On the (In)security of optimized Stern-like signature schemes

André Chailloux, Simona Etinski

▶ **To cite this version:**

André Chailloux, Simona Etinski. On the (In)security of optimized Stern-like signature schemes. Designs, Codes and Cryptography, In press. hal-04320650

HAL Id: hal-04320650
https://inria.hal.science/hal-04320650v1

Submitted on 4 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - ShareAlike 4.0 International License

On the (In)security of optimized Stern-like signature schemes

André Chailloux[1] and Simona Etinski[2]

1. Inria de Paris, andre.chailloux@inria.fr
2. CWI, Amsterdam, simona.etinksi@cwi.nl

Abstract. Stern's signature scheme is a historically important code-based signature scheme. A crucial optimization of this scheme is to generate pseudo-random vectors and permutation instead of random ones, and most proposals that are based on Stern's signature use this optimization. However, its security has not been properly analyzed, especially when we use deterministic commitments. In this article, we study the security of this optimization. We first show that for some parameters, there is an attack that exploits this optimization and breaks the scheme in time $O(2^{\frac{\lambda}{2}})$ while the claimed security is λ bits. This impacts in particular the recent Quasy-cyclic Stern signature scheme [BGKS22]. Our second result shows that there is an efficient fix to this attack. By adding a string $salt \in \{0,1\}^{2\lambda}$ to the scheme, and changing slightly how the pseudo-random strings are generated, we prove not only that our attack doesn't work but that for any attack, the scheme preserves λ bits of security, and this fix increases the total signature size by only 2λ bits. We apply this construction to other optimizations on Stern's signature scheme, such as the use of Lee's metric or the use of hash trees, and we show how these optimizations improve the signature length of Stern's signature scheme.

1 Introduction

As the NIST standardization process [NIS17] for post-quantum cryptography advances, the challenge of building efficient and secure post-quantum signature schemes becomes increasingly demanding. Though lattice-based schemes seem as the most promising candidates, their well-studied alternatives from code-based cryptography are gaining more attention recently. The lack of efficiency of code-based signature schemes, as their major drawback, is addressed in the work of many cryptographers, and different techniques that aim to circumvent this problem were proposed. For some of these techniques, however, it remained unclear if the schemes stayed secure after decreasing the signature size, i.e. if the security reduction still holds in that case.

In this article, we focus on the signature schemes derived from the identification protocol originally proposed by Jacques Stern [Ste94]. Its security is based on the hardness of the syndrome decoding problem [BMvT78] and the Fiat-Shamir transform [FS87], which is well-analyzed with respect to classical and quantum adversaries. The main drawback of Stern's signature scheme in its original form, the same as of the other code-based schemes, is large signature size (order of a few hundreds kilobytes). Different variants of the scheme are thus proposed to mitigate this problem. Some of them, for example, reduce the signature sizes by reducing the cost of each round of the underlying identification protocol. Some versions use the cut and choose technique or even modify the underlying syndrome decoding problem ([Vér97], [AGS11], [CVE11], [GRSZ14], [Beu20], [BBC+20], [BGKS22], [FJR21], [FJR22]).

The goal of this article is to study several variants of optimized Stern's signature scheme and to assess their security. We first discuss here two features that appear in Stern's signature and other schemes that use the Fiat-Shamir transform.

Random seeds for constructing pseudo-random strings. In the original Stern's scheme, the signer must generate random vectors $\mathbf{y}_i$ as well as a random permutations π_i. In the signature, some of these have to be revealed and for others, we only have the commitment to these values. Revealing all these values is very costly and makes the signature very long. The idea is

to replace these random strings by pseudo-random strings. For example, if we want to construct a vector $\mathbf{y} \in \mathbb{F}_q^n$, we use a function $E : \{0,1\}^{l_{\text{seed}}} \to \mathbb{F}_q^n$, generate a random seed $\in \{0,1\}^{l_{\text{seed}}}$ and set $\mathbf{y} = E(\text{seed})$. When one wants to reveal $\mathbf{y}$, it is enough to reveal the seed which is much shorter.

This idea has been used in several proposals but has never been properly proven and we don't even know to what extent this optimization remains secure. In order to maintain λ bits of security[1] should we have $l_{\text{seed}} = 2\lambda$ in order to avoid any collision between the seeds, or can we take $l_{\text{seed}} = \lambda$ which ensures that $\mathbf{y}$ has λ bits of randomness? As we will see, many designs use seeds of size λ which seems at first sight to be enough.

Using commitment from hash functions The following textbook construction builds a commitment scheme from a hash function $\mathscr{H}$: in order to commit to a string s, pick a random string $\rho \in \{0,1\}^{2\lambda}$ and send as a commitment $c = \mathscr{H}(s, \rho) \in \{0,1\}^{2\lambda}$. In order to reveal s, one also reveals the random value ρ. This commitment scheme has λ bits of security against collision attacks, and also has a strong hiding property. For example, in MQDSS [CHR+20] , we can read (where k is the security parameter):

> *In particular the commitment functions now take an additional argument - a random string of length 2k. The reason for this change is that with this additional input it can be shown that the commitment function used in MQDSS is computationally hiding - a property needed to show the EU-CMA security of MQDSS.*

The string they refer to is exactly the string ρ. However, in many proposer schemes, this randomization ρ is omitted from the commitment scheme. Indeed, if one has to reveal a string ρ of size 2λ for each opening, this can greatly increase the size of the signature scheme constructed from this commitment. We will call a commitment with ρ a randomized commitment and a commitment without this ρ a deterministic commitment. So what it the correct thing to do? Is it necessary to use randomized commitments or can we use deterministic commitments?

Strikingly, each proposal that uses these 2 ingredients doesn't give the same answers to these questions, some are quite conservative while others take the maximum amount of risk. For instance

- [BGKS22] in an optimized Stern's signature scheme, uses seeds of size λ and deterministic commitments.
- [BBC+20] relies on restricted syndrome decoding, uses seeds of size 2λ and deterministic commitments.
- As we said, the MQDSS signature scheme uses seeds of size λ bits and commitments of size 2λ. However, the commitments are probabilistic so our attack doesn't work but this is makes the scheme quite inefficient.

1.1 Contributions

Our contributions can be divided in 3 parts:

1. First we show that one has indeed to be careful with some optimizations. In Stern's signature scheme, if one adds seeds of size λ and deterministic commitments, then we only achieve $2^{\lambda/2}$ bits of security instead of λ bits. We show an attack in $O(2^{\lambda/2})$ that recovers the secret key in this setting. This directly attacks the scheme presented in [BGKS22].
2. We present the *salt* + index construction for commitments in order to circumvent the attack by adding a minimal amount of randomness in the commitment scheme. We show that with this addition, Stern's signature scheme has λ bits of security and the cost in terms of signature size is minimal (a total increase of 2λ bits to be precise).
3. We look at other optimizations of Stern's signature scheme: for example by using Lee's metric or by adding Merkle trees, how to improve even more the size of Stern's signature scheme.

We now elaborate each of these contributions.

[1] When we write maintain λ bits of security, we mean that if the original scheme has λ bits of security then the optimized scheme should also have λ bits of security.

Showing the insecurity of some optimized Stern's signature schemes.

Our first contribution is the following

Theorem 1. *If we use seeds of size λ and deterministic commitments in Stern's signature scheme, then there is an attack that recovers the secret key in time $O(2^{\lambda/2})$.*

In other words, the choice of [BGKS22] for instance gives only $\lambda/2$ bits of security and not λ as claimed. We present here the main idea behind our attack, which is quite simple. Assume you construct a pseudo-random string $\mathbf{y}$ using a randomseed st. $E(\text{seed}) = \mathbf{y}$ and that you commit to $\mathbf{y}$ using a deterministic commitment $c = \mathscr{H}(\mathbf{y})$. If you perform such a commitment many times, which happens when you perform many calls to the signing oracle, then you will arrive at a collision wp. $O(2^{\lambda/2})$ (at least each time there is a collision in the seeds) and an adversary can exploit such a collision. Indeed, if for one signature, this commitment is opened and for another signature it isn't opened, the adversary will be able to guess the committed value for this second signature which will break the scheme.

Mitigating the attack: the *salt+* index construction

On multi-HVZK advantage. Before we discuss the way to circumvent the above attack, let's discuss on how come the attack we present exists? Stern's signature scheme is constructed by applying the Fiat-Shamir transform to an identification scheme and this transformation is supposed to be secure. An identification scheme is required to be honest-verifier zero-knowledge and the identification scheme used for instance by [BGKS22] is HVZK. So where is the issue? Actually, if one wants to prove that a signature scheme constructed from an identification scheme is EUF-CMA, we require the identification scheme to be multi-HVZK meaning that it is honest-verifier zero-knowledge even when seeing many transcripts arising from the identification scheme. Normally, this is not a big issue and the HVZK property will imply the multi-HVZK but this needn't be always the case. In [GHHM21], where the authors argue about the security of the Fiat-Shamir transform, they write

> *"In our security proofs, we will have to argue that collections of honestly generated transcripts are indistinguishable from collections of simulated ones. Since it is not always clear whether computational HVZK implies computational HVZK for multiple transcripts, we extend our definition, accordingly: In the multi-HVZK game, the adversary obtains a collection of transcripts (rather than a single one)."*

Here, we are exactly in a setting where this subtlety matters. If we consider one call of Stern's optimized identification scheme, then one can construct a simulator st. any adversary that wants to distinguish the real transcript from a simulated transcript needs average time $O(2^{\lambda})$. However, there is an adversary that runs in time $O(2^{\lambda/2})$ which can recover the secret key with overwhelming probability if he is given $\Omega(2^{\lambda/2})$ independent transcripts. In order words, the t-HVZK property breaks for $t = 2^{\lambda/2}$.

The salt + index construction. The above attack can be circumvented if we use a different hash function $\mathscr{H}$ each time we perform a commitment. This is exactly the role of the string ρ in the randomized commitment scheme. In theory, we could have a counter i and use as a commitment $\mathscr{H}(s, i)$ the i^{th} time we perform a commitment. However, since we run the signing oracle several times, we would need to remember this counter between different signatures and that would make the scheme stateful. There ie however an idea along these lines that will work:

- Each time you sign, pick a random string $salt \in \{0,1\}^{2\lambda}$ that will act as a randomization string. What is important is that this string is the same for all the commitments used in the signature scheme so we will have to reveal only one such string.
- *Within each signature:* add a counter (which we call index) to ensure that within the same call to the signing oracle, we don't call the same function twice.

Since *salt* is the same throughout the signature, the whole signature is increased only by $l_{\text{salt}} = 2\lambda$ bits. One can see that our attack will not work. Even if there are collisions in the seeds, the hashed values will be with different salts or different indices, and this negates our attack. Moreover, we show that the resulting scheme is mutli-HVZK, which ensures that no other attacks of this type exist.

Theorem 2. *The optimized Stern's signature scheme with* salt + *index has*

$$Adv^{t-HVZK}(\mathcal{A}) = Adv^{SD}(|\mathcal{A}|) + O(\frac{t^2}{2^{l_{\text{salt}}}}) + O(\frac{t}{2^{l_{\text{seed}}}}).$$

The way to read this theorem is the following: it is essentially as hard to break the t-HVZK property than to break the syndrome decoding problem (which is the hard computational problem underlying the whole scheme), as long as $l_{\text{salt}} \geq t^2$ and $l_{\text{seed}} \geq t$. t here actually corresponds to the number of calls allowed to the signing oracle, which is often taken to be 2^λ. This motivates the choice $l_{salt} = 2\lambda$ and $l_{\text{seed}} = \lambda$. Using existing results on the security of the Fiat-Shamir transform, we have the following (informal) corollary.

Corollary 1. *Stern's optimized signature scheme with salt + index is EUF-CMA secure*

Let us discuss the technical difficulties we had to overcome in order to prove our theorem. It seems at first strange that the advantage of solving the syndrome decoding problem appears in the statement of our theorem but we will make this clear. We want to construct a simulator that will output transcripts computationally indistinguishable from real transcripts. In the identification scheme, the prover commits to some strings and reveals some of them. How should we simulate the committed values $comm = \mathscr{H}(s, salt, index)$? The natural solution is to simulate this with a random string. Indeed, we model $\mathscr{H}$ as a random function in the random oracle model so the commitment is a random string. This reasoning is not entirely correct. Since *salt* and index are publicly known (*salt* is revealed most often by another opening) it suffices for an adversary to guess s in order to distinguish c from a random string. This shows the difficulty of such zero-knowledge proofs with deterministic commitment, even without pseudo-random strings and seeds. As we mentioned earlier, if we use probabilistic commitments, this problem disappears and we can show that the commitment is statistically indistinguishable from a random string.

Going back to our case, we said that $comm = \mathscr{H}(z, salt, index)$ can be distinguished from a random string if we can guess z. This will actually be the only way to distinguish *comm* from a random string, even in the t-HVZK setting, as long as each *salt* is different (which happens wp. $O(\frac{t^2}{2^{l_{\text{salt}}}})$). Stern's identification scheme consists of many repetitions of an identification scheme where the prover commits to 3 strings z_1, z_2, z_3 and reveals 2 out of the 3 depending on the verifier's challenge. In this context, guessing the string z means guessing the string that has not been revealed. But we show that if the adversary can guess this string, then he can recover the secret key, meaning he can solve the underlying syndrome decoding problem. Regarding seeds, we show that if we have salts, the only way to distinguish the random strings from the pseudo-random strings is to guess a seed value, which happens wp. $2^{-l_{\text{seed}}} = 2^{-\lambda}$ so with the *salt* + index construction, seeds of size λ suffice.

Adding other optimizations Now that we have theoretic foundations that show the security of our optimized Stern's signature scheme. We propose some instantiations where we also add optimizations.

Changing the metric. Stern's identification scheme was originally constructed based on the hardness of the binary syndrome decoding problem but actually, Shamir already proposed a similar scheme based on the permuted kernel problem, which can be seen as a generalization of the binary syndrome decoding. Several variants of this problem: ternary syndrome decoding with large weight [DST19], restricted syndrome decoding, syndrome decoding with Lee's weight [CDE21,WKH+22,HTW21] have been proposed and studied. In [CDE21], the current

authors and Thomas Debris-Alazard presented a cryptanalysis of syndrome decoding using Lee's metric with the idea of proposing Stern's signature scheme Similar schemes but with different optimizations) have also been proposed. Here, we show that Stern's optimized scheme with Lee's metric gives better results than Stern's original scheme.

Hash trees. Once we allow ourselves seeds, there are actually many ways to combine seeds. Here, we show an optimization which is a good compromise between performance and space gain: we regroup 4 rounds of the identification scheme together and we construct the seeds and commitments according to a Merkle tree. We show that this can also lead to some gains.

2 Hard Problems in Code-Based Cryptography

In this work, we focus on two hard problems in code-based cryptography: the Syndrome Decoding (SD) problem and the Permuted Kernel Problem(PKP). The two have many properties in common, namely, both of them are known to be NP-complete [BMvT78,GJ90] and they are believed to be hard on average once the instance of the problem is taken from the appropriate distribution.

2.1 Syndrome Decoding (SD) problem

The canonical hard problem used in code-based cryptography is the syndrome decoding problem, defined as follows.

Problem 1 (Syndrome Decoding, $\mathrm{SD}(n,k,w)$*).*

Input: A matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $n,k \in \mathbb{N}$, a column vector (the syndrome) $\mathbf{s} \in \mathbb{F}_q^{n-k}$, a weight function $wt(\cdot) : \mathbb{F}_q^n \to \mathbb{N}$, and a weight $w \in \mathbb{N}$.

Goal: Find a column vector $\mathbf{e} \in \mathbb{F}_q^n$ that satisfies $\mathbf{He} = \mathbf{s}$ and $wt(\mathbf{e}) = w$.

We call q the alphabet size of the problem. The original version of the problem, which we refer to as the binary syndrome decoding problem, corresponds to the case where $q = 2$ and the weight function is Hamming weight, defined as: $wt_H(\mathbf{e}) = |\{i : e_i \neq 0\}|$. The decision version of this problem, asking whether there exists a vector $\mathbf{e}$ of Hamming weight w satisfying $\mathbf{He} = \mathbf{s}$, is proven to be NP complete[BMvT78]. This is problem is also believed to be hard on average, as summarized by the following conjecture.

Conjecture 1 (Average-case hardness of syndrome decoding).

For suitable choices of $R = \frac{k}{n} = \Theta(1)$ and $\omega = \frac{w}{n} = \Theta(1)$, $SD(n,k,w)$ defined over the Hamming weight $wt(\cdot)$ is hard on average when the input $(\mathbf{H},\mathbf{s})$ is sampled from the distribution $\mathscr{D}_{n,k,w}$ given as:

$$\mathscr{D}_{n,k,w} : \mathbf{H} \xleftarrow{\$} \mathbb{F}_q^{(n-k)\times n}, \mathbf{e} \xleftarrow{\$} S_w, \text{ return } (\mathbf{H}, \mathbf{s} = \mathbf{He}). \tag{1}$$

where S_w is the set of vectors of weight w.

Moreover, the hardness holds even in the presence of quantum computers. One has to be careful with the choice of parameters. Indeed, for constant values $0 < R, \omega < 1$, this problem can be solved in time $poly(n)$ using Prange's algorithm [Pra62] for $\omega \in [(1-R)\frac{q-1}{q}, 1-(1-R)\frac{q-1}{q}]$, while the problem is believed to be exponentially hard outside this zone even for quantum computers, see for instance [KT17].

2.2 Permuted Kernel Problem (PKP)

The Permuted Kernel Problem was introduced by Shamir [Sha89], and has the same linear algebra flavor as the syndrome decoding problem. The main difference is that the weight constraint is replaced with a combinatorical constraint, which seemingly removes the connection with decoding linear codes, while preserving the NP-completeness of the problem. The problem is defined as follows.

Problem 2 (Permuted Kernel Problem $\mathrm{PKP}(n, k, \mathbf{v})$*).*

Input: A matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $n, k \in \mathbb{N}$, a column vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$, a column vector $\mathbf{v} \in \mathbb{F}_q^n$.

Goal: Find a permutation σ acting on $[n]$ that satisfies $\mathbf{H}\sigma(\mathbf{v}) = \mathbf{s}$.

As in the case of the syndrome decoding problem, it is believed that the permuted kernel problem is hard on average for suitable choices of $\mathbf{v}$, i.e. given the instances of the problem is sampled from the distribution $\mathscr{D}_{n,k}(\mathbf{v})$ sampled as follows:

$$\mathscr{D}_{n,k}(\mathbf{v}) : \mathbf{H} \xleftarrow{\$} \mathbb{F}_q^{(n-k)\times n}, \sigma \xleftarrow{\$} Perm_{[n]}, \text{ return } (\mathbf{H}, \mathbf{s} = \sigma(\mathbf{v}), \mathbf{v}).$$

One can see that the permuted kernel problem is similar to the syndrome decoding one in many respects. In fact, if we observe the binary syndrome decoding problem over the Hamming weight, and the corresponding permuted kernel problem over the binary field, we can easily show that the two are equivalent.[2] For other weight functions, however, they are not equivalent as the set of potential solutions of the syndrome decoding problem is generally wider than the set of potential solutions of the permuted kernel problem[3]. Nevertheless, it can be shown that for the properly chosen parameters, finding a solution of the permuted kernel problem yields a solution to the syndrome decoding problem, as summarized in the following proposition.

2.3 Reduction from SD to PKP

Other versions of syndrome decoding problem, namely those with a weight function other than the Hamming weight and an alphabet size greater than 2, start to attract more attention of the cryptographic community in the recent years as they yield potentially harder computational problems and hence more efficient code-based schemes. One example of a weight function that is known from coding theory and that recently gained more attention in cryptography is the Lee weight wt_L, defined as: $\forall \mathbf{e} = (e_0, ..., e_{n-1}) \in \mathbb{F}_q^n, \quad wt_L(\mathbf{e}) = \sum_{i=0}^{n-1} \min(e_i, q - e_i)$. The hardness of $SD(n, k, w)$ using Lee's weight has been studied against classical adversaries as well as the quantum ones [CDE21,WKH+22] in the Information Set Decoding framework and it is shown that problem is computationally hard in both cases.

Definition 1 (Weight function from distance function). *We say that a weight function* $wt : \mathbb{F}_q^n \to \mathbb{N}$ *is constructed from a distance function is there exists a distance function* $d : \mathbb{F}_q \times F_q \to \mathbb{N}$ *st.*

$$\forall \boldsymbol{e} = (e_0, \ldots, e_{n-1}) \in \mathbb{F}_q^n, \quad wt(\boldsymbol{e}) = \sum_{i=0}^{n-1} d(e_i, 0). \tag{2}$$

[2] If the weight constraint from the syndrome decoding problem, $wt_H(\mathbf{e}) = w$, is replaced with the permutation constraint from the permuted kernel problem, $\mathbf{e} = \sigma(\mathbf{v})$ (where $\mathbf{v}$ is given and it satisfies $wt_H(\mathbf{v}) = w$), a solution to first problem yields a solution to the second, and vice versa.

[3] The number of non-binary words of weight w, length n, and alphabet of size q is given as the number of permutations of any word of a given weight and length multiplied by the number of compositions of w into n parts taking values in $\{0, \ldots, q-1\}$. In the binary case, there is only one possible such composition, while in the non-binary case the number of such compositions can be, and in most cases it is, greater than 1.

This definition allows us to generalize the syndrome decoding problem over different metric spaces with potentially better security and/or better efficiency.

Definition 2 (Vector decomposition). *Let $\mathbf{x} \in \mathbb{F}_q^n$. The decomposition of $\mathbf{x}$ is a q-tuple $\mathbf{c} = (c_0, c_1, \ldots, c_{q-1})$ in which c_0 counts the number of zeros in $\mathbf{x}$, c_1 counts the number of ones, ..., and c_{q-1} counts the number of $q-1$ elements in $\mathbf{x}$. In other words:*

$$c_i = |\{j \in \{0, \ldots, n-1\} : x_j = i\}|.$$

Lemma 1 ([Bog83]). *The number of vectors in $\mathbb{F}_q^n$ having the same decomposition $\mathbf{c} = (c_0, c_1, \ldots, c_{q-1})$ is given by the multinomial coefficient $\binom{n}{\mathbf{c}} = \binom{n}{c_0, c_1, \ldots, c_{q-1}} \stackrel{def}{=} \frac{n!}{c_0! c_1! \cdots c_{q-1}!}$.*

We now prove what is essentially a reduction from the SD problem to the PKP problem.

Proposition 1 (Reduction from SD to PKP). *Take a constant alphabet size q. Assume we have an algorithm $\mathcal{A}$ that can solve the $PKP(n, k, \mathbf{v})$ problem in time t. Then, we can use this algorithm to solve randomly chosen instance $(\mathbf{H}, \mathbf{s}) \leftarrow D_{n,k,wt(\mathbf{v})}$ in time t wp. $\geq \Omega(\frac{1}{\mathsf{poly}(q)})$. This is true for any weight function $wt()$ satisfying Definition 1.*

Proof. Let us fix an instance $(\mathbf{H}, \mathbf{s})$ of the syndrome decoding problem sampled from $\mathscr{D}_{n,k,w}$, and let $\mathbf{e} \in S_w$ be the random vector st. $\mathbf{He} = \mathbf{s}$ sampled when constructing $(\mathbf{H}, \mathbf{s})$, as it is done in $\mathscr{D}_{n,k,w}$. Recall that $S_w \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{F}_q^n : wt(\mathbf{x}) = w\}$. Let also C_w be the set of all decompositions of the elements from S_w, namely:

$$C_w \stackrel{\text{def}}{=} \{\mathbf{c} = (c_0, \cdots, c_{q-1}) \in \mathbb{N}^q : \forall i \in \{0, \ldots, q-1\}, \sum_{i=0}^{q-1} c_i = n \text{ and } \sum_{i=0}^{q-1} c_i d(0, i) = w\}.$$

The size of S_w can now be calculated as the number of vectors having a decomposition in C_w, summing over $\mathbf{c} \in C_w$. Using 1, we get

$$|S_w| = \sum_{\mathbf{c} \in C_w} \binom{n}{\mathbf{c}}.$$

Now, let C be the set of all decompositions, namely

$$C \stackrel{\text{def}}{=} \{\mathbf{c} = (c_0, \cdots, c_{q-1}) \in \mathbb{N}^q : \forall i \in \{0, \ldots, q-1\}, \sum_{i=0}^{q-1} c_i = n\}.$$

The size of this set C is known to be $\binom{n+q-1}{q-1}$[Bog83]. Moreover, we have $C = \bigcup_w C_w$ hence $|C_w| \leq |C|$ for each w. This implies

$$|S_w| = \sum_{\mathbf{c} \in C_w} \binom{n}{\mathbf{c}} \leq \binom{n+q-1}{q-1} \max_{\mathbf{c} \in C_w} \binom{n}{\mathbf{c}}.$$

Let $\tilde{\mathbf{c}}$ st. $\max\limits_{\mathbf{c} \in C_w} \binom{n}{\mathbf{c}} = \binom{n}{\tilde{\mathbf{c}}}$, and let $\tilde{\mathbf{v}}$ be a vector that has the decomposition given by $\tilde{\mathbf{c}}$. We then fix an instance of the permuted kernel problem, $\mathrm{PKP}(n, k, \widetilde{v})$, for which the parity check matrix, and the syndrome are the same as for the syndrome decoding instance.

We run our algorithm $\mathcal{A}$ to find a solution of this PKP instance we constructed. A solution to this problem will directly yield a solution to our original SD problem, but it is possible that there is no solution to this PKP problem. Notice however that if the error vector $\mathbf{e}$ (given by the $SD(n, k, w)$ instance) has the decomposition $\tilde{\mathbf{c}}$, then our $PKP(n, k)$ instance will have a solution. Given that $\mathbf{e}$ is sampled uniformly at random from the set S_w, this probability is then given by:

$$\frac{\binom{n}{\tilde{\mathbf{c}}}}{|S_w|} \geq \frac{\binom{n}{\tilde{\mathbf{c}}}}{\binom{n+q-1}{q-1}\binom{n}{\tilde{\mathbf{c}}}} = \frac{1}{\binom{n+q-1}{q-1}} \geq \frac{1}{(n+q-1)^{q-1}} = \Omega\Big(\frac{1}{\mathsf{poly}(n)}\Big),$$

3 Identification and digital signature schemes

3.1 Identification schemes

First definitions In this article, we study 3-round identification schemes, also known as Σ-protocols, defined as follows.

Definition 3 (3-round identification scheme). *A 3-round identification scheme is an interactive protocol consisting of the following 3 algorithms, which should be efficient with respect to some security parameter λ:*

- *a key generation algorithm $Keygen(1^\lambda) \to (pk, sk)$,*
- *the prover's algorithms $P = (P_1, P_2)$ satisfying $P_1(sk) \to (x, St)$ and $P_2(sk, x, c, St) \to z$, where x is the first message, St is some internal state, c is the challenge from the verifier and $z \in R$ the prover's response (i.e. second message),*
- *the verifier's algorithm $V(pk, x, c, z)$ that outputs 1 if the verifier accepts the interaction and 0 otherwise.*

We describe below the main steps of an identification scheme.

Identification scheme IS

Initialization. Let $(pk, sk) \leftarrow Keygen(1^\lambda)$. The prover gets (pk, sk) and the verifier gets pk.
Interaction.

1. The prover generates $(x, St) \leftarrow P_1(sk)$ and sends x to the verifier.
2. The verifier picks a uniform c from the challenge set C and sends c to the prover.
3. The prover generates $z \leftarrow P_2(sk, x, c, St)$ and sends z to the verifier.

Verification. The verifier accepts iff. $V(pk, x, c, z) = 1$.

A transcript consists of the messages exchanged between the prover and the verifier when running IS. In our setting, a transcript is described by (x, c, z). We now define different distribution sof transcripts.

Definition 4 (Distributions of transcripts, honest behavior). *For an identification scheme* IS*, $trans_{\mathrm{IS}}(pk, sk)$, is the distribution of transcripts when both the prover and the verifier are honest. This distribution is sampled as follows:*

$$trans_{IS}(pk, sk) : (x, St) \leftarrow P_1(sk), c \leftarrow C, z \leftarrow P_2(sk, x, c, St), \mathrm{return}(x, c, z).$$

We also consider transcripts in the case an adversary $\mathcal{A}$ who only knows pk tries to impersonate the prover.

Definition 5. *For an identification scheme* IS *and an adversary $\mathcal{A}(pk)$, $trans^{\mathcal{A}}_{\mathrm{IS}}(pk)$, is the distribution of transcripts when an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that only knows the public key interacts with the verifier.*

$$trans^{\mathcal{A}}_{\mathrm{IS}}(pk) : (\tilde{x}, \tilde{St}) \leftarrow \mathcal{A}_1(pk), c \leftarrow C, \tilde{z} \leftarrow \mathcal{A}_2(pk, \tilde{x}, c, \tilde{St}), \mathrm{return}(\tilde{x}, c, \tilde{z}).$$

Completeness,soundness and zero-knowledge An identification must satisfy the completeness, soundness and zero-knowledge properties that we describe below.

Definition 6 (Completeness). *An identification scheme* IS *is perfectly complete*[4] *iff.*

$$\Pr\left[V(pk,x,c,z)=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\(x,c,z)\leftarrow trans_{\mathrm{IS}}(pk,sk)\end{smallmatrix}\right.\right]=1.$$

For the soundness property, we define the soundness advantage as follows

Definition 7 (Soundness Advantage). *Let* IS *be an identification scheme. For any adversary* $\mathcal{A}$

$$Adv_{\mathrm{IS}}(\mathcal{A})=\Pr\left[V(pk,x,c,z)=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\(x,c,z)\leftarrow trans_A(pk)\end{smallmatrix}\right.\right].$$

Notice that we write these definitions in order to perform concrete security statements and not asymptotic statements, where we would only require that an adversary that runs in time $poly(\lambda)$ has soundness advantage $negl(\lambda)$.

The zero-knowledge property ensures that the transcripts generated by the identification scheme do not reveal any information, in particular about the secret key. The way the notion of not revealing any information is formalized is through the existence of an efficient simulator $Sim(pk)$ who can output transcripts indistinguishable from real transcripts. For a specified simulator, $Adv_{\mathrm{IS}}^{HVZK}(\mathcal{A})$ corresponds to the probability that an adversary $\mathcal{A}$ distinguishes a real transcript from a simulated transcript.

Definition 8 (Honest verifier Zero-knowledge advantage). *Let* IS *be an identification scheme for which we specified a simulator* Sim*. For any adversary* $\mathcal{A}$*, we define*

$$Adv_{\mathrm{IS}}^{HVZK}(\mathcal{A})=\left|\Pr\left[b=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\ \tau=(x,c,z)\leftarrow trans_{\mathrm{IS}}(pk,sk)\\ b\leftarrow\mathcal{A}(\tau,pk)\end{smallmatrix}\right.\right]-\Pr\left[b=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\ \tau=(x,c,z)\leftarrow Sim(pk)\\ b\leftarrow\mathcal{A}(\tau,pk)\end{smallmatrix}\right.\right]\right|.$$

As we said in the introduction, an important concept will actually be the notion of multi-HVZK, where we give to an adversary t real or simulated transcripts and we want to consider the distinguishing advantage in this case.

Definition 9 (Multi-Honest verifier Zero-knowledge advantage). *Let* IS *be an identification scheme for which we specified a simulator* Sim*. For any adversary* $\mathcal{A}$*, we define*

$$Adv_{\mathrm{IS}}^{t-HVZK}(\mathcal{A})=\left|\Pr\left[b=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\ \tau_1,\ldots,\tau_t\leftarrow trans_{\mathrm{IS}}(pk,sk)\\ b=\mathcal{A}(\tau_1,\ldots,\tau_t,pk)\end{smallmatrix}\right.\right]-\Pr\left[b=1 \left| \begin{smallmatrix}(pk,sk)\leftarrow Keygen(1^\lambda)\\ \tau_1,\ldots,\tau_t\leftarrow Sim(pk)\\ b=\mathcal{A}(\tau_1,\ldots,\tau_t,pk)\end{smallmatrix}\right.\right]\right|.$$

Parallel repetition of an identification scheme For any identification scheme IS, we define $\mathrm{IS}^{\otimes r}$ as the r-fold parallel repetition of IS, which consists of the following:

Identification scheme $\mathrm{IS}^{\otimes r}$

Initialization. Let $(pk,sk)\leftarrow Keygen(1^\lambda)$. The prover gets (pk,sk) and the verifier gets pk.
Interaction.

1. For each $i\in[r]$, the prover generates $(x^i,St^i)\leftarrow P_1(sk)$ and then sends $\mathbf{x}=x^1,\ldots,x^r$ to the verifier.
2. The verifier picks a random $\mathbf{c}=c^1,\ldots,c^r$ for independently randomly chosen challenges and sends $\mathbf{c}$ to the prover.
3. The prover generates $\mathbf{z}=(z^1,\ldots,z^r)$, where for each $i\in[r]$, $z^i\leftarrow P_2(sk,x^i,c^i,St^i)$, and then sends $\mathbf{z}$ to the verifier.

Verification. The verifier accepts iff. $\forall i\in[r], V(pk,x^i,c^i,z^i)=1$.

[4] In this paper, we construct a scheme that is perfectly complete. In general, this condition can be relaxed so that "almost perfectly complete scheme", where the probability above is very close to 1, can also be considered as a proper identification scheme.

3.2 Signature schemes

A signature scheme S consists of 3 algorithms $(\text{Keygen}_S, \text{Sign}_S, \text{Ver}_S)$:

- $\text{Keygen}_S(1^\lambda) \to (pk, sk)$ is the generation of the public key pk and the secret key sk from the security parameter λ.
- $\text{Sign}_S(m, pk, sk) \to \sigma_m$: generates the signature σ_m of a message m from m, pk, sk.
- $\text{Ver}_S(m, \sigma, pk) \to \{0, 1\}$ verifies that σ is a valid signature of m using m, σ, pk. The output 1 corresponds to a valid signature.

We require from our signature schemes correctness and the EUF-CMA property.

Definition 10. *A signature scheme is correct iff. when we sample* $(pk, sk) \leftarrow \text{Keygen}_S(1^\lambda)$*, we have for each* m

$$\Pr[\text{Ver}_S(m, \text{Sign}_S(m, pk, sk), pk)] = 1.$$

We consider the standard EUF-CMA security for signature schemes. To define the advantage of an adversary $\mathcal{A}$, we consider the following EUF-CMA game

EUF-CMA game
$(pk, sk) \leftarrow \text{Keygen}_S(1^\lambda)$. Run $(m^*, \sigma^*) \leftarrow \mathcal{A}(pk)$, where $\mathcal{A}(pk)$ can perform q sign queries. More precisely, $\mathcal{A}(pk)$ constructs $m_1, \ldots, m_q$ and receives $\sigma_1 \leftarrow \text{Sign}_S(m_1, sk, pk), \ldots, \sigma_q \leftarrow \text{Sign}_S(m_q, sk, pk)$. Then $\mathcal{A}$ performs a procedure to output (m^*, σ^*). $\mathcal{A}$ wins the game iff. $\text{Ver}_S(m^*, \sigma^*, pk) = 1 \wedge \forall i \in [q], m^* \neq m_i$.

The EUF-CMA advantage is defined as follows

Definition 11. *For a signature scheme* S *and an adversary* $\mathcal{A}$*, we define*

$$Adv_S^{EUF\text{-}CMA}(\mathcal{A}) = \Pr[\mathcal{A} \textit{ wins the EUF-CMA } \text{game}].$$

The Fiat-Shamir transform We can directly construct a signature scheme from an identification scheme via the Fiat-Shamir transform [FS87]. From an identification scheme $\text{IS} = (Keygen, P = (P_1, P_2), V)$, we define the following signature scheme $\text{S} = (\text{Keygen}_S, \text{Sign}_S, \text{Ver}_S)$ that uses a random function $\mathscr{H}$:

The Fiat-Shamir transform: constructing S from IS
- $\text{Keygen}_S(1^\lambda) = Keygen(1^\lambda)$ - $\text{Sign}_S(m, pk, sk) : (x, St) \leftarrow P_1(pk), c \leftarrow \mathscr{H}(x, m), z \leftarrow P_2(sk, x, c, St)$, output $\sigma = (x, z)$. - $\text{Ver}_S(m, \sigma = (x, z), pk) = V(pk, x, \mathscr{H}(x, m), z)$.

The transformation is proven to be secure in both classical and quantum random-oracle model, if the identification has good properties. Recently in [GHHM21], the Fiat-Shamir is proven to be secure essentially if the underlying identification scheme has low soundness advantage and low t-HVZK advantage. For Stern's identification scheme, the soundness advantage will be immediate and the main focus of our work will be on the t-HVZK advantage. For instance the attack we present later in the paper uses the fact that one of the proposed variants of Stern's identification has a small HVZK-advantage but a big t-HVZK advantage for t large enough.

3.3 Commitment schemes

A commitment scheme consists of 3 algorithms:

- A commit function $Commit(s) \rightarrow (c, St)$. c is the commitment and St is some internal state (usually some private randomness), used for the opening.
- An open function $Open(s, St) \rightarrow o$ that outputs the opening of the string s.
- A function that verifies the validity of an opening o for a string s and a commitment c: $Ver(c, s, o) \rightarrow \{0, 1\}$.

For a commitment c, (s, o) is valid iff. $Ver(c, s, o) = 1$. Ideally, we want 2 properties from a commitment scheme. First, we want a commitment scheme to be hiding, meaning that any adversary $\mathcal{A}$ receiving only the commitment c shouldn't have information about the committed value. More precisely, $\forall s, s'$, we want the following advantage to be as small as possible:

$$Adv^{Hiding}(\mathcal{A}) = \left| \Pr\left[b = 1 \,\middle|\, \substack{(c,St) \leftarrow Commit(s) \\ b \leftarrow \mathcal{A}(s,c)}\right] - \Pr\left[b = 1 \,\middle|\, \substack{(c,St) \leftarrow Commit(s') \\ b \leftarrow \mathcal{A}(s',c)}\right] \right|$$

Again, we deal with concrete security so we don't specify what condition we want for the hiding advantage. If we dealt with asymptotic security, we would require that for any adversary running in time $poly(\lambda)$ for some security parameter λ, the above advantage is negligible in λ. The second property we want from a commitment scheme is the binding property, which means that an adversary cannot produce a commitment which can be revealed to 2 different values. We therefore want the following advantage to be as small as possible:

$$Adv^{Binding}(\mathcal{A}) = \Pr[Ver(c, s, o) = Ver(c, s', o') = 1 \wedge s \neq s' \,|(c, s, o, s', o') \leftarrow \mathcal{A}()]$$

4 Stern's identification scheme

4.1 Description of the scheme

In it's original form, the protocol is based on the binary syndrome decoding problem over the Hamming weight, where the instance of the syndrome decoding problem is taken from the distribution $\mathscr{D}_{n,k,w}$ defined in the previous section. As mentioned in the introductory part, different variants of the scheme were proposed to mitigate the problem of protocol's high communication cost. The most prominent ones, however, are based on the use of pseudo-random generators, so we focus on them in the next two sections.

We describe the scheme for the case of binary syndrome decoding but our descriptions can be easily extended to the case of PKP with larger alphabet size q. We consider a commitment scheme $(Commit, Open, Ver)$ and let $Perm_n$ be the set of permutations acting on $[n]$. Stern's 1 round identification protocol is the following:

1 ***round Stern's identification scheme, generic commitment,*** IS^1_{Stern}

Initialization. Let $(\mathbf{H}, \mathbf{e}, \mathbf{s} = \mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $sk = \mathbf{e}$, $pk = (\mathbf{H}, \mathbf{s})$. The prover obtains (pk, sk) and the verifier obtains pk.

Interaction.

- The prover picks a random permutation $\pi \in Perm_n$ and a random column vector $\mathbf{y} \xleftarrow{\$} \mathbb{F}_q^n$, and then computes $\mathbf{t} = \mathbf{Hy}$. The prover computes $z_1 = (\pi, \mathbf{t})$, $z_2 = (\pi(\mathbf{y}))$, $z_3 = (\pi(\mathbf{y} + \mathbf{e}))$. For $i \in \{1, 2, 3\}$, he then constructs $(x_i, St_i) \leftarrow Commit(z_i)$ and sends each x_1, x_2 and x_3 to the verifier.

- The verifier sends a random challenge $c \xleftarrow{\$} \{1, 2, 3\}$ to the prover.

- The prover reveals information so that the verifier can recover z_i for each $i \in \{1, 2, 3\} \setminus c$. More precisely:

1. if $c = 1$, the prover computes $o_2 = Open(z_2, St_2), o_3 = Open(z_3, St_3)$ and sends z_2, z_3, o_2, o_3 to the verifier.
2. if $c = 2$, the prover computes $o_1 = Open(z_1, St_1), o_3 = Open(z_3, St_3)$ and sends z_1, z_3, o_1, o_3 to the verifier.
3. if $c = 3$, the prover computes $o_1 = Open(z_1, St_1), o_2 = Open(z_2, St_2)$ and sends z_1, z_2, o_1, o_2 to the verifier.

Verification.

1. if $c = 1$, the verifier checks that $|z_2 + z_3| = w$ and checks the validity of the commitments z_2, z_3 so he checks that $Ver(x_2, z_2, o_2) = Ver(x_3, z_3, o_3) = 1$.
2. if $c = 2$, the verifier gets $z_1 = (\pi, \mathbf{t})$ and $z_3 = \pi(\mathbf{y} + \mathbf{e})$. He checks that $\mathbf{H}(\pi^{-1}(z_3)) = \mathbf{H}(\mathbf{y} + \mathbf{e}) = \mathbf{t} + \mathbf{s}$. He also checks the validity of the commitments z_1 and z_3.
3. if $c = 3$, the verifier gets $z_1 = (\pi, \mathbf{t})$ and $z_2 = \pi(\mathbf{y})$. He checks that $\mathbf{H}(\pi^{-1}(z_2)) = \mathbf{H}(\mathbf{y}) = \mathbf{t}$. He also checks the validity of the commitments z_1 and z_2.

Definition 12. *Stern's identification scheme* IS_{Stern} *is the R-fold parallel repetition of* (IS^1_{Stern}) *described above, namely* $(\mathrm{IS}^1_{Stern})^{\otimes R}$*. When aiming for* λ *bits of classical security, we take* $R = \lambda \log_2(3)$*, and we double the number of rounds* R *if we want* λ *bits of quantum security.*

4.2 Basic properties of Stern's identification scheme

Stern's identification scheme is clearly complete. Soundness and HVZK properties are also known from this scheme and a thorough analysis has been done for instance in [Lei18]. Regarding soundness, we have the following

Proposition 2.

$$Adv_{\mathrm{IS}^1_{Stern}}(\mathcal{A}) \leq \frac{2}{3} + Adv^{Binding}(\mathcal{A}) + Adv^{SD}(\mathcal{A}).$$

where Adv^{SD} *denotes the probability to break the underlying syndrome decoding problem.*

Proof. The way this statement is proved actually by using a notion called 3-special soundness. We show that if an adversary can win the identification scheme simultaneously for the 3 challenges then: or he broke the binding property of the commitment function, or solved the underlying syndrome decoding problem. We refer the reader to [Lei18] for details of this proof.

Moreover, the above scheme is zero-knowledge, if the commitment scheme is hiding

Proposition 3.

$$Adv^{HVZK}_{\mathrm{IS}^1_{Stern}}(\mathcal{A}) \leq Adv^{Hiding}(\mathcal{A}).$$

Proof. We consider the following simulator:

Simulator

Input. A public key $pk = (\mathbf{H}, \mathbf{s})$.
Execution. Pick a random challenge $c \leftarrow \{1, 2, 3\}$.

- If $c = 1$, choose $z_2 \leftarrow \mathbb{F}_q^n, y \leftarrow S_w$ and let $z_3 = z_2 + y$. For $i \in \{2, 3\}$, compute $(x_i, St_i) \leftarrow Commit(z_i)$ and $o_i = Open(z_i, St_i)$.
- If $c = 2$, choose $\pi \leftarrow Perm_n, \mathbf{y} \leftarrow \mathbb{F}_q^n$ and set $\mathbf{t} = \mathbf{H}\mathbf{y}$, $z_1 = (\pi, \mathbf{t})$ and $z_2 = \pi(\mathbf{y})$. For $i \in \{1, 3\}$, compute $(x_i, St_i) \leftarrow Commit(z_i)$ and $o_i = Open(z_i, St_i)$.
- If $c = 3$, choose $\pi \leftarrow Perm_n, \mathbf{y}' \leftarrow \mathbb{F}_q^n$ and set $\mathbf{t} = \mathbf{H}\mathbf{y}' - \mathbf{s}$, $z_1 = (\pi, \mathbf{t})$ and $z_3 = \pi(\mathbf{y}')$. For $i \in \{1, 2\}$, compute $(x_i, St_i) \leftarrow Commit(z_i)$ and $o_i = Open(z_i, St_i)$.

  Compute $(x_c, St_c) \leftarrow Commit(\mathbf{0})$ where $\mathbf{0}$ is the all 0 string.

Output. $\tau = (x_1, x_2, x_3, c, o_{c'}, o_{c''})$ where $c' \neq c''$ and $c', c'' \neq c$.

We also rewrite the procedure that generates real transcripts

Real transcript

Input. A public key $pk = (\mathbf{H}, \mathbf{s})$. A secret key $sk = \mathbf{e}$.
Execution. Pick a random permutation $\pi \in Perm_n$ and a random string $y \in \mathbb{F}_q^n$. Let $\mathbf{t} = \mathbf{Hy}$ and $z_1 = (\pi, \mathbf{t}), z_2 = \pi(\mathbf{y}), z_3 = \pi(\mathbf{y} + \mathbf{e})$. Pick a random challenge $c \leftarrow \{1, 2, 3\}$. For $i \in \{1, 2, 3\}$, let $(x_i, St_i) \leftarrow Commit(z_i)$ and $o_i = Open(z_i, St_i)$.
Output. $\tau = (x_1, x_2, x_3, c, o_{c'}, o_{c''})$ where $c' \neq c''$ and $c', c'' \neq c$.

We now prove that distinguishing these 2 distributions is equivalent to breaking the hiding property of the commitment scheme. First, notice that $c \leftarrow \{1, 2, 3\}$ in both cases. Now, for each challenge, we look at the distribution of $z_{c'}, z_{c''}$ for the 2 values $c', c'' \in \{1, 2, 3\}$ which are different from c.

- If $c = 1$: both for the real and the simulated transcript, z_2 is a random string in $\mathbb{F}_q^n$ and $z_3 - z_2$ is a random string in S_w
- If $c = 2$: both for the real and the simulated transcript, z_3 is a random string in $\mathbb{F}_q^n$. Regarding $z_1 = (\pi, \mathbf{t})$, in both cases π is a random permutation in $Perm_n$ and $\mathbf{t} = \pi^{-1}(z_2) - \mathbf{s}$.
- If $c = 3$, both for the real and the simulated transcript, z_2 is a random string in $\mathbb{F}_q^n$. Regarding $z_1 = (\pi, \mathbf{t})$, in both cases π is a random permutation in $Perm_n$ and $\mathbf{t} = \pi^{-1}(z_2)$.

The same distribution for $z_{c'}, z_{c''}$ imply the same distributions for $(x_{c'}, x_{c''}, o_{c'}, o_{c''})$ given c for real and simulated transcripts. Finally, we are left with x_c. For real transcripts, it is the commitment of z_c and for simulated transcripts, it is the commitment of $\mathbf{0}$. By definition of the hiding property, if we can distinguish these 2 settings then we can break the hiding property of the commitment scheme.

4.3 Optimized schemes

The choice of commitment From a hash function $\mathscr{H}$, it is known how to construct a commitment scheme which is both hiding and binding. We take $Commit(s) = (c = \mathscr{H}(s||\rho), St = \rho)$ for a randomly chosen ρ of large enough length, and the opening consists of revealing the value ρ, from which one can verify that the commitment is well formed. However, this implies that in order to reveal s, one has to send this string ρ each time. We will call this commitment a randomized commitment

In the literature regarding Stern's signature scheme, we most often see another choice for commitment, namely $Commit(s) = (c = \mathscr{H}(s), St = \emptyset)$ and the opening is also empty so in order to open s, you can just reveal s and the verification just checks that $c = \mathscr{H}(s)$. This improves the communication cost. However, this scheme is not hiding anymore. Indeed, an adversary can distinguish $Commit(s)$ from $Commit(s')$ for fixed (and known) s, s' just by recomputing these 2 values (which he couldn't do before because he didn't know ρ). We will call this commitment a deterministic commitment.

We haven't seen the choice of deterministic commitment widely discussed regarding security. It seems that this has been the default choice (except for example in [Lei18] and in MQDSS which is not based on the syndrome decoding problem but for which a similar discussion exists) because of it's higher efficiency and because the lack of hiding property didn't seem to be exploitable. It could be that the hiding property given by randomized commitments is overkill and that deterministic commitments are enough for having a secure scheme. This question is actually at the core of our contributions, where we show that there are security issues regarding deterministic schemes, but that they can be repaired much more efficiently than by using the randomized commitments described above.

Adding seeds Another standard technique for reducing the communication cost of identification protocols is to replace some of the random vectors with pseudo-random ones. We thus

consider the following functions, i.e. pseudo-random generators:

$$E_0 : \{0,1\}^{l_{\text{seed}}} \to \{0,1\}^{l_{\text{seed}}} \times \{0,1\}^{l_{\text{seed}}},\ E_1 : \{0,1\}^{l_{\text{seed}}} \to Perm_n,\ E_2 : \{0,1\}^{l_{\text{seed}}} \to \mathbb{F}_q^n,$$

where $l_{\text{seed}} \in \mathbb{N}$ is the seed size. We present below a state of the art 1 round optimized scheme which uses deterministic commitments and seeds, which is similar to the one presented in [BGKS22]. We denote by $\mathscr{H}_{comm}$ the hash function used for the commitment.

Stern's identification scheme (optimized, 1 round), $\text{IS}^1_{Stern,opt}$

Initialization. Let $(\mathbf{H}, \mathbf{e}, \mathbf{s} = \mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $sk = \mathbf{e}$, $pk = (\mathbf{H}, \mathbf{s})$. The prover obtains (pk, sk) and the verifier obtains pk.

Interaction.

- The prover picks a random seed $\in \{0,1\}^{l_{\text{seed}}}$. Let $(\text{seed}_\pi, \text{seed}_\mathbf{y}) = E_0(\text{seed}), \pi = E_1(\text{seed}_\pi)$ and $\mathbf{y} = (\pi)^{-1}(E_2(\text{seed}_\mathbf{y}))$. It then computes $\mathbf{t} = \mathbf{Hy}$, $z_1 = (\pi, \mathbf{t}), z_2 = \pi(\mathbf{y}), z_3 = \pi(\mathbf{y} + \mathbf{e})$, and computes for $i \in \{1,2,3\}$, $x_i = \mathscr{H}_{comm}(z_i)$. He then sends x_1, x_2 and x_3 to the verifier.
- The verifier sends a random challenge $c \xleftarrow{\$} \{1,2,3\}$ to the prover.
- The prover reveals information so that the verifier can recover x_i for each $i \in \{1,2,3\} \setminus c$. More precisely:
  1. if $c = 1$, the prover sends $l_1 = (o_2, o_3)$ with $o_2 = \text{seed}_\mathbf{y}$, $o_3 = \pi(\mathbf{e})$.
  2. if $c = 2$, the prover sends $l_2 = (o_1, o_3)$ with $o_1 = \text{seed}_\pi$, $o_3 = \mathbf{y} + \mathbf{e}$.
  3. if $c = 3$, the prover sends $l_3 = \text{seed}$.

Verification.

1. if $c = 1$, the verifier recovers z_2, z_3 from o_2, o_3 by taking $z_2 = E_2(o_2)$ and $z_3 = z_2 + o_3$. Then for $i \in \{2,3\}$ checks that $\mathscr{H}_{comm}(x_i) = z_i$ and that $wt(o_3) = w$.
2. if $c = 2$, the verifier computes $\pi = E_1(o_1), z_3 = \pi(o_3)$ then $\mathbf{t} = \mathbf{H}o_3 - \mathbf{s}$ and finally $z_1 = (\pi, \mathbf{t})$. Then for $i \in \{1,3\}$ checks that $\mathscr{H}_{comm}(x_i) = z_i$.
3. if $c = 3$, the verifier computes $(\text{seed}_\pi, \text{seed}_y) = E_0(l_3)$, $\pi = E_1(\text{seed}_\pi)$ and $z_2 = E_2(\text{seed}_\mathbf{y})$, followed by $\mathbf{t} = \mathbf{H}(\pi)^{-1}(z_2)$. From there, he constructs $z_1 = (\pi, \mathbf{t})$. Then for $i \in \{1,2\}$ checks that $\mathscr{H}_{comm}(x_i) = z_i$.

Notice that with deterministic commitments, there are here other optimizations regarding what the prover sends for each challenge. He doesn't reveal the full strings $z_{c'}, z_{c''}$ but the minimal information required to recover them.

Replacing the first message with a global commitment Recall that the full scheme consists of R parallel repetition of the above 1 round schemes. In this case, it is possible to replace the first message by a global commitment of all the x_i. This allows to reveal the commitment only of the values not revealed (since they can be recovered from the openings). The scheme becomes the following.

Stern's identification scheme (optimized, R rounds), $\text{IS}_{Stern,opt}$

Initialization. Let $(\mathbf{H}, \mathbf{e}, \mathbf{s} = \mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $sk = \mathbf{e}$, $pk = (\mathbf{H}, \mathbf{s})$. The prover obtains (pk, sk) and the verifier obtains pk.

Interaction.

– For j from 1 to R: the prover picks a random $\text{seed}^j \in \{0,1\}^{l_{\text{seed}}}$ and lets $(\text{seed}_\pi^j, \text{seed}_{\mathbf{y}}^j) = E_0(\text{seed}^j), \pi^j = E_1(\text{seed}_\pi^j)$ and $\mathbf{y}^j = (\pi^j)^{-1}(E_2(\text{seed}_{\mathbf{y}}^j))$. It then computes $\mathbf{t}^j = \mathbf{H}\mathbf{y}^j$, $z_1^j = (\pi^j, \mathbf{t}^i), z_2 = \pi^i(\mathbf{y}^i), z_3^i = \pi^i(\mathbf{y}^i + \mathbf{e})$, and computes for $i \in \{1,2,3\}$, $x_i^j = \mathscr{H}_{comm}(z_i^j)$.

The prover sends $\widetilde{x} = \mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R)$ to the verifier.

– For j from 1 to R: the verifier sends a random challenge $c^j \xleftarrow{\$} \{1,2,3\}$ to the prover.

– For j from 1 to R: the prover reveals information so that the verifier can recover x_i^j for each $i \in \{1,2,3\} \setminus c$. More precisely:
  1. if $c^j = 1$, the prover sends $l_1^j = (o_2^j, o_3^j)$ with $o_2^j = \text{seed}_{\mathbf{y}}$, $o_3^j = \pi^j(\mathbf{e})$ and x_1^j.
  2. if $c^j = 2$, the prover sends $l_2^j = (o_1^j, o_3^j)$ with $o_1^j = \text{seed}_\pi$, $o_3^j = \mathbf{y}^j + \mathbf{e}$ and x_2^j
  3. if $c^j = 3$, the prover sends $l_3^j = \text{seed}$ and x_3^j.

Verification. For j from 1 to R:

1. if $c^j = 1$, recovers z_2^j, z_3^j from o_2^j, o_3^j by taking $z_2^j = E_2(o_2^j)$ and $z_3^j = z_2^j + o_3^j$. Then for $i \in \{2,3\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$ and that $wt(o_3^j) = w$.
2. if $c^j = 2$, computes $\pi^j = E_1(o_1^j), z_3^j = \pi^j(o_3^j)$ then $\mathbf{t}^j = \mathbf{H}o_3^j - \mathbf{s}$ and finally $z_1^j = (\pi^j, \mathbf{t}^j)$. Then for $i \in \{1,3\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$.
3. if $c^j = 3$, computes $(\text{seed}_\pi^j, \text{seed}_y^j) = E_0(l_3^j)$, $\pi^j = E_1(\text{seed}_\pi^j)$ and $z_2^j = E_2(\text{seed}_{\mathbf{y}}^j)$, followed by $\mathbf{t}^j = \mathbf{H}(\pi^j)^{-1}(\mathbf{y}^j)$. From there, he constructs $z_1^j = (\pi^j, \mathbf{t}^j)$. Then for $i \in \{1,2\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$.

The verifier also checks that $\mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R)$ is equal to the value $\widetilde{x}$ he received.

Here, we described an identification scheme. The resulting signature schemes is constructed using the Fiat-Shamir transform presented in Section 3.2. This means each signature is of the form:

$$\sigma(m) = (\widetilde{x}, x_{c_1}^1, \ldots, x_{c_R}^R, l_{c_1}^1, \ldots, l_{c_R}^R),$$

where $c = (c_1, \ldots, c_R) = \mathscr{H}_{comm}(\widetilde{x}||M)$.

5 Attack on optimized Stern's signature schemes

Here, we show that the combination of deterministic commitment and of seeds with $l_{\text{seed}} = \lambda$ gives only $\lambda/2$ bits of security instead of the claimed λ bits of security from [BGKS22].

Attack on the optimized signature scheme

– The adversary performs $q_s = \lceil \frac{1}{R} 2^{l_{\text{seed}}/2} \rceil$ signature queries, with not necessarily distinct messages, to obtain the signatures $\sigma_{m_1}, \ldots, \sigma_{m_{q_s}}$ from the signer, where each $\sigma(m_k)$ is written as

$$\sigma_{m_k} = \big(\widetilde{x}_k, x_{c_1,k}^1, \ldots, x_{c_R,k}^R, l_{c_1,k}^1, \ldots, l_{c_R,k}^R)\big).$$

– The adversary tries to find a pair $((i,k),(i',k'))$ st.

$$(i,k) \neq (i',k') \wedge x_{2,k}^i = x_{2,k'}^{i'} \wedge c_k^i = 2 \wedge c_{k'}^{i'} \neq 2. \tag{3}$$

If none are found, return to the first step.

– From such a pair of couples $((i,k),(i',k'))$, the adversary recovers the secret key as follows:
  - Since $c_k^i = 2$, we have that from l_2^i, the adversary can recover $\pi_k^i = E_1(z_{1,k}^i)$ and $\pi_k^i(\mathbf{y}_k^i + \mathbf{e}_k) = \pi(z_{3,k}^i)$.
  - Since $c_{k'}^{i'} \neq 2$, then the adversary obtains either $\text{seed}_{\mathbf{y},k'}^{i'}$ or $\text{seed}_{k'}^{i'}$ from which $\text{seed}_{\mathbf{y},k'}^{i'}$ can be calculated. Moreover, as $x_{2,k'}^{i'} = x_{2,k}^i$, with overwhelming probability $\text{seed}_{\mathbf{y},k}^i = \text{seed}_{\mathbf{y},k'}^{i'}$, the adversary learns $\pi_k^i(\mathbf{y}_k^i) = E_2(\text{seed}_{\mathbf{y},k'}^{i'})$.

– From $\pi_k^i,\ \pi_k^i(\mathbf{y}_k^i),\ \pi_k^i(\mathbf{y}_k^i+\mathbf{e})$, the adversary thus recovers the secret key by computing

$$\mathbf{e} = (\pi_k^i)^{-1}\left(\pi_k^i(\mathbf{y}_k^i + \mathbf{e}) - \pi_k^i(\mathbf{y}_k^i)\right) = (\pi_k^i)^{-1}\left(\pi_k^i(\mathbf{e})\right),$$

where $(\pi_k^i)^{-1}$ is the inverse of (π_k^i).

Theorem 3. *The attack presented above finds the secret key,* $\mathbf{e}$*, in time* $O(2^{l_{\text{seed}}/2})$ *doing* $O(\frac{1}{r}2^{l_{\text{seed}}/2})$ *signature queries. The scheme thus preserves at most* $\frac{l_{\text{seed}}}{2}$ *bits of security.*

Proof. We prove that each iteration of the attack described above finds the secret key with a constant probability. Notice that for $q_S = \lceil \frac{1}{R}2^{l_{\text{seed}}/2}\rceil$, the signing oracle generates $Rq_S = R\lceil \frac{1}{R}2^{l_{\text{seed}}/2}\rceil \geq 2^{l_{\text{seed}}/2}$ random seeds $\text{seed}_{\mathbf{y},k}^i \in \{0,1\}^{l_{seed}}$, $i \in [R]$, $k \in [q_S]$. There is a collision in the seeds, i.e. $\exists (i,k),\ (i',k')$ such that $(i,k) \neq (i',k') \ \wedge\ \text{seed}_{\mathbf{y},k}^i = \text{seed}_{\mathbf{y},k'}^{i'}$ wp. $\Omega(\frac{q_S^2}{2^{l_{\text{seed}}}})$ and with constant probability, this happens wp. $c_k^i = 2$ and $c_{k'}^{i'} \neq 2$. So we proved that the probability that the adversary finds a pair of couples $((i,k),(i',k'))$ satisfying Equation 3, which proves our theorem.

As a concrete example, assume we consider the above signature scheme with $\lambda = 128$ which is practice the most common choice. This means that there is an adversary performing $\frac{1}{R}2^{64} \leq 2^{58}$ queries and running in time 2^{64} that will break the scheme with constant probability (here something around $\frac{1}{10}$), while the underlying syndrome decoding problem requires time 2^{128} to break. The attack clearly invalidates the claim of 128 bits of security. Notice also that it is quite standard to allow up to 2^{64} calls to the signing oracle, this is for example what is requried by the NIST for the standardisation of post-quantum signature schemes so our attack fits these guidelines.

Fortunately, this attack can be mitigated with the methods we present in the next section with a close to zero loss in the efficiency, so optimized Stern's schemes can still be used securly with these mild changes.

6 The *salt* + index construction

There are 2 easy ways of mitigating the above attack: by taking $l_{\text{seed}} = 2\lambda$ or by using randomized commitments but both these solutions are very costly in terms of signature size. Here, we present another solution that will very mildly increase the signature size while preserving security: the *salt* + index construction. The idea is very simple: when running the identification scheme (the full R round scheme), we pick a random $salt \in \{0,1\}^{2\lambda}$ which we use as a randomization for all the commitments and the pseudo-random generators. Since the functions are used several times, we add an index which will be like an internal counter. Each time such a function is used, we increment the index and add it to the input, to ensure that each function used is independent from one another.

Stern's identification scheme (optimized, R rounds), $\text{IS}_{Stern,opt}$

Initialization. Let $(\mathbf{H},\mathbf{e},\mathbf{s}=\mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $sk = \mathbf{e}$, $pk = (\mathbf{H},\mathbf{s})$. The prover obtains (pk,sk) and the verifier obtains pk.

Interaction. The prover picks a random $salt \in \{0,1\}^{2\lambda}$ and initializes $index = 0$. Each time index is called, increment index.

- For j from 1 to R: the prover picks a random $\text{seed}^j \in \{0,1\}^{l_{\text{seed}}}$ and lets $(\text{seed}_\pi^j, \text{seed}_\mathbf{y}^j) = E_0(\text{seed}^j, salt, index), \pi^j = E_1(\text{seed}_\pi^j, salt, index)$ and $\mathbf{y}^j = (\pi^j)^{-1}(E_2(\text{seed}_\mathbf{y}^j), salt, index)$. He then computes $\mathbf{t}^j = \mathbf{H}\mathbf{y}^j$, $z_1^j = (\pi^j, \mathbf{t}^i), z_2 = \pi^i(\mathbf{y}^i), z_3^i = \pi^i(\mathbf{y}^i + \mathbf{e})$, and computes for $i \in \{1,2,3\}$, $x_i^j = \mathscr{H}_{comm}(z_i^j, salt, index)$.

The prover sends $\widetilde{x} = \mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R, salt, index)$ to the verifier as well as *salt*.

- For j from 1 to R: the verifier sends a random challenge $c^j \xleftarrow{\$} \{1,2,3\}$ to the prover.

- For j from 1 to R: the prover reveals information so that the verifier can recover x_i^j for each $i \in \{1,2,3\} \setminus c$. More precisely:
  1. if $c^j = 1$, the prover sends $l_1^j = (o_2^j, o_3^j)$ with $o_2^j = \text{seed}_\mathbf{y}$, $o_3^j = \pi^j(\mathbf{e})$ and x_1^j.
  2. if $c^j = 2$, the prover sends $l_2^j = (o_1^j, o_3^j)$ with $o_1^j = \text{seed}_\pi$, $o_3^j = \mathbf{y}^j + \mathbf{e}$ and x_2^j
  3. if $c^j = 3$, the prover sends $l_3^j = \text{seed}$ and x_3^j.

Verification. For j from 1 to R:

1. if $c^j = 1$, recovers z_2^j, z_3^j from o_2^j, o_3^j by taking $z_2^j = E_2(o_2^j, salt, index)$ and $z_3^j = z_2^j + o_3^j$. Then for $i \in \{2,3\}$ checks that $\mathscr{H}_{comm}(x_i^j, salt, index) = z_i^j$ and that $wt(o_3^j) = w$.
2. if $c^j = 2$, computes $\pi^j = E_1(o_1^j, salt, index), z_3^j = \pi^j(o_3^j)$ then $\mathbf{t}^j = \mathbf{H}o_3^j - \mathbf{s}$ and finally $z_1^j = (\pi^j, \mathbf{t}^j)$. Then for $i \in \{1,3\}$ checks that $\mathscr{H}_{comm}(x_i^j, salt, index) = z_i^j$.
3. if $c^j = 3$, computes $(\text{seed}_\pi^j, \text{seed}_y^j) = E_0(l_3^j, salt, index)$, $\pi^j = E_1(\text{seed}_\pi^j, salt, index)$ and $z_2^j = E_2(\text{seed}_\mathbf{y}^j, salt, index)$, followed by $\mathbf{t}^j = \mathbf{H}(\pi^j)^{-1}(\mathbf{y}^j)$. From there, he constructs $z_1^j = (\pi^j, \mathbf{t}^j)$. Then for $i \in \{1,2\}$ checks that $\mathscr{H}_{comm}(x_i^j, salt, index) = z_i^j$.

The verifier also checks that $\mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R, salt, index)$ is equal to the value $\widetilde{x}$ he received. In each function, is known what index is used by the prover (it doesn't depend on some internal randomness) so the verifier uses for each function the same index as the prover when performing the above.

First, notice that this mitigates the above attack. Indeed, the attack used the fact that collision in the seeds between different signatures led to collisions in the commitments which could be detected and used. Here, with the addition of the *salt*, this won't happen anymore (and the index ensures that collisions within the same signature cannot be used). Moreover, notice that in terms of communication, a single string *salt* of size 2λ is added so the increase in the signature size is 2λ.

There could a priori be other attacks on this scheme. We show that this is not the case, by showing that Stern's scheme with the *salt* + index construction is multi-HVZK. Then, one can use the results of [GHHM21] to prove the EUF-CMA security of the resulting signature scheme.

6.1 Zero-knowledge for IS_{Stern} without pseudo-random generators

We first prove a bound on the multi-HVZK advantage when we don't have seeds, but we still have deterministic commitments. We call this scheme $\text{IS}_{DC,NoS,salt}$, which is described as follows:

$\mathrm{IS}_{DC,NoS,salt}$ scheme

Initialization. Let $(\mathbf{H}, \mathbf{e}, \mathbf{s} = \mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $sk = \mathbf{e}$, $pk = (\mathbf{H}, \mathbf{s})$. The prover obtains (pk, sk) and the verifier obtains pk.

Interaction.

- For j from 1 to R: The prover picks a random permutation π^j acting on $\mathbb{F}_q^n$ and a random column vector $\mathbf{y}^j \xleftarrow{\$} \mathbb{F}_q^n$, and then computes $\mathbf{t}^j = \mathbf{H}\mathbf{y}^j$. The prover computes $z_1^j = (\pi^j, \mathbf{t}^j)$, $z_2^j = (\pi^j(\mathbf{y}^j))$, $z_3 = (\pi^j(\mathbf{y}^j + \mathbf{e}))$, and computes for $i \in \{1,2,3\}$, $x_i^j = \mathscr{H}_{comm}(z_i^j || salt, index)$.

The prover sends $\widetilde{x} = \mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R, salt, index)$ to the verifier.

- For j from 1 to R: the verifier sends a random challenge $c^j \xleftarrow{\$} \{1,2,3\}$ to the prover.
- For j from 1 to R: the prover reveals information so that the verifier can recover x_i^j for each $i \in \{1,2,3\} \setminus c$. More precisely:
  1. if $c^j = 1$, the prover sends $l_1^j = (o_2^j, o_3^j)$ with $o_2^j = \pi^j(\mathbf{y}^j)$, $o_3^j = \pi^j(\mathbf{e})$ and x_1^j.
  2. if $c^j = 2$, the prover sends $l_2^j = (o_1^j, o_3^j)$ with $o_1^j = \pi^j$, $o_3^j = \mathbf{y}^j + \mathbf{e}$ and x_2^j
  3. if $c^j = 3$, the prover sends $l_3^j = (\pi^j, \mathbf{y}^j)$ and x_3^j.

Verification. For j from 1 to R:

1. if $c^j = 1$, he recovers z_2^j, z_3^j from l_1^j. Then for $i \in \{2,3\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$ and that $wt(o_3^j) = w$.
2. if $c^j = 2$, he recover z_1^j, z_3^j from l_2^j. Then for $i \in \{1,3\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$.
3. if $c^j = 3$, he recover z_1^j, z_2^j from l_3^j. Then for $i \in \{1,2\}$ checks that $\mathscr{H}_{comm}(x_i^j) = z_i^j$.

The verifier also checks that $\mathscr{H}_{comm}(x_1^1, x_2^1, x_3^1, \ldots, x_1^R, x_2^R, x_3^R, salt, index)$ is equal to the value $\widetilde{x}$ he received.

We prove a tight reduction from the advantage of breaking the t-HVZK condition and the advantage of solving the underlying syndrome decoding problem.

Theorem 4. *For any adversary $\mathcal{A}$, there exists a adversary $\mathcal{B}$ st.*

$$Adv^{\mathrm{t-HVZK}}_{\mathrm{IS}_{DC,NoS,\mathrm{salt}}}(\mathcal{A}) \leq Adv^{\mathrm{SD}}(\mathcal{B}) + O\left(\frac{t^2}{2^{l_{\mathrm{salt}}}}\right).$$

where $Adv^{\mathrm{SD}}(\mathcal{B})$ gives the probability that the adversary breaks the underlying syndrome decoding problem, and the running time of $\mathcal{B}$ is the same as the running time of $\mathcal{A}$.

Before presenting a proof of this theorem, let us perform a small discussion on how to interpret this theorem. In the previous section, we started from a signature scheme st. the underlying syndrome decoding problem requires 2^λ time to break, and we presented an adversary $\mathcal{A}$ running in time $O(2^{\lambda/2})$ that breaks the signature scheme arising from $\mathrm{IS}_{Stern,opt}$ with $t = O(2^{\lambda/2})$ signature queries. One can also interpret our attack as an adversary $\mathcal{A}$ st. running in time $O(2^{\lambda/2})$ st. $Adv^{\mathrm{t-HVZK}}_{\mathrm{IS}_{Stern,opt}}(\mathcal{A})$ is close to 1 with $t = 2^{\lambda/2}$.

Theorem 4 that we will shortly prove, shows that such an attack cannot be possible when in the case where there is salt. Indeed, assume we also had an adversary $\mathcal{A}$ running in time 2^λ st. $Adv^{\mathrm{t-HVZK}}_{\mathrm{IS}_{Stern,opt}}(\mathcal{A})$ is close to 1. Then the above theorem immediately says that there is an adversary $\mathcal{B}$ running in time $2^{\lambda/2}$ that breaks the underlying syndrome decoding problem but we chose the parameters st. such an adversary would require time at least 2^λ hence the contradiction.

We now present the proof of this theorem.

Proof. We will do a game based proof. The first game we consider is

GAME:0 Real transcript

$(pk, sk) \leftarrow Keygen.$
$\forall i \in \{1, \ldots, t\}$:

- Pick a random $salt^i \leftarrow \{0,1\}^{l_{\text{salt}}}$.
- $\forall j \in \{1, \ldots, R\}$:
  - Pick a random permutation π^{ij} and a random string $y^{ij} \in \mathbb{F}_q^n$. Let $\mathbf{t}^{ij} = \mathbf{H}\mathbf{y}^{ij}$. Let $z_1^{ij} = (\pi^{ij}, \mathbf{t}^{ij})$, $z_2^{ij} = (\pi^{ij}(\mathbf{y}^{ij}))$, $z_3^{ij} = (\pi^{ij}(\mathbf{y}^{ij} + \mathbf{e}))$, and compute for $k \in \{1,2,3\}$, $x_k^{ij} = \mathscr{H}_{comm}(z_k^{ij} || salt^i, index)$. Let also

$$\widetilde{x}^i = \mathscr{H}_{comm}(x_1^{i1}, x_2^{i1}, x_3^{i1}, \ldots, x_1^{iR}, x_2^{iR}, x_3^{iR}, salt^i, index).$$

    Pick a random $c^{ij} \leftarrow \{1,2,3\}$
    1. If $c^{ij} = 1$, set $l_1^{ij} = (o_2^{ij}, o_3^{ij})$ with $o_2^{ij} = \pi^{ij}(\mathbf{y}^{ij})$, $o_3^{ij} = \pi^{ij}(\mathbf{e})$.
    2. If $c^{ij} = 2$, set $l_2^{ij} = (o_1^{ij}, o_3^{ij})$ with $o_1^{ij} = \pi^{ij}$, $o_3^{ij} = \mathbf{y}^{ij} + \mathbf{e}$.
    3. If $c^{ij} = 3$, set $l_3^{ij} = (o_1^{ij}, o_2^{ij})$ with $o_1^{ij} = \pi^{ij}$, $o_2^{ij} = \mathbf{y}^{ij}$.
  - $\tau^i = \left(\widetilde{x}^i, salt^i, x_{c^{i1}}^{i1}, \ldots, x_{c^{iR}}^{iR}, l_{c^{i1}}^{i1}, \ldots, l_{c^{iR}}^{iR}\right)$.

$\mathcal{A}$ wins the game iff. $\mathcal{A}(\tau^1, \ldots, \tau^t, pk) = 1$.

In this game, we construct t transcripts constructed according to $trans_{\text{IS}_{Stern,DC,NoS}}(pk, sk)$. The game is won if the adversary outputs 1 given these real transcripts as input.

We now consider another game where we construct the same transcripts but in a different way. For each choice of c^{ij}, it is possible to construct the responses l^{ij} independently of the secret key and to make only the strings $x_{c^{ij}}^{ij}$ dependent of the secret key. Let S_w be set the of words of weight w.

GAME:1 Real transcript with openings depending only on pk

$(pk, sk) \leftarrow Keygen.$
$\forall i \in \{1, \ldots, t\}$:

- Pick a random $salt^i \leftarrow \{0,1\}^{l_{\text{salt}}}$.
- $\forall j \in \{1, \ldots, R\}$:
  - Pick a random challenge $c^{ij} \leftarrow \{1,2,3\}$.
    1. If $c^{ij} = 1$, set $l_1^{ij} = (o_2^{ij}, o_3^{ij})$ with $o_2^{ij} \leftarrow \mathbb{F}_q^n$ and $o_3^{ij} \leftarrow S_w$. Pick a random π^{ij} st. $\pi^{ij}(o_3^{ij}) = \mathbf{e}$, and let $x_1^{ij} = \mathscr{H}_{comm}\left(\pi^{ij}, \mathbf{t}^{ij} = \mathbf{H}\pi^{ij}(o_2^{ij}), \text{seed}, index\right)$.
    2. If $c^{ij} = 2$, set $l_2^{ij} = (o_1^{ij}, o_3^{ij})$ with $o_1^{ij} \leftarrow Perm_n$, $o_3^{ij} \leftarrow \mathbb{F}_q^n$. Pick $x_2^{ij} = \mathscr{H}_{comm}\left(\pi^{ij}(\mathbf{y}^{ij}), salt^i, index\right)$.
    3. If $c^{ij} = 3$, set $l_3^{ij} = (o_1^{ij}, o_2^{ij})$ with $o_1^{ij} \leftarrow Perm_n$, $o_2^{ij} \leftarrow \mathbb{F}_q^n$. Pick $x_3^{ij} = \mathscr{H}_{comm}\left(\pi^{ij}(\mathbf{y}^{ij} + \mathbf{e}), salt^i, index\right)$.

    Let also

$$\widetilde{x}^i = \mathscr{H}_{comm}(x_1^{i1}, x_2^{i1}, x_3^{i1}, \ldots, x_1^{iR}, x_2^{iR}, x_3^{iR}, salt^i, index).$$

  - $\tau^i = \left(\widetilde{x}^i, salt^i, x_{c^{i1}}^{i1}, \ldots, x_{c^{iR}}^{iR}, l_{c^{i1}}^{i1}, \ldots, l_{c^{iR}}^{iR}\right)$.

$\mathcal{A}$ wins the game iff. $\mathcal{A}(\tau^1, \ldots, \tau^t, pk) = 1$.

Lemma 2.

$$\Pr[\mathcal{A} \textit{ wins } \text{GAME:0}] = \Pr[\mathcal{A} \textit{ wins } \text{GAME:1}].$$

Proof. This is true since the distribution of each τ^i is the same for both games. This can be seen using the same arguments as in Proposition 3, where we proved that the ideal simulator has the same distribution of openings as the real transcript.

We now change GAME:1 into GAME:2 where we replace the commitments x^{ij} with uniformly random strings. We put the difference between the games in red.

GAME:2 Simulated transcript

$(pk, sk) \leftarrow Keygen.$
$\forall i \in \{1, \ldots, t\}$:

- Pick a random $salt^i \leftarrow \{0,1\}^{l_{\text{salt}}}$.
- $\forall j \in \{1, \ldots, R\}$:
  - Pick a random challenge $c^{ij} \leftarrow \{1, 2, 3\}$.
    1. If $c^{ij} = 1$, set $l_1^{ij} = (o_2^{ij}, o_3^{ij})$ with $o_2^{ij} \leftarrow \mathbb{F}_q^n$ and $o_3^{ij} \leftarrow S_w$.
    2. If $c^{ij} = 2$, set $l_2^{ij} = (o_1^{ij}, o_3^{ij})$ with $o_1^{ij} \leftarrow Perm_n$, $o_3^{ij} \leftarrow \mathbb{F}_q^n$.
    3. If $c^{ij} = 3$, set $l_3^{ij} = (o_1^{ij}, o_2^{ij})$ with $o_1^{ij} \leftarrow Perm_n$, $o_2^{ij} \leftarrow \mathbb{F}_q^n$.
  - Let $x_{c^{ij}}^{ij} \leftarrow \{0,1\}^{l_{\text{comm}}}$. Let also

$$\widetilde{x}^i = \mathscr{H}_{comm}(x_1^{i1}, x_2^{i1}, x_3^{i1}, \ldots, x_1^{iR}, x_2^{iR}, x_3^{iR}, salt^i, index).$$

  - $\tau^i = \left(\widetilde{x}^i, salt^i, x_{c^{i1}}^{i1}, \ldots, x_{c^{iR}}^{iR}, l_{c^{i1}}^{i1}, \ldots, l_{c^{iR}}^{iR}\right)$.

$\mathcal{A}$ wins the game iff. $\mathcal{A}(\tau^1, \ldots, \tau^t, pk) = 1$.

Lemma 3. *We have*

$$|\Pr[\mathcal{A} \text{ wins GAME:2}] - \Pr[\mathcal{A} \text{ wins GAME:1}]| \leq Adv^{\text{SD}}(\mathcal{B}) + \Theta(\frac{t^2}{2^{l_{\text{salt}}}}).$$

where $\mathcal{B}$ is an adversary st. $|\mathcal{B}| \approx |\mathcal{A}|$ and $Adv^{\text{SD}}(\mathcal{B})$ is the probability that $\mathcal{B}$ breaks the underlying syndrome decoding problem.

Proof. Consider an adversary $\mathcal{A}$ for GAME:2. This adversary can perform queries $\mathscr{H}_{comm}(\cdot)$. The only difference between GAME:1 and GAME:2 is in the strings $x_{c^{ij}}^{ij}$. In GAME:1, if all the salts seed_{salt}^i are pairwise distinct, $x_{c^{ij}}^{ij} = \mathscr{H}_{comm}(z_{c^{ij}}^{ij}, salt^i, index)$ is queried only once to while $\mathscr{H}_{comm}(*, salt^i, index)$ is not queried in GAME:2. The probability that all the salts are different is $\Theta(\frac{t^2}{2^{l_{\text{salt}}}})$.

$\mathcal{A}$ can query this function and the only way the output distribution of $\mathcal{A}$ is different in GAME:2 and in GAME:1 is $\mathcal{A}$ queries exactly $z_{c^{ij}}^{ij}$. This means

$$|\Pr[\mathcal{A} \text{ wins GAME:2}] - \Pr[\mathcal{A} \text{ wins GAME:1}]| \leq \Pr\left[\exists i, j \text{ st. } \mathcal{A} \text{ queries } x_{c^{ij}}^{ij}\right] + \Theta(\frac{t^2}{2^{l_{\text{salt}}}}).$$

Moreover, from the openings o^{ij}, the adversary can recover the 2 other strings $z_{c'}^{ij}, z_{c''}^{ij}$ for $c' \neq c''$ and $c', c'' \neq c^{ij}$. From the 3 strings z_1^{ij}, z_2^{ij}, z_3^{ij} one can recover in polynomial time the secret key sk using some extraction algorithm (we described how to do this in Section 5 for example). We can therefore construct from $\mathcal{A}$ an algorithm $\mathcal{B}$ that solves the syndrome decoding problem.

Algorithm $\mathcal{B}$: run $\mathcal{A}$. Each time a query $\mathscr{H}_{comm}(\cdot)$ is done (and the values seed_{salt}^i and c^{ij} are publicly known from the transcript), consider the responses z^{ij} from the transcript. From them, one can recover the values $x_{c'}^{ij}, x_{c''}^{ij}$ for $c' \neq c''$ and $c', c'' \neq c^{ij}$, and check, using the extractor Ext and by arranging the inputs depending on the value of c^{ij}, whether from $Ext(x_{c'}^{ij}, x_{c''}^{ij})$ one can recover the secret key sk. Since Ext is efficient, the running time of $\mathcal{B}$ is essentially the running time of $\mathcal{A}$. This means that

$$\Pr\left[\exists i, j \text{ st. } \mathcal{A} \text{ queries } x_{c^{ij}}^{ij}\right] \leq Adv^{\text{SD}}(\mathcal{B})$$

with $|\mathcal{B}| \approx |\mathcal{A}|$.

Putting the 2 inequalities together, we have

$$|\Pr[\mathcal{A} \text{ wins GAME:2}] - \Pr[\mathcal{A} \text{ wins GAME:1}]| \leq Adv^{\text{SD}}(|\mathcal{A}|) + \Theta(\frac{t^2}{2^{l_{\text{salt}}}}).$$

Finally, notice that the τ^{ij} constructed in GAME:2 follows exactly the distribution of simulated transcripts. This means

$$Adv^{\mathrm{t-HVZK}}_{\mathrm{IS}_{DC,NoS,salt}}(\mathcal{A}) = \left|\Pr[\mathcal{A} \text{ wins GAME:2}] - \Pr[\mathcal{A} \text{ wins GAME:0}]\right|.$$

Putting everything together, we obtain

$$Adv^{\mathrm{t-HVZK}}_{\mathrm{IS}_{DC,NoS,salt}}(\mathcal{A}) \leq Adv^{\mathrm{SD}}(|A|) + O\left(\frac{t^2}{2^{l_{\mathrm{salt}}}}\right).$$

6.2 Zero-knowledge for IS_{Stern} using pseudo-random generators

Here, we show that using pseudo-random strings instead of random strings $\pi, \mathbf{y}$ doesn't change the security of the scheme. Recall that we showed in the previous section that if one isn't careful using seeds of size λ and deterministic commitments, there are attacks on the identification scheme. We show here that our *salt + index* construction. In the ROM, we have the following:

Proposition 4. *Consider a random function $E : \{0,1\}^{l_{\mathrm{seed}}} \to I$ modeled as a random function in the random oracle model, and consider the following distinguishing game:*

GAME:$Indist(E)$
$b \leftarrow \{0,1\}$.
If $b = 0$, $\mathrm{seed} \leftarrow \{0,1\}^{l_{\mathrm{seed}}}, y = E(\mathrm{seed})$.
If $b = 1$, $y \leftarrow I$.
$b' \leftarrow A(y)$.
The game is won iff. $b = b'$.

Consider an adversary $\mathcal{A}$ that performs q queries to E. Then

$$\Pr[\mathcal{A} \textit{ wins } \text{GAME:}Indist(E)] \leq \frac{1}{2} + O(\frac{q}{2^{l_{\mathrm{seed}}}}).$$

Moreover, the above proposition is true even if $\mathcal{A}$ has some auxiliary information, *as long as it is independent of the function E*[5].

Proposition 5. *Consider the 2 following games*

GAME:*0 Real transcript*
$(pk, sk) \leftarrow Keygen(1^\lambda)$.
$\forall i \in \{1, \ldots, t\}, \tau^1, \ldots, \tau^t \leftarrow trans_{\mathrm{IS}_{DC,NoS,\mathrm{salt}}}(pk, sk)$.
$\mathcal{A}$ *wins the game iff.* $\mathcal{A}(\tau^1, \ldots, \tau^t) = 1$.

Notice that this game is exactly GAME:0 *from Section 6.1. Consider the second game*

GAME:*0' Real transcript with seeds*
$(pk, sk) \leftarrow Keygen$.
$\forall i \in \{1, \ldots, t\}, \tau^1, \ldots, \tau^t \leftarrow trans_{\mathrm{IS}_{Stern,opt}}(sk)$.
$\mathcal{A}$ *wins the game iff.* $\mathcal{A}(\tau^1, \ldots, \tau^t) = 1$.

where $\mathrm{IS}_{Stern,opt}$ is the full optimized scheme with seeds and salt described in Section 6. For any adversary performing q queries to E, we have

$$\Pr[\mathcal{A} \text{ wins GAME:0}] - \Pr[\mathcal{A} \text{ wins GAME:0'}] \leq O(\frac{q}{2^{l_{\mathrm{seed}}}}) + O(\frac{t^2}{2^{l_{\mathrm{salt}}}}).$$

[5] This last part is what motivates the use of the salt + index construction for seeds.

Proof. The only difference between the 2 games is the way the strings π^{ij} and $\pi^{ij}(\mathbf{y}^{ij})$ are generated. Notice that for each i, j we use a different function E as long as the salts are all different (more precisely, we use $E(*||salt^i, index)$ so we actually use the same function but with a different suffix). Take the permutations π^{ij}, the adversary can distinguish a pseudo-random π^{ij} from a random π^{ij} wp. $O(\frac{q^{ij}}{2^{l_{\text{seed}}}})$ where q^{ij} is the number of queries done to the function E used to construct π^{ij}. Similarly, let r^{ij} be the number of queries done to the function E used to construct $\pi^{ij}(\mathbf{y}^{ij})$. Because each of these functions is different, we have $\sum_{ij} q_{ij} + r_{ij} \leq q$ where recall that q is the total number of queries done by $\mathcal{A}$. Moreover, we can use Proposition 4 and therefore have

$$\Pr[\mathcal{A} \text{ wins GAME:0}] - \Pr[\mathcal{A} \text{ wins GAME:0}'] \leq \sum_{ij} O(\frac{q_{ij} + r_{ij}}{2^{l_{\text{seed}}}}) + O(\frac{t^2}{2^{l_{\text{salt}}}}) \leq O(\frac{q}{2^{l_{\text{seed}}}}) + O(\frac{t^2}{2^{l_{\text{salt}}}}).$$

7 Reduction of Signature Size using Hash Trees

A common technique for further reducing the signature size is to utilize a technique based upon seeded hash trees – small seed expansions associated to Merkel trees. This technique is used, for example, in [Beu20] and promises to have a huge advantage in efficiency. We show here how this technique can reduce the length of the signatures arising from Stern's signature scheme.

Stern's identification protocol, optimized fusion of** 4 **rounds

Initialization. **Keygenerator** samples $(\mathbf{H}, \mathbf{e}, \mathbf{s} = \mathbf{He}) \xleftarrow{\$} \mathscr{D}_{n,k,w}$ 1, $\mathbf{sk} = \mathbf{e}$, $\mathbf{pk} = (\mathbf{H}, \mathbf{s})$.

Interaction.

Prover: picks $\text{seed}_1^{1234}, \text{seed}_2^{1234} \in \{0,1\}^{l_{\text{seed}}}$. For $j \in \{1, 2\}$:

$$\begin{aligned}(\text{seed}_j^{12}, \text{seed}_j^{34}) &= J(\text{seed}_j^{1234}||salt||index)\\ (\text{seed}_j^{1}, \text{seed}_j^{2}) &= J(\text{seed}_j^{1234}||salt||index)\\ (\text{seed}_j^{1}, \text{seed}_j^{2}) &= J(\text{seed}_j^{1234}||salt||index)\end{aligned}$$

For $i \in \{1, 2, 3, 4\}$, we construct $\pi^i = E(\text{seed}_1^i||salt||index)$ and $\pi^i(\mathbf{y}^i) = E(\text{seed}_1^i||salt||index)$. Let $\mathbf{t}^i = \mathbf{H}\mathbf{y}^i$, $\mathbf{z}_1^i = (\pi^i, \mathbf{t}^i)$. He then construct the commitments $\mathbf{x}_j^i = \mathscr{H}_{comm}(z_j^i||salt||index)$, and constructs

$$\begin{aligned}\mathbf{x}_j^{12} &= L((\mathbf{x}_j^1, \mathbf{x}_j^2)||salt||index))\\ \mathbf{x}_j^{34} &= L((\mathbf{x}_j^3, \mathbf{x}_j^4)||salt||index))\\ \mathbf{x}_j^{1234} &= L((\mathbf{x}_j^{12}, \mathbf{x}_j^{34})||salt||index))\end{aligned}$$

Prover calculates random *challenges* as $(c^1, c^2, c^3, c^4) = \mathscr{H}_{FSH}(\mathbf{x}^{1234}, m)$.

The prover and the verifier run a 4 round optimized scheme with the following optimizations:

- For $j \in \{1, 2\}$ If the prover needs to reveal seed_j^1 and seed_j^2 (resp. seed_j^3 and seed_j^4) he reveals seed_j^{12} instead (resp. seed_j^{34}) and the verifier uses J to recover each seed. If the prover needs to reveal seed_j^{12} and seed_j^{34} he reveals seed_j^{1234} instead and uses J again

– For $j \in \{1,2,3\}$ If the prover needs to reveal x_j^1 and x_j^2 (resp. x_j^3 and X_j^4) he reveals x_j^{12} instead (resp. x_j^{34}) and the verifier uses L to recover each commitment. If the prover needs to reveal x_j^{12} and x_j^{34} he reveals x_j^{1234} instead and uses L again.

Stern's identification scheme with hash trees consists of 55 parallel repetition of the above scheme (which each has 4 rounds), except that the same salt is used for all the repetition. This gives a total of 220 rounds.

Proposition 6. *The signature size of the signature scheme constructed from the above identification scheme is (in bits):*

$$SIZE = l_{\text{salt}} + 55\left(2C_1 + 4\left(\frac{1}{3}s_w + \frac{1}{3}n\log_2(q) + \frac{1}{3}l_{\text{comm}}\right)\right).$$

where

$$C_1 = \frac{128}{81}l_{\text{seed}} + \frac{89}{81}l_{\text{comm}}.$$

and $s_w = \log_2\left(|\{x \in \mathbb{F}_q^n, wt(x) = w\}\right)$.

Proof. Let us compute the communication cost in this setting. Let C_1 be the communication cost corresponding to $j = 1$. This means we count here the costs of the revealed seed_1^i as well as x_1^i. For each of the 4 rounds, we have a probability $\frac{2}{3}$ of revealing the seed and probability $\frac{1}{3}$ of revealing the commitment. Moreover, we use our seed and commitment regrouping. For example, if we reveal the 4 seeds (which happens wp. $(\frac{2}{3})^4$), we only have to reveal seed_1^{1234} which costs l_{seed} bits. Considering each of these case, we obtain:

$$\begin{aligned} C_1 = (\frac{2}{3})^4 s_{\text{seed}} + 4(\frac{2}{3})^3\frac{1}{3}\left(2s_{\text{seed}} + s_{comm}\right) + 4(\frac{2}{3})^2(\frac{1}{3})^2\left(2s_{\text{seed}} + 2s_{comm}\right) \\ + 2(\frac{2}{3})^2(\frac{1}{3})^2\left(s_{\text{seed}} + s_{comm}\right) + 4\frac{2}{3}(\frac{1}{3})^3\left(s_{\text{seed}} + 2s_{comm}\right) + (\frac{1}{3})^4 s_{comm} \end{aligned}$$

which gives

$$C_1 = \frac{128}{81}s_{\text{seed}} + \frac{89}{81}s_{comm}.$$

The same calculation can be done for $j = 2$. For the final case, we reveal the commitment wp. $\frac{1}{3}$. If this doesn't happen, we reveal $\pi(e)$ or $\pi(y+e)$ which requires respectively s_w and $n\log_2(q)$ bits to send. Finally, we have to send also the salt used (the indexes can be recomputed by the verifier). From there, we have that the total size is

$$SIZE = l_{\text{salt}} + 55\left(2C_1 + 4\left(\frac{1}{3}s_w + \frac{1}{3}n\log_2(q) + \frac{1}{3}l_{\text{comm}}\right)\right).$$

7.1 Proposal with Lee's metric

We use our scheme with the salt + index construction and with the above hash trees. Wa also consider Lee's metric, defined as follows. For $\mathbf{e} = (e_1, \ldots, e_n) \in F_q^n$, we define

$$wt_L(\mathbf{e}) = \sum_{i=1}^{n} \min\{e_i, q - e_i\}.$$

The best known algorithms for solving the syndrome decoding problem with this metric were proposed in [CDE21]. Plugging these exponents to our construction, we obtain the following signature sizes.

Non-optimized version					
q	2	3	5	7	13
$wt_H(\cdot)$	253.05 kB	116.54 kB	138.54 kB	126.468 kB	113.227 kB
$wt_L(\cdot)$	253.05 kB	116.54 kB	95.48 kB	90.935 kB	79.2733 kB

Optimized version					
q	2	3	5	7	13
$wt_H(\cdot)$	26.2105 kB	21.8122 kB	27.6249 kB	28.291 kB	29.3769 kB
$wt_L(\cdot)$	26.2105 kB	21.8122 kB	21.4108 kB	22.7065 kB	23.2905 kB

Hash-tree version (4 levels)					
q	2	3	5	7	13
$wt_H(\cdot)$	24.7223 kB	20.304 kB	26.1432 kB	26.8123 kB	27.9032 kB
$wt_L(\cdot)$	24.7223 kB	20.304 kB	19.9007 kB	21.2023 kB	21.789 kB

8 Conclusion

In this work, we studied the security and insecurity of some optimized Stern's signature schemes. The main takeaway from our work is that one has to be careful when considering deterministic commitments and seeds of size λ but that there is a way to make work with minimal deterioration of the efficiency. Similar analysis could be done for other schemes, and we hope to extend this fully to MQDSS. There doesn't seem to be any special difficulty but each scheme has to be studied with care. Moreover, it would be nice to extend this framework to 5 round schemes, which is an optimization that we didn't consider in this work.

References

AGS11. Carlos Aguilar, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop*, pages 648–652, 2011.

BBC+20. Marco Baldi, Massimo Battaglioni, Franco Chiaraluce, Anna-Lena Horlemann-Trautmann, Edoardo Persichetti, Paolo Santini, and Violetta Weger. A new path to code-based signatures via identification schemes with restricted errors. *CoRR*, abs/2008.06403, 2020.

Beu20. Ward Beullens. Sigma protocols for mq, pkp and sis, and fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, pages 183–211, 2020.

BGKS22. Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Nicolas Sendrier. Quasi-cyclic stern proof of knowledge. In *2022 IEEE International Symposium on Information Theory (ISIT)*, page 1459–1464. IEEE Press, 2022.

BMvT78. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

Bog83. K. P. Bogart. *Introductory Combinatorics*. Pitman Publishing Inc., 1983.

CDE21. André Chailloux, Thomas Debris-Alazard, and Simona Etinski. Classical and quantum algorithms for generic syndrome decoding problems and applications to the lee metric. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 44–62, Cham, 2021. Springer International Publishing.

CHR+20. Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. Mqdss specifications, 2020.

CVE11. Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In *SAC*, pages 171–186, 2011.

DST19. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 21–51, Cham, 2019. Springer International Publishing.

FJR21. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. *IACR Cryptol. ePrint Arch.*, page 1576, 2021.

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. *IACR Cryptol. ePrint Arch.*, page 188, 2022.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

GHHM21. Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the qrom. Springer-Verlag, 2021.

GJ90. Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., USA, 1990.

GRSZ14. Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zemor. Ranksign: An efficient signature algorithm based on the rank metric. In Michele Mosca, editor, *Post-Quantum Cryptography*, pages 88–107, 2014.

HTW21. Anna-Lena Horlemann-Trautmann and Violetta Weger. Information set decoding in the lee metric with applications to cryptography. *Advances in Mathematics of Communications*, 15(4):677–699, 2021.

KT17. Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography*, pages 69–89, Cham, 2017. Springer International Publishing.

Lei18. Dominic Leichtle. Post-quantum signatures from identification schemes, masters thesis, 2018.

NIS17. NIST. Nist post-quantum standardization, https://csrc.nist.gov/projects/post-quantum-cryptography, 2017.

Pra62. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

Sha89. Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 606–609, New York, NY, 1989. Springer New York.

Ste94. Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 13–21, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

Vér97. Pascal Véron. Improved identification schemes based on error-correcting codes. 8(1):57–69, jan 1997.

WKH+22. Violetta Weger, Karan Khathuria, Anna-Lena Horlemann, Massimo Battaglioni, Paolo Santini, and Edoardo Persichetti. On the hardness of the lee syndrome decoding problem. *Advances in Mathematics of Communications*, page doi: 10.3934/amc.2022029, 2022.