A Code-Based Group Signature Scheme

Quentin Alamélou[1,2] Olivier Blazy[1] Stéphane Cauchie[2]
Philippe Gaborit[1],
[1] Université de Limoges, XLIM-DMI, Limoges, France,
{quentin.alamelou, olivier.blazy, philippe.gaborit}@xlim.fr,
[2] Worldline, R&D Department, Seclin, France,
{quentin.alamelou, stephane.cauchie}@worldline.com.

August 24, 2016

Abstract

This work is the extended version of [1] which proposed the first code-based group signature. The new group signature scheme we present here has numerous advantages over all existing post-quantum constructions and even competes (in terms of properties) with pairing based constructions: it allows to add new members during the lifetime of the group (*dynamic*). Plus, it appears that our scheme might be extended into a traceable signature according to the definition of Kiayias, Tsiounis and Yung [2] (KTY model) while handling membership revocation. Our 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*. The main idea of our scheme consists in building an offset 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 to which a fixed syndrome is added. 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 [4] of Courtois, Finiasz and Sendrier. If one denotes by N the number of group members, CFS leads to signatures and public keys sizes in $N^{1/\sqrt{log(N)}}$. Along with this work, we also introduce a new kind of proof of knowledge, *Testable weak Zero Knowledge* (TwZK), implicitly covered in the short version of this paper [1]. TwZK proofs appear particularly well fitted in the context of group signature schemes: it allows a verifier to test whether a specific witness is used without learning anything more from the proof. Under the Random Oracle Model (ROM), we ensure the security of our scheme by defining the *One More Syndrome Decoding* problem, a new code-based problem related to the Syndrome Decoding problem [5].

Keywords. Code-Based Cryptography, Group Signature, Proof of Knowledge, Random Oracle Model.

MSC Codes. 81P94, 94A60.

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 [6], group signatures have been extensively studied. Bellare et al. [7] (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 [8, 9, 10] were proposed but only proven secure in a relaxation security of [7]. Delerablée and Pointcheval [11] proposed the first practical scheme fully fitting BSZ in the random oracle model (ROM) whereas Groth [12] 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 [2]. In addition to classic properties of a group signature scheme, a traceable signature enables the opening authority to delegate its revoking (or opening) 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 [13].

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 [14] 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 [15, 16, 17, 18] proposed more efficient lattice-based constructions where both sizes the group public keys and signatures are proportional to $log(N)$. In a concurrent and posterior work, Ezerman et al. [19] also designed a code based group signature that suffers weaker features in terms of size of parameters and properties. Plus, it is interesting to notice that, with the recent exception of [20], all lattice and code based constructions base their security on the static model of [7] meaning that our scheme constituted the first post-quantum dynamic group signature scheme.

Because of a restriction for adding new users (procedure $Join$), our scheme ensures security properties of traceability, anonymity and non-frameability in a relaxation of the BSZ model. Indeed, the security of our protocol is ensured only when an adversary can add honest users (oracle $joinP$) that may be corrupted later while the BSZ model requires to fulfill security even in presence of an adversary adding already corrupted users: it led us to define our construction as *weakly dynamic*.

The main idea of our scheme consists in building an offset 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 to which a fixed syndrome is added. 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 contributions In this work, we propose a generic construction for designing the first code-based group signature. In a concurrent and independent work, [19] proposed a group signature scheme based on coding assumptions but only fitting the limited BMW model and with signatures and public key sizes linear in the number of group members.

Our security is based on a relaxation of the restrictive 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 enjoys nice features such as traceability in the sense of the KTY model and membership revocation. When instantiated with the CFS scheme [4], it leads to signatures and public keys sizes proportional to $N^{1/\sqrt{log(N)}}$ for N the number of group members, which is greater than a logarithmic complexity but asymptotically smaller than $N^{1/d}$ for any d. Plus, the proposed scheme can easily be extended into a traceable signature (according to the KTY model) handling membership revocation. In order to reach our goal, we introduce a new kind of proof of knowledge, *Testable weak Zero Knowledge* (TwZK), implicitly covered in [1]. It can be seen as a weaker version of zero-knowledge proofs [21]: it allows a verifier to test whether a specific witness is used without learning anything more from the proof. This new notion, that could be of independent interest, appears particularly well-fitted in the context of group signature schemes. Under the Random Oracle Model (ROM), we ensure the security of our scheme by defining the *One More Syndrome Decoding* problem, a new code-based problem related to the Syndrome Decoding problem [5].

Organization The following Section is dedicated to notations and background; Section 3 deals with the building blocks of our construction. In Section 4, we give formal definition for group signature while our construction is described in Section 5. Section 6 is concerned with the security of our scheme and we finally give an example and parameters for building an instance of our scheme in Section 7.

2 Preliminaries

In this section, we first present the notation used throughout this work and then make a focus on code-based cryptography.

2.1 Notation

μ denotes some randomness and we use the symbol $\|$ for concatenation. *crs* denotes the common string shared by involved parties in the Common Reference String (CRS) model.
For a vector (resp. a string) v, $v[r]$ denotes the r-th coordinate (resp. symbol) of v. 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 rows and n columns. S^n_ω is the set of vectors of weight ω lying in $\mathbb{F}^n_2$. Σ_n denotes the set of permutations over $[n]$.

We denote by λ a security parameter. For a protocol, l_λ denotes the number of iterations needed to reach the level of security required by λ.
$\mathcal{H}, \mathcal{H}_\lambda : \mathbb{F}^*_2 \longrightarrow \{0,1,2\}^{l_\lambda}$, $h : \mathbb{F}^*_2 \longrightarrow \mathbb{F}^n_2$ and $h' : \mathbb{F}^*_2 \longrightarrow \mathcal{M}_{k\times n}(\mathbb{F}_2)$ model random oracles. Except stated otherwise, *log* denotes the logarithm in base 2. For any entity $\mathcal{E}$, we denote by $\mathcal{E}(I)$ the fact that $\mathcal{E}$ has knowledge of I. $\mathcal{A}(z : \mathcal{O})$ denotes that entity $\mathcal{A}$ has knowledge of z and access to oracle $\mathcal{O}$. 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 $x \in \mathbb{F}^n_q$. The product Hx^T is called *a syndrome* and $\omega t(x)$ refers to the Hamming weight of x.

For an instance of a group signature scheme, we denote by N the number of group members.

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 [22].

Syndrome Decoding Problem The Syndrome Decoding problem (SD-problem) is a problem based on coding theory shown NP-complete [5]. It consists in finding a small weight word for a given syndrome s.

Definition 1. *Let H be a random matrix from $\mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, 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 $He^T = s$.*

The case where one is asked to find a small solution where the weight is approximated within a constant was also proven to be computationally intractable in [23].

One More Syndrome Decoding problem We now define a related problem, that we call the One-More Syndrome Decoding (OMSD) problem which we believe difficult. Let H a $(n-k) \times n$ random binary matrix, s a random syndrome and l vectors $x_1, x_2, \ldots, x_l$ of weight ω such as for any $i = 1 \ldots l$, $Hx_i^T = s$.

Question Is it possible to find a $(l+1)$-th vector x_{l+1} such as $x_{l+1} \neq x_i$ for any $i = 1 \ldots l$, of weight ω and verifying $Hx_{l+1}^T = s$?

We denote $(H, s, \omega, \{x_i\}_{i=1\ldots l})$ such an OMSD instance where the goal is to find a valid x_{l+1}.

Discussion on the problem This problem corresponds to a code version of problems which are well known in number theory based cryptography [24]. In the case of coding theory and the Syndrome Decoding problem, the best known attacks are direct retrieval of independent syndromes, and it is widely believed that having an oracle which give you information on l independent syndromes does not help the attacker in finding a $(l+1)$-th solution to a syndrome decoding problem.

Stern's protocol With security based on the SD-problem, Stern first proposed an efficient code-based ZK protocol [25]. 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, ω).

Remark 1. *The protocol presented here is the original one as introduced in [25]. It suffers a distinguishability issue since a verifier with knowledge of z is able to check whether this given witness z was actually involved during the protocol. Indeed, when $ch = 0$ (respectively $ch = 1$): in addition to checking c_1 and c_2 (resp. c_1 and c_3), such a verifier $\mathcal{V}(z)$ can also compute $c_3 = h(\pi(z + u))$ (resp. $c_2 = h(\pi(z + u + z))$ and then deduces involment of z. To fix this distinguishability issue, Stern then proposed a randomized version of its protocol for which commitments c_1, c_2, c_3 are computed by concatenating inputs to random seeds [26].*

This ability to somewhat *trace* a user in the original version will constitute a central point for designing our group signature for which security will be ensured by defining what we call *Testable weak Zero-Knowledge*(TwZK) proofs (Definition 3).

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

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

1. $\mathcal{P}$ computes commitments as follows:
   $\pi \xleftarrow{\$} \Sigma_n$, $u \xleftarrow{\$} \in \mathbb{F}_2^n$
   $c_1 = h(\pi || Hu^T), c_2 = h(\pi(u)), c_3 = h(\pi(z+u))$
2. $\mathcal{P}$ sends c_1, c_2, c_3
3. $\mathcal{V}$ sends $ch \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cases for $\mathcal{P}$:
   $ch = 0$: $\mathcal{P}$ sends π, u
   $ch = 1$: $\mathcal{P}$ sends $\pi, z+u$
   $ch = 2$: $\mathcal{P}$ sends $\pi(z), \pi(u)$
5. 3 cases $\mathcal{V}$:
   $ch = 0$: $\mathcal{V}$ checks c_1, c_2
   $ch = 1$: $\mathcal{V}$ checks c_1, c_3
   $ch = 2$: $\mathcal{V}$ 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

Fiat-Shamir Paradigm Fiat and Shamir proposed in [27] 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 cmt a commitment from the prover, ch a random challenge from the verifier, and rsp the response to ch), and then turn it into a digital signature scheme with the help of a random oracle $\mathcal{H}$. Indeed, to sign a message m, the signer (who knows the secret) produces a valid transcript (cmt, ch, rsp) of the interactive protocol where $ch = \mathcal{H}(cmt, m)$.

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 7).

Definition 2. *A trapdoor matrix family is a couple of polynomial algorithms* $(TrapGen, Inv)$ *such that:*

- $TrapGen(1^\lambda)$*: outputs a pair* $(Q, trk) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times \mathcal{TRK}$ *according to security parameter* λ*;*
- $Inv(Q, trk, s, \omega)$*: outputs, with non negligible probability, some* $x \in S_n^\omega$ *such as* $Qx^T = s$ *assuming* $(Q, trk) \leftarrow TrapGen(1^\lambda)$*,* $s \in \mathbb{F}_2^{n-k}$ *and* $\omega \in \mathbb{N}$*.* x *has to appear random in* S_n^ω*. If no such solution is found, it returns* $\perp$*;*
- *(Correctness): for all* (Q, trk) *output by* $TrapGen(1^\lambda)$*, and all* $s \in \mathbb{F}_2^{n-k}$*, we have that* $Q(Inv(Q, trk, s, \omega))^T = s$*;*
- *(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 $TrapGen$.

3 Cryptographic Primitives Revisited

In this section, we first introduce a Stern-like protocol whose contribution could be of independent interest. Secondly, we define a new kind of proof of knowledge which aims at capturing the traceability issue of deterministic Stern-like protocols as highlighted in Remark 1. After exhibiting how our new protocol can satisfy this new definition, we prove its security.

3.1 A variation on Stern's Protocol: Concatenated Stern's protocol

We propose a variation of Stern's protocol, referred as Concatenated Stern's protocol ($\mathcal{CSP}$) and depicted in Figure 2. The main 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 $Hz^T = s \Leftrightarrow Rx^T + Qy^T = s$. We additionally assume that the verifier $\mathcal{V}$ can be given some y'. As exhibited at step 6 of Figure 2, he is then able to check in two out of three cases ($ch = 0$ and $ch = 1$) if $y' = y$ without learning x.

Public data: $(H, s, \omega) \in \mathcal{M}_{(n-k)\times 2n}(\mathbb{F}_2) \times \mathbb{F}_2^{n-k} \times \mathbb{N}$
$\mathcal{P}$ proves knowledge of a valid $z = (x\|y) \in \mathbb{F}_2^{2n}$ to $\mathcal{V}(y')$

1. $\mathcal{P}$ computes commitments as follows:
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$c_1 = h(\pi||Ru^T + Qv^T), c_2 = h(\pi(u)), c_3 = h(\pi(x + u))$
$d_1 = h(\sigma||Ru^T + Qv^T), d_2 = h(\sigma(v)), d_3 = h(\sigma(y + v))$
2. $\mathcal{P}$ sends $c_1, c_2, c_3, d_1, d_2, d_3$.
3. $\mathcal{V}$ sends $ch \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cases for $\mathcal{P}$:
$ch = 0$. $\mathcal{P}$ sends $\{\pi, \sigma, u, v\}$
$ch = 1$. $\mathcal{P}$ sends $\{\pi, \sigma, x + u, y + v\}$
$ch = 2$. $\mathcal{P}$ sends $\{\pi(x), \pi(u), \sigma(y), \sigma(v)\}$

5. 3 cases for $\mathcal{V}$:
$ch = 0$: $\mathcal{V}$ checks c_1, c_2, d_1, d_2
$ch = 1$: $\mathcal{V}$ checks c_1, c_3, d_1, d_3
$ch = 2$: $\mathcal{V}$ checks c_2, c_3, d_2, d_3
and $\omega t(\pi(x)) = \omega t(\sigma(y)) = \omega$
If all checks passed, $\mathcal{V}$ accepts $\mathcal{P}$'s knowledge.

6. Additional checks for $\mathcal{V}(y')$:
$ch = 0$: checks if $d_3 = h(\sigma(y' + v))$
$ch = 1$: checks if $d_2 = h(\sigma(y' + (y + v)))$
$ch = 2$: no additional check
If additional check passed, $\mathcal{V}$ accepts y'.

Figure 2: Concatenated Stern's Protocol ($\mathcal{CSP}$)

Compared to classic Stern's protocol, the interest of $\mathcal{CSP}$ is twofold: it enables to check independently the weight of each half of a small weight secret while a part of the secret can be compromised without revealing the entire secret. From now, by $\mathcal{CSP}$-instance, we refer to (H, s, ω) for which one is asked to find $z = (x\|y)$ such as $Hz^T = s \wedge \omega t(x) = \omega t(y) = \omega$. Relying on Stern's protocol (Figure 1), we prove the security of $\mathcal{CSP}$ in subsection 3.4.

3.2 Testable weak Zero Knowledge

Introduced in [21], ZK proofs allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the veracity of a statement $\mathcal{S}$, without leaking any additional information on the proof but its length. Since its introduction, this kind of proofs has been widely applied to digital identification protocols, and by extension signature schemes using the Fiat-Shamir paradigm [27]. These proofs satisfy the following three properties:

- Completeness: if $\mathcal{S}$ is true, the honest verifier will be convinced except with negligible probability,
- Soundness: if $\mathcal{S}$ is false, no cheating prover can convince the honest verifier that it is true except with negligible probability,
- Zero-Knowledge: anything that is feasibly computable from the proof is also feasibly computable from the assertion itself.

A classical variant of ZK proofs generally used is Witness-Indistinguishable (WI) proofs [28], where the ZK requirement is roughly weakened into "seeing the proof does not provide any information on the witness used". More formally, assuming a language $\mathcal{L}$ and witnesses z_0, z_1 satisfying the language, WI requires distributions D_0 and D_1 to be indistinguishable where $D_b = \{(z_0, z_1, Prove(z_b, \mathcal{L}; \mu))$, for random string $\mu\}$, while not requiring proof simulatability anymore.

Nevertheless, our construction is going to need somewhat opposite requirements: we want to be able to build a simulator capable of simulating proofs, while allowing anyone possessing information to test if this piece of information is related to the witness used for generating the proof. Proofs of knowledge meeting these requirements will be called Testable weak Zero-Knowledge (TwZK for short). The general idea behind TwZK is that there is a little information leaking compared to classical ZK: the fact that an attacker can test whether a particular information is related to the witness used in a proof. If this value is not known, an attacker should either test all possible values or find a particular one which, in both cases has to be hard. Hence an attacker who breaks the security of our model either breaks ROM security or is able to solve a computational problem.

As an example, considering Stern's protocol (Figure 1), this particular information about the witness is the witness itself. Indeed, as exhibited in Remark 1, if $\mathcal{V}$ knows the prover's secret z, he is able to trace this witness z by checking more requirements for cases $ch = 0$ and $ch = 1$.

We then propose the following definition.

Definition 3 (Testable weak Zero-Knowledge Proofs)**.** *A Testable weak Zero-Knowledge proof is defined through the following algorithms:*

- $TSetup(1^\lambda)$*: generates the public parameters of the system among which some function* f *and possibly a simulation trapdoor* stk*;*
- $Prove(z, \mathcal{L}; \mu)$*: generates a proof* Π *that a word* z *is in the language* $\mathcal{L}$ *using some random string* μ*;*
- $TVerif(\Pi, \mathcal{L})$*: checks that the validity of a proof* Π *with respect to the language* $\mathcal{L}$*;*
- $TestWit(y', \Pi, \mathcal{L})$*: for a valid proof* $\Pi \leftarrow Prove(z, \mathcal{L}; \mu)$ *and* f *generated during* $TSetup$*, this algorithm checks, with negligible failure probability, if* $y' = f(z)$*;*
- $SProve(stk, \mathcal{L}; \mu)$*: generates a simulated proof of knowledge of a word in* $\mathcal{L}$ *using a trapdoor* stk*.*

In addition to classical correctness and soundness properties, we want weak indistinguishability of proof simulation:
for a non empty language $\mathcal{L}$, a proof generated using a witness (case S_0) must be indistinguishable from one using the trapdoor (case S_1) meaning that $S_0 = \{Prove(z, \mathcal{L}; \mu) | w \in \mathcal{L}\}$ and $S_1 = \{SProve(stk, \mathcal{L}; \mu)\}$ have to be indistinguishable.

$\mathsf{Exp}^{wS-b}_{\mathcal{P},\mathcal{A}}(\lambda)$
1.*params* $\leftarrow TSetup(1^\lambda)$
2.if $(b = 0)$, $\Pi^* \leftarrow Prove(z, \mathcal{L}; \mu)$
3.else $\Pi^* \leftarrow \{SProve(stk, \mathcal{L}; \mu)\}$
4.$b' \leftarrow \mathcal{A}(\Pi^*, TestWit(.))$
5.Return b'

Compared to what might be expected for Classical Zero-Knowledge, the previous property can only be achieved for some languages. Namely those where guessing the correct related information $f(z)$ is hard (either because there is no efficient way to search through the possible set, or simply because the set of potential $f(z)$ is hard to sample).

Original Stern's protocol is TwZK We consider here the particular case of the non randomized version of Stern's protocol to exhibit that it is TwZK instead of ZK. We do not formally describe here how Stern's original protocol fulfills Definition 3 since the goal is only to highlight that it satisfies the property of TwZK encompassed in weak indistinguishability of proof simulation described above. To cope with step 6 of protocol $\mathcal{CSP}$ (Figure 2), we propose the following additional checks for a verifier, with additional knowledge a vector z', to Stern's protocol (Figure 1):

- case $ch = 0$. $\mathcal{V}(z')$ checks if commitment $c_3 = h(\pi(z' + u))$;
- case $ch = 1$. $\mathcal{V}(z')$ checks if commitment $c_2 = h(\pi(z + u + z'))$;
- case $ch = 2$. No additional test in this case.

In two out of three cases ($ch = 0$ and $ch = 1$), an additional check can be processed so that this step has a failure probability $\leqslant 1/3$. Hence, in the non-interactive case where several repetitions of the protocol are demanded, this failure probability can be made as small as wished to design an algorithm $TestWit$. For a dishonest prover, there are 3 different cheating strategies depending on the value computed for c. We are going to focus on $c = 0$, the others are left to the sagacity of the reader. (It should be noted that for $c = 2$ no test is possible so the indistinguishability proof is even easier.)

We use the Random Oracle programmability on $\mathcal{H}$ to force the hash value to be 0. Following Stern's framework, in this case the $SProve$ algorithm would have to pick an honest π and u, and a random vector z of weight ω (not satisfying the equation $Hz^T = s$), and proceeds as expected. The difference in distributions between honestly generated proofs and simulated one comes from $h(\pi(z + u))$, which in the first case is correctly generated while in the other one is not. Assuming h is also a random oracle the value $h(\pi(z + u))$ is random compared to the rest of the view of the adversary.

Now the only way the adversary could possibly distinguish a value $h(\pi(z + u))$ computed for a z in the language from a random value, would be by querying $\pi(z + u)$ to the ROM. (Until now, the ROM programmability allows us to argue that the two visions are indistinguishable).

Using Random Oracle Observability, one can then monitor the calls to the random oracles made by the adversary, and for the π, u used in the challenge, parse his calls to define values z_is. In the eventuality the adversary queries the ROM on a z_i in the language (i.e. $Hz_i^T = s$ and $\omega t(z_i) = \omega$), a simulator generating a random proof without knowing a word in the language can use the adversary's query to solve the Syndrome Decoding challenge.

Formal transform into TwZK proofs and security of protocol $\mathcal{CSP}$ are respectively studied in subsection 3.3 and 3.4. Similar results for Stern's non randomized protocol could then easily be deduced.

3.3 From CSP to Testable weak Zero Knowledge Proofs

Even if some points may appear straightforward, we detail here how our protocol $\mathcal{CSP}$ (Figure 2) can be seen as TwZK proofs satisfying Definition 3.

Description of *TSetup* According to a security parameter λ, $TSetup$ algorithm generates public parameters among which an $\mathcal{CSP}$-instance (H, ω, s) and some function f. The $\mathcal{CSP}$-instance is meant to be chosen difficult and defines the following language $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$

$Hz^t = s \wedge \omega t(x) = \omega t(y) = \omega\}$. The function f stipulates how much information $f(z)$ could be tested about z while knowing some side information (algorithm $TestWit$).

TSetup(1^λ)
1. $(H, s, \omega, f) \xleftarrow{\$} 1^\lambda$ where:
   -$H = (R\|Q)$ with R, $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$
   -$s \in \mathbb{F}_2^{n-k}$, $\omega \in \mathbb{N}$
   -$f : \mathbb{F}_2^{2n} \to \mathbb{F}_2^n$ such as $f(x\|y) := y$
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
   $Hz^t = s \wedge [\omega t(x) = \omega t(y) = \omega]\}$

Return $(H, s, \omega, f, \mathcal{L})$.

Figure 3: TSetup protocol

In the following, we denote by $params$ the quantity $(H, s, \omega, f, \mathcal{L}) \leftarrow TSetup(1^\lambda)$.

Description of *Prove* and *TVerif* Algorithm $Prove$ enables anyone that possesses an element of $\mathcal{L}$ to prove such a knowledge without revealing it. Nevertheless, contrary to ZK proofs, some controlled information can be leaked and later be tested through algorithm $TestWit$. Through Fiat-Shamir paradigm, algorithm $Prove$ reaps benefit of the interactive protocol $\mathcal{CSP}$ to generate a proof of knowledge on an element of $\mathcal{L}$. It then processes as follows: it first pre-computes $6 \times l_\lambda$ commitments to be stored in cmt on which the random oracle is then applied. Finally, for each ternary symbol $ch[j]$, algorithm $Prove$ sets the corresponding response $rsp[j]$, as specified by protocol $\mathcal{CSP}$, and outputs the proof (cmt, rsp) that will later be given to $TVerif$.

This latter unfolds the proof, computes $ch = \mathcal{H}_\lambda(cmt)$ and checks consistency according to interactive protocol $\mathcal{CSP}$. We recall here that l_λ is the number of times protocol $\mathcal{CSP}$ should be repeated to ensure negligible error probability. Algorithms $Prove$ and $TVerif$ are depicted in Figure 4.

Description of *TestWit* Contrary to classical ZK proofs, the newly introduced concept of Testable weak Zero Knowledge (Definition 3) states that some controlled information can leak. Indeed, algorithm $TestWit$ will be able to decide if some side information (y' here) is related to the secret witness (z here), as stipulated by f, used to generate a valid proof $\Pi \leftarrow Prove(z, \mathcal{L}; \mu)$. Similarly to the non randomized version of Stern's protocol (see Remark 1), it is possible to trace proofs outputted by $Prove$. Before depicting the algorithm $TestWit$, we first give the general idea. Let us consider a witness $z = (x\|y)$ used to generate a proof. Now, using f defined such as $f(z) = y$ (Figure 3), algorithm $TestWit$ can check if $f(z) = y'$ without knowing z by running additional checks when unfolding a proof (cmt, rsp). Indeed, according to the value of $ch[j]$, $TestWit$ works as follows:

- $ch[j] = 0$, parse $rsp[j] = \{\pi, \sigma, u, v\}$ and check if $d_3 = h(\sigma(y' + v))$;
- $ch[j] = 1$, parse $rsp[j] = \{\pi, \sigma, x + u, y + v\}$ and check if $d_2 = h(\sigma(y' + y + v))$;
- $ch[j] = 2$, no such additional check is possible in this case.

(a) ***Prove***$(z, \mathcal{L}; \mu)$
Parse $z = (x \| y)$.
1. Commitments Generation
For $j = 1 \dots l_\lambda$
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$c_1 = h(\pi || Ru^T + Qv^T), c_2 = h(\pi(u))$,
$c_3 = h(\pi(x + u)), d_1 = h(\sigma || Ru^T + Qv^T)$,
$d_2 = h(\sigma(v)), d_3 = h(\sigma(y + v))$
$cmt[j] \leftarrow \{c_1, c_2, c_3, d_1, d_2, d_3\}$
2. Challenge Oracle Generation
$ch := \mathcal{H}_\lambda(cmt) \in \{0, 1, 2\}^{l_\lambda}$
3. Response Generation
For $j = 1 \dots l_\lambda$
3 cases:
$ch[j] = 0$: $rsp[j] = \{\pi, \sigma, u, v\}$
$ch[j] = 1$: $rsp[j] = \{\pi, \sigma, x + u, y + v\}$
$ch[j] = 2$: $rsp[j] = \{\pi(x), \pi(u), \sigma(y), \sigma(v)\}$
4. $\Pi \leftarrow (cmt, rsp)$
Return $\Pi = (cmt, rsp)$.

(b) ***TVerif***$(\Pi, \mathcal{L})$:
1. Parse $\Pi = (cmt, rsp)$
2. Challenge Oracle Generation
$ch := \mathcal{H}_\lambda(cmt) \in \{0, 1, 2\}^{l_\lambda}$
3. Commitments and Responses Checks
For $j = 1 \dots l_\lambda$
Parse $cmt[j] = \{c_1, c_2, c_3, d_1, d_2, d_3\}$
3 cases according to $ch[j]$:
$ch[j] = 0$: $\mathcal{V}$ uses $rsp[j]$ to check c_1, c_2, d_1, d_2
$ch[j] = 1$: $\mathcal{V}$ uses $rsp[j]$ to check c_1, c_3, d_1, d_3
$ch[j] = 2$: $\mathcal{V}$ uses $rsp[j]$ to check c_2, c_3, d_2, d_3
and $\omega t(\pi(x)) = \omega t(\sigma(y)) = \omega$
4. If all checks passed, Return 1.
Else Return 0.

Figure 4: Algorithms $Prove$ and $TVerif$

These additional checks pass if and only if $y = y'$ which means that some verifier $\mathcal{V}(y)$ can trace a user owning the secret witness $z = (x \| y)$. As in Stern's protocol, an additional check is possible in two out of three cases ($ch = 0$ and $ch = 1$) whereas no such test in proposed when $ch = 2$. The failure probability of this additional checking step is then $\leqslant 1/3$. In the non-interactive case where several repetitions of the protocol are demanded, this failure probability can be made as small as wished to design algorithm $TestWit$. We refer the reader to Figure 5 (a) for a description of $TestWit$.

Remark 2. *As it will be proven in next subsection, $\mathcal{CSP}$ is a prover verifier protocol with cheating probability of $2/3$. To provide convincing proofs, with negligible error probability, we then have to set $l_\lambda = \lambda / log(3/2)$. On the other hand, additional checks do not apply only when $ch = 2$ so that proofs simulating l_λ iterations of $\mathcal{CSP}$ will also also lead to a negligible error probability in the case of $TestWit$.*

Description of *SProve* It enables anyone with the trapdoor simulation stk to program the random oracle to generate a proof without knowledge of the secret key z that appears indistinguishable to a fair one i.e. as if it was an output of $Prove$.

According to the SD-problem, it is difficult to find a vector z both of small weight 2ω and verifying $Hz^t = s$. Nevertheless, it is easy to independently find a $z_1 = (x_1 || y_1)$, with no weight constraint, such as ${Hz_1}^T = s$ and a $z_2 = (x_2 || y_2)$ such as $\omega t(x_2) = \omega t(y_2) = \omega$. Let us now consider a simulator not knowing a valid secret that can set the value of ch^* by programming the random oracle. Following algorithm $Prove$, the simulator randomly picks π, σ, u and v and accordingly sets the value cmt^*.

- $ch^* = 0$: the simulator computes commitments with z_1 or z_2 playing the role of z and stores the following values in rsp^*: π, σ, u, v.

- $ch^* = 1$: the simulator computes commitments with z_1 playing the role of z and stores the following values in rsp^*: π, σ, $x_1 + u$, $y_1 + v$
- $ch^* = 2$: the simulator computes commitments with z_2 playing the role of z and stores the following values in rsp^*: $\pi(x_2)$, $\pi(u)$, $\sigma(y_2)$, $\sigma(v)$.

When called by Algorithm $TVerif$, because of programmability the random oracle $\mathcal{H}_\lambda$ will output ch^* on input cmt^* so that the simulated proof Π^* set to (cmt^*, rsp^*) clearly passes algorithm $TVerif$.

Algorithm $SProve$ is formally described in Figure 5 (b).

(a) ***TestWit***$(y', \Pi, \mathcal{L})$
1. Check proof validity
If $TVerif(\Pi, \mathcal{L}) = 0$, return $\perp$.
2. Challenge Oracle Generation
$ch := \mathcal{H}_\lambda(cmt) \in \{0,1,2\}^{l_\lambda}$
3. Additional Checks
Parse $\Pi = (cmt, rsp)$
For $j = 1 \dots l_\lambda$
Parse $cmt[j] = \{c_1, c_2, c_3, d_1, d_2, d_3\}$
3 cases according to $ch[j]$:
$ch[j] = 0$: parse $rsp[j] = \{\pi, \sigma, u, v\}$
If $d_3 = h(\sigma(y' + v))$, return 1.
Else, return 0.
$ch[j] = 1$: parse $rsp[j] = \{\pi, \sigma, x + u, y + v\}$
If $d_2 = h(\sigma(y' + y + v))$, return 1.
Else, return 0.
$ch[j] = 2$: does not apply
4. In any other case, return $\perp$.

(b) ***SProve***$(stk, \mathcal{L}; \mu)$
Parse $params = (H, s, \omega, f, \mathcal{L})$.
1. For $j = 1 \dots l_\lambda$
1.1. Fake secrets Generation
$z_1 = (x_1 \| y_1) \xleftarrow{\$} \mathbb{F}_2^{2n}$ such as $Hz_1^T = s$
$z_2 = (x_2 \| y_2) \xleftarrow{\$} \mathbb{F}_2^{2n}$ such as $\omega t(x_2) = \omega t(y_2) = \omega$
1.2. Simulated Proof Generation
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$ch^*[j] \xleftarrow{\$} \{0,1,2\}$
Following Step 1 and Step 3 of Figure 4 (a):
$ch^*[j] = 0$: compute commitments $cmt^*[j]$ using z_1 or z_2
and set $rsp^*[j] = \{\pi, \sigma, u, v\}$
$ch^*[j] = 1$: compute commitments $cmt^*[j]$ using z_1
and set $rsp^*[j] = \{\pi, \sigma, x_1 + u, y_1 + v\}$
$ch^*[j] = 2$: compute commitments $cmt^*[j]$ using z_2
and set $rsp^*[j] = \{\pi(x_2), \pi(u), \sigma(y_2), \sigma(v)\}$
2. Use stk to program the random oracle to have:
$\mathcal{H}_\lambda(cmt^*) = ch^* \in \{0,1,2\}^{l_\lambda}$
Return $\Pi^* = (cmt^*, rsp^*)$.

Figure 5: Algorithms $TestWit$ and $SProve$

3.4 Security of the Concatenated Stern's protocol

In this subsection we show that protocol $\mathcal{CSP}$ presented in Figure 2 is a TwZK protocol that ensures completeness, soundness and testable weak zero-knowledge. This protocol is a prover-verifier protocol with cheating probability equal to $2/3$ that needs to be repeated several times to decrease this cheating probability close to 0.

By design, it is straightforward that proving the security of protocol $\mathcal{CSP}$ enables to prove that algorithms defined in previous subsection fulfill Definition 3. Indeed, completeness and soundness of protocol $\mathcal{CSP}$ imply that algorithms $Prove$, $TVerif$ and $TestWit$ are well defined while showing that proofs outputted by $Prove$ and $SProve$ are indistinguishable ensures TwZK of protocol $\mathcal{CSP}$. Let us notice that we extend methodology of subsection 3.2 about TwZK of non randomized Stern's protocol to prove the TwZK of protocol $\mathcal{CSP}$.

Theorem 1. *The Concatenated Stern's protocol $\mathcal{CSP}$ is a prover verifier Testable weak Zero-Knowledge protocol with cheating probability $2/3$ verifying properties of completeness, soundness and Testable weak Zero-Knowledge in the ROM and assuming the hardness of the SD-problem.*

Proof We prove Theorem 1 through 3 lemmas in which we respectively discuss completeness, soundness and Testable weak Zero-Knowledge.

Completeness To ensure the completeness of our scheme, we exhibit that for a prover and a verifier honestly proceeding $\mathcal{CSP}$, it always succeeds.

Lemma 1. *If $\mathcal{P}$ and $\mathcal{V}$ honestly execute $\mathcal{CSP}$, we have for any round that $Pr[\mathcal{CSP}_{\mathcal{P},\mathcal{V}} = Accept] = 1$.*

Proof. The proof of the completeness is straightforward. The only subtlety might be to notice that z is a valid secret, which means that $Rx^T + Qy^T = s$. Then, for checking c_1 and c_2 in the case $ch = 1$, we have to use that $R(x+u)^T + Q(y+v)^T - s = Rx^T + Ru^T + Qy^T + Qv^T - s = Ru^T + Qv^T$. □

Soundness To ensure the soundness of our scheme, we ensure that a cheating prover $\mathcal{P}$ cannot convince $\mathcal{V}$ that he knows a valid secret z when he does not.

Lemma 2. *If a cheating prover $\mathcal{P}$ and an honest verifier $\mathcal{V}$ execute $\mathcal{CSP}$, then for any round, we have that: $Pr[\mathcal{CSP}_{\mathcal{P},\mathcal{V}} = Accept] = 2/3$.*

Proof. We will prove that $\mathcal{P}$ has a maximum probability probability $2/3$ to cheat. For $\mathcal{P}$ to cheat, he must be able to answer correctly to any challenge ch at each round whereas he does not know a valid z. To accept the protocol, $\mathcal{V}$ should be able to verify all the hash values at the end of the interaction.

- $ch = 0$: $\mathcal{P}$ reveals two permutations π_1, σ_1 and two vectors u and v respectively simulating values of π, σ, u and v in a fair protocol (we do not make this precision in the following).
- $ch = 1$: $\mathcal{P}$ reveals two permutations π_2, σ_2 and two vectors u' and v'.
- $ch = 2$: $\mathcal{P}$ reveals vectors X, U, Y, V with $\omega t(X) = \omega t(Y) = \omega$

Those values must verify hash values involved in the protocol. Let us consider a knowledge extractor $\mathcal{E}$ rewinding the random oracle. Since h is a random oracle (one cannot find a collision on it), we have the following straightforward results: $\pi_1 = \pi_2$, $\sigma_1 = \sigma_2$ and $Ru^T + Qv^T = Ru'^T + Qv'^T - s$ which leads to $R(u' - u)^T + Q(v' - v)^T = s$.
By exploiting consistency between different forms of c_2 and d_2, $\mathcal{E}$ gets that: $U = \pi(u)$ and $V = \sigma(v)$. Thanks to d_3 and c_3, we get: $\pi(u') = X + U$ and $\sigma(v') = Y + v$. Finally, by setting $z^* = (u' - u||v' - v)$, $\mathcal{P}$ enabled $\mathcal{E}$ to find z^* verifying $Hz^{*T} = s$ and $\omega t(z^*) = 2\omega$ i.e. $\mathcal{E}$ solved the SD-problem.

Then, $\mathcal{P}$ cannot anticipate the 3 challenges and the protocol clearly outputs $Accept$ with a maximum probability of $2/3$ (cases $ch = 0$ and $ch = 1$ or cases $ch = 0$ and $ch = 2$ according to the strategy of the cheating prover).

□

Testable weak Zero-Knowledge We will now prove the Testable weak Zero-Knowledge property of our scheme. The idea is to prove that a verifier cannot learn nothing more from a fair execution than prover's secret is correct and whether the auxiliary value he knows is related to prover's secret or not.

Lemma 3. *The Concatenated Stern's protocol $\mathcal{CSP}_{\mathcal{P},\mathcal{V}}$ depicted in Figure 2 is a Testable weak Zero-Knowledge proof in the random oracle model assuming the hardness of the SD-problem.*

Proof. Given a valid proof Π^* (i.e. for which $TVerif$ outputs 1), there are two possibilities for an adversary $\mathcal{A}$: algorithm $TestWit$ either outputs 0 or 1. In the first case, honestly generated proofs and simulated ones are clearly random since commitments stored in cmt are obtained via the random oracle h and values to be revealed, stored in rsp, are designed to appear random (the security of all Stern like protocols relies on this fact).

We now focus on the case where case $TestWit$ outputs 1.

If Π^* was generated by algorithm $SProve$, there are three different cheating strategies:

- $ch = 0$: the simulator runs $SProve$ to output Π^*. While generating this proof (see Figure 5 (b)), the simulator implicitly defined the SD-instance (Q, s_1, ω) where $s_1 = Qy_1^T$ for a certain y_1 with no constraint on the weight ($\omega t(y_1) \neq \omega$ a priori). The adversary will first check the proof by running $TVerif$ and he additionally runs $TestWit$ using its auxiliary knowledge y' to check the consistency of d_3. Thanks to RO observability, the simulator can monitor the calls made by the adversary to h. Now when $\mathcal{A}$ calls h to run $TestWit$, the simulator can parse these calls to identify y'. Since y' is such as $TestWit(y', \Pi^*, \mathcal{L}) = 1$, the simulator is able to find a vector y' of weight ω such as $s_1 = Qy'^T$ and thus solving the SD-instance (Q, s_1, ω).

- $ch = 1$: this time, the simulator implicitly defined the SD-instance (Q, s_2, ω) while generating Π^*, where $s_2 = Qy_2^T$ for a certain y_2 with $Qy_2^T \neq Qy^T$ a priori. Once again, the RO observability enables the simulator to monitor the calls made by the adversary to h. Now when the adversary calls h to run $TestWit$ to additionally check d_2, the simulator can parse these calls to identify y'. Since y' is such as $TestWit(y', \Pi^*, \mathcal{L}) = 1$, the simulator is able to find a vector y' of weight ω such as $s_2 = Qy'^T$ and thus solving the SD-instance (Q, s_2, ω).

- $ch = 2$: the algorithm $TestWit$ does not apply in this case and indistinguishability is straightforward.

Now, if this valid proof Π^* was generated by algorithm $Prove$, we consider the following game.

$\mathcal{G}$ The challenger supposed to run $Prove$ will attribute random values to commitments stored in cmt except for the values of d_3 in the case $ch = 0$ and d_2 in the case $ch = 1$ for which he computes values according to $\mathcal{CSP}$.

Because of h modeled as a random oracle, to $\mathcal{A}'$s view, this game $\mathcal{G}$ is indistinguishable from an honest $Prove$ procedure. Plus, for the same reason, it is also indistinguishable from the previous case with $SProve$ (all commitments stored in cmt always appear random).

Hence, we can consider that $\mathcal{A}$ will never provide a word y' such as $TestWit(y', \Pi^*, \mathcal{L}) = 1$ since it leads to breaking the SD-problem. Finally, under the ROM and the SD-problem, we can consider that distributions S_0 and S_1 are indistinguishable.

□

4 Definition and Security Model

In this section, we give the definition of the group signature we will rely on and the associated security model.

A group signature scheme [6] 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. [7] 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 [8, 9, 10] 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 [14, 15, 16, 17, 18] only satisfy the static model of [7], we propose a scheme that we will define as *weakly dynamic* with the classic properties of *anonymity*, *traceability* and *non-frameability*.

4.1 Definition

Let us 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 5) without impacting security (see Section 6). Consequently, the algorithm *Judge* present in dynamic models does not appear in our model. Adapting the BSZ model, we propose the following definition.

Definition 4. *A group signature scheme* $\mathcal{GS} = (GSetup, Join, GSign, GVerif, Open)$ *is a sequence of protocols such as:*

- $GSetup(1^\lambda)$*: this algorithm generates public parameters of the system* $gparams$*, the group public key* gpk *and the group manager secret key* $gmsk = (trk, skO)$ *made of a trapdoor key* trk *and the opening key* skO*;*
- $Join(\mathcal{U}_i)$*: this is an interactive protocol between a user* $\mathcal{U}_i$ *and the group manager owning* trk*. 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*;*
- $GSign(gpk, sk[i], m; \mu)$*: 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*;*
- $GVerif(gpk, m, \Sigma)$*: 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;*
- $Open(skO, gpk, m, \Sigma)$*: for a valid signature* Σ *with respect to* gpk*, the group manager can provide the signer identity : it thus outputs the user* $\mathcal{U}_i$ *when it succeeds and 0 otherwise.*

4.2 Security Model

We define our security notions in a game-based way defined in Figures 6 and 7. 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 oracle $joinP$ (passive join) to create an honest user for whom it does not know the secret key: the index i is added to the HU (Honest Users) list. 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 $S[m]$;
- the oracle $open$ which, on input (m, Σ) returns $Open(skO, gpk, m, \Sigma)$.

(a) Experiment $Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(gpk, gmsk = (trk, skO)) \leftarrow GSetup(1^\lambda)$
2. $(m, \Sigma) \leftarrow \mathcal{A}(gpk : joinP, corrupt, sign, open)$
3. If $GVerif(gpk, m, \Sigma) = 0$, return 0.
4. If $\exists j \notin CU \cup S[m]$,
   $Open(skO, gpk, m, \Sigma) = j$,
   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. $(gpk, gmsk = (trk, skO)) \leftarrow GSetup(1^\lambda)$
2. $(m, \Sigma) \leftarrow \mathcal{A}(gpk, skO : joinP, corrupt, sign)$
3. If $GVerif(gpk, m, \Sigma) = 0$, return 0.
4. If $\exists i \in HU \setminus S[m]$,
   $Open(skO, gpk, m, \Sigma) = i$,
   Return 1.
5. Else return 0.

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

Figure 6: Unforgeability Notions

Traceability and Non-Frameability

Traceability (Figure 6 (a)) says that nobody should be able to produce a valid signature that cannot be opened in a convincing way. Furthermore, non-frameability (Figure 6 (b)) guarantees that no dishonest player (even the authorities, i.e. the Group Manager GM) 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.$(gpk, gmsk = (trk, skO)) \leftarrow GSetup(1^\lambda)$
2.$(m, i_0, i_1) \leftarrow \mathcal{A}(gpk : joinP, corrupt, sign, open)$
3.$\Sigma \leftarrow GSign(gpk, i_b, m, sk[i])$
4.$b' \leftarrow \mathcal{A}(\Sigma : joinP, corrupt, sign)$
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 7: 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 $GSign$, 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 (Figure 7). We will see that our anonymity notion can be related to *selfless-anonymity* introduced in [9] where a user can check if he is the signer of a given signature and learns nothing about it else. The *full-anonymity* notion, required by the BSZ model, means that anonymity is guaranteed even if the adversary is granted access to all users' private keys (even the challenged one) and the oracle $open$ (excepted on the challenge signature).

A *weakly* dynamic model The BSZ model [3] basically defines two ways for an adversary $\mathcal{A}$ to add new group members: a passive one where an honest user is added and might be corrupted later while a kind of active join protocol enables $\mathcal{A}$ to add an already corrupted member (this user is immediately added to CU), for which he knows its secret key. In this latter case, the BSZ model can still provide non-frameability. Giving such an active join to an adversary $\mathcal{A}$ in our case is equivalent to assuming that $\mathcal{A}$ knows the trapdoor key trk. Nevertheless, in Section 6, we ensure the security of our scheme in a scenario in which $\mathcal{A}$ should never access to trk. This explains why $\mathcal{A}$ is only given skO instead of $gmsk$ in the non-frameability case. This difference between our model and the BSZ model led us to define our scheme as *weakly-dynamic*.

Definition 5. *A group signature scheme verifying security notions of Figures 6 and 7 is said to be securely weakly-dynamic.*

5 Our Code-Based Group Signature

We now present our group-signature scheme satisfying Definition 4. To fix ideas, we first present an high level overview and then, we describe more precisely the operation of the different algorithms.

5.1 High Level Overview

Actors Our scheme brings into play:

- a group manager (*GM*): the single authority of our scheme. It runs the *GSetup* 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 *GSign*);
- outsiders: they do not belong to the group but can verify a signature granted the group public key gpk (algorithm *GVerif*).

The main idea of our scheme consists in building an offset collision of two syndromes associated to two different matrices to instantiate $\mathcal{CSP}$. We recall here that the trapdoor matrix Q enables to invert syndromes so that a classic scenario could be as follows.

First Step GM generates Q a trapdoor matrix with trk the corresponding trapdoor key and a random syndrome s_G. $\mathcal{U}$ 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, via algorithm Inv, y such as: $Qy^T = s + s_G$ and $\omega t(y) = \omega$. If this steps fails, $\mathcal{U}$ chooses a new random value for x until a valid antecedent y can be found.
Then, GM returns y to $\mathcal{U}$ who finally 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 $\mathcal{U}$ owns a secret key $z = (x\|y)$ such as $\omega t(x) = \omega t(y) = \omega$ and $Hz^T = Rx^T + Qy^T = s_G$. This situation fits within the model of the protocol $\mathcal{CSP}$. Every group member then owns its secret key which leads to the group common syndrome s_G, as depicted in Figure 8.

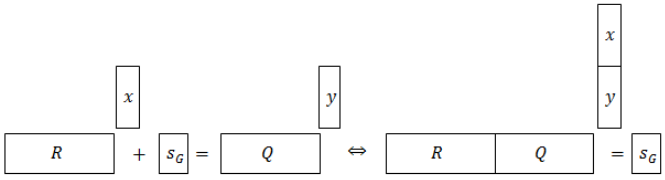

Figure 8: High level Overview

Third Step In case of doubt, GM should be able to revoke anonymity of group signatures. In the present case, GM knows y, the second part of the secret key of every user $\mathcal{U}$; this point will enable him to open signatures (algorithm *Open*).

5.2 Operation of our Scheme

In the CRS model, anyone is able to to generate $R = h'(crs)$.

We first describe algorithms $GSetup$, $Join$ and summed them up in Figure 9.
$GSetup$ 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.

(a) ***GSetup***(1^λ)

1. $gparams = (\lambda, k, n, \omega) \in \mathbb{N}^4 \leftarrow 1^\lambda$
2. $(Q, trk) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times \mathcal{TRK} \leftarrow TrapGen(1^\lambda)$
3. $s_G \xleftarrow{\$} \mathbb{F}_2^{n-k}$, $R = h'(crs)$
4. $gpk \leftarrow (H = (R\|Q), s_G, \omega)$
5. $skO \leftarrow \emptyset$
6. $gmsk \leftarrow (trk, skO)$
7. Output $(gparams, gpk, gmsk)$.

(b) ***Join***(1^λ)

	$user : \mathcal{U}_i$		GM
1.	$x_i \xleftarrow{\$} S_\omega^n$ $s_i = Rx_i^T$	$\xrightarrow{s_i}$	
2.			Run $Inv(Q, trk, s_i + s_G, \omega)$ *If* $Inv(Q, trk, s_i + s_G, \omega) = \perp$, go to 1 *If* $\exists i_0 : skO[i_0] = y_i$, go to 1 *Else* $skO[i] = y_i$
3.	$sk[i] = (x_i\|y_i)$	$\xleftarrow{y_i}$	

Figure 9: *GSetup* and *Join* algorithms

On adapting our TwZK protocol Our group signature scheme will mainly rely on the TwZK proofs described in Section 3. To cope with Fiat-Shamir paradigm in the context of a group signature scheme, we adapt algorithms $Prove$, $TVerif$ and $TestWit$ (Figures 4 and 5 (a)) so that step 2 of all these algorithms takes into account the message to sign m during the challenge generation. More precisely, algorithms $Prove$, $TVerif$ and $TestWit$ will take as additional input the message m so that ch is now equal to $ch = \mathcal{H}_\lambda(m, cmt)$ instead of simply $\mathcal{H}_\lambda(cmt)$. We respectively denote these (slightly) modified algorithms $Prove^{gs}(m, ., .; .)$, $TVerif^{gs}(m, ., .)$ and $TestWit^{gs}(m, ., .)$.

GSetup The $GSetup$ algorithm is executed by the group manager GM taking as input a security parameter λ. It randomly generates a *trapdoor matrix* (according to Definition 2) Q and the corresponding trapdoor key trk. It also chooses a random syndrome $s_G \in \mathbb{F}_2^{n-k}$ that will constitute the group public identity and initializes $gmsk = (trk, skO)$ where skO will be its opening key. Finally, GM publishes global parameters $gparams = (\lambda, n, k, \omega) \in \mathbb{N}^4$ and the group public key $gpk = (H = (R\|Q), s_G, \omega)$ where R and $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$. This algorithm implicitly covers the run of a $TSetup$ algorithm.

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: $s_i + s_G = Qy_i^T$ 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 $skO[i] = y_i$. The reader should notice that this methodology may fail for two reasons. On the one hand, algorithm Inv can fail when computing antecedent y_i. On the other hand, when such an y_i is successfully computed, GM should first check that y_i was not already attributed to another user (GM ensures that all y_i's are different to further revoke anonymity). In both cases, GM tells $\mathcal{U}_i$ to choose another secret x_i.

In the end of this protocol, each user is given a secret key $z_i = (x_i\|y_i)$ of weight 2ω and verifying $Hz_i^T = Rx_i^T + Qy_i^T = s_i + s_i + s_G = s_G$.

***GSign* and *GVerif* Algorithms** $\mathcal{CSP}$ (Figure 2) is an interactive TwZK protocol during which $\mathcal{P}$, which in fact consists, here, in the group G, proves to $\mathcal{V}$ the knowledge of a valid secret ensuring him that he belongs to the group associated to the common syndrome s_G.

As already mentioned, we use Fiat-Shamir paradigm to get a signature scheme. Signing then basically consists in outputting a proof of knowledge on a secret key that is linked to the message to sign m. Namely a group member $\mathcal{U}_i$ has to produce a signature Σ seen as a transcript $\Sigma = (cmt, rsp)$ of the protocol $\mathcal{CSP}$ executed on public key gpk and small weight secret $sk[i]$ for which each tuple $(cmt[j], ch[j], rsp[j])$ simulates a fair execution of $\mathcal{CSP}$. It then suffices to run protocol $Prove^{gs}$ on inputs m, $sk[i]$ and $\mathcal{L}$.

For verification (algorithm *GVerif*), one has only to check a signature outputted by $GSign$ which means checking the validity of some $\Sigma \leftarrow Prove^{gs}(m, sk[i], \mathcal{L}; \rho)$. We then apply algorithm $TVerif^{gs}$ since it precisely aims at checking the validity of a proof outputted by $Prove^{gs}$.

***Open* Algorithm** We first recall that GM's opening key skO consists in the pool of y_is constituting half parts of users' secret keys made of $(x_i||y_i)$. According to Figure 5 (a), the knowledge of such values permits to run (potentially successfully) algorithm $TestWit$ and then to decide if the tested value was the one used for generating the proof. The algorithm $Open$ is then straightforward: given a message m and a signature $\Sigma = (cmt, rsp)$, GM reads through the opening key skO until for some index i_0, $sk[i_0] = y_{i_0}$ passes algorithm $TestWit^{gs}$. In this case, GM is ensured that the actual signer is $\mathcal{U}_{i_0}$.

Remark 3. *By design of our scheme, any group member will then be able to open its own signatures which was referred to selfless-anonymity in [9].*

Algorithms *GSign*, *GVerif* and *Open* are depicted in Figure 10.

(a) ***GSign***$(gpk, sk[i], m, l_\lambda; \mu)$
1. Parse $gpk = (H, s_G, \omega)$.
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^T = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
3. $\Sigma = (cmt, rsp) \leftarrow Prove^{gs}(m, sk[i], \mathcal{L}; \mu)$.
4. Return (m, Σ).

(b) ***GVerif***$(gpk, m, \Sigma, l_\lambda)$
1. Parse $gpk = (H, s_G, \omega)$.
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^T = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
3. $b \leftarrow TVerif^{gs}(m, \Sigma, \mathcal{L})$
4. Return b.

(c) ***Open***(skO, gpk, m, Σ)
1. If (*GVerif*$(gpk, m, \Sigma, l_\lambda) = 0$), return 0.
2. Parse $gpk = (H, s_G, \omega)$
3. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^T = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
4. For $i = 1 \ldots N$
$b \leftarrow TestWit^{gs}(m, skO[i], \Sigma, \mathcal{L})$
If $(b = 1)$, return $\mathcal{U}_i$.
5. If none $skO[i]$ succeeds,
return 0.

Figure 10: *GSign*, *GVerif* and *Open*

5.3 Additional Properties

In this subsection, we briefly explain how the design of our group signature scheme might enable us to turn it into a traceable signature following the definition of [2] and how to handle revocation.

A traceable signature Extending group signatures schemes, Kiayias et al. suggested traceable signatures schemes in [2]. Such constructions enable the opening authority to delegate its opening or *tracing* capability to sub-openers so that they can trace suspicious users without letting them

trace others. Usually, such a scheme provides two additional protocols compared to a classic group signature scheme: $Trace$ and $Claim$.

Due to its construction, our scheme can easily be extended into traceable signature. Indeed if GM wants a sub-opener So to trace or look after $\mathcal{U}_{i_0}$ with secret key $z = (x_{i_0}||y_{i_0})$, he just needs to reveal him y_{i_0}. Now, whenever So is given a signature Σ, he uses y_{i_0} to apply algorithm $Trace$ on Σ: it outputs 1 if and only if the issuer was indeed $\mathcal{U}_{i_0}$. In fact, he has to apply the methodology of algorithm *Open* but since he does not know skO, he cannot look over all users' tracing keys, then he will only be able to open the signature if it was issued by $\mathcal{U}_{i_0}$.

Furthermore, as pointed when defining anonymity, our scheme provides selfless-anonymity since any user will be able to prove that he is the actual signer of his own signature: this is exactly what algorithm $Claim$ is meant to do.

Membership Revocation A crucial requirement for group signature schemes should be membership revocation. Indeed, once a group has been set up, one must be able to keep on trusting a group even in presence of misbehaving users; else, it means a new group should be regenerated to exclude them. When it comes to post-quantum constructions, only few lattice-based schemes [16, 18] 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. We follow the same methodology to provide membership revocation. As for the case of traceable signature, this revocation information, in our case, simply consists in the revoked user's tracing keys y_is.

We neither provide formal definitions nor detailed proofs here but the conversion of our scheme into in a traceable signature with membership revocation is rather straightforward. In particular, we refer the reader to the non-frameability property of our scheme to ensure that information released to potentials sub-openers or in the revocation list does not impact the security of our scheme.

6 Formal Security Analysis

In this section, we study the three requirements of anonymity, traceability and non-frameability previously defined (Figures 6 and 7) in order to claim secure our scheme (Definition 5). In the following, we consider $\mathcal{A}$ an adversary to our scheme and $\mathcal{B}$, an adversary to a difficult problem using advantage of $\mathcal{A}$'s possibilities.

Recalls on our group signature A signature consists in a transcript representing several repetitions of our Concatenated Stern's protocol (Figure 2). Then a signature Σ is a couple (cmt, rsp) such as:

$$(cmt, rsp) = (cmt[1], \ldots, cmt[l_\lambda], rsp[1], \ldots, rsp[l_\lambda]) \tag{1}$$

- $cmt[j]$ consists in commitments of our Concatenated Stern's protocol (Figure 2) generated at j-th iteration;
- $ch[j]$ the j-th symbol of $ch = \mathcal{H}_\lambda(m, cmt) \in \{0,1,2\}^{l_\lambda}$. It plays the role of the random challenge sent by the verifier in the interactive version of $\mathcal{CSP}$ at j-th iteration;
- $rsp[j]$ is the answer to the challenge $ch[j] \in \{0,1,2\}$.

How to generate a simulated signature The methodology for a simulator, without knowledge of a secret key, to generate a signature Π^* that appears valid (i.e. that passes algorithm $GVerif$) is the one used in $SProve$. For each iteration $j = 1 \dots l_\lambda$, the simulator chooses a value for $ch[j]$ and stores commitments and responses respectively in cmt and rsp according to methodology described in protocol $SProve$. Finally, the simulator programs the RO to set $\mathcal{H}_\lambda(m, cmt)$ to ch and outputs $\Sigma^* = (cmt, rsp)$.

Hence, such a simulator will then produce a transcript looking fair to any verifier, without knowing any valid secret z.

6.1 Anonymity

We begin with the anonymity property.

Theorem 2. *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme (Figure 7), then there exists an adversary $\mathcal{B}$ that can break the Testable weak Zero-Knowledge (TwZK) property of our Concatenated Stern's protocol $\mathcal{CSP}$.*

Proof. Through a sequence of games, we will exhibit that an adversary against our anonymity property would be able to break the TwZK property of our scheme.

$\mathcal{G}_0$ $\mathcal{B}$ first runs $GSetup(\lambda)$. He gives gpk to $\mathcal{A}$ who has also access to oracles *joinP*, *corrupt*, *sign* and *open*. Now $\mathcal{B}$ and $\mathcal{A}$ act as described in $Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda)$ (Figure 7). In this game, $\mathcal{B}$ will honestly challenge the adversary $\mathcal{A}$ on $b = 0$. At some point, $\mathcal{A}$ produces (m, i_0, i_1). $\mathcal{B}$ will then behave honestly by signing m outputting $\Sigma_0 = GSign(gpk, sk[i_0], m; \mu)$.

$\mathcal{G}_1$ In this game, $\mathcal{A}$ and $\mathcal{B}$ behave the same way than in $\mathcal{G}_0$ with same knowledge and oracles provided to $\mathcal{A}$ with the exception of queries about $\mathcal{U}_{i_0}$. Indeed, when $\mathcal{A}$ produces (m, i_0, i_1), instead of generating Σ_0, $\mathcal{B}$ generates a simulated signature Σ^* by programming the random oracle $\mathcal{H}_\lambda$ accordingly.

$\mathcal{G}_2$ This game is the version of game $\mathcal{G}_0$ in which $\mathcal{B}$ challenges $\mathcal{A}$ on challenge $b = 1$; the rest consists in $Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ (Figure 7). So, when $\mathcal{A}$ produces (m, i_0, i_1), $\mathcal{B}$ responds $\Sigma_1 = GSign(gpk, sk[i_1], m; \mu)$.

Since the Concatenated Stern's protocol $\mathcal{CSP}$ is TwZK, we have, on the one hand, Σ_0 and Σ^* are statistically close to each other. On the other hand, for the exact same reason, Σ^* and Σ_1 are statistically close then Σ_1 is statistically close to Σ_0. Finally, we can deduce that $Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda)$ and $Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ are indistinguishable which leads to that $Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda) = Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda) - Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible. This terminates the proof. □

6.2 Soundness

The soundness analysis consists in proving traceability and non-frameability.
Through a methodology similar to [19], we will first begin to show how forging a signature could lead to breaking a computational problem.

Forging a signature

Let us suppose that $\mathcal{A}$ can forge a signature on user $\mathcal{U}_{i_0}$ supposed to be uncorrupted. It follows that $\mathcal{A}$ can produce $\Sigma = (cmt, rsp)$ such as $GVerif(gpk, m, \Sigma) = 1$ without knowing $sk[i_0]$. As recalled in (1), Σ is the following transcript $(cmt[1], \ldots, cmt[l_\lambda], rsp[1], \ldots, rsp[l_\lambda])$ simulating l_λ fair iterations of protocol $\mathcal{CSP}$.

If $\mathcal{A}$ can produce such a forgery, then $\mathcal{A}$ must have been able to successfully run l_λ iterations of $\mathcal{CSP}$ without knowing a valid secret whereas the cheating probability of $\mathcal{CSP}$ is 2/3 (Lemma 2). Then $\mathcal{A}$ has either broken the soundness of $\mathcal{CSP}$ or enabled the design of a knowledge extractor reaping benefits of the forgery to produce a valid solution z of the related SD instance.

Soundness Since traceability and non-frameability are two notions closely related, we treat them simultaneously. Indeed, both notions require the adversary $\mathcal{A}$ to produce a valid forgery Σ verifying $GVerif(gpk, m, \Sigma) = 1$. Nevertheless, breaking traceability implies for $\mathcal{A}$ to produce Σ such as the group manager could not trace it back to any group member whereas non-frameability requires to produce a signature that does trace back to an actual group member.

More precisely, if we consider that $\mathcal{A}$ attacks an honest user $\mathcal{U}_{i_0}$: to attack traceability, $\mathcal{A}$ should produce a forgery Σ on m such as:

$$GVerif(gpk, m, \Sigma) = 1 \wedge Open(skO, gpk, m, \Sigma) = 0, \tag{2}$$

whereas to attack non-frameability its forgery Σ should verify:

$$GVerif(gpk, m, \Sigma) = 1 \wedge Open(skO, gpk, m, \Sigma) = i_0, \tag{3}$$

with the obvious constraint in both cases that $\Sigma \neq GSign(gpk, sk[i_0], m; \mu)$.

We now prove the traceability and the non-frameability of the proposed group signature scheme.

Theorem 3. *If there exists an adversary $\mathcal{A}$ against the traceability (Figure 6 (a)) (resp. the non-frameability, Figure 6 (b)) of the scheme, then we can build an adversary $\mathcal{B}$ that can either break the security of the Concatenated Stern's protocol $\mathcal{CSP}$ or the OMSD (resp. SD) problem.*

Proof. Let $\mathcal{A}$ be a PPT adversary attacking the traceability (resp. non-frameability) property of our scheme with advantage $\epsilon(\lambda)$. Through a sequence of games, we will show that if $\mathcal{A}$ is efficient, then it is possible for a simulator $\mathcal{B}$ to solve a difficult problem with non negligible probability related to ϵ.

$\mathcal{G}_0$ Following our group signature scheme, a simulator $\mathcal{B}$ runs algorithm $GSetup(\lambda)$. He then chooses a user $\mathcal{U}_{i_0}$ on which $\mathcal{A}$ will be challenged. The following consists in the traceability (resp. non-frameability) game defined in Figure 6 (a) (resp. Figure 6 (b)); $\mathcal{B}$ then provides $gpk = ((R||Q), \omega)$ (resp. $gpk = ((R||Q), \omega)$ and skO) to $\mathcal{A}$, which has also access to oracles *joinP*, *sign*, *open* and *corrupt* (resp. *joinP*, *sign* and *corrupt* since in this case, $\mathcal{A}$ knows skO and has no need for oracle *open*). For any query of $\mathcal{A}$, $\mathcal{B}$ responds honestly but the game aborts if $\mathcal{A}$ tries to corrupt $\mathcal{U}_{i_0}$.

At some point, $\mathcal{A}$ produces a forgery (m, Σ) under the condition that for all $i \in CU$, signatures on m were never queried. As supposed earlier, the probability for $\mathcal{A}$ to have $GVerif(gpk, m, \Sigma)$ outputting 1 is ϵ.

$\mathcal{G}_1$ In this game, $\mathcal{B}$ still runs algorithm $GSetup(\lambda)$ and chooses a user $\mathcal{U}_{i_0}$. $\mathcal{A}$ still knows gpk (resp. gpk and skO) with the same respective oracle accesses. The only difference from previous game is that whenever $\mathcal{A}$ queries oracle *sign* on user $\mathcal{U}_{i_0}$, $\mathcal{B}$ generates a simulated valid signature. Like previously, the game aborts $\mathcal{A}$ tries to corrupt $\mathcal{U}_{i_0}$.

At some point, $\mathcal{A}$ produces a forgery (m, Σ) under the condition that for all $i \in CU$, signatures on m were never queried. Similarly to anonymity game (subsection 6.1), signatures honestly produced on behalf of $\mathcal{U}_{i_0}$ in game $\mathcal{G}_0$ are indistinguishable from random ones produced in this game.

Hence, to $\mathcal{A}$'s view, games $\mathcal{G}_0$ and $\mathcal{G}_1$ are indistinguishable and we get that the probability for $\mathcal{A}$ to have $GVerif(gpk, m, \Sigma) = 1$ is still ϵ.

Under the soundness of $\mathcal{CSP}$, we now treat separately traceability (game $\mathcal{G}_1^{trac}$) and non-frameability (game $\mathcal{G}_1^{nf}$).

$\mathcal{G}_1^{trac}$ $\mathcal{A}$ has been given a forgery Σ verifying (2). Under the soundness of $\mathcal{CSP}$, $\mathcal{B}$ could thus apply the knowledge extractor algorithm $\mathcal{E}$ on Σ to generate a vector z^* passing protocol $\mathcal{CSP}$. All through this game, $\mathcal{A}$ may have queried for oracle *corrupt* on all users except $\mathcal{U}_{i_0}$ meaning that he may have obtained many secret keys solving the instance $(H, s_G, 2\omega)$. In other words, $\mathcal{A}$ has obtained the following OMSD instance $(H, s_G, 2\omega, \{usk[i]\}_{i \in CU})$. If $z^* \notin \{usk[i]\}_{i \in CU}$, $\mathcal{A}$ has then enabled $\mathcal{B}$ to find one more solution to the previous OMSD instance with probability directly related to ϵ. Else, $\mathcal{A}$ replays the game until $\mathcal{B}$ gets a new solution to the aforesaid OMSD instance.

$\mathcal{G}_1^{nf}$ In the case of non-frameability, $\mathcal{A}$ has also been given a forgery Σ verifying (3). This time, the knowledge of skO provides more information to $\mathcal{A}$ about users' secret keys. Indeed skO consists in all tracing keys (y_is), that is, half of every user's secret key. Writing $usk[i] = (x_i||y_i)$ for all i, we have that $\mathcal{A}$ can compute $s_i = Qy_i^T (= Rx_i^T)$. Through oracle *corrupt*, $\mathcal{A}$ can learn the entire z_i for every user (different from $\mathcal{U}_{i_0}$) he might corrupt. We recall here that every s_i, originally computed during the join procedure by $Rx_i^T = s_i$ appears random since every x_i is randomly chosen in $\mathcal{S}_\omega^n$. In fact, $\mathcal{A}$ has obtained the following unsolved SD-instances $(R, s_i, \omega)_{i \in HU}$ containing the particular one (R, s_{i_0}, ω).

Now, under the soundness of $\mathcal{CSP}$, $\mathcal{B}$, by programming the ROM, exploits Σ to get a vector $z^* = (x^*||y^*)$ from which he can issues signatures verifying (3) just like Σ. In particular, it means that applying algorithm $Open$ on signatures issued with z^* returns i_0. It leads to $y^* = y_{i_0}$ and then that x^* is a solution to the SD instance (R, s_{i_0}, ω).

At the end of game $\mathcal{G}_1$, $\mathcal{A}$ has either broken the soundness of $\mathcal{CSP}$ or been able to solve a computational problem with non negligible probability related to ϵ. This terminates the proof.

□

7 Instantiation with the CFS scheme

Our scheme is generic and can be used with any trapdoor matrix. For coding theory based on Hamming metric, a possible trapdoor function is the CFS signature algorithm [4].

7.1 CFS Distinguishability and Security

The main idea of the CFS signature is to hide a Goppa code matrix with parameters $[2^m, 2^m - m\omega, 2\omega + 1]$ correcting up to ω errors. Parameters are chosen such that the probability to invert a syndrome is in $1/\omega!$; in practice, ω is of order $O(log(2^m)) = O(m)$. From the security discussion, the parameters have to be chosen so that the Syndrome Decoding problem for the matrix $H = (R\|Q)$ is difficult for a weight 2ω.

A recent result has proven that the public matrix of the CFS scheme was distinguishable [29] from a random matrix. Nevertheless, this result did not give rise to any attack on the scheme which then remains usable. Indeed, in practice and despite the aforesaid distinguisher, best attacks to the CFS problem are generic and treat the CFS matrix as a random one.

CFS-based problems and augmented-CFS matrix To be claimed secure (Section 6), our generic scheme does not require the public matrix to be indistinguishable from random but only to be secure against some computational problems defined in Section 2 (namely SD and OMSD problems). In the case of random codes, these problems are either proven hard (SD problem) or assumed to be (OMSD problem). Assuming the putative security of the CFS scheme, we also consider that these problems are hard when instantiated with a CFS public matrix which can then be seen as a trapdoor matrix satisfying Definition 2.

This leads us to deal with a global public matrix of the form $\tilde{H} = (R\|Q)$ where R is random and Q is CFS public matrix; we then define *augmented-CFS* matrices as follows:

Definition 6. *Let $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$ a CFS public matrix and $R \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$ a random binary matrix. We say that the following matrix $\tilde{H} = (R\|Q)$ is an augmented-CFS matrix.*

Since computational problems defined in Section 2 are also assumed to be hard using the CFS scheme, we can naturally extend this hardness to *augmented-CFS* matrices.

Remark 4. *K. P. Mathew et al. recently proposed a provably secure code-based signature scheme [30]. Their work roughly consists in masking the public matrix of the original CFS scheme so that there exists no distinguisher with random matrices. Nevertheless, since their scheme leads to greater keys sizes and ours does not require indistinguishabilty from random to fulfill security (Section 6), we focus on the classic CFS scheme when studying parameters.*

7.2 Parameters

We then instantiate our generic scheme with *augmented-CFS* matrices and consider, relying on previous subsection, the choice of parameters as if we were dealing with random ones. From the best known attacks [31, 32] we can choose $m = 20$ and $\omega = 10$. It leads to a matrix $\tilde{H}$ 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 decreases 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.

The number of syndromes $2^{m\omega}$, for a $[2^m, 2^m - m\omega, 2\omega + 1]$ Goppa Code, gives an upper bound on the number of users. In fact, this number of users corresponds to the number of decodable

random syndromes which is equivalent to $2^{m\omega}/\omega!$. In practice, one considers $\omega = O(m)$; hence, from the Stirling formula, it gives $2^{O(m^2)}$ possible users. Since public keys matrices and signatures sizes are proportional to the length of code 2^m, this leads sizes of order $N^{1/\sqrt{log(N)}}$, for N the number of group members.

Finally, parameters as chosen above lead to a signature (a transcript of proof of knowledge of a small word associated to $\tilde{H}$) 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 $Join$ process so that, on 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.

As a final remark, let us also notice that in the quantum setting, it would be required to consider a security parameter λ greater than 80. Even if theoretically feasible, this may to lead extreme sizes of parameters in the case of CFS.

8 Conclusion

This work is the extended version of the code-based group signature scheme proposed in [1]. To fulfill security, we introduced a new problem in code-based cryptography referred as the *One More Syndrome Decoding* problem and a new kind of proof of knowledge, referred as *Testable weak Zero-Knowledge*, for which a verifier is able to test whether some specific information is related to the prover's secret, without learning anything more from the proof. The main idea of our work was to build an offset 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 to which a fixed syndrome is added. Applying a variation of Stern's protocol on these two small weight vectors led us to design our group signature scheme, secure under the ROM, through the use of Fiat-Shamir paradigm. In spite of great sizes of keys, common to all post-quantum group signatures, our instantiation proposes satisfying asymptotic performances since group public key and signatures sizes are proportional to $N^{1/\sqrt{log(N)}}$. In particular, our scheme compares well with [19] which is based on the static BMW model and presents size of parameters linear in the number of users. Finally, we assume that the elegance, the simplicity and the large range of properties it fulfills make this scheme a good alternative to all other post-quantum constructions. Indeed, not only does it allow to dynamically add new members but it should be noted that its extension into a traceable signature (KTY model) handling member revocation appears rather straightforward.

Acknowledgments The authors would like to thank the anonymous reviewers for their helpful comments that led to notably improve the present work.

References

[1] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A Code-Based Group Signature Scheme. In Jean-Pierre Tillich Pascale Charpin, Nicolas Sendrier, editor, *The 9th International Workshop on Coding and Cryptography 2015 WCC2015*, Proceedings of the 9th International Workshop on Coding and Cryptography 2015 WCC2015, Paris, France, April 2015.

[2] 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.

[3] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 136–153, 2005.

[4] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 157–174, 2001.

[5] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Information Theory*, 24(3):384–386, 1978.

[6] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, pages 257–265, 1991.

[7] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 614–629, 2003.

[8] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.

[9] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 168–177, 2004.

[10] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 56–72, 2004.

[11] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, pages 193–210, 2006.

[12] Jens Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 164–180, 2007.

[13] Benoît Libert and Moti Yung. Efficient traceable signatures in the standard model. In *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, pages 187–205, 2009.

[14] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 395–412, 2010.

[15] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 41–61, 2013.

[16] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 345–361, 2014.

[17] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 427–449, 2015.

[18] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 401–426, 2015.

[19] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. *IACR Cryptology ePrint Archive*, 2015:479, 2015.

[20] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. *IACR Cryptology ePrint Archive*, 2016:101, 2016.

[21] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

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

[23] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optimia in lattices, codes, and systems of linear equations. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 724–733, 1993.

[24] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.

[25] Jacques Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 13–21, 1993.

[26] Jacques Stern. A new paradigm for public key identification. *IEEE Trans. Information Theory*, 42(6):1757–1768, 1996.

[27] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[28] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[29] Jean-Charles Faugère, Valérie Gauthier-Umaña, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate mceliece cryptosystems. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 282–286, 2011.

[30] K. Preetha Mathew, Sachin Vasant, and C. Pandu Rangan. *A Provably Secure Signature and Signcryption Scheme Using the Hardness Assumptions in Coding Theory*, pages 342–362. Springer International Publishing, Cham, 2014.

[31] 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 *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 520–536, 2012.

[32] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 88–105, 2009.