Cross-Unlinkable Hierarchical Group Signatures[⋆⋆⋆]

Julien Bringer[1], Hervé Chabanne[12], and Alain Patey[12]

[1] Morpho
[2] Télécom ParisTech
Identity & Security Alliance (The Morpho and Télécom ParisTech Research Center)

Abstract. We introduce the notion of Cross-Unlinkability for group signature schemes. Considering groups organized in a tree structure, where belonging to the parent group is required to join a new group, Cross-Unlinkability enables a cascade revocation process that takes into account the underlying tree structure, while ensuring anonymity for non-revoked users, in particular, towards the managers of the other groups. We show how to achieve Cross-Unlinkability using the Verifier-Local Revocation group signature scheme of Bringer and Patey at Secrypt 2012, by exploiting its property of Backward Unlinkability.

Keywords: Anonymity, Unlinkability, Group Signatures

1 Introduction

Group signatures [9] enable authorized users to sign anonymously on behalf of a group. We consider in the following the case of VLR (Verifier-Local Revocation) group signatures. The VLR property [5] guarantees that only the public parameters and a revocation list RL are required to check a signature. Concretely, when a user is revoked, a revocation token that is derived from his signing key is added to RL. This token is used by verifiers to prevent revoked users from further signing.

In this paper we consider a scenario where users have access to several groups, equipped with group signatures, that have some dependencies between them: the set $\mathbb{G}$ of these groups is partially ordered and can be represented as a tree. When one wants to apply for new signing keys in a group $\mathcal{G}_l$, one has to own valid signing keys for the parent group $\mathcal{G}_k$ in the tree $\mathbb{G}$. This organization also requires that it should be possible to revoke automatically across different groups. To this aim, the new signing key is derived from the key of the same member for $\mathcal{G}_k$ in

[⋆] This work is partially funded under the European FP7 FIDELITY project (SEC-2011-284862). All information is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The European Commission has no liability in respect of this document, which merely represents the authors view.

[⋆⋆] This is the extended version of the article presented at EuroPKI 2012.

order to maintain a link. One important issue in our model is then to ensure the privacy of this link.

This scenario and the associated security properties are particularly adapted to identity management systems. In this setting, a user owns several identities derived from a strong identity (e.g. the national identity) while maintaining privacy and unlinkability between the different identities, even towards to the providers of the other identities.

We address this problem of derivation of group signatures keys from other group signature keys with privacy properties in mind. We in particular want to ensure that a given group manager cannot retrieve – except in case of revocations – the link between a signature in his group and a signature, issued by the same user, in any of his children groups. Our goal is to parallelize several instances of VLR group signatures while fulfilling the following additional requirements:

- A user registered into a given group should be able to sign anonymously on behalf of this group;
- When a user asks for registering to a new group, he has to prove that he can sign on behalf of the parent group, and the new keys delivered by the group manager should be derived from the pre-required keys;
- The derivation process should be compatible with a revocation process that echoes downwards, i.e. when a user M_i is revoked from a given group $\mathcal{G}_l$, he must also be revoked from all the groups that are below $\mathcal{G}_l$ in the tree $\mathbb{G}$.
- Despite these revocation and derivation processes, only the manager of a given group $\mathcal{G}_l$ (and the signer) can learn information on the signer when looking at a signature for the group $\mathcal{G}_l$, provided this signer is not revoked from $\mathcal{G}_l$. Particularly, the other group managers learn nothing more than any observer and thus cannot link the signer to the members of their groups. This property, that we name *Cross-Unlinkability*, is an essential feature of our proposition.

Recall that, when a user is revoked, a revocation token that is derived from his signing key is added to RL. It enables to reject the further signatures of this user but it may also give a way to identify his previously made signatures. To prevent this, some VLR group signatures (e.g. [8,15–17,19]) enjoy an additional property called *Backward Unlinkability* (BU). The usual mechanism to enable BU is to split the time into several periods and derive revocation tokens associated to each period so that two revocation tokens for the same user for different periods are unlinkable.

We here adapt the derivation process for revocation tokens, no longer to enjoy BU, but to derive keys between the different group signature schemes. In our context defined above, the direction of the derivation is from a parent group $\mathcal{G}_k$ to a child group $\mathcal{G}_l$ in $\mathbb{G}$. In a sense, such a child group is seen as a time period for the VLR signature with BU associated to $\mathcal{G}_k$, unlinkability between different time periods in the original schemes with BU is transformed into unlinkability between the different children. We adapt the BU derivation process so that the new keys are not known by the parent group manager, while satisfying the requirement for the revocation process to be echoed to the lower levels.

For instance, consider the group tree described in Figure 1. We assume that a science faculty sets up a system using groups signatures, used for instance for access control. In this example, applying for a key for the Bioinformatics Team requires to previously own a key for the Computer Science Department. We also wish that, when one signs on behalf of, e.g., the Mechanics Department, anonymity of the signer is guaranteed against the managers of all other groups, including the managers of the parent group (Science Faculty), the children groups (Fluid Dynamics and Solid Mechanics) or the sibling groups (Computer Science Dept.).

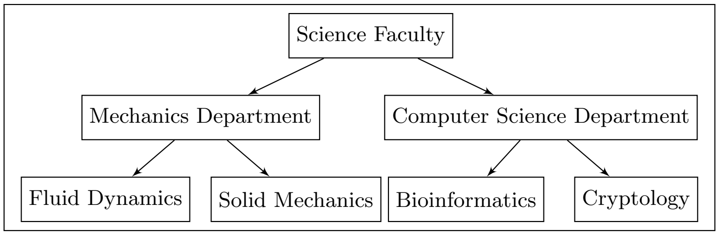

Fig. 1. An example of a group tree $\mathbb{G}$

Other settings with several parallel group signatures have already been introduced. Multi-group signatures [1, 3] enable a user to sign on behalf of either a single group or several groups to which he belongs. The notion of hierarchy between group signatures has been introduced in [14], where having a key for an upper group allows to sign on behalf of lower groups. Hierarchical Group Signatures [18] define a group organization that is close to ours: the managers are organized in a tree structure, but all of them do not manage signers, some only manage groups of managers; anonymity and (a weaker notion of) unlinkability between the users of different groups are considered but there is no possibility of revocation. Attribute-based group signatures [12, 13], anonymous proxy signatures [11] and delegatable credentials [2] are also related notions. None of the above constructions however considers at the same time group hierarchy, unlinkability across the groups and revocation through the groups.

The process described in this paper is instantiated with the Bringer and Patey group signature [8] for a better readability but it can easily be adapted to other group signature schemes enjoying BU (e.g. [15–17, 19]).

In [6], Bringer et al. introduce the concept of cross-unlinkability for anonymous authentications, adapting the biometric-based remote authentication protocol of [7] to the hierarchical setting. In this paper, we formalize further this concept with the model of cross-unlinkable group signatures. We detail a construction and give its security properties. We finally prove that our construction satisfies these properties.

2 VLR Group Signatures

Group signatures [9] are a particular case of digital signatures where authorized members of a group are allowed to sign anonymously on behalf of the group. The anonymity can only be ended by the Group Manager who can also revoke misbehaving users (or users wanting to leave). In the particular case of Verifier-Local Revocation (VLR) [5], anyone knowing the public parameters of the group can verify the signatures (including revocation checks). Groups involved in VLR group signatures are dynamic: users can be revoked (voluntarily or not) at any time. The revocation process consists in adding revocation tokens to a public revocation list which is taken into account by the verifiers when they check a signature.

2.1 Components

The following algorithms are the components of a VLR group signature scheme with Backward Unlinkability. In the context of group signatures with BU, time is divided into time periods $j \in [1, \dots, T]$. We denote by $\mathcal{G}$ the group associated to the group signature. If the signature does not enable BU, then the algorithms do not depend on time periods.

KeyGen$_{GS}$**:** Generates the public parameters for the system and the public/secret keys of the group manager GM. It takes as input a security parameter μ and outputs the secret key msk of GM, its public counterpart mpk, an empty global revocation list RL and empty period revocation lists RL_j's, for $j \in [1, \dots, T]$, and the public parameters gpk. msk is kept secret by the GM, the other elements are published.

Join$_{GS}$**:** Creates keys for a new user M_i and allows him to produce group signatures. It outputs the user key sk_i and the corresponding revocation tokens rt_i (global revocation token) and rt_{ij} (period), for all $j \in [1, \dots, T]$. sk_i is stored by the user and GM stores a part of sk_i and revocation tokens.

Sign$_{GS}$**:** Takes as input a message m, a time period j and a signer's key sk_i. Returns a signature σ of user M_i on the message m at period j.

Verify$_{GS}$**:** Takes as input a time period j, a message m, a signature σ and the public parameters gpk of the system and the current Revocation Lists RL (global) and RL_j (period). Checks if the message has been well-signed by an unrevoked group member without revealing the signer's identity.

Open$_{GS}$**:** Takes a signature σ as input and reveals the identity of the signer. Can only be performed by the GM, since it requires to know the revocation tokens of all users.

Revoke$_{GS}$**:** Revokes a user M_i from the group at a time period j. His revocation token rt_{ij} is added to RL_j. (For a global revocation, the same process is executed with rt_i and RL.)

2.2 Security properties

We describe the security properties that can be required from a VLR group signature scheme with BU. Our description is a slight variant of the one of [8] but it is also fulfilled by the [8] scheme. In particular, we do not require the games to follow a chronological order, since it is not necessary in the proofs of security of [8].

Correctness: Every check of a well-formed signature returns valid if the user who has issued it is not revoked.

The Traceability property ensures that no attacker (or group of attackers) is able to forge a signature that can not be traced to one of the corrupted users which participated in its forgery.

Traceability: Let us consider the following Traceability game played by an adversary **A**.
Setup: The challenger **C** runs the **KeyGen** algorithm, playing the role of the GM. He obtains gpk, mpk and msk. He provides **A** with gpk and mpk.
Queries: **A** can make the following queries to the challenger, provided that it well specifies the time period j:

- *Join:* **A** requests the enrolment of a member M_i to $\mathcal{G}$. **C** obtains the keys for M_i for $\mathcal{G}$. **A** obtains nothing.
- *Sign:* **A** requests that a member M_i of $\mathcal{G}$ signs a message m for the current period j. The challenger computes the signature σ of M_i on m for j. **A** obtains σ.
- *Corruption:* **A** requests the corruption of a given registered member M_i. He obtains the secret key of M_i for $\mathcal{G}$. The member M_i is revoked at all time periods.
- *Revocation:* **A** requests the revocation of a user M_i from $\mathcal{G}$ for period j. He learns the revocation token of M_i that is disclosed during this phase.

Output: **A** outputs a challenge message m, a period j and a signature σ on m and wins if:

1. $\mathbf{Verify}_{GS}(m,\sigma,j,gpk,mpk,RL_j)=\mathsf{valid}$
2. **A** did not obtain σ by making a *Sign* Query on m.

The scheme is said to satisfy *Traceability* if no polynomial probabilistic adversary is able to win the above game with a non-negligible probability.

The *Backward Unlinkability* property is an extension of the notion of *Selfless-Anonymity* [5] to the multi-period setting. Selfless-Anonymity implies that only the signer and the Group Manager learn information on the producer of a given signature, provided that the signer is not revoked. Backward Unlinkability moreover ensures that valid signatures remain anonymous even after the signer's revocation. Revoked users can also come back after their revocation into the group and use their previous keys without any loss of anonymity. That is why we require time to be divided into several periods so that the signer uses different parameters for the different periods.

However notice that when one is revoked at a given period, he loses anonymity on all his signatures issued at this particular period, even if they were produced before the revocation. Anonymity is only guaranteed on the signatures made at previous periods where the user was not revoked.

The Selfless-Anonymity definition can be obtained from the definition of Backward Unlinkability by applying it to the case where only one time period is available.

Backward Unlinkabilty: Let us consider the following BU game played by an adversary $\mathbf{A}$:

Setup: The challenger $\mathbf{C}$ runs the **KeyGen** algorithm, playing the role of the GM. He obtains gpk, mpk and msk. He provides $\mathbf{A}$ with gpk and mpk.

Queries: $\mathbf{A}$ can make the following queries to the challenger, provided that it well specifies the time period j:

- *Join:* $\mathbf{A}$ requests the enrolment of a member M_i to $\mathcal{G}$. $\mathbf{C}$ obtains the keys of M_i for $\mathcal{G}$. $\mathbf{A}$ obtains nothing.
- *Sign:* $\mathbf{A}$ requests that a member M_i of $\mathcal{G}$ signs a message m for the current period j. The challenger computes the signature σ of M_i on m for j. $\mathbf{A}$ obtains σ.
- *Corruption:* $\mathbf{A}$ requests the corruption of a given registered member M_i. He obtains the secret key of M_i for $\mathcal{G}$.
- *Revocation:* $\mathbf{A}$ requests the revocation of a user M_i from $\mathcal{G}$ for period j. He learns the revocation token of M_i that is disclosed during this phase.

Challenge: $\mathbf{A}$ outputs a challenge message m, a period j and two different members M_0 and M_1, such that:

1. M_0 and M_1 are both registered to $\mathcal{G}$;
2. $\mathbf{A}$ corrupted neither M_0 nor M_1;
3. M_0 and M_1 are not revoked from $\mathcal{G}$ at period j

$\mathbf{C}$ chooses a random bits $b \in_R \{0,1\}$ and runs $\mathbf{Sign}_{GS}$ for M_b at period j using message m. The obtained signature σ^* is transmitted to $\mathbf{A}$.

Restricted Queries: $\mathbf{A}$ can make the same queries as in the *Queries* phase, as long as this does not contradict the above requirements 1 to 3 of the Challenge phase.

Output: $\mathbf{A}$ outputs a guess $\beta \in \{0,1\}$ on b.

The scheme satisfies *Backward-Unlinkability* if the probability $|\Pr(\beta = b) - 1/2|$ is negligible.

Exculpability Nobody, even the Group Manager, is able to produce another user's signature.

(This property is not always satisfied by VLR group signature schemes. We refer the reader to [8,10] for a formal definition that is adapted to the schemes used in the following.)

2.3 The CL and BP Schemes

As an example and for a better understanding, we use the scheme of Bringer and Patey [8], that we denote by BP, that fulfils all the above security requirements. Particularly, it enables Backward Unlinkability using the usual technique of dividing the time into several periods and deriving revocation tokens for the members that depend on the time and on the secret key of the user but that cannot be linked with each other. We moreover use the patched version of the scheme of Chen and Li [10], also described in [8], that we denote by CL, and that is merely the BP scheme without BU. In particular, we can use the same parameters and keys for both schemes.

We first describe the CL scheme. Notice that, since it does not enable BU, algorithms are independent of the time period.

KeyGen$_{CL}(\mu)$ Choose bilinear groups $G_1, G_2, G_{\mathcal{T}}$ of order a μ-bit prime number p that is safe, a prime number q and a pairing $e : G_1 \times G_2 \rightarrow G_{\mathcal{T}}$. Let g_1, g_2 be generators of G_1 and G_2. Choose a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. Choose $\tilde{g}_1, \hat{g}_1 \in_R G_1, \gamma \in_R \mathbb{Z}_p^*$, and compute $w = g_2^{\gamma}$. Compute $T_1 = e(g_1, g_2)$, $T_2 = e(\tilde{g}_1, g_2)$, $T_3 = e(\hat{g}_1, g_2)$ and $T_4 = e(\hat{g}_1, w)$. Output: $gpk = (G_1,\ G_2,\ G_{\mathcal{T}},\ e,\ p,\ g_1,\ g_2,\ \tilde{g}_1,\ \hat{g}_1,\ w,\ H,\ T_1,\ T_2,\ T_3,\ T_4$ and $msk = \gamma$.

Join$_{CL}(M_i, msk, gpk, mpk)$ GM sends a nonce $n_i \in \{0,1\}^k$ to M_i. M_i chooses $f_i \in_R \mathbb{Z}_p$ and computes $F_i = \tilde{g}_1^{f_i}$. He chooses $r_f \in_R \mathbb{Z}_p$ and computes $R = \tilde{g}_1^{r_f}$. He computes $c = H(gpk||F_i||R||n_i)$ then $s_f = r_f + cf_i$. M_i sends $comm = (F_i, c, s_f)$ to GM. GM computes $R' = \tilde{g}_1^{s_f} F_i^{-c}$ and checks that $s_f \in \mathbb{Z}_p$ and $c = H(gpk||F||R'||n_i)$. He chooses $x_i \in_R \mathbb{Z}_p$ and computes $A_i = (g_1 F_i)^{1/(x_i+\gamma)}$. GM sends (A_i, x_i) to M_i, using a secure channel. M_i checks that $e(A_i, wg_2^{x_i}) = e(g_1\tilde{g}_1^{f_i}, g_2)$ and outputs $sk_i = (f_i, x_i, A_i)$. The global revocation token for M_i is $rt_i = x_i$. GM outputs x_i, A_i and the revocation tokens.

Sign$_{CL}(m, sk_i, gpk, mpk)$ Choose $B \in_R G_1$ and compute $J = B^{f_i}$, $K = B^{x_i}$. Choose $a \in_R \mathbb{Z}_p$, compute $b = ax_i$ and $T = A_i\hat{g}_1^a$. Choose $r_f, r_x, r_a, r_b \in_R \mathbb{Z}_p$. Compute $R_1 = B^{r_f}$, $R_2 = B^{r_x}$, $R_4 = K^{r_a}B^{-r_b}$, $R_3 = e(T, g_2)^{-r_x}T_2^{r_f}T_3^{r_b}T_4^{r_a}$. Compute c=$H(gpk||B||J||K||T||R_1||R_2|\ |R_3||R_4||m)$. Compute $s_f = r_f + cf_i$, $s_x = r_x + cx_i$, $s_a = r_a + ca$ and $s_b = r_b + cb$. Output: $\sigma = (B, J, K, T, c, s_f, s_x, s_a, s_b, \mathbf{s_1}, \ldots, \mathbf{s}_\lambda)$.

Verify$_{CL}(m, \sigma, gpk, mpk, RL)$ 1. *Signature Check:* Check that $B, J, K, T \in G_1$ and $s_f,\ s_x,\ s_a,\ s_b,\ \mathbf{s_1}, \ldots, \mathbf{s}_\lambda \in \mathbb{Z}_p$. Compute $R_1' = B^{s_f}J^{-c}$, $R_2' = B^{s_x}K^{-c}$, $R_3' = e(T, g_2)^{-s_x}T_2^{s_f}T_3^{s_b}T_4^{s_a}T_1^c e(T, w)^{-c}$ and $R_4' = K^{s_a}B^{-s_b}$. Check that c=$H(gpk||B||J||K||T||R_1'||R_2'||R_3'||R_4'|\ |m)$.

2. *Revocation Check:* Check that $\forall rt_i \in RL, K \neq B^{rt_i}$. Output valid if all checks succeed. Otherwise output invalid.

Revoke$_{CL}(RL, rt_i)$ Add the member's revocation token rt_i to the current revocation list RL and publish the thus updated RL.

Open$_{CL}(\sigma, gpk, mpk, msk, \{rt_i|M_i \text{ is in the group}\})$ For every member $M_i \in \mathcal{G}$, use the *Revocation Check* algorithm on the signature σ with a revocation list set as $RL = \{rt_i\}$. When the test fails, output the corresponding M_i.

We now describe the BP scheme. Notice that, since it enables BU, algorithms depend on the time period.

KeyGen$_{BP}(\mu)$ Run **KeyGen**$_{CL}(\mu)$. Furthermore, choose a security parameter λ for the proofs of knowledge involving double logarithms. Pick $h_1, \ldots, h_T \in_R \mathbb{Z}_q^*$ and add λ and the h_j's to gpk

Join$_{BP}(M_i, msk, gpk, mpk)$ Run **Join**$_{CL}(M_i, msk, gpk, mpk)$. Moreover, the revocation token for M_i at period j is $rt_{ij} = h_j^{x_i}$.

Sign$_{BP}(m, j, sk_i, gpk, mpk)$ Run the **Sign**$_{CL}(m, sk_i, gpk, mpk)$ algorithm with some adaptations: in addition to B, J and K, compute $L = B^{h_j^{x_i}}$ and add j and L in the input of the hash function to compute c. Moreover, pick $r_1, \ldots, r_\lambda \in_R \mathbb{Z}_p$. Compute $V_l = B^{r_l}$ and $W_l = B^{h_j^{r_l}}$, $\forall l = 1 \ldots \lambda$. Compute $d = H(c||(V_l, W_l)_{l=1\ldots\lambda})$. $\forall l = 1 \ldots \lambda$, let b_l be the l^{th} bit of d. Set $s_l = r_l - b_l x$. Add L, d, $s_1, \ldots, s_\lambda$ to the output.

Verify$_{BP}(m, \sigma, j, gpk, mpk, RL, RL_j)$
1. *Signature Check:*

   Run the *Signature Check* of **Verify**$_{CL}(m, \sigma, gpk, mpk, RL)$ with some adaptations: check that $L \in G_1$ and that $s_1, \ldots, s_\lambda \in \mathbb{Z}_p$, and add j and L in the input of the hash function to compute c. $\forall l = 1 \ldots \lambda$, let b_l be the l^{th} bit of d.. Compute $V'_l = B^{s_l} K^{b_l}$ and $W'_l = (B^{1-b_l} L^{b_l})^{h_j^{s_l}}$. Check that $d = H(c'||(V'_l, W'_l)_{l=1\ldots\lambda})$.
2. *Revocation Check:*

   Run the *Revocation Check* of **Verify**$_{CL}(m, \sigma, gpk, mpk, RL)$. Moreover, check that $\forall rt_{ij} \in RL_j, L \neq B^{rt_{ij}}$. Output valid if all checks succeed. Otherwise output invalid.

Revoke$_{BP}(j, RL, RL_j, rt_i, rt_{ij})$ For a global revocation, run **Revoke**$_{CL}(RL, rt_i)$. For a period revocation, add the member's revocation token rt_{ij} to the current revocation list RL_j and publish the thus updated RL_j.

Open$_{BP}(\sigma, j, gpk, mpk, msk, \{rt_{ij}|M_i \text{ is in the group}\})$ For every member $M_i \in \mathcal{G}$, use the *Revocation Check* algorithm on the signature σ with a revocation list set as $RL_j = \{rt_{ij}\}$. When the test fails, output the corresponding M_i.

Notice that the **Revoke**$_{GS}$ and **Open**$_{GS}$ are standard procedures of VLR group signature schemes and are not specific to the BP and CL schemes.

We recall the security results given in [8, 10], where DDH refers to the Decisional Diffie-Hellman assumption, DL to the Discrete Logarithm assumption and q-SDH refers to the q-Strong Diffie-Hellman assumption [4]. The adapted DHH is an adaptation of the DDH assumption described in [8].

Theorem 1 (Security of the BP and CL Schemes).

In the random oracle model, the CL scheme described above achieves Correctness, Selfless Anonymity (under the DDH assumption), Exculpability (under the DL assumption) and Traceability (under the q-SDH assumption).

In the random oracle model, the BP scheme described above achieves Correctness, Backward Unlinkability (under the adapted DDH assumption), Exculpability (under the DL assumption) and Traceability (under the q-SDH assumption).

3 Our Model of Cross-Unlinkable Hierarchical Group Signatures

We here describe our model for a hierarchical group signature setting where groups follow a tree hierarchy. Our goal is to constrain that, when one wants to acquire a group signature key for a group $\mathcal{G}_k$, one has to prove that one belongs to the parent group in the tree $\mathbb{G}$. Our model does not change much the way the members of a particular group use group signatures. It focuses on the way the member keys for the different groups are linked to enable at the same time a cascade derivation process and unlinkability between signatures issued by the same user for different groups, in particular towards the group managers. In our model, only the signer and the group manager of the concerned group (and particularly not the other GM's) are able to identify the producer of a given signature. We also precise that, within a group, the way we use signatures does not enable us to enjoy BU. Nevertheless, we achieve Selfless-Anonymity [5] as for usual VLR group signature schemes, where, still, only the signer and the GM are able to tell who produced a particular signature but where, once a member has been revoked, he loses his anonymity on all his signatures.

3.1 Setting

We assume that there are several groups $\mathcal{G}_k$ organized as a tree $\mathbb{G}$ with a root $\mathcal{G}_0$. Each group $\mathcal{G}_l$ has a group manager GM_l and we will denote by $k \dashv l$ the fact that the group $\mathcal{G}_k$ is a parent of the group $\mathcal{G}_l$. The functionalities of our protocol are the following.

KeyGen is an extension of the **KeyGen**$_{GS}$ algorithm to the hierarchical group setting, it specifies how the parameter choices of the different group managers should be related.

KeyGen(λ)**:** This is run by the GM's. It takes as input a security parameter λ. GM_0 first returns the public parameters gpk for all the group signatures. Then each GM_l creates a secret/public key pair (msk^l, mpk^l) and publishes mpk^l.

The **Enrolment** algorithm specifies how a group manager GM_l and a user M_i applying to join $\mathcal{G}_l$ interact to provide M_i with a key for $\mathcal{G}_l$. If $\mathcal{G}_l \neq \mathcal{G}_0$, this algorithm calls the **Derivation** algorithm, that we describe next.

Enrolment$(M_i, \mathcal{G}_l)$**:** For a group $\mathcal{G}_l$, this algorithm is jointly run by the group manager GM_l and a user M_i. The input for GM_l is his secret key msk^l, it also requires the result of the **Derivation** algorithm if $\mathcal{G}_l \neq \mathcal{G}_0$. It outputs a key sk_i^l for member M_i for the group signature of $\mathcal{G}_l$ and the associated revocation token rt_i^l.

The **Derivation** algorithm is a key step of our setting. A member M_i applying to a group $\mathcal{G}_l$, child of $\mathcal{G}_k$ in $\mathbb{G}$, interacts with the manager GM_l of $\mathcal{G}_l$.

M_i proves that he owns keys for $\mathcal{G}_k$ and, if the proof has been successful, the interaction enables $\mathcal{G}_l$ to derive a key for M_i for $\mathcal{G}_l$ that depends on his key for $\mathcal{G}_k$ (but without learning the latter key). Using the thus derived key, GM_l can finalize the **Enrolment** algorithm and provides M_i with his new key.

Derivation($M_i, \mathcal{G}_k, \mathcal{G}_l$): For a group $\mathcal{G}_l$ such that $k \dashv l$, this algorithm is jointly run by a user M_i requiring to get group signature keys for the group $\mathcal{G}_l$ and the group manager GM_l of $\mathcal{G}_l$. The input for M_i is his group signature key for the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$ in $\mathbb{G}$ and the input for GM_l is his secret key msk^l. It returns a new secret key for M_i for $\mathcal{G}_l$ if M_i successfully proves to GM_l that he is a non revoked member of $\mathcal{G}_k$.

The **Sign** and **Verify** algorithms perform the same functionalities as the **Sign**$_{GS}$ and **Verify**$_{GS}$ algorithms for a given group $\mathcal{G}_l$, using group signatures without Backward Unlinkability, such as the CL scheme for instance. Notice that there are no time periods and the signatures are consequently independent of the time.

Sign($M_i, m, \mathcal{G}_l$): For a group $\mathcal{G}_l$, this is run by a user M_i. The input of M_i is his secret sk_i^l and a message m. It returns a group signature σ issued by M_i on behalf of the group $\mathcal{G}_l$.

Verify($\sigma, m, \mathcal{G}_l$): For a group $\mathcal{G}_l$, this is run by anyone knowing gpk and mpk^l. The inputs are a signature σ and a message m. It checks if σ is a legitimate signature of m by an unrevoked member of $\mathcal{G}_l$.

The **Revocation** algorithm answers to what we expect from our cascade revocation capability. The goal of the downwards revocation process it to ensure that once a user has been revoked from a given group $\mathcal{G}_l$, this user is also revoked from all groups that are children of $\mathcal{G}_l$ in $\mathbb{G}$, the children of these children, and so on. The optional upwards revocation is there to give the possibility for a group manager to report to the parent group manager that a user has been revoked. If this is not executed, GM_k does not learn anything on the identity of the user revoked by GM_l.

Revocation($M_i, \mathcal{G}_l$): This recursive algorithm is run by the group manager GM_l of $\mathcal{G}_l$ who wants to revoke a member M_i of $\mathcal{G}_l$. It takes as input the revocation token rt_i^l of the user M_i and the revocation list RL_l of $\mathcal{G}_l$.

1. *Local Revocation:* It returns an updated RL_l where the revocation token rt_i^l of M_i for $\mathcal{G}_l$ has been added.
2. *Downwards Revocation* (compulsory): The newly published revocation token rt_i^l is sent to the GM's of the groups $\mathcal{G}_m$ that are children of $\mathcal{G}_l$, that then run the **Revocation($M_i, \mathcal{G}_m$)** algorithm, after a computation enabling them to retrieve rt_i^m from rt_i^l.
3. *Upwards Revocation* (optional): GM_l sends an information $rt_i^{k \dashv l}$ to the GM_k of the group $\mathcal{G}_k$ that is the parent of $\mathcal{G}_l$, who can then decide to revoke (in that case we will say that the upwards revocation has been accepted) or not the user, using $rt_i^{k \dashv l}$ to retrieve the user's revocation token rt_i^k for $\mathcal{G}_k$.

3.2 Requirements

We here describe the security properties that we expect from cross-unlinkable group signatures. Correctness is the same property as in the mono-group setting. Traceability and Cross-Unlinkability are adaptations of Traceability and Selfless-Anonymity to our hierarchical group signature setting, with privacy issues in mind.

Correctness: The signature of a member M_i who is registered to the group $\mathcal{G}_l$ and who is not revoked from this group, using the **Sign** algorithm is accepted by any verifier that follows the protocol.

Traceability ensures that if a signature σ for the group $\mathcal{G}_l$ is checked as valid, then the manager GM_l of $\mathcal{G}_l$ is able to find who made σ, and it is impossible to mislead him.

Traceability: Setup: The challenger **C** runs the **KeyGen** algorithm, playing the role of all the GM's. He obtains gpk and (mpk^l, msk^l), for each $\mathcal{G}_l$. He provides **A** with all the mpk^l's and gpk.

Queries: **A** can make the following queries to the challenger:

- *Enrol to $\mathcal{G}_0$:* **A** requests the enrolment of a member M_i to $\mathcal{G}_0$. **C** obtains the keys for M_i for $\mathcal{G}_0$. **A** obtains nothing.
- *Derivation:* **A** requests the enrolment of a member M_i to $\mathcal{G}_l$, provided that M_i is already registered to the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$. **C** obtains the keys for M_i for the group $\mathcal{G}_l$. **A** obtains the keys only if M_i is corrupted. **A** also obtains all the informations that were exchanged on public channels during the derivation process.
- *Sign:* **A** requests that a member M_i of a group $\mathcal{G}_l$ signs for $\mathcal{G}_l$ using a chosen challenge message m. The challenger computes the signature σ of M_i on m. **A** obtains σ.
- *User Corruption:* **A** requests the corruption of a given member M_i. He obtains all the secret keys of M_i for all the groups to which he is registered. M_i is revoked from $\mathcal{G}_0$, and, through Downwards Revocation, from every group to which he belongs.
- *GM Corruption:* **A** requests the corruption of the manager GM_l of a group $\mathcal{G}_l$. He obtains the secret key msk^l of GM_l, and all the informations stored by GM_l: enrolment informations and the keys (and consequently the revocation tokens rt_i^l) of the members of $\mathcal{G}_l$.
- *Revocation:* **A** requests the revocation of a user M_i from a group $\mathcal{G}_l$ (and consequently, from all group under $\mathcal{G}_l$ in $\mathbb{G}$, through Downwards Revocation). He learns all the revocation tokens of M_i that are disclosed during this phase. He can optionally request upwards revocations and, in this case, he learns the informations sent by GM_l to the managers of the parent group. If he wants, he can then do a *Revocation* request on this user for this parent group.

Output: **A** outputs a challenge message m, a signature σ and a group $\mathcal{G}_l$ such that GM_l has not been corrupted. **A** wins the game if:

1. **Verify**(σ,m,$\mathcal{G}_l$)=valid;
2. **A** did not obtain σ by making a signing query on m.

The scheme is said to satisfy *Traceability* if no polynomial probabilistic adversary is able to win the above game with a non-negligible probability.

The *Cross-Unlinkability* property is an extension of the Selfless-Anonymity property to the hierarchical group setting. It is not an extension of the Backward Unlinkability property, since we only use one period in the **Sign** algorithm.

The CU property ensures that a signature issued for the group $\mathcal{G}_l$ remains anonymous even for the GM's of other groups, for instance the parent or the sibling groups in $\mathbb{G}$.

We also insist on the fact that, in case of a revocation, if GM_l does not inform the manager GM_k of the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$ that a given user is revoked from $\mathcal{G}_l$, the manager of $\mathcal{G}_k$ is not able to know about the identity of this user.

Cross-Unlinkability: Consider the following CU game played by an adversary **A**:

Setup: The challenger **C** runs the **KeyGen** algorithm, playing the role of all the GM's. He obtains gpk and (mpk^l, msk^l), for each $\mathcal{G}_l$. He provides **A** with all the mpk^l's and gpk.

Queries: **A** can make the following queries to the challenger:

- *Enrol to $\mathcal{G}_0$:* **A** requests the enrolment of a member M_i to $\mathcal{G}_0$. **C** obtains the keys for M_i for $\mathcal{G}_0$. **A** obtains nothing.
- *Derivation:* **A** requests the enrolment of a member M_i to $\mathcal{G}_l$, provided that M_i is already registered to the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$. **C** obtains the keys of M_i for the group $\mathcal{G}_l$. **A** obtains these keys only if M_i is corrupted. **A** also obtains all the informations that were exchanged on public channels during the derivation process.
- *Sign:* **A** requests that a member M_i of a group $\mathcal{G}_l$ signs for $\mathcal{G}_l$ using a chosen challenge message m. The challenger computes the signature σ of M_i on m. **A** obtains σ.
- *User Corruption:* **A** requests the corruption of a given member M_i. He obtains all the secret keys of M_i for all the groups to which he is registered.
- *GM Corruption:* **A** requests the corruption of the manager GM_l of a group $\mathcal{G}_l$. He obtains the secret key msk^l of GM_l, and all the informations stored by GM_l: enrolment informations and the keys (and consequently the revocation tokens rt_i^l) of the members of $\mathcal{G}_l$.
- *Revocation:* **A** requests the revocation of a user M_i from a group $\mathcal{G}_l$ (and consequently, from all group under $\mathcal{G}_l$ in $\mathbb{G}$, through Downwards Revocation). He learns all the revocation tokens of M_i that are disclosed during this phase. He can optionally request upwards revocations and, in this case, he learns the informations sent by GM_l to the managers of the parent group. If he wants, he can then do a *Revocation* request on this user for this parent group.

Challenge: **A** outputs two challenge messages m and m', two different members M_0 and M_1 and two groups $\mathcal{G}_k$ and $\mathcal{G}_l$. They must be such that:

1. M_0 and M_1 are registered to both $\mathcal{G}_k$ and $\mathcal{G}_l$;
2. **A** corrupted neither M_0 nor M_1;
3. **A** corrupted at most one among GM_k and GM_l. And none is corrupted if $\mathcal{G}_k = \mathcal{G}_l$;
4. M_0 and M_1 are revoked from at most one (and the same for both) group, and, if this is the case, the GM of the other group is not corrupted. If $\mathcal{G}_k = \mathcal{G}_l$, neither M_0 nor M_1 is revoked from $\mathcal{G}_k$.

This implies in particular that in the case where M_0 or M_1 is revoked from $\mathcal{G}_l$ (resp. $\mathcal{G}_k$) and if $\mathcal{G}_k$ is the parent of $\mathcal{G}_l$ (resp. $l \dashv k$), upwards revocation should not be executed and accepted by the parent group manager.

C chooses two random bits $b, b' \in_R \{0,1\}$ and signs for M_b on behalf of $\mathcal{G}_k$ using message m and for $M_{b'}$ on behalf of $\mathcal{G}_l$ using message m'. The respective signatures σ^* and σ'^* are transmitted to **A**

Restricted Queries: **A** can make the same queries as in the *Queries* phase, as long as he does not contradict the above requirements 1 to 4 of the Challenge phase.

Output: **A** outputs a guess $\beta' \in \{0,1\}$ on the boolean $\beta = (b == b')$.

The scheme is said to satisfy *Cross-Unlinkability* if the probability $|\Pr(\beta = \beta') - 1/2|$ is negligible.

We can also define a multi-group version of **Exculpability**, as a straightforward transposition of the definition of Exculpability for VLR group signatures (see [8, 10]) to our hierarchical setting.

4 A Construction of Hierarchical Cross-Unlinkable Group Signatures

In this section we describe our proposal for cross-unlinkable group signatures that follows the model described in the previous section and fulfils the required properties. This proposal is presented using the CL and BP schemes [8] but could use any VLR group signature satisfying Backward Unlinkability such as [15–17, 19]. We also give the proofs that our construction meets the security requirements described in Section 3.2.

4.1 The Protocol

We recall that groups are organized as a tree $\mathbb{G}$ with a root $\mathcal{G}_0$. We use the BP construction with the requirement that each group signature linked to a group $\mathcal{G}_k$ to have one period per child $\mathcal{G}_l$ of $\mathcal{G}_k$ in $\mathbb{G}$, called the "$k \dashv l$" period, that will be used in the Derivation process, but never in the **Sign** algorithm, where we use a CL signature with the same parameters.

KeyGen(λ) GM_0 runs the **KeyGen**$_{CL}$ algorithm to generate the public parameters $gpk^0 = gpk$. Then each GM_l, including GM_0, creates a CL/BP group key pair (mpk^l, msk^l) using the same group parameters $gpk^l = gpk$.

The msk^l's are kept secret by the GM's. gpk and all the mpk^l's are published. Moreover, all GM_l's agree on a random choice of period tokens. In every group $\mathcal{G}_k$,one token $h_{k \dashv l}$ per child $\mathcal{G}_l$ is required for the "$k \dashv l$" periods. All theses tokens are made public.

For the **Enrolment** phase, we assume that M_i has fulfilled all the conditions to acquire a key for $\mathcal{G}_l$. We distinguish two cases of enrolments, either a first enrolment to a group of $\mathbb{G}$, or an enrolment taking place after a derivation process.

Enrolment$(M_i, \mathcal{G}_l)$ M_i and GM_l jointly run the $\textbf{Join}_{BP}(M_i, msk^l, gpk, mpk^l)$ algorithm of the BP group signature of $\mathcal{G}_l$, following two cases:

- *Enrolment to* $\mathcal{G}_0$*:* GM and M_i follow the BP protocol, in particular x_i^0 is randomly chosen by GM. It outputs a secret key $sk_i^0 = (f_i^0, x_i^0, A_i^0)$, a global revocation token $rt_i^0 = x_i^0$ and period revocation tokens $rt_i^{0 \dashv m} = (h_{0 \dashv m})^{x_i^0}$, for each child $\mathcal{G}_m$ of $\mathcal{G}_0$ in $\mathbb{G}$.
- *Enrolment after a Derivation:* GM uses the output of the **Derivation** as the choice for x_i^l. It outputs a secret key $sk_i^l = (f_i^l, x_i^l, A_i^l)$, a global revocation token $rt_i^l = x_i^l$ and period revocation tokens$rt_i^{l \dashv m} = (h_{l \dashv m})^{x_i^l}$, for each child $\mathcal{G}_m$ of $\mathcal{G}_l$ in $\mathbb{G}$.

At the end of this algorithm, GM_l stores x_i^l, A_i^l, rt_i^l and the period revocation tokens. M_i gets $sk_i^l = (f_i^l, x_i^l, A_i^l)$. This phase is partially done in a secure way so that no eavesdropper can learn the keys (x_i^l, A_i^l) that are sent by GM_l to M_i.

We now explain how to derive signing keys. Let $\mathcal{G}_k$ be the parent group of $\mathcal{G}_l$ in $\mathbb{G}$ and let us assume that a user M_i owns keys for $\mathcal{G}_k$ and wants to acquire keys for the group $\mathcal{G}_l$. M_i has to engage a specific authentication process with the group manager GM_l of $\mathcal{G}_l$.

First, the user authenticates to GM_l by signing on behalf of $\mathcal{G}_k$, parent of $\mathcal{G}_l$ in $\mathbb{G}$, to prove that he is allowed to join $\mathcal{G}_l$. This signature is associated to the period "$k \dashv l$", dedicated to the derivation from $\mathcal{G}_k$ to $\mathcal{G}_l$. In addition, M_i sends his revocation token $rt_i^{k \dashv l}$ associated to the "$k \dashv l$" period.

The group manager GM_l then acts as a verifier for the group signature of $\mathcal{G}_l$ and first checks the validity of the signature and the fact that M_i is not revoked by executing $\textbf{Verify}_{CL}$ (see Section 2.3) on the signature, using the revocation list RL_k containing the global revocation tokens of all the revoked members of $\mathcal{G}_k$. He then checks that the revocation token $rt_i^{k \dashv l}$ is the one associated to M_i for the period "$k \dashv l$" by executing $\textbf{Verify}_{BP}$ on the signature using a period revocation list set as $\{rt_i^{k \dashv l}\}$; if the Revocation Check fails, then $rt_i^{k \dashv l}$ is valid.

Then, a key derivation is executed in such a way that the expected revocation mechanisms would apply while satisfying the requirements defined in Section 3.2. We are in fact exploiting the BU property, without introducing periods that really represent time. We only use the BU part to the purpose of Cross-Unlinkability while ensuring the possibility of later downwards and upwards revocations. Indeed, the group $\mathcal{G}_l$ will be in a sense seen as a time period for the group signature associated to $\mathcal{G}_k$.

Once GM_l is sure that M_i is an unrevoked member of $\mathcal{G}_k$, M_i and GM_l can run the enrolment procedure for the group $\mathcal{G}_l$ with one requirement: x_i^l is derived from $rt_i^{k \dashv l}$ using a hash function and the secret key msk^l of GM_l: x_i^l =Hash$(msk^l||rt_i^{k \dashv l})$. This will be used by our cascade revocation process, since x_i^l is also the revocation token rt_i^l of M_i for the group $\mathcal{G}_l$. As said before, this enrolment phase must be partially executed in a secure environment.

This derivation process is described in Figure 2.

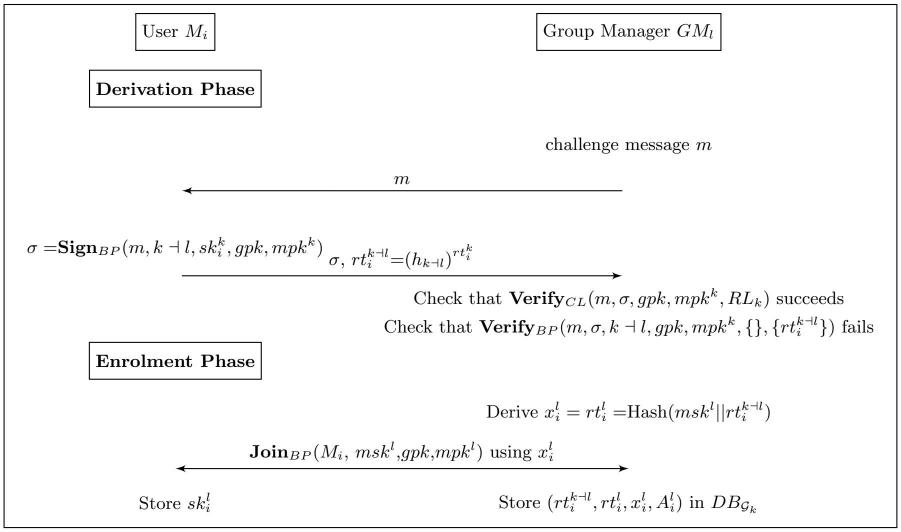
User M_i
Group Manager GM_l
Derivation Phase
challenge message m
m
$\sigma = \mathbf{Sign}_{BP}(m, k \dashv l, sk_i^k, gpk, mpk^k)$
σ, $rt_i^{k \dashv l} = (h_{k \dashv l})^{rt_i^k}$
Check that $\mathbf{Verify}_{CL}(m, \sigma, gpk, mpk^k, RL_k)$ succeeds
Check that $\mathbf{Verify}_{BP}(m, \sigma, k \dashv l, gpk, mpk^k, \{\}, \{rt_i^{k \dashv l}\})$ fails
Enrolment Phase
Derive $x_i^l = rt_i^l$ =Hash$(msk^l||rt_i^{k \dashv l})$
$\mathbf{Join}_{BP}(M_i,\ msk^l, gpk, mpk^l)$ using x_i^l
Store sk_i^l
Store $(rt_i^{k \dashv l}, rt_i^l, x_i^l, A_i^l)$ in $DB_{\mathcal{G}_k}$

Fig. 2. The derivation process

Derivation$(M_i, \mathcal{G}_k, \mathcal{G}_l)$ GM_l sends a message m. M_i signs the challenge message m by executing $\mathbf{Sign}_{BP}(m,\ k \dashv l,\ sk_i^k,\ gpk,\ mpk^k)$. He also sends his associated revocation token $rt_i^{k \dashv l}$.

GM_l checks the signature by executing $\mathbf{Verify}_{CL}(m,\ \sigma,\ gpk,\ mpk^k,\ RL_k)$. In addition, he checks that $rt_i^{k \dashv l}$ has been used in the signature, by checking that the revocation check of $\mathbf{Verify}_{BP}(m, \sigma, k \dashv l, gpk, mpk^k, \{\}, \{rt_i^{k \dashv l}\})$ fails.

M_i and GM_l run the **Enrolment**$(M_i,\mathcal{G}_l)$ algorithm using the *Enrolment after a Derivation* setting with x_i^l =Hash$(msk^l||rt_i^{k \dashv l})$ as input.

GM_l stores in a dedicated database $DB_{\mathcal{G}_l}$ the couple $(rt_i^{k \dashv l}, rt_i^l)$, which will be used for the revocation process.

The **Sign** and **Verify** algorithms are direct applications of the CL scheme algorithms. The CL scheme is dedicated to the signature and is not used for derivation. We will also see in the **Revocation** description that we only use revocation lists made of global revocation tokens. Therefore, we use the $\mathbf{Verify}_{CL}$

algorithm as **Verify** algorithm, **Verify**$_{BP}$ being only used in the derivation process.

Sign$(M_i, m, \mathcal{G}_l)$ M_i runs the **Sign**$_{CL}$ algorithm of the group signature associated to $\mathcal{G}_l$, *i.e.* he executes **Sign**$_{CL}(m,sk_i^l,gpk,mpk^l)$.
Verify$(\sigma, m, \mathcal{G}_l)$ The verifier runs **Verify**$_{CL}(m,\ \sigma,\ gpk,\ mpk^l,RL_l)$.

Let us assume that the group manager GM_l of $\mathcal{G}_l$ wants to revoke a member M_i. He proceeds as follows.

Revocation$(M_i, \mathcal{G}_l)$ **Local Revocation:** GM_l runs the **Revoke**$_{CL}(RL_l, rt_i^l)$ algorithm. The updated RL_l is published.

Downwards Revocation: This direction is automatic. All managers for the children groups $(\mathcal{G}_m)_{m\in M}$ of $\mathcal{G}_l$ in $\mathbb{G}$ learn the revocation token rt_i^l. They all compute $(h_{l\dashv m})^{rt_i^l}$ and look in their databases $DB_{\mathcal{G}_m}$'s if this token is present. If it is, they start the **Revocation**$(M_i, \mathcal{G}_m)$ algorithm for the associated user, using the revocation token rt_i^m associated to the couple containing $(h_{l\dashv m})^{rt_i^l}$ in $DB_{\mathcal{G}_m}$.

Upwards Revocation: We recall that this part of the **Revocation** algorithm is optional. GM_l can report to the manager of the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$ the user M_i if he thinks that GM_k should revoke him too. He sends to GM_k the item $rt_i^{k\dashv l}$ associated to M_i in $DB_{\mathcal{G}_l}$. If GM_k wishes to discover to whom it corresponds, he computes $(h_{k\dashv l})^{rt_{i'}^k}$ for all the $M_{i'}$'s that belong to $\mathcal{G}_k$. When $(h_{k\dashv l})^{rt_{i'}^k} = rt_i^{k\dashv l}$, the associated user $M_{i'}$ is the user M_i that was revoked by GM_l. GM_k can then, if he desires, revoke $M_{i'}$ from $\mathcal{G}_k$.

Notice that, when GM_k does not revoke M_i, if other GM's, for instance the siblings of $\mathcal{G}_l$ in $\mathbb{G}$, have access to the token $rt_i^{k\dashv l}$ sent by GM_l, they cannot link the corresponding member to one of their users, thanks to BU, since $rt_i^{k\dashv l}$ is only a period revocation token and thus cannot be linked to the revocation token of the same user for another period.

4.2 Security Analysis

We now prove that our construction satisfies the requirements defined in Section 3.2. We rely on the fact that the BP scheme satisfies the properties defined in Section 2.2, i.e. Correctness, Backward-Unlinkability, Exculpability and Traceability, as stated in Theorem 1. Therefore, our construction also relies on the q-Strong Diffie-Hellman [4], the adapted Diffie-Hellman [8] and the Discrete Logarithm assumptions.

Theorem 2. *The protocol defined in Section 4.1 achieves Correctness.*

The proof of Correctness is straightforward, as it only requires that all the underlying BP group signatures achieve Correctness.

Theorem 3. *In the random oracle model and under the adapted DDH assumption, the protocol defined in Section 4.1 achieves Cross-Unlinkability.*

Proof (Cross-Unlinkability). Let us assume that there is an adversary **A** that is able to win the CU game with non-negligible probability, we describe how to build an adversary **B** that is able to win the BU game of the BP group signature with non negligible probability. Let $\mathcal{G}^*$ be the group considered in the BP BU game, the challenger **C** considered in the following is the challenger of the BU game and we use a ^{BU} notation to denote a step of the BU game. We model the hash function Hash as a random oracle. When **B** uses it, he picks a uniformly random number while preserving consistency.

B proceeds as follows to play the BU game with **A**.

Setup: The challenger **C** runs the $\mathbf{KeyGen}_{CL}$ algorithm, playing the role of the GM GM^* of $\mathcal{G}^*$. He obtains gpk^*, mpk^* and msk^*. He provides **B** with gpk^* and mpk^*. **B** then builds a group tree $\mathbb{G}$ and picks a random node of this tree to be the one for $\mathcal{G}^*$. He fixes the public parameters for $\mathbb{G}$ to be the public parameters mpk^* of $\mathcal{G}^*$, then chooses secret and public keys for the other groups and provides **A** with the public keys.

We assume that the number of period of the group signature of the BU is bigger than the number of children of $\mathcal{G}^*$ in $\mathbb{G}$. We use, in addition to the $k \dashv l$ periods, another period called "0".

B answers to the Queries of **A** in the following way:

- *Enrol to $\mathcal{G}_0$:* If $\mathcal{G}_0 \neq \mathcal{G}^*$, **B** follows the protocol. Otherwise, he makes a $Join^{BU}$ request to **C** for the considered user M_i.
- *Derivation:* When **B** requests the enrolment of a member M_i to $\mathcal{G}_l$, provided that M_i is already registered to the parent group $\mathcal{G}_k$ of $\mathcal{G}_l$, if $\mathcal{G}_l \neq \mathcal{G}^*$ and $\mathcal{G}_k \neq \mathcal{G}^*$, **B** follows the protocol.
    - If $\mathcal{G}_k = \mathcal{G}^*$, **B** makes a $Sign^{BU}$ request to **C** on M_i for the period "$k \dashv l$". He then makes a $Revocation^{BU}$ query on M_i for the same period to obtain $rt_i^{k \dashv l}$. (This revocation does not impact the sequel, since the user signs at most once for the derivation from $\mathcal{G}_k$ to $\mathcal{G}_l$. Moreover, thanks to BU, there is no impact on anonymity.) He sends both signature and periodic revocation token to **A** and enrols M_i to $\mathcal{G}_l$ following the protocol.
    - If $\mathcal{G}_l = \mathcal{G}^*$, **B** follows the protocol to issue the signature and the periodic revocation token. He then makes a $Join^{BU}$ request to **C** on M_i. As the hash function used to derive keys is modelled as a random oracle, it is impossible for **A** to distinguish between the random choice x_i of **C** and what should have been really issued with msk and $rt_i^{k \dashv l}$. Moreover, consistency of the random oracle is preserved with overwhelming probability (it is almost impossible that **B** also invokes it on $msk || rt_i^{k \dashv l}$).
- *Sign:* If $\mathcal{G}_l \neq \mathcal{G}^*$, **B** follows the protocol. Otherwise, **B** makes a $Sign^{BU}$ request to **C** for period 0 and member M_i and, after having removed the BU part (the items in bold font in the description of Section 2.3), sends the obtained signature σ to **A**.
- *User Corruption:* **B** sends to **A** all the secret keys of M_i for all the groups ($\neq \mathcal{G}^*$) to which M_i is registered. If M_i belongs to $\mathcal{G}^*$, **B** also makes a $Corruption^{BU}$ request on M_i to GM^*, obtains the key of M_i for $\mathcal{G}^*$ and sends it to **A**.

- *GM Corruption:* If $\mathcal{G}_l \neq \mathcal{G}^*$, **B** follows the protocol. Otherwise, he aborts.
- *Revocation:* **B** follows the protocol as far as $\mathcal{G}^*$ is not concerned by the revocation process. If it is, **B** requests the global revocation token of the user from **C** and publishes it. For the Upwards Revocation case, **B** follows the protocol (he is able to do it, since he knows all derivation informations).

Challenge: **A** outputs two challenge messages m and m', two different members M_0 and M_1 and two groups $\mathcal{G}_k$ and $\mathcal{G}_l$ fulfilling the conditions mentioned in the CU game. If $\mathcal{G}_k \neq \mathcal{G}^*$ and $\mathcal{G}_l \neq \mathcal{G}^*$, **B** aborts. Otherwise, we assume w.l.o.g. that $\mathcal{G}_k = \mathcal{G}^*$. **B** chooses a random bit b', signs m' for $M_{b'}$ for the group $\mathcal{G}_l$ to obtain σ'^*. He then proceeds to the $Challenge^{BU}$ phase with **C** with a choice m, M_0, M_1 and period “0”. (It is easy to check that we fulfil the requirements of the $Challenge^{BU}$ phase.) **B** obtains a signature σ^*. Let us denote by b the random bit chosen by **C**. σ^* and σ'^* are transmitted to **A**.

Restricted Queries: **A** and **B** interact as in the *Queries* phase, as long as it does not contradict the requirements of the *Challenge* phase.

Output: **A** outputs a guess $\beta' \in \{0,1\}$ on the boolean $\beta = (b == b')$. If $(\beta' = 1)$, **B** outputs b', else it outputs $1 - b'$.

We can see that the cases of abortion by **B** happen with non-overwhelming probability. Since the advantage of **A** against the CU game is non negligible, so is the advantage of **B** against the BU game, which would contradict the fact that the BP scheme is backward-unlinkable.

Thus, our protocol achieves Cross-Unlinkability. □

Theorem 4. *In the random oracle model and under the q-SDH assumption, the protocol defined in Section 4.1 achieves Traceability.*

Proof (Traceability). Let us assume that there is an adversary **A** that is able to win the Traceability game with non-negligible probability, we describe how to build an adversary **B** that is able to win the Traceability game of the BP group signature with non negligible probability. Let $\mathcal{G}^*$ be the group considered in the BP Traceability game, the challenger **C** considered in the following is the challenger of the BP Traceability game. We model the hash function Hash as a random oracle. When **B** uses it, he picks a uniformly random number while preserving consistency.

We proceed exactly as in the CU proof for the Setup and the Queries phases. We now explain how to manage the Output phase.

Output: **A** outputs a message m, a signature σ and group $\mathcal{G}_l$. If $\mathcal{G}_l \neq \mathcal{G}^*$, abort. Otherwise, **B** outputs m, σ and period “0” to **C**. Then, as the requirements for both considered games are equivalent, if **A** wins the BP Traceability game, **B** wins the Traceability game.

We can see that the cases of abortion by **B** happen with non-overwhelming probability. Since the advantage of **A** against the BP Traceability game is non negligible, so is the advantage of **B** against the Traceability game, which would contradict the fact that the BP scheme is traceable.

Thus, our protocol achieves Traceability. □

The proof that Exculpability of the underlying BP group signature scheme implies Exculpability of the protocol is straightforward.

5 Application to Anonymous Authentication

In [7], Bringer *et al.* suggest to use VLR group signatures to build a biometric anonymous authentication scheme. Their scheme is based on the [5] scheme but can easily be instantiated using any VLR scheme. It can also easily be extended to a non-biometric setting. We describe it with the notations of Section 2.3 for the users keys.

In the [7] setting, members of a group $\mathcal{G}$ authenticate to service providers P (who are different from GM) while remaining anonymous within $\mathcal{G}$. Moreover, secret keys of the users are derived from an acquisition of a biometric trait. When a user M_i applies to join $\mathcal{G}$, an acquisition b of a biometric trait B is made. The group manager and the user then run the $\mathbf{Join}_{GS}$ algorithm with the additional requirement that $f_i = H(b)$ where H is a hash function. M_i then stores b and x_i, A_i on a device such as a smart-card.

When M_i wants to authenticate to a service provider P, he connects his device to a trusted sensor. He is acquired a fresh biometric trait b'. The sensor also gets b and A_i from the device. If b and b' match, then the sensor signs a challenge message sent by P using $\mathbf{Sign}_{GS}$ with $f_i = H(b)$, x_i and A_i. P then checks the signature using $\mathbf{Verify}_{GS}$ and accepts the authentication if the signature is valid. This authentication process is summed up in Figure 3.

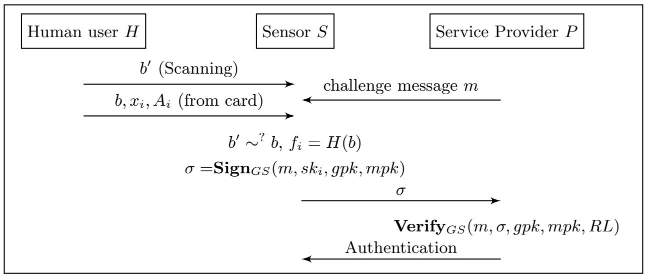
Human user H
Sensor S
Service Provider P
b' (Scanning)
challenge message m
b, x_i, A_i (from card)
$b' \sim^? b$, $f_i = H(b)$
$\sigma = \mathbf{Sign}_{GS}(m, sk_i, gpk, mpk)$
σ
$\mathbf{Verify}_{GS}(m, \sigma, gpk, mpk, RL)$
Authentication

Fig. 3. The [7] authentication scheme.

We can adapt our hierarchical setting to the [7] setting. In this case, several hierarchical groups are available to users and they can anonymously authenticate towards service providers requiring belonging to one or several of these groups. The group signatures associated to these groups are then cross-unlinkable hierarchical group signatures as described in this paper. The adaptation is straightforward and the use of biometrics does not impact our constructions, since we had no requirements on the f_i^l parts of the secret keys of the users.

This adaptation of the [7] authentication scheme can for instance be a basis for an identity management system where the groups are the identity domains and where users get identities for these domains, which are not linkable to each other, except in case of revocation.

A more detailed presentation of the adaptation of cross-unlinkable group signatures to biometric identity management can be found in [6].

References

1. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Financial Cryptography*, pages 196–211, 1999.
2. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO*, pages 108–125, 2009.
3. V. Benjumea, S. G. Choi, J. Lopez, and M. Yung. Fair traceable multi-group signatures. In *Financial Cryptography*, pages 231–246, 2008.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177, 2004.
6. J. Bringer, H. Chabanne, and A. Patey. An application of a group signature scheme with backward unlinkability to biometric identity management. In *SECRYPT*, 2012.
7. J. Bringer, H. Chabanne, D. Pointcheval, and S. Zimmer. An application of the Boneh and Shacham group signature scheme to biometric authentication. In *IWSEC*, pages 219–230, 2008.
8. J. Bringer and A. Patey. VLR group signatures: How to achieve both backward unlinkability and efficient revocation checks. In *SECRYPT*, 2012. Full version available on the IACR Cryptology ePrint Archive: `http://eprint.iacr.org/2011/376`.
9. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.
10. L. Chen and J. Li. VLR group signatures with indisputable exculpability and efficient revocation. In *SocialCom/PASSAT*, pages 727–734, 2010.
11. G. Fuchsbauer and D. Pointcheval. Anonymous proxy signatures. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 201–217. Springer, 2008.
12. D. Khader. Attribute based group signature with revocation. *IACR Cryptology ePrint Archive*, 2007:241, 2007.
13. D. Khader. Attribute based group signatures. *IACR Cryptology ePrint Archive*, 2007:159, 2007.
14. S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *ISW*, pages 273–281, 1997.
15. B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS*, pages 498–517, 2009.
16. T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT*, pages 533–548, 2005.

17. T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *IWSEC*, pages 17–32, 2006.
18. M. Trolin and D. Wikström. Hierarchical group signatures. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 446–458. Springer, 2005.
19. S. Zhou and D. Lin. Shorter verifier-local revocation group signatures from bilinear maps. In *CANS*, pages 126–143, 2006.