary conditions. The set $\{m_i\}$ returned by the adversary must indeed also satisfy the following two conditions:

1. $\forall k\in[1,q_R]$, $\{m_i\}_{i\in\mathcal{I}} \nsubseteq \{m_{k,i}\}_{i=1}^{n}$;
2. $\forall \ell\in[1,q_D]$ either $\mathcal{I}\neq\mathcal{I}_\ell$ or $\{m_i\}_{i\in\mathcal{I}} \neq \{m_{\ell,i}\}_{i\in\mathcal{I}_\ell}$.

We here need to distinguish two cases:

- Case 1: $[\widetilde{\sigma}] = 0$;
- Case 2: $[\widetilde{\sigma}] \neq 0$.

Case 1. Here, we can only consider the first equation of the verification process and get the following relation:

$$[\sigma_1](x+\sum_{i\in\mathcal{I}} y^i\cdot m_i) = [\sigma_2]$$

As there is no monomial of degree 2 in x in $[\sigma_2]$, we can conclude that $\alpha_{4,k,2} = \alpha_{5,\ell,2} = 0\ \forall\ k\in[1,q_R]$ and $\ell\in[1,q_D]$. Similarly, there are no elements of the form $x\cdot y^u$ for $u>0$, which implies that $\alpha_{1,i}=0$, $\forall i>0$, and that $\alpha_{5,\ell,3}=0\ \forall\ell\in[1,q_D]$. We can then significantly simplify $[\sigma_1]$:

$$[\sigma_1] = \alpha_{1,0} + \sum_{k=1}^{q_R}\alpha_{4,k,1}\cdot r_k + \sum_{\ell=1}^{q_D}\alpha_{5,\ell,1}\cdot r'_\ell$$

Now, we note that any monomial of degree 1 in x in $[\sigma_2]$ also involves some r_k or r'_ℓ. We must then have $\alpha_{1,0}=0$, meaning that any monomial on the left hand side also involves some r_k or r'_ℓ. This allows us to conclude that $\beta_{1,i}=0$ $\forall i$ and that $\beta_{5,\ell,3}=0\ \forall\ell\in[1,q_D]$. Finally, the factor $(x+\sum_{i\in\mathcal{I}} y^i\cdot m_i)$ in the previous relation means that any monomial in $[\sigma_2]$ is at least of degree 1 in x or y. This means that $\beta_{4,k,1}=\beta_{5,\ell,1}=0$, $\forall\ k\in[1,q_R]$ and $\ell\in[1,q_D]$. We can then also simplify $[\sigma_2]$:

$$[\sigma_2] = \sum_{k=1}^{q_R}\beta_{4,k,2}\cdot r_k(x+\sum_{i=1}^{n} y^i\cdot m_{k,i}) + \sum_{\ell=1}^{q_D}\beta_{5,\ell,2}\cdot r'_\ell(x+t_\ell+\sum_{i=1}^{n} y^i\cdot m_{\ell,i})$$

In our case, there are no terms in t_ℓ in the left member of the equation, which means that $\beta_{5,\ell,2} = 0\ \forall \ell \in [1, q_D]$. We can therefore also remove the monomials involving r'_ℓ in $[\sigma'_1]$.

At this stage, we have:

- $[\sigma_1] = \sum_{k=1}^{q_R} \alpha_{4,k,1} \cdot r_k$;
- $[\sigma_2] = \sum_{k=1}^{q_R} \beta_{4,k,2} \cdot r_k(x + \sum_{i=1}^{n} y^i \cdot m_{k,i})$.

This implies that $\alpha_{4,k,1} = \beta_{4,k,2}$, $\forall k \in [1, q_R]$. Moreover, for any k such that $\alpha_{4,k,1} \neq 0$, we must have $m_i = m_{k,i}\ \forall i \in \mathcal{I}$, in which case σ would not be a valid forgery. The only other possibility is $\alpha_{4,k,1} = 0\ \forall k \in [1, q_R]$ but, here again, this would mean that σ is not a valid forgery. Hence, no adversary can output a valid forgery in Case 1.

Case 2. Let us now consider the second case, and more specifically the second equation of the verification process (which is not trivial in this case):

$$[\sigma_3] = [\widetilde{\sigma}](\sum_{i \in \mathcal{I}} c_i \cdot y^{n+1-i})$$

where $c_i = \mathtt{H}(\sigma_1 || \sigma_2 || \widetilde{\sigma} || \mathcal{I} || i)$. As there are no terms in $x \cdot y^u$, for $u > 0$, in $[\sigma_3]$, we can conclude that $\delta_2 = 0$. This means that the coefficients of all monomials involving x in $[\sigma_3]$ are zero: $\gamma_{4,k,2} = \gamma_{5,\ell,2} = 0,\ \forall\ k \in [1, q_R]$ and $\ell \in [1, q_D]$. Similarly, we note that we can remove the monomials involving r_k or r'_ℓ as there are not present in the right member. We thus get:

- $[\sigma_3] = \sum_{i \in [0,n] \cup [n+2, 2n]} \gamma_{1,i} \cdot y^i + \sum_{\ell=1}^{q_D} \gamma_{5,\ell,3} \cdot \sum_{i \in \mathcal{I}_\ell} c_{\ell,i} [y^{n+1-i} \cdot t_\ell + \sum_{j \in \overline{\mathcal{I}}_\ell} y^{n+1-i+j} \cdot m_j]$;
- $[\widetilde{\sigma}] = \sum_{i=0}^{n} \delta_{3,i} \cdot y^i + \sum_{\ell=0}^{q_D} \delta_{5,\ell}(t_\ell + \sum_{j \in \overline{\mathcal{I}}_\ell} y^j \cdot m_{\ell,j})$.

Let us define, for all $i \in [0, n]$, $\delta'_{3,i} = \delta_{3,i} + \sum_{\ell \in [1, q_D] : i \in \overline{\mathcal{I}}_\ell} \delta_{5,\ell} \cdot m_{\ell,i}$. Then, $[\widetilde{\sigma}]$ can be written as follows:

$$[\widetilde{\sigma}] = \sum_{i=0}^{n} \delta'_{3,i} \cdot y^i + \sum_{\ell=0}^{q_D} \delta_{5,\ell} \cdot t_\ell.$$

We note that the coefficient of the monomial y^{n+1} of $[\widetilde{\sigma}](\sum_{i \in \mathcal{I}} c_i \cdot y^{n+1-i})$ is exactly $\sum_{i \in \mathcal{I}} c_i \cdot \delta'_{3,i}$. As there is no monomial of degree $n+1$ in $[\sigma_3]$, we know that this sum is necessarily 0 but we need to show that this can be true only if $\delta'_{3,i} = 0\ \forall i \in \mathcal{I}$. This will be done at the end of this proof where we will show that the event $\sum_{i \in \mathcal{I}} c_i \cdot \delta'_{3,i} = 0$ and $\exists j \in \mathcal{I}$ such that $\delta'_{3,j} \neq 0$ is very unlikely.

Right now, we assume this result and so consider that $\delta'_{3,i} = 0$, $\forall i \in \mathcal{I}$:

$$[\widetilde{\sigma}] = \sum_{i \in [0,n] \setminus \mathcal{I}} \delta'_{3,i} \cdot y^i + \sum_{\ell=0}^{q_D} \delta_{5,\ell} \cdot t_\ell$$

We can now focus on the first equation of verification:

$$[\sigma_1](x + [\widetilde{\sigma}] + \sum_{i \in \mathcal{I}} y^i \cdot m_i) = [\sigma_2]$$

Although the relation is not the same as in Case 1 because $[\widetilde{\sigma}]$ is not necessarily 0, some parts of the previous analysis are still relevant. In particular, for the same reasons as above, we have:

- $\alpha_{4,k,2} = \alpha_{5,\ell,2} = \alpha_{5,\ell,3} = 0$, $\forall\ k \in [1, q_R]$ and $\ell \in [1, q_D]$;
- $\alpha_{1,i} = 0$, $\forall i \geq 0$;
- $\beta_{1,i} = 0\ \forall i$ and $\beta_{5,\ell,3} = 0\ \forall \ell \in [1, q_D]$.

We can then simplify both σ_1 and σ_2 as follows:

- $[\sigma_1] = \sum_{k=1}^{q_R} \alpha_{4,k,1} \cdot r_k + \sum_{\ell=1}^{q_D} \alpha_{5,\ell,1} \cdot r'_\ell$
- $[\sigma_2] = \sum_{k=1}^{q_R} (\beta_{4,k,1} \cdot r_k + \beta_{4,k,2} \cdot r_k(x + \sum_{i=1}^{n} y^i \cdot m_{k,i})) + \sum_{\ell=1}^{q_D} (\beta_{5,\ell,1} \cdot r'_\ell + \beta_{5,\ell,2} \cdot r'_\ell(x + t_\ell + \sum_{i=1}^{n} y^i \cdot m_{\ell,i}))$

We must now distinguish two subcases:

- Case 2.1: $\exists \ell \in [1, q_D]$ such that $\delta_{5,\ell} \neq 0$.
- Case 2.2: $\forall \ell \in [1, q_D], \delta_{5,\ell} = 0$.

In the first case, we have $\alpha_{4,k,1} = 0$ for all $k \in [1, q_R]$ as there are no terms in $r_k \cdot t_\ell$ in $[\sigma_2]$. We must therefore also have $\beta_{4,k,1} = \beta_{4,k,2} = 0$, $\forall k \in [1, q_R]$. This means that, $\forall \ell \in [1, q_D]$,

$$\alpha_{5,\ell,1} \cdot r'_\ell(x + \sum_{i \in [0,n] \setminus \mathcal{I}} \delta'_{3,i} \cdot y^i + \delta_{5,\ell} \cdot t_\ell + \sum_{i \in \mathcal{I}} y^i \cdot m_i) = \beta_{5,\ell,1} \cdot r'_\ell + \beta_{5,\ell,2} \cdot r'_\ell(x + t_\ell + \sum_{i=1}^{n} y^i \cdot m_{\ell,i})$$

which implies that $\alpha_{5,\ell,1} = \beta_{5,\ell,2}$. Hence, $m_{\ell,i} = m_i$, $\forall i \in \mathcal{I}$, and $\delta_{5,\ell} = 1$ (assuming that $\alpha_{5,\ell,1} \neq 0$ as this is trivial otherwise). σ can then be a valid forgery only if $\mathcal{I} \neq \mathcal{I}_\ell$, for such a ℓ. However, we recall that $[\sigma_3]$ and $[\widetilde{\sigma}]$ must satisfy:

$$[\sigma_3] = [\widetilde{\sigma}](\sum_{i \in \mathcal{I}} c_i \cdot y^{n+1-i})$$

On the right hand side, the monomials involving t_ℓ are exactly $c_i \cdot y^{n+1-i} \cdot t_\ell$, for $i \in \mathcal{I}$. On the left hand side, they are $\gamma_{5,\ell,3} \cdot c_{\ell,i} \cdot y^{n+1-i} \cdot t_\ell$ for $i \in \mathcal{I}_\ell$. Since $\mathtt{H}$ returns non-zero scalars, this equation can be satisfied only if $\mathcal{I} \subset \mathcal{I}_\ell$. However, since $\mathcal{I} \neq \mathcal{I}_\ell$, we know that there is at least one index i^* such that $i^* \in \mathcal{I}_\ell$ and $i^* \notin \mathcal{I}$. For this index, the monomial $\gamma_{5,\ell,3} \cdot c_{\ell,i^*} \cdot y^{n+1-i^*} \cdot t_\ell$ has no counterpart in the right member of the equation, which means that $\gamma_{5,\ell,3} = 0$

and therefore that $\beta_{5,\ell,2} = 0$. In case 2.1, the adversary can then succeed only by using $[\sigma_1] = 0$, which makes the forgery σ invalid.

Let us now consider the case 2.2. Here, the situation is simpler as we have:

- $[\sigma_1] = \sum_{k=1}^{q_R} \alpha_{4,k,1} \cdot r_k + \sum_{\ell=1}^{q_D} \alpha_{5,\ell,1} \cdot r'_\ell$;
- $[\sigma_2] = \sum_{k=1}^{q_R} (\beta_{4,k,1} \cdot r_k + \beta_{4,k,2} \cdot r_k(x + \sum_{i=1}^{n} y^i \cdot m_{k,i})) + \sum_{\ell=1}^{q_D} \beta_{5,\ell,1} \cdot r'_\ell$;
- $[\widetilde{\sigma}] = \sum_{i \in [0,n] \setminus \mathcal{I}} \delta'_{3,i} \cdot y^i$.

From $[\sigma_1](x + [\widetilde{\sigma}] + \sum_{i \in \mathcal{I}} y^i \cdot m_i) = [\sigma_2]$, we can deduce that $\alpha_{4,k,1} = \beta_{4,k,2}$, $\forall k \in [1, q_R]$. Therefore, either $m_{k,i} = m_i$ for all $i \in \mathcal{I}$ or $\alpha_{4,k,1} = 0$. The former case means that the forgery is invalid so we can assume that $\alpha_{4,k,1} = \beta_{4,k,2} = 0$, $\forall k \in [1, q_R]$. There are then no longer terms involving x in $[\sigma_2]$, which implies that $[\sigma_1]$ must be zero and so that σ is also invalid.

We now need to bound the probability of some bad events that would make our simulation in the generic group model incorrect. The first one is the event where the adversary returns a forgery $(\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ for a set $\mathcal{I}$, where $\widetilde{\sigma}$ was generated using some scalars $\delta'_{3,i}$, as described above, such that

1. $\exists j \in \mathcal{I}$ such that $\delta'_{3,j} \neq 0$;
2. $\sum_{i \in \mathcal{I}} c_i \cdot \delta'_{3,i} = 0$, with $c_i \leftarrow \mathtt{H}(\sigma_1 || \sigma_2 || \widetilde{\sigma} || \mathcal{I} || i) \in \mathbb{Z}_p^*$.

We distinguish two cases. Either $\mathcal{A}$ has queried $\sigma_1 || \sigma_2 || \widetilde{\sigma} || \mathcal{I} || i$, $\forall i$ such that $\delta'_{3,i} \neq 0$, or there is at least one such index for which the corresponding string has not been queried to the random oracle before $\mathcal{A}$ outputs its forgery. In the second case, at least one random $c_i \in \mathbb{Z}_p^*$ is generated after the adversary has returned the forgery σ, which means that the second condition can only be satisfied with probability at most $\frac{1}{p}$.

So let us focus on the first case. In the generic group model, each set of values $\{\delta'_{3,i}\}_{i \in \mathcal{I}}$ generates a different element $\widetilde{\sigma}$. As $\widetilde{\sigma}$ is taken as input by the random oracle $\mathtt{H}$ to generate the random, non-zero scalars c_i, this means that $\{\delta'_{3,i}\}_{i \in \mathcal{I}}$ is distributed independently of $\{c_i\}_{i \in \mathcal{I}}$. The probability that $\sum_{i \in \mathcal{I}} c_i \cdot \delta'_{3,i} = 0$ with some non-zero $\delta'_{3,i}$ is then at best $\frac{1}{p}$ for a given set of values $\{\delta'_{3,i}\}_{i \in \mathcal{I}}$. Now, we note that the sum $\sum_{i \in \mathcal{I}} c_i \cdot \delta'_{3,i}$ needs at least two non-zero terms to be equal to zero because $c_i \neq 0$. This means that any tentative by the adversary to satisfy both conditions consume at least two queries to the random oracle. We can therefore bound the probability of $\mathcal{A}$ succeeding in this second case by $\frac{q_H}{2p}$, which is negligible as long as $\mathcal{A}$ makes only a polynomial number of queries to the random oracle. In the end, the probability that this bad event occurs is at most $\frac{q_H+2}{2p}$.

The second event we need to bound is the one where two of the formal polynomials used above evaluate to the same value. Indeed, the two elements associated with these polynomials would be equal in practice but would be considered as

different by our simulation. The number of polynomials involved in our proof is $3n + 2 + 2q_R + 4q_D + q_G$, each of them being of degree at most $2n$. Using the Schwartz-Zippel lemma, we can bound the probability of such an event by $\frac{2n\cdot(3n+2+2q_R+4q_D+q_G)^2}{2p}$, which is negligible.

Proof of Unlinkability. We here prove that a derived signature $\sigma_{\mathcal{I}}$ on $\{m_i\}_{i\in\mathcal{I}}$ is distributed independently of the original signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ and of the set of redacted messages $\{m_i\}_{i\in\overline{\mathcal{I}}}$.

Let $h = g^v$ be some random element of $\mathbb{G}_1$, t be a random scalar and s be such that $\sigma_1 = g^s$. We can then define $u = t + \sum_{j\in\overline{\mathcal{I}}} y^j \cdot m_j$ along with:

- $\sigma'_1 \leftarrow h$;
- $\sigma'_2 \leftarrow h^{x+\sum_{i\in\mathcal{I}} y^i\cdot m_i} \cdot h^u$;
- $\sigma'_3 \leftarrow [\prod_{i\in\mathcal{I}} Y^{c_i}_{n+1-i}]^u$;
- $\widetilde{\sigma}' \leftarrow \widetilde{g}^u$.

We note that:

$$(\prod_{i\in\mathcal{I}} Y^{c_i}_{n+1-i})^u = \prod_{i\in\mathcal{I}} (Y^u_{n+1-i})^{c_i} = \prod_{i\in\mathcal{I}} [Y^t_{n+1-i} \cdot \prod_{j\in\overline{\mathcal{I}}} Y^{m_j}_{n+1-i+j}]^{c_i}.$$

Therefore $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3, \widetilde{\sigma}')$ is exactly the derived signature that one would get by running the `Derive` algorithm with scalars t and $r = \frac{v}{s}$. Moreover, σ' is distributed as a valid output of this algorithm since both t and r are random.

Now, we note that the scalar u is random because t is random, so the four elements of $\sigma'_{\mathcal{I}}$ are distributed independently of σ and $\{m_i\}_{i\in\overline{\mathcal{I}}}$, which concludes the proof.

4 Group Signature with Time-Bound Keys

In this section, we recall and extend the definition of a group signature scheme with time-bound keys from [13]. There are three main differences with the latter paper. Firstly, as we explain in the introduction, we associate each signing key usk_k with a set of *active* time periods $\mathcal{T}_k$ and not just an expiry time. This means that the user k can issue valid group signatures for all time periods $t \in \mathcal{T}_k$, which are not necessarily contiguous. Concretely, the user can be considered as revoked at some time period $t \notin \mathcal{T}_k$ and then be automatically reinstated at a later period $t' \in \mathcal{T}_k$. This definition clearly generalizes the previous ones [11,13] that only consider expiry time. Our second difference is that we allow $\mathcal{O}$`Open` queries for the adversary during the anonymity game. Such queries were indeed forbidden in [13], resulting in a weaker notion of anonymity. Finally, our group manager does not need to provide the so-called "expiration information" for each period, which simplifies both the management process and the signature algorithm. Our group members indeed only need to know the current time period (and their original signing key) to issue a signature and in particular do not need to update their signing key with such information at each time period.

4.1 Syntax

As in [13], our group signature is composed of the following algorithms, involving three types of entities: the group manager, the users and the verifiers.

- `GKeygen`: on input a security parameter 1^λ and a bound n on the number of time periods, this algorithms generates the group key pair $(\mathsf{gsk}, \mathsf{gpk})$ and initializes a register `Reg`. The group public key `gpk` is then considered as known by all the other parties.
- `Join`: this is a two-party interactive protocol between the group manager and a new group member k. Both of them take as input `gpk` along with the set of active time periods $\mathcal{T}_k$[2] for this user. The group manager additionally takes as input `gsk` along with the current version of `Reg`. After successful completion of this protocol, the user obtains his group signing key usk_k that contains in particular his secret key sk_k and $\mathcal{T}_k$ whereas the group manager stores a revocation token rt_k and $\mathcal{T}_k$ in $\mathtt{Reg}[k]$.
- `Sign`: on input a group signing key usk_k, the group public key `gpk`, the current time period t and a message m, this algorithm returns a group signature σ.
- `Revoke`: on input $(\mathsf{gsk}, \mathsf{gpk})$, `Reg`, a time period t and a set $\mathcal{R}$ of users to revoke, this algorithm returns a revocation list $\mathtt{RL}_t$ specific to this period.
- `Verify`: on input `gpk`, a time period t, a revocation list $\mathtt{RL}_t$, a group signature σ and a message m, this algorithms returns either 1 (valid) or 0 (invalid).
- `Open`: on input `gsk`, `Reg`, a group signature σ, a message m and a period t, this algorithm returns either a user identifier k or a failure message $\perp$.

Remark 5. Previous works [11,13] distinguish two kinds of revocation, the natural revocation that automatically excludes the users from the group once their expiry time has passed and the premature revocation that is called to exclude users before their expiry time. This terminology is coherent with their setting but not with ours where each user may alternate between periods of activity and inactivity. We then rather consider, for each time period t, a set of *active* users (*i.e.* those such that $t \in \mathcal{T}_k$) and a set of inactive ones. A user k is then said to be *revoked* at period t only if the `Revoke` algorithm has been run on a set $\mathcal{R} \ni k$ and t, which corresponds to the premature revocation in [11,13]. In particular, the revocation list $\mathtt{RL}_t$ is independent of the set of inactive users.

4.2 Security Model

A group signature should achieve correctness, anonymity, traceability and non-frameability as defined below. Our definitions are adapted from [13] and include the differences discussed at the beginning of this section. We refer to [13] for a formal definition of correctness but it intuitively requires that an honestly generated signature, issued by an active and non-revoked group member, is considered as valid by the verification algorithm and can be opened by the group manager.

[2] We associate each time period with an integer t in $[1, n]$. We can then talk of "time period t" without loss of generality.

As in [13], we ensure anonymity of users as long as their secret key do not leak. This is quite standard for group signature with revocation lists as the revocation tokens for a user can usually be generated from his secret key. This therefore corresponds to the notion of selfless-anonymity [6]. We nevertheless consider an anonymity definition that extends the notion of backward unlinkability to our setting. That is, we even allow revocation of challenge users (those targeted by the adversary) at any time period t different from the one of issuance of the challenge group signature. We thus ensure both backward and forward unlinkability.

Our traceability and non-frameability notions are standard. Intuitively, the former ensures that any group signature valid for a period t can be linked back to a group member k *active* at this time period whereas the latter ensures that k has indeed taken part in the generation of this signature.

All these security definitions make use of the following oracles that define $\mathcal{H}$ (resp. $\mathcal{C}$) as the set of honest (resp. corrupt) users. As the corrupt entities differ according to the experiments, we consider several oracles to add new members to the groups. All of them associate a group member to a unique index k and then return $\perp$ if they are run on an existing k.

- $\mathcal{O}\mathtt{Add}(k, \mathcal{T}_k)$ is an oracle that can be used to add a new honest user k for the set of time periods $\mathcal{T}_k$. It runs `Join` with the corresponding inputs, thus generating a group signing key along with the associated revocation tokens. It does not return any data but adds k to $\mathcal{H}$.
- $\mathcal{O}\mathtt{J}_U(k, \mathcal{T}_k)$ is an oracle that plays the user's side of the `Join` protocol. It can be used by an adversary $\mathcal{A}$ playing the role of a corrupt group manager to add a new user k for the time periods $\mathcal{T}_k$.
- $\mathcal{O}\mathtt{J}_{GM}(k, \mathcal{T}_k)$ is the counterpart of the $\mathcal{O}\mathtt{J}_U$ oracle that can be used by a corrupt user to join the group. k is then added to $\mathcal{C}$.
- $\mathcal{O}\mathtt{Cor}(k)$ is an oracle that returns the group signing key of an honest user k. k is then removed from $\mathcal{H}$ and added to $\mathcal{C}$.
- $\mathcal{O}\mathtt{Sign}(i, m, t)$ is an oracle that returns $\mathtt{Sign}(\mathsf{gpk}, \mathsf{usk}_i, m, t)$, provided that k is an honest user that has already joined the group.
- $\mathcal{O}\mathtt{Open}(\sigma, m, t)$ is an oracle that returns $\mathtt{Open}(\mathsf{gsk}, \mathsf{gpk}, \mathtt{Reg}, \sigma, m, t)$.
- $\mathcal{O}\mathtt{Revoke}(\mathcal{R}, t)$ is an oracle that returns $\mathtt{Revoke}(\mathsf{gsk}, \mathsf{gpk}, \mathtt{Reg}, \mathcal{R}, t)$. The adversary may adaptively run several times this algorithm for the same time period t.
- $\mathcal{O}\mathtt{Ch}_b(i_0, i_1, m, t)$ is an oracle that takes as inputs the index of two honest users active at the time period t and returns $\mathtt{Sign}(\mathsf{gpk}, \mathsf{usk}_{i_b}, m, t)$.

In both the traceability and the non-frameability experiments, by including `Reg` in the adversary input we mean that it has read access to this register. We could also provide write access but this would require a more intricate `Join` protocol where the new user would sign the entry of the register with a specific key, as in [3] for example. We believe this would unnecessarily complicate our security model and so only consider this simpler scenario in our paper. Should the need arise, it would be straightforward to upgrade our model and our construction to handle write access.

$\mathtt{Exp}^{an}_{\mathcal{A}}(1^\lambda)$ – Anonymity Security Game
1. $b \xleftarrow{\$} \{0,1\}$
2. $(\mathsf{gsk}, \mathsf{gpk}, \mathtt{Reg}) \leftarrow \mathtt{GKeygen}()$
3. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Sign},\mathcal{O}\mathtt{Open},\mathcal{O}\mathtt{Revoke},\mathcal{O}\mathtt{Ch}_b}(\mathsf{gpk})$
4. If $\mathcal{O}\mathtt{Open}$ is queried on the output of $\mathcal{O}\mathtt{Ch}_b$, then return 0
5. If $\mathcal{O}\mathtt{Cor}$ has been run on an input of $\mathcal{O}\mathtt{Ch}_b$, then return 0
6. If both $\mathcal{O}\mathtt{Ch}_b(k_0, k_1, m, t)$ and $\mathcal{O}\mathtt{Revoke}(\mathcal{R}, t)$ are queried, with $\{k_0, k_1\} \cap \mathcal{R} \neq \emptyset$, then return 0
7. Return $(b = b^*)$

$\mathtt{Exp}^{tra}_{\mathcal{A}}(1^\lambda)$ – Traceability Security Game
1. $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathtt{GKeygen}()$
2. $(\sigma, m, t) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{J}_{GM},\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Open},\mathcal{O}\mathtt{Revoke},\mathcal{O}\mathtt{Sign}}(\mathsf{gpk}, \mathtt{Reg})$
3. If $0 \leftarrow \mathtt{Verify}(\mathsf{gpk}, \mathtt{RL}_t, \sigma, m, t)$, then return 0
4. If $\perp \leftarrow \mathtt{Open}(\mathsf{gsk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m, t)$, then return 1
5. If $k \leftarrow \mathtt{Open}(\mathsf{gsk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m, t)$ and $t \notin \mathcal{T}_k$, then return 1
6. Return 0

$\mathtt{Exp}^{nf}_{\mathcal{A}}(1^\lambda)$ – Non-Frameability Security Game
1. $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathtt{GKeygen}(pp)$
2. $(\sigma, m, t) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{J}_U,\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Open},\mathcal{O}\mathtt{Revoke},\mathcal{O}\mathtt{Sign}}(\mathsf{gsk}, \mathtt{Reg})$
3. $k \leftarrow \mathtt{Open}(\mathsf{gsk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m, t)$
4. If $0 \leftarrow \mathtt{Verify}(\mathsf{gpk}, \mathtt{RL}_t, \sigma, m, t)$, then return 0
5. If $\mathcal{O}\mathtt{Sign}$ returned σ, then return 0
6. If $k \notin \mathcal{H}$, then return 0
7. Return 1

Fig. 2. Security games for group signature

Let $\mathcal{A}$ be a probabilistic polynomial adversary. A group signature scheme with time-bound keys is (Fig. 2)

- anonymous if $\mathtt{Adv}^{an}(\mathcal{A}) = |\Pr[\mathtt{Exp}^{an}_{\mathcal{A}}(1^\lambda) = 1] - 1/2|$ is negligible for any $\mathcal{A}$;
- traceable if $\mathtt{Adv}^{tra}(\mathcal{A}) = \Pr[\mathtt{Exp}^{tra}_{\mathcal{A}}(1^\lambda) = 1]$ is negligible for any $\mathcal{A}$;
- non-frameable if $\mathtt{Adv}^{nf}(\mathcal{A}) = \Pr[\mathtt{Exp}^{nf}_{\mathcal{A}}(1^\lambda) = 1]$ is negligible for any $\mathcal{A}$.

5 Instantiation from Redactable Signature

5.1 Our Construction

Intuition. The core idea behind our construction is rather simple once we have identified redactable signature as a suitable building block for group signature with time-bound keys. However, there are still some issues to address to get a scheme achieving the strong security properties of Sect. 4.

Indeed, the core idea is that any user k joining the group for a set of time periods $\mathcal{T}_k$ receives a redactable signature on a set of n messages $\{m_i\}_{i=1}^n$ where $m_i = 0$ if $i \notin \mathcal{T}_k$. This leads to a very compact group signing key that essentially consists of this signature. To issue a group signature at a time period i, a member

only has to run the $\mathtt{Derive}$ algorithm on the subset $\mathcal{I} = \{i\}$ to get a redacted signature σ'. The latter can be straightforwardly used by the verifier to check if this member is active at this time period, as this signature would be valid on "0" otherwise. To achieve non-frameability while allowing revocation, it might be tempting to follow [22] and simply add a signature on a secret value sk_k. As in [20] the knowledge of $\widetilde{g}^{\mathsf{sk}_k}$ would then be enough to revoke (or, alternatively, open) signatures issued by k. Unfortunately, this would prevent revocation for specific time periods and in particular backward and forward unlinkability. We therefore proceed differently and define $m_i = \mathsf{sk}_k$ for any $i \in \mathcal{T}_k$. This way we can revoke a user for any time period i (and only for this time period) by simply providing a way to test whether the redacted signature σ' (derived on $\{i\}$) is valid on sk_k.

The Scheme. Our construction heavily relies on the redactable signature scheme described in Sect. 3.2 that we refer to as Σ.

- $\mathtt{GKeygen}(1^\lambda, n)$: This algorithms runs $\Sigma.\mathtt{Keygen}(1^\lambda, n)$ to get a pair (x, y) along with $\mathsf{pk} = (\mathtt{H}, g, \widetilde{g}, \{Y_i\}_{i=1}^{n}, \{Y_i\}_{i=n+2}^{2n}, \widetilde{X}, \{\widetilde{Y}_i\}_{i=1}^{n})$. It then sets $\mathsf{gsk} = (x, y)$ and $\mathsf{gpk} = \mathsf{pk}$, and initializes a register $\mathtt{Reg}$.
- $\mathtt{Join}$: To join the group for the set of time periods $\mathcal{T}_k$, a new user k first selects a random secret key $\mathsf{sk}_k \xleftarrow{\$} \mathbb{Z}_p$ and sends $(g^{\mathsf{sk}_k}, \widetilde{g}^{\mathsf{sk}_k})$ to the group manager. He then proves knowledge of sk_k using for example the Schnorr's algorithm [24]. If the proof is valid and if $e(g^{\mathsf{sk}_k}, \widetilde{g}) = e(g, \widetilde{g}^{\mathsf{sk}_k})$, it selects a random scalar r and returns $(\sigma_1, \sigma_2) \leftarrow (g^r, [g^x \cdot (g^{\mathsf{sk}_k})^{\sum_{j \in \mathcal{T}_k} y^j}]^r)$. Note that this is a valid redactable signature on a set $\{m_i\}_{i=1}^{n}$, with $m_i = \mathsf{sk}_k$ if $i \in \mathcal{T}_k$ and 0 otherwise. The user is then able to check the validity of (σ_1, σ_2) by running the $\Sigma.\mathtt{Verify}$ algorithm. It then sets usk_k as $\{\mathsf{sk}_k, (\sigma_1, \sigma_2), \mathcal{T}_k\}$. In the meantime, the group manager stores $\widetilde{g}^{\mathsf{sk}_k}$ and $\mathcal{T}_k$ in $\mathtt{Reg}[k]$.
- $\mathtt{Sign}$: to sign a message m for the current time period t, the user runs $\Sigma.\mathtt{Derive}(\mathsf{pk}, (\sigma_1, \sigma_2), \{m_i\}_{i=1}^{n}, \{t\})$ with $\{m_i\}_{i=1}^{n}$ defined as above. It then gets a derived $\sigma_{\mathcal{I}} = (\sigma'_1, \sigma'_2, \sigma'_3, \widetilde{\sigma}')$ for $\mathcal{I} = \{t\}$ and must now prove that it is valid on a message $m_t = \mathsf{sk}_k \neq 0$. As the second equation defined in $\Sigma.\mathtt{Verify}$ can be tested with the sole knowledge of $\sigma_{\mathcal{I}}$, this means that he must simply prove knowledge of sk_k such that:

$$e(\sigma_1, \widetilde{X} \cdot \widetilde{\sigma} \cdot \widetilde{Y}_t^{\mathsf{sk}_k}) = e(\sigma_2, \widetilde{g})$$

  Concretely, it generates $a \xleftarrow{\$} Z_p$ and computes $K = e(\sigma_1, \widetilde{Y}_t)^a$ along with $c \leftarrow \mathtt{H}(K, \sigma_{\mathcal{I}}, m)$ and $s = a + c \cdot \mathsf{sk}_k$. It then outputs the group signature $\sigma \leftarrow (\sigma_{\mathcal{I}}, c, s)$.
- $\mathtt{Revoke}$: For each user k to revoke for the time period t, the group manager recovers $\widetilde{g}^{\mathsf{sk}_k}$ from $\mathtt{Reg}[k]$ and adds $(\widetilde{g}^{\mathsf{sk}_k})^{y^t}$ to $\mathtt{RL}_t$.
- $\mathtt{Verify}$: To verify a group signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma}, c, s)$ on a message m for a time period t, the verifier first checks whether the second equation of $\Sigma.\mathtt{Verify}$ is satisfied. That is, $(\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ must verify

$$e(\sigma_3, \widetilde{g}) = e(Y_{n+1-t}^{c_t}, \widetilde{\sigma})$$

where $c_t \leftarrow \mathtt{H}(\sigma_1||\sigma_2||\widetilde{\sigma}||\{t\}||t)$. It then checks whether $m_t = 0$ by testing if the following equation holds:

$$e(\sigma_1, \widetilde{X} \cdot \widetilde{\sigma}) = e(\sigma_2, \widetilde{g})$$

in which case it returns 0. Else, it verifies the proof of knowledge by computing $K' \leftarrow e(\sigma_1, \widetilde{Y}_t)^s \cdot [e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-1}, \widetilde{X} \cdot \widetilde{\sigma})]^{-c}$ and checking if $c = \mathtt{H}(K', \sigma_{\mathcal{I}}, m)$. If the proof is correct, then the signature is valid but it could have been generated with a revoked key.
For each element $\widetilde{h}_k$ in $\mathtt{RL}_t$, this algorithm then tests whether $e(\sigma_1, \widetilde{h}_k) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-1}, \widetilde{X} \cdot \widetilde{\sigma})$. If one of these conditions is satisfied, the algorithm returns 0. Else it returns 1.

– `Open`: For each active user k at time period t, this algorithm recovers $\widetilde{g}^{\mathsf{sk}_k}$ from $\mathtt{Reg}[k]$ and tests whether $e(\sigma_1, \widetilde{g}^{\mathsf{sk}_k}) = [e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-1}, \widetilde{X} \cdot \widetilde{\sigma})]^{y^{-t}}$ until it gets a match, in which case it returns the corresponding identifier k.

Remark 6. As the size of the group signature is the usual benchmark in the literature, we chose to present the version of our construction that optimizes it. However, this version requires to perform operations in $\mathbb{G}_T$ to generate the NIZK in the group signature. This may be a problem if one favours computational complexity as operations in $\mathbb{G}_T$ are notoriously less efficient than their counterparts in $\mathbb{G}_1$. In such a case, one can use a standard trick (*e.g.* [2,4]) which consists in adding the element $\sigma_1^{\mathsf{sk}_k} \in \mathbb{G}_1$ to the group signature and then proving knowledge of sk_k directly in $\mathbb{G}_1$. This shifts all computations from $\mathbb{G}_T$ to $\mathbb{G}_1$, improving efficiency at the cost of a small increase of the group signature size. The security proofs can be straightforwardly adapted to this new version.

Remark 7. We note that we are in a very favourable case when evaluating the Σ.`Derive` algorithm in the `Sign` protocol. Indeed, we have $|\mathcal{I}| = |\{t\}| = 1$ and all the involved messages are either 0 or sk_k. Computing $\widetilde{\sigma}$ thus only requires two exponentiations in $\mathbb{G}_2$ as we have $\widetilde{\sigma} = \widetilde{g}^u \cdot (\prod_{j \in \mathcal{T}_k \setminus \{t\}} \widetilde{Y}_j)^{\mathsf{sk}_k}$. Moreover, the latter product $P_t = \prod_{j \in \mathcal{T}_k \setminus \{t\}} \widetilde{Y}_j$ can be efficiently updated from one active time period i to the next one i' as $P_{i'} = P_i \cdot \widetilde{Y}_i \cdot \widetilde{Y}_{i'}^{-1}$. For the same reasons, σ_3 can be computed with essentially 2 exponentiations in $\mathbb{G}_1$. This complexity is therefore much lower than the generic one of our `Derive` algorithm. We provide more details in the next section.

5.2 Security Analysis

The security of our group signature scheme is stated by the next theorem, proved below.

Theorem 8. *In the random oracle model, our group signature is*

– *non-frameable under the* SDL *assumption;*
– *anonymous under the* EDDH *assumption and the non-frameability of our construction;*
– *traceable under the unforgeability of the redactable signature scheme* Σ.

Proof of Anonymity. The unlinkability property of a redactable signature scheme Σ implies that the set of redacted messages $(\{m_i\}_{i\in\overline{\mathcal{I}}})$ is perfectly hidden but does not provide any guarantee for the messages in $\{m_i\}_{i\in\mathcal{I}}$. In each of our group signatures, the latter set is exactly $\{\mathsf{sk}_k\}$, where sk_k is the group member's secret key. We can't therefore only rely on the unlinkability of Σ to prove anonymity and will then build a more intricate proof.

Game 1. Our first game is exactly the anonymity experiment where the adversary $\mathcal{A}$ is expected to win with probability ϵ.

Game 2. In our second game, we proceed as usual except that our reduction $\mathcal{R}$ makes a guess on the time period t^* targeted by the adversary. If $\mathcal{O}\mathtt{Ch}_b$ is run on a different time period, then $\mathcal{R}$ aborts. The new success probability is then at least $\frac{\epsilon}{n}$.

Game 3. In this game, the reduction $\mathcal{R}$ now makes a guess on the user k^* targeted by the adversary. If $\mathcal{O}\mathtt{Ch}_b$ is run with $k_b \neq k^*$, then $\mathcal{R}$ aborts. The new success probability is then at least $\frac{\epsilon}{n\cdot q}$, where q is a bound on the number of group members.

Game 4. In this game, $\mathcal{R}$ proceeds as in the previous game except that it stores all the signatures issued on behalf of k^* by $\mathcal{O}\mathtt{Sign}$ in a specific list $\mathtt{L}$ and deletes the information contained in $\mathtt{Reg}[k^*]$. Upon receiving an $\mathcal{O}\mathtt{Open}$ query, it first proceeds as usual but then tests whether the signature to open belongs to $\mathtt{L}$. In such a case, it returns k^*. Else, it returns $\perp$.

Game 5. In this last game, $\mathcal{R}$ generates two random elements σ_1 and σ_2 and defines $\widetilde{\sigma} \leftarrow \widetilde{g}^s$ and $\sigma_3 \leftarrow Y_{n+1-t^*}^{s\cdot c_i}$, where $c_i \leftarrow \mathtt{H}(\sigma_1'||\sigma_2'||\widetilde{\sigma}'||\mathcal{I}||t^*)$ and s is random. It then returns $(\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ along with a simulated proof (c, s') of validity when $\mathcal{A}$ queries the oracle $\mathcal{O}\mathtt{Ch}_b$ for the time period t^*. As this group signature is perfectly independent of the users targeted by $\mathcal{A}$, the latter can only succeed with probability negligibly close to $\frac{1}{2}$.

Game 3 → Game 4. The only difference between Game 3 and Game 4 is the opening of signatures issued on behalf of k^*. As we keep track of all signatures returned by $\mathcal{O}\mathtt{Sign}$, there can be a problem only if the adversary manages to forge a valid group signature which can be traced back to k^*. In such a case our reduction will fail to correctly simulate the opening process as it will return $\perp$ instead of k^*.

However, such an adversary can be trivially converted into an adversary against non-frameability. Indeed, the latter property provides at least the same oracle queries and inputs than in the anonymity experiment. If we denote by ϵ_4 the advantage of the adversary in Game 4, we then get $\frac{\epsilon}{n\cdot q} \leq \mathtt{Adv}^{nf}(\mathcal{A}) + \epsilon_4$.

Game 4 → Game 5. Let us consider a EDDH instance $(\{g^{a\cdot c^i}\}_{i=0}^{2n^2}, \{g^{b\cdot c^i}\}_{i=0}^{n-1}, \{g^{c^i}\}_{i=1}^{3n^2}, \{g^{d\cdot c^i}\}_{i=1}^{2n^2}) \in \mathbb{G}_1^{7n^2+n+1}$ and $(\{\widetilde{g}^{a\cdot c^i}\}_{i\in[1,2n^2]\setminus]n^2-n,n^2+n[}, \{\widetilde{g}^{c^i}\}_{i=1}^{2n^2}, \widetilde{g}^d) \in \mathbb{G}_2^{4n^2-2n+1}$ along with g^z. We show that any adversary able to distinguish Game 4 from Game 5 can be used to solve the associated problem. In our proof, we will implicitly define $\mathsf{sk}_{k^*} = a$ and proceed as usual for all the other users. We therefore only need to explain how to generate the group public key and to deal with oracle queries involving this user.

To avoid confusion with the elements provided in the EDDH instance, we will denote the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ in our group public key by respectively Y_0 and $\widetilde{Y}_0$. Let $k = \lfloor \frac{n^2}{t^*} \rfloor$. We note that we have $k \geq n$ and $0 < n^2 - kt^* < n$ as $t^* \in [1, n]$. This in particular implies that $n^2 + k(t - t^*) \notin]n^2 - n, n^2 + n[\ \forall t \neq t^*$.

First, the reduction $\mathcal{R}$ generates a random $x \in \mathbb{Z}_p$ and define the elements of the public key as follows:

- $(Y_0, \widetilde{Y}_0) \leftarrow (g^{c^{n^2-kt^*}}, \widetilde{g}^{c^{n^2-kt^*}})$
- $\widetilde{X} \leftarrow (\widetilde{g}^{c^{n^2-kt^*}})^x$;
- $\widetilde{Y}_i \leftarrow \widetilde{g}^{c^{n^2+k(i-t^*)}}$, $\forall 1 \leq i \leq n$;
- $Y_i \leftarrow g^{c^{n^2+k(i-t^*)}}$, $\forall i \in [1, n] \cup [n+2, 2n]$.

We note that all the necessary elements are provided in the EDDH instance, no matter the value of t^*, thanks to the definition of k. $\mathcal{R}$ can now answer all the oracle queries involving k^*:

- $\mathcal{O}\mathtt{Add}$: As explained above, $\mathcal{R}$ implicitly defines $\mathsf{sk}_{k^*} = a$ and then issues a signature $(\sigma_1, \sigma_2) \leftarrow ((Y_0)^r, [(Y_0)^x \cdot (\prod_{i\in\mathcal{T}_k} Y_i)^a]^r)$ for some random r. All the involved elements are provided in the EDDH instance.
- $\mathcal{O}\mathtt{Cor}$: Since Game 3, we know that this oracle is not used on k^*, so $\mathcal{R}$ can answer any such queries.
- $\mathcal{O}\mathtt{Sign}$: A group signature issued at time period i on behalf of k^* is a derived signature for $\mathcal{I} = \{i\}$ along with a non-interactive proof of knowledge of $\mathsf{sk}_{k^*} = a$. As the latter can be simulated, it only remains to explain how to generate the derived signature $\sigma_{\mathcal{I}} = (\sigma'_1, \sigma'_2, \sigma'_3, \widetilde{\sigma}')$.
  The reduction generates a random r and s and acts as if the scalar $t = s - \sum_{j\in\mathcal{T}_k\setminus\{i\}} a \cdot c^{k\cdot j}$, leading to:
  - $\sigma'_1 \leftarrow (Y_0)^r$;
  - $\sigma'_2 \leftarrow (Y_0^{(s+(x+a\cdot c^{k\cdot i})}$)^r;
  - $\widetilde{\sigma}' \leftarrow (\widetilde{Y}_0)^t \cdot \prod_{j\in\mathcal{T}_k\setminus\{i\}} \widetilde{Y}_j^a = (\widetilde{Y}_0)^s$
  - $\sigma'_3 \leftarrow [Y_{n+1-i}^t \cdot \prod_{j\in\mathcal{T}_k\setminus\{i\}} Y_{n+1-i+j}^a]^{c_i} = Y_{n+1-i}^{s\cdot c_i}$

  where $c_i \leftarrow \mathtt{H}(\sigma'_1||\sigma'_2||\widetilde{\sigma}'||\mathcal{I}||i)$. All these values can be generated from our EDDH instance. We note that both r and t are correctly distributed as s is random. Our reduction then returns the resulting signature $\sigma_{\mathcal{I}}$ along with a simulated proof of knowledge (c, s').

- $\mathcal{O}\texttt{Revoke}$: Since Game 3, we know that this oracle is not queried on a list of users containing k^* for the time period t^*. If $\texttt{RL}_t$ does not contain k^*, then $\mathcal{R}$ proceeds as usual. Else, this means that k^* is revoked for a time period $t \neq t^*$ and $\mathcal{R}$ then returns $\widetilde{Y}_t^a = \widetilde{g}^{a \cdot c^{n^2+k(t-t^*)}}$.
- $\mathcal{O}\texttt{Open}$: we proceed as explained in Game 4.
- $\mathcal{O}\texttt{Ch}_b$: Since Game 3, we know that $\mathcal{R}$ must return a group signature on behalf of k^* for the time period t^*. We set $\sigma_1' = (Y_0)^b$ (which implicitly defines $r = b$) and acts as if $t = \frac{d}{c^{n^2-kt^*}} - \sum_{j \in \mathcal{T}_k \setminus \{t^*\}} a \cdot c^{k \cdot j}$. This gives us:
  - $\widetilde{\sigma}' \leftarrow (\widetilde{Y}_0)^t \cdot \prod_{j \in \mathcal{T}_k \setminus \{t^*\}} \widetilde{Y}_j^a = \widetilde{g}^d$;
  - $\sigma_3' \leftarrow [Y_{n+1-t^*}^t \cdot \prod_{j \in \mathcal{T}_k \setminus \{t^*\}} Y_{n+1-t^*+j}^a]^{c_{t^*}} = (g^{c^{k(n+1-t^*)}})^{d \cdot c_{t^*}}$.

  Moreover we set $\sigma_2' = (Y_0^b)^x \cdot g^z$.
  First, note that σ_1' is perfectly distributed. All the required elements are provided in the EDDH instance, in particular because $n^2 - k \cdot t^* < n$. There are then only two cases. If $z = a \cdot b \cdot c^{n^2} + b \cdot d$, then we have $\sigma_2' = (\sigma_1')^{t+(x+a \cdot c^{k \cdot t^*} + \sum_{j \in \mathcal{T}_k \setminus \{t^*\}} a \cdot c^{k \cdot j})}$ and we are playing Game 4. If z is random then σ_1' and σ_2' are random and independent elements and we are playing Game 5. We then have $\epsilon_4 \leq \texttt{Adv}^{\mathsf{EDDH}}(\mathcal{A}) + \epsilon_5$.

In the end, we have $\frac{\epsilon}{n \cdot q} \leq \texttt{Adv}^{nf}(\mathcal{A}) + \texttt{Adv}^{\mathsf{EDDH}}(\mathcal{A}) + \epsilon_5$ with a negligible ϵ_5, which concludes our proof.

Proof of Non-frameability. Intuitively, as our group signatures contain a proof of knowledge of the user's secret key, any adversary framing an honest user must have first recovered his secret key. We will use this fact to solve the SDL problem.

Indeed, let $(g, g^a) \in \mathbb{G}_1$ and $(\widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_2$ be a SDL instance. We make a guess on k^*, the honest user that $\mathcal{A}$ aims to frame and implicitly sets his secret key $\mathsf{sk}_{k^*} = a$. We abort if this guess is wrong.

By using g^a and $\widetilde{g}^a$, and by simulating the proof of knowledge of a, we can perfectly simulate the $\texttt{Join}$ protocol. We then explain how we can deal with $\mathcal{O}\texttt{Sign}$ queries for a time period i. Our reduction $\mathcal{R}$ generates random $u, v \xleftarrow{\$} \mathbb{Z}_p$ and computes $\sigma_1' \leftarrow g^u$ and $\sigma_2' \leftarrow g^{u(v+x)} \cdot (g^a)^{u \cdot y^i}$, which sets $t = v - \sum_{j \in \mathcal{T}_k \setminus \{i\}} a \cdot y^j$ in the $\texttt{Derive}$ process. It then remains to send $\widetilde{\sigma}' = \widetilde{g}^v$ and $\sigma_3' = Y_{n+1-i}^{v \cdot c_i}$, where $c_i \leftarrow \texttt{H}(\sigma_1' || \sigma_2' || \widetilde{\sigma}' || \mathcal{I} || i)$ along with a simulated proof of knowledge.

Eventually, the adversary returns a group signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma}, c, s)$ for a time period i that can be traced back to k^* which means that $e(\sigma_1, \widetilde{g}^a) = [e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-1}, \widetilde{X} \cdot \widetilde{\sigma})]^{y^{-i}}$. Therefore, we have $e(\sigma_1, \widetilde{Y}_i^a) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-1}, \widetilde{X} \cdot \widetilde{\sigma})$ and thus $e(\sigma_1, \widetilde{X} \cdot \widetilde{\sigma} \cdot \widetilde{Y}_i^a) = e(\sigma_2, \widetilde{g})$. The proof of knowledge contained in σ is then a proof of knowledge of a that $\mathcal{R}$ can extract to solve the SDL problem.

Any adversary $\mathcal{A}$ succeeding against the non-frameability of our scheme with probability ϵ can then be used to solve the SDL problem with probability $\frac{\epsilon}{q}$, where q is a bound on the number of group members.

Proof of Traceability. A successful adversary $\mathcal{A}$ against traceability returns a valid group signature for a time period t that either foils the opening process (*i.e.* $\texttt{Open}$ returns $\perp$) or that is traced back to a user k that is inactive at this time period (*i.e.* $t \notin \mathcal{T}_k$). We show in this proof that both cases lead to a forgery against the underlying redactable signature scheme Σ.

We construct a reduction $\mathcal{R}$ that acts as an adversary in the unforgeability experiment for Σ, with the goal to output a valid forgery by using $\mathcal{A}$. $\mathcal{R}$ then gets a public key pk that it sets as the group public key. To answer $\mathcal{O}\mathtt{J}_{GM}$ query, $\mathcal{R}$ simply extracts the user's secret signing key from the proof of knowledge and then queries the signing oracle for Σ. All the other queries are trivial to address as $\mathcal{R}$ does not need to know the secret key (x, y). In the end, $\mathcal{A}$ returns a valid group signature $\sigma = (\sigma_{\{t\}}, c, s)$ that corresponds to one of the two cases mentioned above. In all cases, the fact that σ is valid implies that $\sigma_{\{t\}}$ is a valid derived signature for the subset $\{t\}$ and on some message m_t whose knowledge is proven by (c, s). We can then extract m_t from this proof and distinguish two cases. In all cases $m_t \neq 0$, otherwise the group signature would be rejected by the verification algorithm.

- Case 1: $\texttt{Open}$ returns $\perp$ on σ. As this algorithm tests whether $m_t = \mathsf{sk}_k$ for every registered member k, this means that $m_t \notin \mathtt{L}_t = \{0, \{\mathsf{sk}_k\}_k\}$. However, every query to the signing oracle of Σ was on sets of messages $\{m'_i\}_{i=1}^n$ with $m'_i \in \mathtt{L}_t$. This means that $\sigma_{\{t\}}$ and m_t constitute a valid forgery against Σ.
- Case 2: $\texttt{Open}$ returns k with $t \notin \mathcal{T}_k$. This case means that $m_t = \mathsf{sk}_k$. However, as $t \notin \mathcal{T}_k$, $\mathcal{R}$ has queried for this user a signature on a set $\{m'_i\}_{i=1}^n$ with $m'_t = 0$. Therefore, here again, $\sigma_{\{t\}}$ and m_t necessarily constitute a forgery against Σ.

Both cases then lead to a forgery against Σ. An adversary against the traceability of our construction can then be used to attack the unforgeability of Σ with the same success probability, which concludes our proof.

6 Efficiency

6.1 Redactable Signature

We compare in Table 1 the efficiency of our unlinkable redactable signature with the recent one from [22]. Regarding the public key and the signature sizes, the comparison is provided both in terms of group elements and in bits, by implementing our bilinear groups using the BLS12 curve from ZCash [7]. The latter corresponds to a 128-bits security level, yielding elements of 256 bits ($\mathbb{Z}_p$), 382 bits ($\mathbb{G}_1$), 763 bits ($\mathbb{G}_2$) and 4572 bits ($\mathbb{G}_1$).

Table 1 shows that the main features of these two schemes are essentially the same (constant size derived signature, $O(k)$ verification complexity, etc.) except in the case of the public key. In [22], pk contains $O(n^2)$ elements whereas it only contains $O(n)$ elements in our scheme. The concrete improvement is thus extremely significant, except for very small values of n.

Table 1. Complexity of our redactable rignature scheme and the one from [22]. The costs of `Derive` and `Verify` are provided for a set $\{m_i\}_{i\in\mathcal{I}}$ of k elements. Here, H denotes the evaluation of a hash function, r_i denotes the generation of a random element in $\mathbb{G}_i$, e_i denotes an exponentiation in $\mathbb{G}_i$, for $i \in \{1, 2\}$, and p_k denotes an equation involving k pairings.

	pk	σ	Sign	Derive	Verify
[22]	$\frac{n^2+n+2}{2}\mathbb{G}_1 + n\mathbb{G}_2$ $= 191(n^2+n+2)$ bits	$2\mathbb{G}_1 + 2\mathbb{G}_2 = 2290$ bits	$1\mathsf{r}_2 + 1\mathsf{e}_2$	$2(n-k+1)\mathsf{e}_1$ $+3\mathsf{e}_2$	$k\mathsf{e}_1 + 2\mathsf{p}_2$
Ours	$(1+2n)\mathbb{G}_1 + (n+1)\mathbb{G}_2$ $= 382(4n+3)$ bits	$3\mathbb{G}_1 + 1\mathbb{G}_2$ $= 1909$ bits	$1\mathsf{r}_1 + 1\mathsf{e}_1$	$2(n+1)\mathsf{e}_1 + k\mathsf{H}$ $+(n-k+1)\mathsf{e}_2$	$k(\mathsf{e}_1 + \mathsf{e}_2 + \mathsf{H})$ $+2\mathsf{p}_2$

Table 2. Size complexities of our group signature scheme and [13]. `Update` here refers to the information (called expiration information in [13]) required to update the group signing key at each time period whereas `RL` is the size of the revocation list (only required by the verifiers) to revoke R users. We use the same notations as in the previous table and define $m = \log n$.

	pk	usk	Update	RL	σ
[13]	$8\mathbb{G}_1 + 3\mathbb{G}_2 = 5345$ bits	$1\mathbb{Z}_p +$ $(1\mathbb{G}_1 + 2\mathbb{Z}_p)m$ $= 256 + 894m$ bits	$1\mathbb{G}_1 +$ $(1\mathbb{G}_1 + 2\mathbb{Z}_p)m$ $= 382 + 894m$ bits	$R(2\mathbb{G}_1 + 1\mathbb{G}_2)$ $= 1527R$ bits	$6\mathbb{G}_1 + 1\mathbb{G}_2 + 11\mathbb{Z}_p$ $= 5871$ bits
Ours	$(1+2n)\mathbb{G}_1$ $+(n+1)\mathbb{G}_2$ $= 382(4n+3)$ bits	$2\mathbb{G}_1 + \mathbb{Z}_p = 1020$ bits	-	$R\mathbb{G}_2 1.9\,cm$ $= 763R$ bits	$3\mathbb{G}_1 + 1\mathbb{G}_2 + 2\mathbb{Z}_p$ $= 2421$ bits

6.2 Group Signature with Time-Bound Keys

We compare in Tables 2 and 3 the efficiency of our construction with the one of the most efficient scheme [13] from the state-of-the-art, using the same bilinear groups as above. We nevertheless note that this comparison has some limitations as our schemes do not have exactly the same features. For example, we allow $\mathcal{O}$`Open` queries in our anonymity experiment (hence achieving so-called CCA anonymity) contrarily to [13], but have a less efficient opening process. Similarly, our signing key can be associated with any set of time periods (thus offering a better granularity) and do not need to be updated, contrarily to [13], but we need a larger group public key. We also rely on different computational assumptions. In all cases, these tables show that our whole authentication process (issuance of group signature and verification) is more efficient regarding both computational complexity and the size of the elements. We believe this is a very interesting feature of our scheme as this authentication process is the usual bottleneck of use-cases involving group signatures.

As several parameters depend on n, it might be interesting to evaluate the concrete complexity in real-world scenarios. For example, in the case of a transit pass, it seems reasonable to consider time periods of one day for efficient revocation. A value $n \sim 1000$ allows then to issue group signing keys for the next three years. Such a value of n only impacts our public key, that would represent 191 KB, which is within the reach of any smartphone. If this is a problem in more constrained environments, we note that this large public key is only necessary

Table 3. Computational complexities of our group signature scheme and [13].

	`Sign`	`Verify`
[13]	$23\mathsf{e}_1 + 1\mathsf{e}_2 + 1\mathsf{p}_6 + 1\mathsf{p}_5 + \mathsf{H}$	$24\mathsf{e}_1 + 1\mathsf{p}_2 + 2\mathsf{p}_8 + \mathsf{H}$
Ours	$6\mathsf{e}_1 + 2\mathsf{e}_2 + 2\mathsf{H} + 1\mathsf{p}_1$	$3\mathsf{e}_1 + 1\mathsf{p}_3 + 2\mathsf{H} + 2\mathsf{p}_2$

to run `Derive`. In an alternative solution, the group manager could simply send, for each time period, the elements $\prod_{j\in\overline{\mathcal{I}}} \widetilde{Y}_j$, $\prod_{j\in\overline{\mathcal{I}}} Y_{n+1-i+j}$ and Y_{n+1-i} that are sufficient, for all group members, to run this algorithm. This would dramatically reduce the size of the public key for the group members and for the verifiers. This would not be a disadvantage compared to [13] as the latter already requires to send (larger) update information at each time period.

Conclusion

In this paper, we have proposed an extension of group signature with time-bound keys that allows for a better revocation granularity while removing the need to update member's secret keys at each time period. This has two important practical consequences. Firstly, by providing a way to precisely limit the signing rights of a group member we make group signature even more suitable for real-world applications, as most of the latter are unlikely to grant unlimited access to users. This in particular limits the need for premature revocation whose complexity increases with the number of revoked members. Secondly, this simplifies key management for both the group manager (that no longer needs to publish expiration information at the beginning of each period) and the group members (that no longer need to recover this information).

We have also shown in this paper that we can implement such a complex primitive with remarkable efficiency. Our group signatures are indeed only 300 Bytes long, which is rather surprising as (efficient) revocation for group signature is usually quite hard to achieve. This was made possible by a variant of a recent redactable signature scheme that we have introduced and that we believe to be of independent interest.

Acknowledgement. The author thanks R. Kabaleeshwaran for pointing out that a previous version of the EDDH assumption did not hold. The author is grateful for the support of the ANR through project ANR-16-CE39-0014 PERSOCLOUD and project ANR-18-CE-39-0019-02 MobiS5.

References

1. AlLee, G.: EPID for IoT Identity (2016). https://img.en25.com/Web/McAfeeE10BuildProduction/a6dd7393-63f8-4c08-b3aa-89923182a7e5_EPID_Overview_Public_2016-02-08.pdf

2. Barki, A., Desmoulins, N., Gharout, S., Traoré, J.: Anonymous attestations made practical. In: Noubir, G., Conti, M., Kasera, S.K. (eds.) WiSec 2017 (2017)
3. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: the case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_11
4. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N.P., Warinschi, B.: Anonymous attestation with user-controlled linkability. Int. J. Inf. Secur. **12**, 219–249 (2013)
5. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 435–464. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_15
6. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 168–177. ACM Press, October 2004
7. Bowe, S.: BLS12-381: New zk-SNARK Elliptic Curve Construction (2017). https://electriccoin.co/blog/new-snark-curve/
8. Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_3
9. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: definitions and practical constructions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 262–288. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_11
10. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
11. Chu, C.-K., Liu, J.K., Huang, X., Zhou, J.: Verifier-local revocation group signatures with time-bound keys. In: Youm, H.Y., Won, Y. (eds.) ASIACCS 2012, pp. 26–27. ACM Press, May 2012
12. Desmoulins, N., Lescuyer, R., Sanders, O., Traoré, J.: Direct anonymous attestations with dependent basename opening. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 206–221. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12280-9_14
13. Emura, K., Hayashi, T., Ishida, A.: Group signatures with time-bound keys revisited: a new model and an efficient construction. In: Karri, R., Sinanoglu, O., Sadeghi, A.-R., Yi, X. (eds.) ASIACCS 2017, pp. 777–788. ACM Press, April 2017
14. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discret. Appl. Math. **156**(16), 3113–3121 (2008)
15. Libert, B., Mouhartem, F., Nguyen, K.: A lattice-based group signature scheme with message-dependent opening. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 137–155. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_8
16. Libert, B., Peters, T., Yung, M.: Group signatures with almost-for-free revocation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 571–589. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_34
17. Libert, B., Peters, T., Yung, M.: Scalable group signatures with revocation. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 609–627. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_36

18. McDonald, K.L.: The landscape of pointcheval-sanders signatures: Mapping to polynomial-based signatures and beyond. IACR Cryptology ePrint Archive 2020:450 (2020)
19. Navigo (2020). https://www.iledefrance-mobilites.fr/titres-et-tarifs/liste?d=forfaits
20. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_7
21. Pointcheval, D., Sanders, O.: Reassessing security of randomizable signatures. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 319–338. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_17
22. Sanders, O.: Efficient redactable signature and application to anonymous credentials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 628–656. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_22
23. Sanders, O.: Improving revocation for group signature with redactable group signature (full version of this work). IACR Cryptology ePrint Archive 2020:856 (2020)
24. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 688–689. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_68
25. TCG (2015). https://trustedcomputinggroup.org/authentication/