A Code-Based Group Signature Scheme

Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, Philippe Gaborit

A Code-Based Group Signature Scheme

Quentin Alamélou[1], Olivier Blazy[1], Stéphane Cauchie[2], and Philippe Gaborit[1]

[1]XLIM-DMI, Université de Limoges, France
{quentin.alamelou,olivier.blazy,philippe.gaborit}@xlim.fr
[2]RD Worldline, Seclin, France, stephane.cauchie@worldline.com

March 16, 2015

Abstract

In this work we propose the first code-based group signature. As it will be described below, its security is based on a relaxation of the model of Bellare, Shi and Zhang [3] (BSZ model) verifying the properties of *anonymity*, *traceability* and *non-frameability*. Furthermore, it has numerous advantages over all existing post-quantum constructions and even competes (in terms of properties) with pairing based constructions: it allows to dynamically add new members and signature and public key sizes are constant with respect to the number of group members. Last but not least, our scheme can be extended into a traceable signature according to the definition of Kiayias, Tsiounis and Yung [19] (KTY model) and handles membership revocation. The main idea of our scheme consists in building a collision of two syndromes associated to two different matrices: a random one which enables to build a random syndrome from a *chosen* small weight vector; and a *trapdoor matrix* for the syndrome decoding problem, which permits to find a small weight preimage of the previous random syndrome. These two small weight vectors will constitute the group member's secret signing key whose knowledge will be proved thanks to a variation of Stern's authentication protocol. For applications, we consider the case of the code-based CFS signature scheme [11] of Courtois, Finiasz and Sendrier.

1 Introduction

A group signature scheme allows members of a group to issue signatures on behalf of the group in an anonymous but revocable way: an opener is able to revoke anonymity of the actual signer in case of abuse. Since its introduction by Chaum and van Heyst [10], group signature has been extensively studied. Bellare et al. [2] (BMW model) first gave formal security properties of group signature. Later, Bellare, Shi and Zhang [3] extended this model to dynamic groups (BSZ model). Numerous efficient group signatures such as [7, 8, 9] were proposed but only proven secure in a relaxation security of [2]. Delerablée and Pointcheval [12] proposed the first practical scheme fully fitting BSZ in the random oracle model (ROM) whereas Groth [18] also provided such a scheme but secure in the standard model. Then, as an improvement of group signatures, Kiayias, Tsiounis and Yung,

suggested traceable signatures schemes in [19]. In addition to classic properties of a group signature scheme, a traceable signature enables the opening authority to delegate its tracing capability to sub-openers but only against specific users. This gives two crucial advantages: sub-openers can run in parallel and authorities can monitor misbehaving users and then preserve honest users anonymity. The first efficient traceable signatures, provably secure in the standard model, were introduced by Libert and Yung in [22].

All these aforesaid schemes are pairing-based constructions. It was then worth looking for alternative since their security might collapse in front of quantum computers and that they involve heavy computations. Thus, many lattice-based constructions have been proposed such as [17] who first designed a lattice-based group signature scheme with both public key and signature size linear in the number of group members N. Recently, numerous works such as [20, 21, 25, 24] proposed more efficient lattice-based constructions where both sizes of the group public key and the signature are proportional to $log(N)$. Another crucial requirement for group signature schemes is the membership revocation: only few lattice-based constructions, namely [21, 24], handle this property. Plus, it is interesting to notice that all lattice-based group signature schemes base their security on the static model of [2] meaning that, in our knowledge, there exists no post-quantum dynamic group signature scheme.

In this work, we present the first code-based group signature scheme which furthermore addresses many of these issues. Indeed, our work can be seen as the first (weakly) dynamic post-quantum traceable signature with membership revocation. Lattice-based constructions such as [20, 21, 25, 24] are secure in the sense of the BMW model then providing full-anonymity and traceability. In our scheme, we provide traceability, non-frameablity and only anonymity but in a stronger model since it allows to dynamically add new group members. Indeed, basing our security notions on the BSZ model, we prove the security of our scheme through the use of only one oracle for achieving a passive *Join* whereas the BSZ model requires two oracles respectively proceeding to a passive join and an active join; this led us to define our scheme as *weakly dynamic*.

The main idea of our scheme consists in building a collision of two syndromes associated to two different matrices: a random one which enables to build a random syndrome from a *chosen* small weight vector; and a *trapdoor matrix*, which permits to find a small weight preimage of the previous random syndrome. These two small weight vectors will constitute the group member's secret signing key whose knowledge will be proved thanks to a variation of Stern's protocol.

Our contribution In this work, we propose a generic construction for designing the first code-based group signature. As explained above, its security is based on a relaxation of the BSZ model with the properties of *anonymity*, *traceability* and *non-frameability*. Furthermore, it has numerous advantages over all existing post-quantum constructions and even some pairing based constructions: it allows to dynamically add new members (*weakly dynamic*) and signature and public key sizes are constant with respect to the number of group members. Last but not least, our scheme may be extended into a traceable signature according to the KTY model and handles membership revocation. For applications, we consider the CFS signature [11] for an instantiation of our scheme.

Organization In Section 2, we give necessary background for the well understanding of our work; Section 3 deals with the variation we brought to Stern's protocol in order to design our group signature scheme; we then present our group signature scheme in Section 4 where its extension to a *traceable signature* in the KTY model and the case of revocation are also highlighted. Section 5 is concerned with the security of our scheme and finally we give an example and parameters for building an instance of our scheme in Section 6.

2 Notation and Background

In this section, we first present the notation used throughout this work, we then make a brief focus on code-based cryptography and finally provide group signature definitions.

2.1 Notation

All through this work, we use the following notations.
μ denotes some randomness and we use the symbol $\|$ for concatenation.
$v[r]$ denotes the $r-th$ coordinate of the vector v and s_r denotes the $r-th$ symbol of the string s. The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. $\mathbb{F}_q$ denotes the finite field of cardinality q. $\mathcal{M}_{m\times n}(\mathbb{F}_q)$ denotes matrices over $\mathbb{F}_q$ of m lines and n columns. S_ω^n is the set of vectors of weight ω lying in $\mathbb{F}_2^n$. Σ_n denotes the set of permutations over $[n]$. $\mathcal{H}$ denotes a generic random oracle. $h : \mathbb{F}_2^* \longrightarrow \mathbb{F}_2^n$, $h' : \mathbb{F}_2^* \longrightarrow \mathcal{M}_{k\times n}(\mathbb{F}_2)$ and $h'' : \mathbb{F}_2^* \longrightarrow \{0, 1, 2\}^l$ model random oracles.

For any entity $\mathcal{E}$, we denote by $\mathcal{E}(I)$ the fact that $\mathcal{E}$ has knowledge of I. For protocols, we denote by $\mathcal{P}$ and $\mathcal{V}$ respectively the prover and the verifier.

We use usual coding theory notation, where G and H respectively denote generator and parity check matrices of a code. Let $H \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$ and $x \in \mathbb{F}_q^n$. The product Hx^T is called *a syndrome* and $\omega t(x)$ refers to the Hamming weight of x.

2.2 Code-based Cryptography Background

In this subsection, we only give necessary recalls for the well understanding of our work; for more details on coding theory, see [23].

Syndrome Decoding Problem The Syndrome Decoding problem (SD-problem) is a problem based on coding theory shown NP-complete [4]. It consists in finding a small weight word for a given syndrome s.

Definition 1 *Let H be a random matrix from $\mathcal{M}_{m\times n}(\mathbb{F}_q)$, ω an integer and $s \in {\mathbb{F}_q}^{n-k}$. The Syndrome Decoding problem consists in finding e of weight below or equal to ω such as $He^T = s$.*

Minimum Distance Problem The Minimum Distance problem (MD-problem) is also a problem based on coding theory shown NP-complete in [27]. It consists in finding a small weight word whose syndrome is null.

Definition 2 *Let H be a random matrix from $\mathcal{M}_{k\times n}(\mathbb{F}_q)$, ω an integer. The Minimum Distance problem consists in finding e of weight below or equal to ω such as $He^T = 0$.*

We say that any tuple with coherent dimensions $(H,\ s,\ \omega)$ where s is a syndrome by H is an SD-instance. When $s = 0$, this is an MD-instance and we denote it $(H,\ \omega)$.

Stern's protocol Whereas zero-knowledge (ZK) protocols were proposed by [16], Stern [26] first proposed such a scheme based on codes. Stern's protocol (Figure 1) is a 3-pass prover-verifier protocol with cheating probability equal to 2/3 during which $\mathcal{P}$ makes a zero-knowledge proof to $\mathcal{V}$ on a small weight secret z solving an SD-instance (H, s). The security on the scheme relies on the SD-problem.

Public data: $(H, s, \omega) \in \mathcal{M}_{k\times n}(\mathbb{F}_2) \times {\mathbb{F}_2}^{n-k} \times \mathbb{N}$

$\mathcal{P}$ proves knowledge of $z \in \mathbb{F}_2^n$ such as $Hz^T = s$ and $\omega t(z) = \omega$

1. $\mathcal{P}$ computes:
   $\pi \xleftarrow{\$} \Sigma_n$
   $u \xleftarrow{\$} \mathbb{F}_2^n$
   $c_1 = h(\pi||Hu^T), c_2 = h(\pi(u)), c_3 = h(\pi(z+u))$
2. $\mathcal{P}$ sends c_1, c_2, c_3
3. $\mathcal{V}$ sends $c \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cases:
   $c = 0$: $\mathcal{P}$ sends π, u
   $c = 1$: $\mathcal{P}$ sends $\pi, z + u$
   $c = 2$: $\mathcal{P}$ sends $\pi(z), \pi(u)$
5. 3 cases:
   $c = 0$: $\mathcal{V}$ checks c_1, c_2
   $c = 1$: $\mathcal{V}$ checks c_1, c_3
   $c = 2$: $\mathcal{V}$ checks c_2, c_3
   and $\omega t(\pi(z)) = \omega$
6. $\mathcal{V}$ outputs *Accept* if all checks passed
   $\perp$ otherwise.

Figure 1: Stern's protocol

Remark 3 *Remark on Stern's protocol*

If $\mathcal{V}$ knows the secret z, he can check more assumptions while executing Stern's protocol. Indeed, for the case $c = 0$, $\mathcal{V}(z)$ receives π and u so he can also check that $c_3 = h(\pi(z + u))$. For the case $c = 1$, $\mathcal{V}(z)$ receives π and $z + u$ so he can check that $c_2 = h(\pi(z + u + z))$. Then, we can say that a prover whose secret is known by the verifier can be *traced* while executing Stern's protocol.

Fiat-Shamir Paradigm Fiat and Shamir proposed in [14] a general paradigm for designing a signature scheme from a secure identification scheme. The idea is to start from a secure 3-round public coin identification scheme (with α a commitment from the prover, β a random challenge from the verifier, and *ans* the response to β sent by the prover), and then turn into a digital signature scheme with the help of the random oracle $\mathcal{H}$. Indeed, to sign a message m, the signer (who knows the secret) produces a valid transcript (α, β, ans) of the interactive protocol where $\beta = \mathcal{H}(\alpha, m)$.

Trapdoor Matrix In this work, we propose a generic construction of a code-based group signature scheme through the use of what we call a *trapdoor matrix*. Such a matrix is actually hard to find and the only current candidate for instantiating our construction is the CFS matrix (see section 5).

Definition 4 *A trapdoor matrix family is a tuple of polynomial algorithms* $(TrapGen, Eval, Inv)$ *such that:*

- $TrapGen(1^\lambda)$*: outputs a pair* $(Q, trk) \in \mathcal{M}_{k\times n}(\mathbb{F}_2) \times \mathcal{TRK}$ *with* Q *indistinguishable from random;*
- $Inv(Q, trk, s, \omega)$ *outputs some* $x \in \mathbb{F}_2^n$ *such as* $Qx^T = s$ *and* $\omega t(x) = \omega$ *assuming* $(Q, trk) \leftarrow TrapGen(1^\lambda)$, $s \in \mathbb{F}_2^{n-k}$ *and* $\omega \in \mathbb{N}$*;*
- *(Correctness) For all* (Q, trk) *output by* $TrapGen(1^\lambda)$*, and all* $s \in \mathbb{F}_2^n$*. We have* $Q(Inv(Q, trk, s, \omega))^T = s$*;*
- *(One-wayness) For all polynomial adversary* $\mathcal{A}$*, the following is negligible:* $Pr[(Q, trk) \leftarrow Gen(1^\lambda); s \in \mathbb{F}_2^{n-k}; x \leftarrow \mathcal{A}(1^\lambda, Q, s, \omega) : (Qx^T = s \, and \, \omega t(x) = \omega)]$.

We say that Q is a *trapdoor matrix* and trk a *trapdoor key* if (Q, trk) was generated by *TrapGen*.

2.3 Group Signature

A group signature scheme [10] is a protocol which allows members of a group to individually issue signatures on behalf of the group in an anonymous but revocable way: an opener is able to revoke anonymity of the actual signer in case of abuse. Several steps have been made in the study of those protocols: Bellare et al. [2] first gave formal security properties of group signature. Later, Bellare, Shi and Zhang (BSZ model) [3] extended this model to dynamic groups, emphasizing the importance of unforgeability and anonymity. Numerous efficient group signatures schemes such as [7, 8, 9] using bilinear maps were proposed but only secure in a relaxation of these models. While recent post-quantum group signature schemes, namely lattice-based, such as [17, 20, 21, 25, 24] only satisfy the static model of [2], we propose a scheme that we will define as *weakly dynamic* with the classic properties of *anonymity*, *traceability*, *non-frameability* for which key and signature sizes are independent of the number of group members.

All throughout this section, we adapted our definition and our security model from [5] which was the closest one to ours.

2.3.1 Definition

We first precise that our scheme only involves three entities with a single authority: the group manager. Indeed, in our model, the group manager will both participate in issuing users' secret keys and revoking anonymity (see section 4) without impacting security (see section 5). Consequently, the algorithm *Judge* present in dynamic models [3, 5] does not appear in our model. This difference apart, we adapt [5] to propose the following definition.

Definition 5 *A group signature scheme* $\mathcal{GS} = (Setup, Join, Sign, Verif, Open)$ *is a sequence of protocols such as:*

- $Setup(1^\lambda)$*: this algorithm generates global public parameters of the system params, the group public key gpk and the group manager secret key gmsk encompassing the opening key, skO;*

– *Join*($\mathcal{U}_i$)*: this is an interactive protocol between a user* $\mathcal{U}_i$ *and the group manager. At the end of the protocol, the user obtains a secret signing key* $sk[i]$*. The group manager adds the new user* $\mathcal{U}_i$ *and updates* skO*;*

– *Sign*($gpk, sk[i], m; \mu$)*: to sign a message* m*, the user uses his secret key* $sk[i]$ *and some randomness* μ *to output a signature* σ *valid under the group public key* gpk*;*

– *Verif*(gpk, m, σ)*: anybody should be able to verify the validity of the signature* σ *on the message* m *with respect to* gpk*. It thus outputs* 1 *if the signature is valid, and* 0 *otherwise;*

– *Open*(skO, gpk, m, σ)*: for a valid signature* σ *with respect to* gpk*, the group manager can provide the signer identity. It thus outputs the user* $\mathcal{U}_i$*.*

2.3.2 Security Model

Following [5], we define our security notions in a game-based way defined in Figures 2 and 3. To be claimed secure, a group signature scheme has to prove its *correctness* and fulfill three properties: *anonymity*, *traceability* and *non frameability.*

Correctness The *correctness* notion guarantees that honest users should be able to generate valid signatures, and the opener should then be able to revoke anonymity of the signers.

Unforgeability Informally, the unforgeability notion guarantees that no one can produce a valid signature that cannot be opened in convincing way (traceability) and that no one can produce a signature on behalf of some group member (non-frameability).

In the following experiments, to join the group, an adversary runs the $joinP$-oracle (passive join), which means that it creates an honest user for whom it does not know the secret key: the index i is added to the HU (Honest Users) list.

Remark 6 *The BSZ model defines two Join oracles: a passive one as described above and an active one where an adversary can proceed the Join with a user already corrupted. The adversary then knows user's secret key from the beginning. Theses two oracles are necessary to be dynamic in the sense of the BSZ model.*

For users whose secret keys are known to the adversary, we let the adversary play on their behalf. For users whose secret keys are known to the adversary, we let the adversary play on their behalf. For honest users, the adversary can interact with them, granted some oracles:

– *corrupt*(i), if $i \in HU$, provides the secret key $sk[i]$ of this user. The adversary can now control it. The index i is then moved from HU to the list of corrupted users CU;

– *sign*(i, m), if $i \in HU$, plays as the honest user $\mathcal{U}_i$ would do in the signature process. Then i is appended to the list $S[m]$.

We also define the *open*-oracle which, on input (m, σ) returns *Open*(skO, gpk, m, σ).

(a) Experiment $Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(gpk, gmsk, skO) \leftarrow Setup(1^\lambda)$
2. $(m, \sigma) \leftarrow \mathcal{A}(gpk : joinP, corrupt, sign, open)$
3. if $Verif(gpk, m, \sigma) = 0$, return 0
4. if $\exists j \notin CU \cup S[m]$,
   $Open(gmsk, gpk, m, \sigma) = j$
   return 1
5. else return 0

$$Adv^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

(b) Experiment $Exp^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(gpk, gmsk, skO) \leftarrow Setup(1^\lambda)$
2. $(m, \sigma) \leftarrow \mathcal{A}(gpk, gmsk : joinP, corrupt, sign)$
3. if $Verif(gpk, m, \sigma) = 0$ return 0
4. if $\exists i \in HU \setminus S[m]$,
   $Open(gmsk, gpk, m, \sigma) = i$
   return 1
5. else return 0

$$Adv^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

Figure 2: Unforgeability Notions

Traceability and Non-Frameability

Traceability (see Figure 2 (a)) says that nobody should be able to produce a valid signature that cannot be opened in a convincing way. Furthermore, non-frameability (see Figure 2 (b)) guarantees that no dishonest player (even the authorities, i.e. the Group Manager GM, hence the keys when $gmsk$ is provided to the adversary) will be able to frame an honest user: an honest user that does not sign a message m should not be convincingly declared as a possible signer, non-frameability also shows that the group manager cannot cheat. We thus say that:

- $\mathcal{GS}$ is *traceable* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible;
- $\mathcal{GS}$ is *non-frameable* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible.

In both games, the adversary generates a signature σ on a message m of its choice. In the latter game, the adversary itself can play the role of the opener, trying to frame an honest user i.

Anonymity

Given two of honest users i_0 and i_1, the adversary should not have any significant advantage in guessing which one of them have issued a valid signature.

Experiment $Exp^{anon-b}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(gpk, gmsk, skO) \leftarrow (1^\lambda)$
2. $(m, i_0, i_1) \leftarrow \mathcal{A}(FIND, gpk : joinP, corrupt, sign)$
3. $\sigma \leftarrow Sign(gpk, i_b, m, sk[i])$
4. $b' \leftarrow \mathcal{A}(GUESS, \sigma : joinP, corrupt, sign)$
5. if $i_0 \notin HU$ or $i_1 \notin HU$ return 0
6. return b'

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

Figure 3: Anonymity Notion

The adversary can interact with honest users as before (with *sign* and *corrupt*), but the challenge signature is generated using the interactive signature protocol

Sign, where the adversary plays the role of the corrupted users, but honest users are activated to play their roles.

$\mathcal{GS}$ is *anonymous* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible. The *full-anonymity* notion means that anonymity is guaranteed even if the adversary is granted access to the *open*-oracle (excepted on the challenge signature).

Definition 7 *A group signature scheme verifying security notions of Figures 2 and 3 is said to be weakly-dynamic.*

3 A Variation on Stern's Protocol

For designing our group signature, we proposed a variation of Stern's protocol where the idea consists in splitting the small weight secret z the prover will be challenged on and to run two related instances of Stern's protocol in parallel. Let us consider the following SD-instance: $(H, s, 2\omega)$ for which z is said to be a *valid* solution if $z = (x\|y)$ where x and y have the same length and $\omega t(x) = \omega t(y) = \omega$. Then H can also be written as $H = (R\|Q)$ such as $Hz^T = s \Leftrightarrow Rx^T + Qy^T = s$.

Public data: $(H, s, \omega) \in \mathcal{M}_{k\times 2n}(\mathbb{F}_2) \times \mathbb{F}_2^{n-k} \times \mathbb{N}$

$\mathcal{P}$ proves knowledge of a valid $z = (x\|y) \in \mathbb{F}_2^{2n}$ to $\mathcal{V}$

1. $\mathcal{P}$ generates $rnd := gen(\mu)$ and $\alpha := com(H, z, rnd)$
where *gen* and *com* are defined by:
   $(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$
   $(u, v) \xleftarrow{\$} \mathbb{F}_2^n \times \mathbb{F}_2^n$
   $rnd \leftarrow \{\pi, \sigma, u, v\}$
   $c_1 = h(\pi||Ru^T + Qv^T), c_2 = h(\pi(u)), c_3 = h(\pi(x + u))$
   $d_1 = h(\sigma||Ru^T + Qv^T), d_2 = h(\sigma(v)), d_3 = h(\sigma(y + v))$
   $\alpha \leftarrow \{c_1, c_2, c_3, d_1, d_2, d_3\}$
2. $\mathcal{P}$ sends α
3. $\mathcal{V}$ sends $c \xleftarrow{\$} \{0, 1, 2\}$
4. $\mathcal{P}$ computes $ans := resp(z, c, rnd)$
where *resp* is defined by:
   If $c = 0$: $ans \leftarrow \{\pi, \sigma, u, v\}$
   If $c = 1$: $ans \leftarrow \{\pi, \sigma, x + u, y + v\}$
   If $c = 2$: $ans \leftarrow \{\pi(x), \pi(u), \sigma(y), \sigma(v)\}$
   $\mathcal{P}$ sends ans
5. $\mathcal{V}$ computes $b := check(\alpha, c, ans)$
where *check* is defined by:
   If $c = 0$: $\mathcal{V}$ checks c_1, c_2, d_1, d_2
   If $c = 1$: $\mathcal{V}$ checks c_1, c_3, d_1, d_3
   If $c = 2$: $\mathcal{V}$ checks c_2, c_3, d_2, d_3
   and $\omega t(\pi(x)) = \omega t(\sigma(y)) = \omega$
   If all checks passed
   $b \leftarrow 1$
   Else
   $b \leftarrow 0$
6. $\mathcal{V}$ outputs *Accept* if $b = 1$
   $\perp$ otherwise

Figure 4: Identification protocol ($\mathcal{IP}$)

Security of our identification protocol This protocol is a prover-verifier protocol with cheating probability equal to 2/3 and needs to be repeated several times to decrease this cheating probability close to 0.

Theorem 8 *The identification protocol $\mathcal{IP}$ is an honest-prover verifier zero-knowledge (HPVZK) protocol with cheating probability* 2/3 *thus verifying properties of completeness, soundness and zero-knowledge.*

Idea of proof The identification protocol $\mathcal{IP}$ is a straightforward application of Stern's protocol which thus implies its security. A detailed proof will appear in the extended version of this paper.

We will see in the following how to turn this *HPVZK* identification scheme into a group signature scheme through Fiat-Shamir paradigm.

4 Code-Based Group Signature Scheme

In this section, we reuse notation defined in subsection 2.3 concerning group signature. To fix ideas, we first present an high level overview of our scheme and secondly, we describe precisely the operations of the different algorithms required to cope with our group signature definition (Definition 5).

Let us notice that our scheme requires that all participants have access to a public random matrix R which will constitute half of the group public key gpk; we therefore assume that we are in the CRS model [6] with crs the common reference string. R is obtained by applying h' to crs.

Actors Our scheme brings into play:

- **a group manager (*GM*)**: the single authority of our scheme. It runs the *Setup* algorithm, adds new members to the group (algorithm *Join* protocol) and opens signatures (algorithm *Open*);
- **group members** also referred as **users** who can sign on behalf of the group (algorithm *Sign*);
- **outsiders** which do not belong to the group but can verify a signature granted the group public key gpk (algorithm *Verif*).

4.1 High Level Overview

We give an overview of our scheme by following a classic scenario: first, a candidate $\mathcal{U}$ for joining the group G solicits GM to get a secret signing key; secondly, $\mathcal{U}$ must be able to sign on behalf of the group while finally the group manager must be able to revoke its anonymity.

First Step GM generates Q a trapdoor matrix with trk the corresponding trapdoor key. It is important to notice that Q (and trk) will never change during lifetime of the group ensuring that the public key size is independent of the number of group members.
$\mathcal{U}$ chooses a random vector x of weight ω and computes $s = Rx^T$. $\mathcal{U}$ then sends s to GM who uses its trapdoor key to compute y such as: $Qy^T = s$ and $\omega t(y) = \omega$. GM returns y to $\mathcal{U}$ who then forms its secret signing key $z = (x\|y)$. It is important to notice that half of the secret key, namely x, is only known by $\mathcal{U}$ himself; it will ensure non-frameability of our scheme.

Second Step As summed up in Figure 5, $\mathcal{U}$ owns a secret key z which fits within the model of the protocol $\mathcal{IP}$ for a null syndrome. Indeed since $s = Rx^T = Qy^T$, we have that $Rx^T + Qy^T = 0$ and every group member possesses a secret key verifying $Hz^T = 0$ and $\omega t(x) = \omega t(y) = \omega$.

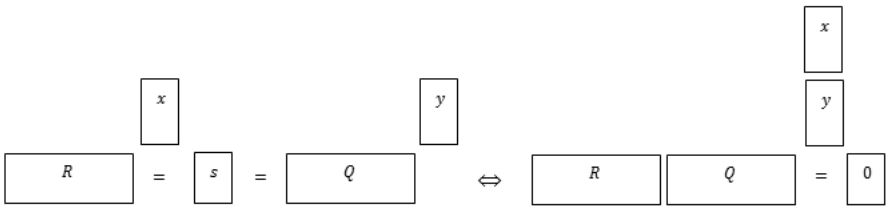

Figure 5: High level Overview

Third Step Due to our construction, *GM* knows y the second part of the secret key of every user $\mathcal{U}$; this point added to remark 6 will enable him to run *Open* algorithm.

4.2 Operation of our Scheme

We first describe algorithms *Setup*, *Join* and summed them up in Figure 6.
Setup is performed by the group manager while $Join(\mathcal{U}_i)$ is an interactive protocol between a candidate $\mathcal{U}_i$ for joining the group and the group manager *GM*.

We recall that we are in the CRS model which enables anyone to generate $R = h'(crs)$.

Setup The *Setup* algorithm is executed by the group manager *GM* taking as input a security parameter λ. It generates a *trapdoor matrix* (according to Definition 7) Q and the corresponding trapdoor key *trk*. It also initializes $gmsk = (trk, skO)$ where *skO* will be its opening key.
Finally, *GM* publishes global parameters $params = (\lambda, n, k, \omega) \in \mathbb{N}^4$ and the group public key $gpk = (H, \omega)$ where R and $Q \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$.

Join To proceed the *Join* protocol, *GM* and $\mathcal{U}_i$ behave as following: $\mathcal{U}_i$ randomly chooses a vector $x_i \in S_\omega^n$ and computes $s_i = Rx_i{}^T$. Then, he sends s_i to *GM* who uses its trapdoor key *trk* to compute y_i verifying: $s_i = Qy_i{}^T$ and $\omega t(y_i) = \omega$. *GM* responds y_i to $\mathcal{U}_i$. Finally, $\mathcal{U}_i$ forms $sk[i] = (x_i \| y_i)$ and *GM* updates $skO[i] = y_i$. When *GM* computes y_i, he first checks that y_i was not already attributed to another user; if so, *GM* tells $\mathcal{U}_i$ to choose another secret x_i (since *GM* cannot check values for x_i's, he has to ensure that all y_i's are different so that two users cannot share the same secret key).

(a) ***Setup***(1^λ)

1. $(\lambda, k, n, \omega) \in \mathbb{N}^4 \leftarrow params(1^\lambda)$
2. $(Q, trk) \in \mathcal{M}_{k\times n}(\mathbb{F}_2) \times \mathcal{TRK} \leftarrow GenTrap(params)$
3. $gpk \leftarrow ((R\|Q), \omega)$
4. $skO \leftarrow \emptyset$
5. $gmsk \leftarrow (trk, skO)$
6. Output $(params, gpk)$

(b) ***Join***(1^λ)

$user : \mathcal{U}_i$		GM
$x_i \xleftarrow{\$} S_\omega^n$		
$s_i = Rx_i{}^T$		
	$\xrightarrow{s_i}$	
		$y_i = Inv(Q, trk, s_i, \omega)$
		If $\exists i_0 : skO[i_0] = y_i$, *Abort*
		Else $skO[i] = y_i$
$sk[i] = (x\|y)$	$\xleftarrow{y_i}$	

Figure 6: *Setup* and *Join* algorithms

***Sign* and *Verif* Algorithms** $\mathcal{IP}$ is an interactive zero-knowledge protocol during which $\mathcal{P}$, which in fact consists in the group G, proves to $\mathcal{V}$ the knowledge of a valid secret ensuring $\mathcal{V}$ that he belongs to the group associated to the null identity. We now describe how we used $\mathcal{IP}$ to build primitives *Sign*, *Verif* and *Open* of our group signature scheme.

General idea As already mentioned, our group signature scheme is obtained from the identification protocol $\mathcal{IP}$ (Figure 4) through the use of Fiat-Shamir paradigm (subsection 2.2). To sign a message m (algorithm *Sign*), a group member $\mathcal{U}_i$ produces a transcript $Tr = (\alpha, \beta, ans)$ of the protocol $\mathcal{IP}$ executed on public key gpk and small weight secret $sk[i]$ simulating the interaction through the use of a random oracle. For verification (algorithm *Verif*), one tries to regenerate the same transcript Tr using $\sigma = (\alpha, ans)$ published along with m by the signer. When, m and the signature σ are valid, the verifier can generate β from m and α and then checks integrity of the message m and the signer's group membership. We detail these algorithms in Figure 7; we reused primitives *gen*, *com*, *resp*, *check* defined in Figure 4 and we defined l_λ as the number of necessary iterations to reach the required level of security λ. h'' must have been chosen such as the size of the output l is greater than l_λ.

(a) ***Sign***$(gpk, sk[i], m, l_\lambda; \mu)$

$\alpha \leftarrow \emptyset$, $rnd \leftarrow \emptyset$, $ans \leftarrow \emptyset$ and $r \leftarrow 0$
1. While $(r < l_\lambda)$
   $rnd[r] \leftarrow gen(\mu)$
   $\alpha[r] \leftarrow com(gpk, sk[i], rnd[r])$
   $r \leftarrow r + 1$
2. $\beta = h''(m, \alpha)$
3. $r \leftarrow 0$
4. While $(r < l_\lambda)$
   $ans[r] \leftarrow resp(sk[i], \beta_r, rnd[r])$
   $r \leftarrow r + 1$
5. Output (m, σ) where $\sigma = (\alpha, ans)$

(b) ***Verif***$(gpk, m, l_\lambda, \sigma)$

1. Split $\sigma = (\alpha, ans)$
   $\beta = h''(m, \alpha)$
2. $r \leftarrow 0$
   While $(r < l_\lambda)$
      $b \leftarrow check(\alpha[r], \beta_r, ans[r])$
      If $(b == 0)$
         Output 0
      $r \leftarrow r + 1$
3. Output 1

Figure 7: *Sign* and *Verif*

***Open* Algorithm** We first deal with the interactive point of view: by splitting the secret (here $sk[i] = (x_i \| y_i)$) into two parts in the protocol $\mathcal{IP}$, it is as if a user $\mathcal{U}_i$ were executing two instances of Stern's protocol: $(R, Rx_i{}^T, \omega)$ and $(T, Ty_i{}^T, \omega)$.

Now, recalling Remark 3 about Stern's protocol and that the manager secret key skO consists in the pool of all the y_i's, if $\mathcal{U}_i$ and GM are executing the identification protocol $\mathcal{IP}$ (Figure 4), GM will be able check the validity of more commitments than a classic verifier would do in two on three cases. Indeed, in function of the challenge c received at step 3 of $\mathcal{IP}$ protocol, GM may check *additional requirements* at step 5:

- if $c = 0$: GM can also check that $d_3 = h(\sigma(y_i + v))$ in addition to c_1, c_2, d_1, d_2;
- if $c = 1$: GM can also check that $d_2 = h(\sigma(y_i + v + y_i))$ in addition to c_1, c_3, d_1, d_3.

We then define the primitive $addCheck(y_i, \alpha, c, ans)$ which acts like *check* but also checks *additional requirements* described above. This primitive can only be run by GM (or possibly by a user) since it requires knowledge of $y_i's$.

How to open a signature ?
The algorithm (Figure 8) is straightforward : given a message m and a signature $\sigma = (\alpha, ans)$, GM proceeds as follows: for each user, he tries to reproduce a valid transcript as in algorithm *Verif* but, as explained above, he also has to check *additional requirements* at during step 2 (Figure 8). If it exists a user i such as all iterations on r are successful, the signer can only be i.

Open(skO, gpk, m, σ)
1. Split $\sigma = (\alpha, ans)$
   $\beta = h''(m, \alpha)$
2. $i \leftarrow 1$
   While $(i \leq nbUsers)$
     $r \leftarrow 0$
     While $(r < l_\lambda)$
       $b \leftarrow addCheck(skO[i], \alpha[r], \beta_r, ans[r])$
       If $(b == 0)$
        $i \leftarrow i + 1$
        $r \leftarrow 0$
       Else if $(b == 1$ *and* $r == l_\lambda - 1)$
        Output i
       Else
        $r \leftarrow r + 1$
3. Output $\perp$

Figure 8: *Open* algorithm

When i is not the signer of m, $addCheck(skO[i], \alpha[r], \beta_r, ans[r])$ has a probability to output 1 equal to 1/3 at step 2 (the case $\beta_r = 2$ is the only one he can pass); This probability is less than the cheating probability of 2/3 of $\mathcal{IP}$ protocol so we are sure that the l_λ rounds required for algorithms *Sign* and *Verify* ensure an opening probability close to 0 for anyone else but GM.

4.3 Extension to a Traceable Signature

In this subsection, we briefly explain how our group signature scheme can be turned into a traceable signature following the definition of [19] and how the same argument enables our scheme to handle revocation.

A traceable signature Extending group signatures schemes, Kiayias et al. suggested traceable signatures schemes in [19]. Such signatures enable the opening authority to delegate its tracing capability to sub-openers so that they can trace suspicious users without letting them trace others.

Due to its construction, our scheme can easily be extended into such a scheme. Indeed if GM wants a sub-opener So to look after $\mathcal{U}_{i_0}$, he just needs to reveal him y_{i_0}, which can be defined as the *tracing key*. From now, when So wants to check a signature he has to apply the methodology of the *Open* algorithm. Since he does not know skO, he cannot look over all users' tracing keys, then he will

only be able to open signatures issued by users he knows tracing keys.

Anonymity revocation A crucial requirement for group signature schemes is the membership revocation; when it comes to lattice-based constructions, only few schemes [21, 24] handle this property by proceeding with verifier local revocation (VLR). VLR requires the verifiers to possess some up-to-date revocation information, but not the signers. With the same argument as the case above, this revocation information, in our case, consists in the revoked user's tracing keys.

5 Formal Security of our Scheme

In this section, we study the three requirements previously defined (Figures 2 and 3) in order to claim secure our scheme. Due to lack of space, complete proofs will appear in the full version of this paper.

Anonymity We now study the anonymity property.

Theorem 9 *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme, then there exists an adversary $\mathcal{B}$ that can break either the Syndrome Decoding Problem for matrix H and weight $2w$, or the Zero-Knowledge property of the Stern's proof.*

Idea of Proof Intuitively, under the Syndrome Decoding assumption / ZK property of the proof the adversary is not able to test which key is used in the Signing algorithm, and as such he is not able to distinguish a signature done honestly by a signature with a simulated proof on random values.

To fit the model, we should do that for a message signed by a user i_0, end on a random proof, say that this random proof can be assimilated to a signature by the user i_1 and then come back to a proper signature by the same user i_1. By doing so, we can deduce the validity of the previous theorem.

Soundness The soundness analysis consists in proving traceability and non-frameability.

Theorem 10 *If there exists an adversary $\mathcal{A}$ against the soundness of the scheme, then we can build an adversary $\mathcal{B}$ that can either break a computational problem (Syndrome Decoding for matrix H and weight $2w$), or the* Simulation-Soundness *of the proof.*

Sketch of proof Once again the proof is straightforward, receiving a Syndrome Decoding challenge, the simulator $\mathcal{B}$ will produce a sequence of games where he will process to substitute the framed user public key by the challenge value, answer various signing queries by simulating the Extended Stern proof (by programming the Random Oracle), and conclude by arguing that under the *Simulation-Soundness* of the proof, the valid new signature produced by the adversary implies the knowledge of a small weight word whose associated syndrom is the challenge, and then extract the value with a quick rewinding / reprogrammation of the Random Oracle (by memorizing the Random Oracle queries made by the adversary, $\mathcal{B}$ is able to know what the preimages of the hashes involved in the Challenge Signature are, and thus can learn the exact values of the challenge solution).

Proof. Non-frameability and traceability are very closely related, we will treat both simultaneously. We now define a (x_i, y_i, s_i) as being a *valid tuple* if $Rx_i^T = Qy_i^T = s_i$. There are two ways to cheat the soundness of our scheme: either by creating a new *valid tuple* without interacting with the group manager ($\mathcal{G}_1$) which induces a traceability attack, or by using an existing valid tuple but on a new message ($\mathcal{G}_2$) which breaks the non-frameability.

We study the security of the unhashed version of the proofs scheme (because of the perfect soundness of the proofs and the memory of the ROM). We will construct two different games, in the first one ($\mathcal{G}_1$), we assume the adversary is able to forge a signature by generating a new *valid tuple*, in the second one ($\mathcal{G}_2$) the adversary is able to forge a new x_i for a given s_i and so break the tracing procedure.

$\mathcal{G}_1$ Let us be given a (one more) Minimum Distance Challenge $(R||Q, (z_i)_{i\in[NQ]})$. We build an adversary $\mathcal{B}$ able to solve this challenge, from $\mathcal{A}$ that breaks the soundness of our scheme by generating a new balanced-small weight z. (Assuming the z_i are not balanced but properly distributed, we need approximately a sample 4 times bigger than the expected number of query)

To answer the i-th join queries, as it is passive, $\mathcal{B}$ directly splits z_i in $(x_i\|y_i)$.

After at most NQ join queries, $\mathcal{A}$ is able to output a new signature with a fresh certificate tuple with non-negligible probability. As $\mathcal{B}$ controls the Random Oracle he can look up the preimage of each computed values, and obtain a fresh (x, y) such that $(R\|Q)(x\|y)^T = 0)$ and so he is able to answer the challenge instance.

$\mathcal{G}_2$ Let us be given a Syndrome Decoding challenge (R, s). We build an adversary $\mathcal{B}$ answering this challenge, from an adversary $\mathcal{A}$ breaking the soundness of our scheme by forging a new *valid tuple*.

$\mathcal{B}$ generates a new $gmsk$, he then gives $gmsk$ to $\mathcal{A}$, together with the public parameters. $\mathcal{B}$ can answer any $join$ queries as he knows $gmsk$, the user on which we expect the attack (the challenge user) will have a *valid tuple* corresponding to one with y as a secret key. (Specifically $Rx^T = s$). $\mathcal{A}$ can corrupt any user, if he tries to corrupt the challenge user, the simulation fails. As all uncorrupted user looks the same, with non-negligible probably the simulation continues. Thanks to the random oracle, $\mathcal{B}$ can simulate all answers to the signing queries.

After at most NQ signing queries, $\mathcal{A}$ succeeds in breaking the non-frameability with non-negligible probability by generating a new proof, on an uncorrupted user. As uncorrupted users are indistinguishable, with non negligible probability this user is the challenge one, and so $\mathcal{B}$ is able to produce a small weight word y, which breaks the Syndrome Decoding assumption.

6 Example and Parameters

Our scheme is generic and can be used with any trapdoor function $Inv()$, for coding theory based on Hamming metric a possible trapdoor function is the CFS signature algorithm [11], which for Q a $(2^m \times \omega m)$ masked dual matrix of a Goppa

code, is able to associate, with probability $1/\omega!$, a word of weight ω to a random syndrome. From the security discussion, the parameters have to be chosen so that the Syndrome Decoding problem for the matrix $H = (R\|Q)$ (which has double length of the CFS matrix Q, and is formed from a random matrix R and the CFS public matrix Q) is difficult for a weight 2ω. Notice that even if a recent result has proven that public matrix of the CFS scheme was distinguishable [13] from a random matrix, it did not give rise to an attack on the scheme, hence for choice of parameters we consider H as a random matrix. From the best known attacks [15, 1] we can choose $m = 20$ and $\omega = 10$. It leads to a matrix H of length 2^{21} and dimension 200, with a security of 80 bits. The level of security can be improved over 80 bits by taking parameters with $m = 20$ and $\omega = 11$. The fact that the searched small weight vectors have a particular form since they have to be of same weight ω on the two parts of the matrix only increases the complexity of the attacks at the margin. Indeed it decreases the number of possible solutions to the Syndrome Decoding problem only by a small factor, since random solutions of weight 2ω to the problem have a good probability to be of equal weight on each part of the matrix.

These parameters lead to a signature length for our scheme (a transcript of proof of knowledge of a small word associated to H) of length roughly 20 megaBytes and a public key of size 2.5 megaBytes.

Notice that what takes time in the protocol is the computation of the CFS signature by the signer, but this is done only once for each member of the group when he enters the groups, and hence it is less important that this signature takes a little more time than usual signatures. At last the CFS signature scheme cannot find a preimage for any syndrome s, it does it only with probability $1/\omega!$, this fact can be managed through the sending of $\omega!$ different syndromes s in the Set-up process of entering the group, so that, on the average, a preimage y by the CFS public matrix Q is found with a small failure probability, in which case the set-up process is started over, since the syndrome s is computed randomly, it does not affect the security of the scheme.

7 Conclusion

In this work, we proposed the first code-based group signature scheme. The main idea of our work was to build a collision of two syndromes associated to two different matrices: a random one which enables to build a random syndrome from a *chosen* small weight vector; and a *trapdoor matrix*, which permits to find a small weight preimage of the previous random syndrome. Applying a variation of Stern's protocol on these two small weight vectors led us to design our group signature scheme through the use of Fiat-Shamir paradigm. We assume that its elegance, its simplicity and the large range of properties it fulfills make this scheme a good alternative to all other post-quantum, namely lattice-based, constructions. Indeed, not only did we design a scheme ensuring *anonymity*, *traceability* and *non-frameability* but it also allows to dynamically add new members (*weakly-dynamic*), handles member revocation and can be turned into a traceable signature (KTY model) while group public key and signatures sizes remain independent of the number of group members. Finally, our construction can be seen as the first first *weakly* dynamic post-quantum traceable signature with membership revocation.

References

[1] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in 2 n/20: How 1+1=0 improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer Berlin Heidelberg, 2012.

[2] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer Berlin Heidelberg, 2003.

[3] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer Berlin Heidelberg, 2005.

[4] E. Berlekamp, R.J. McEliece, and H.C.A. Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384–386, May 1978.

[5] Olivier Blazy. *Preuves de connaissance interactives et non-interactives. Part 1, Chapter 3.* PhD thesis, University Paris VII – Denis Diderot, September 2012.

[6] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin Heidelberg, 2004.

[8] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of CCS 2004*, pages 168–177. ACM Press, 2004.

[9] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. pages 56–72. Springer-Verlag, 2004.

[10] David Chaum and Eugne van Heyst. Group signatures. In DonaldW. Davies, editor, *Advances in Cryptology EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Berlin Heidelberg, 1991.

[11] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174. Springer, 2001.

[12] Cecile Delerablee and David Pointcheval. Dynamic fully anonymous short group signatures. In PhongQ. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer Berlin Heidelberg, 2006.

[13] J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems. In *ITW 2011*, pages 282–286, Paraty, Brazil, October 2011.

[14] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[15] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer Berlin Heidelberg, 2009.

[16] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989.

[17] S.Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer Berlin Heidelberg, 2010.

[18] Jens Groth. Fully anonymous group signatures without random oracles. In *In ASIACRYPT 2007, volume 4833 of LNCS*, pages 164–180. Springer, 2007.

[19] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer Berlin Heidelberg, 2004.

[20] Fabien Laguillaumie, Adeline Langlois, Benot Libert, and Damien Stehl. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer Berlin Heidelberg, 2013.

[21] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In Hugo Krawczyk, editor, *Public-Key Cryptography PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer Berlin Heidelberg, 2014.

[22] Benoit Libert and Moti Yung. Efficient traceable signatures in the standard model. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 187–205. Springer Berlin Heidelberg, 2009.

[23] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[24] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices.

[25] K. Nguyen S. Ling and H. Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *In PKC 2015 (to appear)*.

[26] Jacques Stern. A new identification scheme based on syndrome decoding. In DouglasR. Stinson, editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer Berlin Heidelberg, 1994.

[27] A. Vardy. The intractability of computing the minimum distance of a code. *Information Theory, IEEE Transactions on*, 43(6):1757–1766, Nov 1997.