EXPLOITING SIGNATURE LEAKAGES: BREAKING ENHANCED PQSIGRM

THOMAS DEBRIS–ALAZARD[1], PIERRE LOISEL[1], AND VALENTIN VASSEUR[2]

ABSTRACT. Enhanced pqsigRM is a code-based hash-and-sign scheme proposed to the second National Institute of Standards and Technology call for post-quantum signatures. The scheme is based on the $(U, U+V)$-construction and it enjoys remarkably small signature lengths, about 1KBytes for a security level of 128 bits. Unfortunately we show that signatures leak information about the underlying $(U, U+V)$-structure. It allows to retrieve the private-key with $100,000$ signatures.

1. Introduction

Code-Based Signature Schemes. Until very recently, no digital signature scheme based on the hardness of decoding a random linear code which could compete with widespread schemes like DSA or RSA were known. Contrary to those signatures, code-based schemes would provide quantum-safe solutions, no efficient quantum algorithm is known to efficiently decode a random linear code. A glaring illustration of this situation was given by the first National Institute of Standards and Technology (NIST) call from '17 for post-quantum cryptographic primitives: all code-based signatures were broken during the first of the four rounds [HBPL18, DT18, ASP18]. However and fortunately the situation has drastically changed. Many safe code-based signatures are currently known. This rapidly changing situation may also once again be illustrated by the efforts of the NIST which had recently called for an additional standardizing process dedicated to quantum-safe digital signatures (in order to increase the diversity of the future standards). Many schemes whose security inherits from coding problems were proposed and they can be sorted according to two main approaches. The first one uses the Fiat-Shamir transform [FS87] to turn (i) zero-knowledge authentication schemes [BBB+23b, BBB+23a, CNP+23] or (ii) MPC in the head protocols [ABB+23c, ABB+23b, ABB+23a, ABB+23d, AMFG+23] into signatures while the second one is known as hash-and-sign [CNL+23, RBK+23, BCC+23]. Both approaches enjoy dual advantages and drawbacks. Using the Fiat-Shamir transform leads to large signature lengths but small public-keys while hash-and-sign schemes usually turn out to benefit from short signatures and fast verification but big public-keys. However, even if both approaches turn out to be dual in terms of advantages and drawbacks, it has been historically much more difficult to build secure and efficient code-based hash-and-sign schemes. It is again illustrated by the current NIST standardization: only three code-based hash-and-sign schemes were submitted [CNL+23, RBK+23, BCC+23].

First Attempt to Design Code-Based Hash-and-Sign. The first approach to design a code-based hash-and-sign primitive was given by the so-called CFS

[1] Inria and Laboratoire LIX, École Polytechnique, Palaiseau, France
[2] Thales, Gennevilliers, France
E-mail addresses: thomas.debris@inria.fr, pierre.loisel@inria.fr, valentin.vasseur@thalesgroup.com.

scheme [CFS01]. It consisted in finding parity-check matrices $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ such that the solution $\mathbf{e}$ of smallest Hamming weight of the equation,

$$\mathbf{e}\mathbf{H}^\top = \mathbf{s} \tag{1}$$

could be found for a non-negligible proportion of $\mathbf{s} \in \mathbb{F}_2^{n-k}$. This task was achieved by using high rate Goppa codes but it has unfortunately two main drawbacks: (i) high rates Goppa codes are distinguishable from random codes [FGO+11, CMT23] (avoiding to reduce the security to decoding a random linear code as aimed in code-based cryptography) and (ii) security scales only weakly super-polynomially in the key-size for polynomial-time signature generation making the scheme unpractical (public-key size of terabytes, signature time of hours for quantum-safe security levels). Facing these difficulties, one may say that CFS approach is asking too much. Particularly, it seems natural to relax for hash-and-sign purposes the task of having a non-negligible fraction of syndromes that can be *decoded* (it constraints a lot the choice of the code). One way to proceed is by only asking parity-check matrices $\mathbf{H}$ such that Equation (1) admits for most of the $\mathbf{s}$'s a solution $\mathbf{e}$ of moderately small weight which is enough to design a hash-and-sign scheme. Interestingly, there exists many code families achieving such task. These codes are not used in error-correction but in lossy source coding or source-distortion theory where the problem is to find codes with an associated "decoding" algorithm which can approximate any word of the ambient space by a close enough codeword. Convolutional, LDGM or polar codes are examples of such codes but unfortunately they are distinguishable from random codes [LJ12, PT16, BCD+16] avoiding their use for designing signature schemes.

SURF: the First Code-Based Signature Based on Source-Distortion Theory. Despite the aforementioned difficulties, basing hash-and-sign construction on source-distortion theory turned out to be fruitful. A first attempt based on this approach was proposed in '17: SURF [DST17b] which introduced $(U, U+V)$ codes in this context. The latter are just a way of building codes of length n from two codes U and V of length $n/2$. It consists in,

$$(U, U+V) \stackrel{\text{def}}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U, \mathbf{v} \in V\}.$$

These codes are extremely attractive as even by choosing codes U and V as random (putting as few as possible structure on the whole code in order to make it indistinguishable from a random code), then a simple and polynomial-time algorithm exploiting the structure can be used to solve Equation (1) for any $\mathbf{s}$ as input and for weights such that the best generic decoding algorithm is still exponential (when the parity-check matrix $\mathbf{H}$ is regarded as random). However, the decoding algorithm works for weights such that there are an exponential number of solutions opening the possibility of security issues. Indeed, any signature (here the solution $\mathbf{e}$) is intended to be made public. In such case, the decoding algorithm computes one of the solutions and, if no precautions are taken, it can choose one of them in a way that reveals the underlying structure. This issue was clearly identified in the case of SURF [DST17b, §5.1]. Fortunately by tweaking the decoding algorithm and performing an appropriate rejection sampling, it was shown [DST17b, §5] how to produce signatures whose distribution is independent of the underlying $(U, U+V)$-structure, *i.e.,* the secret. It was the key to prove that SURF security relies on the hardness (i) of decoding a random linear codes and (ii) distinguishing a permuted $(U, U+V)$-code from a random code. Unfortunately there is a fatal issue. The second assumption turns out to be false: it is easy to distinguish a permuted $(U, U+V)$-code from a random code in the regime of parameters needed for the SURF scheme to work (codes U and V have to be binary codes and their respective dimensions k_U and k_V need to verify $k_U > k_V$). The key observation [DST17a] was that the hull (the intersection of the primal code with its dual) of permuted $(U, U+V)$-codes does not behave as the hull of random codes, thus providing a distinguisher. But even worse, the hull is full of permuted codewords $(\mathbf{u}, \mathbf{u})$ where $\mathbf{u} \in U$ from which the secret permutation can easily be retrieved and then breaking the scheme.

(Enhanced) pqsigRM: a Hash-and-Sign Based on the $(U, U+V)$-Codes. In light of the history of SURF, it may seem that using a $(U, U+V)$-structure to design hash-and-sign schemes is a dead-end. However, this idea has been pursued. In '17, it has been submitted to the NIST call for post-quantum cryptographic primitives the code-based hash-and-sign pqsigRM scheme [LKLN17]. It also takes advantage of some $(U, U+V)$-structure to decode. However, to avoid SURF pitfall, authors of pqsigRM [LKLN17] proposed to add some structure: choosing some variation of Reed-Muller codes for U and V as well as puncturing some positions of the code. These two choices were made to avoid classical attacks against Reed-Muller codes and to increase the size of the hull in such a way that permuted $(\mathbf{u}, \mathbf{u})$ codewords become hard to find. This approach enabled to avoid the attack breaking SURF even if the public-code remains distinguishable from random codes for which the hull has typically dimension 0. However, pqsigRM has been broken in an official comment [ASP18] of the NIST forum. It has been first shown that one can locate the punctured positions using the hull of the code. Then it has been noticed that this information enables to run attacks that originally broke McEliece cryptosystem instantiated with Reed-Muller codes [Sid94]. Ultimately, this attack broke pqsigRM by only using the public-key. Despite this, it has been few years later proposed a variation, Enhanced pqsigRM [NCL+22]. Some other variation of the used $(U, U+V)$-code were proposed to defeat the attack from [ASP18]. This novel hash-and-sign has then been submitted [CNL+23] to the second NIST call for standardizing signature schemes.

Our Contribution: Breaking Enhanced pqsigRM by Exploiting Signature Leakage. Our contribution is that, despite the fact that Enhanced pqsigRM was designed to thwart known attacks exploiting the structure of the public-code, we show how to retrieve from the public-key for a security level of 128 bits of classical security and 100,000 signatures an equivalent secret key that would enable to forge new signatures. Our key observation has been that signatures in Enhanced pqsigRM are coming from a $(U, U+V)$-decoder which has not been designed to ensure that signatures distribution is independent of the secret-key. Therefore, as outlined in the case of SURF for which a proper rejection sampling was designed, signatures of pqsigRM are biased and they reveal information on the secret-key as its associated decoding algorithm is based on the $(U, U+V)$-structure. Our attack is based on simple empirical probabilities and we have developed two scripts available at `https://github.com/vvasseur/pqsigRM`. The first script is designed for rapid execution. It effectively illustrates the biases in the signature distribution. The second script, though more time-consuming, leverages these biases to reveal almost all the whole $(U, U+V)$-structure of the secret.

2. Preliminaries

Basic Notation. The notation $x \overset{\text{def}}{=} y$ means that x is being defined as equal to y. We let $[\![a, b]\!]$ denote the set of the integers between a and b. Vectors are in row notation and are written with bold letters (such as $\mathbf{x}$). Uppercase bold letters are used to denote matrices (such as $\mathbf{H}$). Given a matrix $\mathbf{H} = (H_{i,j})_{i,j}$

with n columns and a permutation σ of $[\![1, n]\!]$, we let $\mathbf{H}^{\sigma}$ denote the matrix whose columns are permuted according to σ, i.e., $\mathbf{H}^{\sigma} = (H_{i,\sigma(j)})$. The canonical inner product $\sum_{i=1}^{n} x_i y_i$ between two vectors $\mathbf{x}$ and $\mathbf{y}$ of $\mathbb{F}_2^n$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$ where $\mathbb{F}_2$ denotes the binary field. The concatenation of two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $(\mathbf{x}, \mathbf{y})$. The Hamming weight of a vector $\mathbf{x} \in \mathbb{F}_2^n$ is defined as the number of its non-zero coordinates,

$$|\mathbf{x}| \stackrel{\text{def}}{=} \sharp \{i \in [\![1, n]\!],\ x_i \neq 0\}$$

where $\sharp\mathcal{A}$ stands for the cardinality of a finite set $\mathcal{A}$.

Coding Theory. A binary linear code $\mathcal{C}$ of length n and dimension k is a subspace of the vector space $\mathbb{F}_2^n$ of dimension k. We say that it has parameters $[n, k]$ or that it is an $[n, k]$-code. A generator matrix $\mathbf{G}$ for $\mathcal{C}$ is a full rank $k \times n$ matrix over $\mathbb{F}_2^{k \times n}$ such that,

$$\mathcal{C} = \left\{ \mathbf{x}\mathbf{G} :\ \mathbf{x} \in \mathbb{F}_2^k \right\}.$$

In other words, the rows of $\mathbf{G}$ form a basis of $\mathcal{C}$. A parity-check matrix $\mathbf{H}$ for $\mathcal{C}$ is a full-rank $(n-k) \times n$ matrix over $\mathbb{F}_2^{(n-k) \times n}$ such that,

$$\mathcal{C} = \left\{ \mathbf{c} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{c}^{\top} = \mathbf{0} \right\}.$$

In other words, $\mathcal{C}$ is the null space of $\mathbf{H}$. The code whose generator matrix is the parity-check matrix of $\mathcal{C}$ is called the dual code of $\mathcal{C}$. It is an $[n, n-k]$-code defined as,

$$\mathcal{C}^{\perp} \stackrel{\text{def}}{=} \{\mathbf{h} \in \mathbb{F}_2^n : \forall \mathbf{c} \in \mathcal{C},\ \langle \mathbf{h}, \mathbf{c} \rangle = 0\}.$$

The hull of a code $\mathcal{C}$ is defined as the intersection with its dual, i.e., $\text{hull}(\mathcal{C}) \stackrel{\text{def}}{=} \mathcal{C} \cap \mathcal{C}^{\perp}$.

3. The Enhanced pqsigRM Signature Scheme

We recall in this section basic facts about Enhanced pqsigRM [CNL+23]. It is roughly speaking a hash-and-sign scheme: the message $\mathbf{m}$ to be signed is first hashed by a hash function Hash and then the signature is equal to $f^{-1}(\text{Hash}(\mathbf{m}))$ where f is a trapdoor one-way function. Therefore, the pair $(\mathbf{m}, f^{-1}(\text{Hash}(\mathbf{m})))$ forms a valid signature. Code-based cryptography provides the following one way-function,

$$f_{\mathbf{H}} :\ \mathbf{e} \in \mathcal{S}_{\leq w} \longmapsto \mathbf{e}\mathbf{H}^{\top} \in \mathbb{F}_2^{n-k}$$

where $\mathcal{S}_{\leq w}$ denotes the words of $\mathbb{F}_2^n$ of Hamming weight $\leq w$ and $\mathbf{H}$ is a parity-check matrix of size $(n-k) \times n$. To introduce a trapdoor in $f_{\mathbf{H}}$ authors of [CNL+23] proposed to use parity-check matrices from a family of codes derived from Reed-Muller codes. The proposed codes are rather involved in order to avoid structural attacks, i.e., recovering from $\mathbf{H}$ its underlying structure enabling to invert $f_{\mathbf{H}}$. Our aim in the following two subsections is to describe how the matrix $\mathbf{H}$ is chosen in the reference implementation of [CNL+23][1].

3.1. Reed-Muller Codes.

The Reed-Muller codes are a family of $(U, U+V)$-codes which are constructed in a recursive way. First we will introduce the $(U, U+V)$ construction which is a way to construct a new code based on two given codes U and V of the same length.

Definition 1 ($(U, U+V)$-Codes)**.** *Let U, V be linear binary codes of length $n/2$ and dimension k_U, k_V. We define the subset of $\mathbb{F}_2^n$:*

$$(U, U+V) \stackrel{\text{def}}{=} \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) \text{ such that } \mathbf{u} \in U \text{ and } \mathbf{v} \in V\}$$

[1]It differs slightly from the specifications [CNL+23, §3.3] without incidence on our attack (our attack relies on signatures produced by the reference implementation).

which is a linear code of length n and dimension $k = k_U + k_V$. A generator matrix of such a code is given by,

$$\begin{pmatrix} \mathbf{G}_U & \mathbf{G}_U \\ \mathbf{0} & \mathbf{G}_V \end{pmatrix}$$

where $\mathbf{G}_U \in \mathbb{F}_2^{k_U \times n/2}$ (resp. $\mathbf{G}_V \in \mathbb{F}_2^{k_V \times n/2}$) is a generator matrix of U (resp. V).

We are now ready to recall how Reed-Muller codes are built by recursively applying the $(U, U+V)$ construction.

Definition 2. *Let $m \in \mathbb{N}$ and $r \in [\![0, m]\!]$. The Reed-Muller code $\mathrm{RM}(r,m)$ is the $[2^m, k]$-code, where,*

$$k \overset{def}{=} \sum_{j=0}^{r} \binom{m}{j},$$

built recursively as a $(U, U+V)$-code with generator matrix,

$$\mathbf{G}(r,m) \overset{def}{=} \begin{bmatrix} \mathbf{G}(r, m-1) & \mathbf{G}(r, m-1) \\ 0 & \mathbf{G}(r-1, m-1) \end{bmatrix} \tag{2}$$

$$\mathbf{G}(0,m) \overset{def}{=} \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \in \mathbb{F}_2^{2^m} \quad \textit{and} \quad \mathbf{G}(m,m) \overset{def}{=} \mathbf{I}_{2^m}$$

3.2. **Key Generation.** It is proposed in Enhanced pqsigRM [CNL+23] to choose a parity-check matrix of some code $\mathcal{C}$ after making a series of modification over $\mathrm{RM}(6, 13)$. The code $\mathcal{C}$ is defined from a generator matrix which is built as follows.

1) Pick two permutations σ_1 and σ_2 of $[\![1, n/4]\!]$ and consider

$$\begin{bmatrix} \mathbf{G}(6,11)^{\sigma_1} & \mathbf{G}(6,11)^{\sigma_1} & \mathbf{G}(6,11)^{\sigma_1} & \mathbf{G}(6,11)^{\sigma_1} \\ 0 & \mathbf{G}(5,11) & 0 & \mathbf{G}(5,11) \\ 0 & 0 & \mathbf{G}(5,11) & \mathbf{G}(5,11) \\ 0 & 0 & 0 & \mathbf{G}(4,11)^{\sigma_2} \end{bmatrix}$$

Notice that it is not anymore a generator matrix of a Reed-Muller code $\mathrm{RM}(6,13)$ but it is still a generator matrix of a $(U, U+V)$-code. The design rationale of this permutation choice is to make the dimension of $\mathrm{hull}(\mathcal{C}) \setminus \mathrm{RM}(r,m)$ big.

2) All the matrices $\mathbf{G}(6,11)$'s appearing above are decomposed via the recursive $(U, U+V)$-construction. After these five recursive steps, the top 2^6 rows of the matrix consist of $4 \cdot 2^5 = 2^7$ permuted (according to σ_1) identity matrices. Then each of them are replaced by the same random matrix $\mathbf{M} \in \mathbb{F}_2^{(2^6-2)\times 2^6}$ with the property that the dual of the code generated by $\mathbf{M}$ contains at least one vector with odd Hamming weight. Notice that the resulting matrix has $k-2$ rows where k denotes the dimension of $\mathbf{G}(6,13)$.

3) Ultimately, two rows are appended to the matrix. They are chosen as random vector coordinates (linearly independent of the existing rows) with one of them having an odd weight.

After these steps, the following matrix is obtained,

$$\mathbf{G}_f \overset{\text{def}}{=} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{U} & \mathbf{U} \\ 0 & \mathbf{V} \end{bmatrix} \in \mathbb{F}_2^{k \times 2^{13}} \text{ with } k \overset{\text{def}}{=} \sum_{j=0}^{6} \binom{13}{j} \tag{3}$$

and where blocks $\mathbf{A}$ and $\mathbf{B}$ contain 2^6 rows. Therefore, the code contains a $(U, U+V)$-code and the U, V codes are themselves $(U, U+V)$-codes. This fact is important as the decoding algorithm and our attack are based on these two properties.

Secret-Key. A parity-check matrix $\mathbf{H}_f$ of the code generated by $\mathbf{G}_f$ is computed. Then a random permutation $\mathbf{P}$ is picked up to the moment that $\mathbf{H}_f\mathbf{P}$ can be put in systematic form, i.e., there exists a non-singular matrix $\mathbf{S}$ such that $\mathbf{S}\mathbf{H}_f\mathbf{P} = (\mathbf{I}, \mathbf{H}')$ where denote $\mathbf{I}$ the identity matrix of the right size. The secret-key consists in the invertible matrix $\mathbf{S}$, the permutation $\mathbf{P}$ and the above permutations σ_1, σ_2 as well as the two appended rows to build $\mathbf{G}_f$.

Public-Key. It consists of $\mathbf{H} \overset{\text{def}}{=} (\mathbf{I}_d, \mathbf{H}')$ and $w = 1370$.

3.3. **Signing Process, i.e., Inverting $f_{\mathbf{H}}$.** We now briefly describe how the decoding algorithm works. We refer to [CNL+23] for more details, in particular the proof of its correctness. Our aim in this subsection is to show how this decoding algorithm (based on [Dum04] approach) takes advantage of the $(U, U+V)$-structure.

The decoding algorithm takes as input the public parity-check matrix $\mathbf{H}$, the syndrome to decode and the secret-key. It uses as main subroutine the function RecursiveDecode using the recursive $(U, U+V)$-structure of the code. In order to be consistent, we made the choice to describe RecursiveDecode as in [CNL+23]. Particularly, vectors' coordinates are seen as elements of $\{-1, 1\}$ and all operations are done in $\mathbb{Q}$. We let $\star$ denote the component-wise multiplication. The terminating cases use a function MaximumDecoding corresponding to a decoding algorithm which can be a majority voting (when the underlying code is the repetition-code) or a naive decoding algorithm using linear algebra, *e.g.* one iteration of Prange algorithm [Pra62] (when the code is random). The permutation being used in

Algorithm 1 Tweaked decoding algorithm

```
1: function Decode($\mathbf{s}, \mathbf{H}$)
2:     Compute via linear algebra $\mathbf{r}$ such that $\mathbf{r}\mathbf{H}^\top = \mathbf{s}$
3:     while True do
4:         $\mathbf{c} \leftarrow$ Random codeword from the code with parity-check matrix $\mathbf{H}$
5:         $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{c}$
6:         $\mathbf{c}' \leftarrow$ RecursiveDecode($\mathbf{r}\mathbf{P}^{-1}, 6, 13$)$\mathbf{P}$
7:         if $|\mathbf{r} + \mathbf{c}'| \leq w$ then return $\mathbf{r} + \mathbf{c}'$
8: function RecursiveDecode($\mathbf{y}$,$r$,$m$):
9:     $\mathbf{y} \leftarrow \mathbf{y}^{\sigma^{-1}}$            ▷ $\sigma$ is $\sigma_1$, $\sigma_2$ or Id depending on the current block
10:    if $r = 0$ or $r = m$ then
11:        Maximum decoding on RM $(r, m)$
12:                                  ▷ With potentially the $2^6$ modified rows.
13:    else
14:        Write $\mathbf{y} \leftarrow (\mathbf{y}', \mathbf{y}'')$
15:        $\mathbf{y}^V \leftarrow \mathbf{y}' \cdot \mathbf{y}''$
16:        $\mathbf{v} \leftarrow$ RecursiveDecode($\mathbf{y}^V, r-1, m-1$)
17:        $\mathbf{y}^U \leftarrow \frac{\mathbf{y}' + \mathbf{y}'' \star \mathbf{v}}{2}$
18:        $\mathbf{u} \leftarrow$ RecursiveDecode($\mathbf{y}^U, r, m-1$)
19:        return $(\mathbf{u}, \mathbf{u} \star \mathbf{v})^\sigma$
```

the last recursive step of RecursiveDecode is the identity. Using notation of Algorithm 1, let (in additive notation),

$$\mathbf{e}_U \overset{\text{def}}{=} \mathbf{y}^V - \mathbf{v} \quad \text{and} \quad \mathbf{e}_V \overset{\text{def}}{=} \mathbf{y}^V - \mathbf{u}$$

where $\mathbf{y}^V, \mathbf{y}^U, \mathbf{u}, \mathbf{v}$ are the computed vectors in this last iteration.

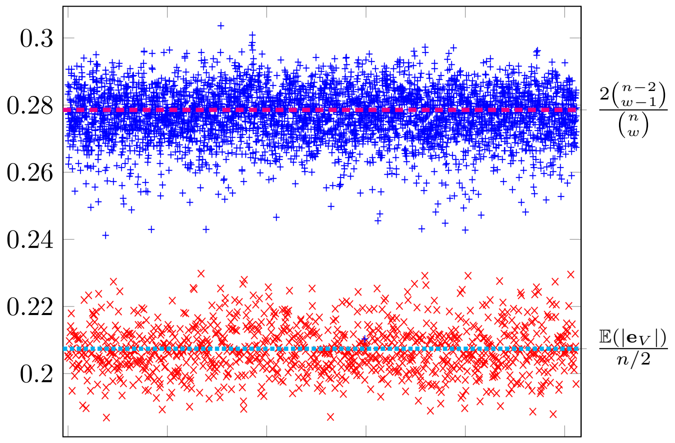

FIGURE 1. Empirical estimation of the $\mathbb{P}(e_i \neq e_j)$'s by using 5,000 signatures. For 1,024 points i in the x-axis, we draw empirical estimations $\mathbb{P}(e_i \neq e_j)$ for $j = i + n/2$ (in red) and four other random coordinates (in blue).

As we will show now, our attack crucially uses the fact that an output signature has the form,

$$(\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)\mathbf{P}.$$

4. Signature Leakage

Produced signatures in [CNL+23], which are intended to be made public, are outputs of Algorithm 1. Our key observation is that they have the form $\mathbf{eP}$, where (here $n = 2^{13}$)

$$\mathbf{e} \overset{\text{def}}{=} (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V) \in \mathbb{F}_2^n$$

and $(\mathbf{e}_U, \mathbf{e}_V) \in \mathbb{F}_2^{n/2}$ are the output vectors of the recursive decoding algorithm before its last step. These signatures leak information about the secret permutation $\mathbf{P}$: they enable to recover the image by $\mathbf{P}$ of the pairs

$$(i, i + n/2) \quad \text{or} \quad (i + n/2, i)$$

that we will call *matched pairs*. Indeed,

$$\mathbb{P}\left(e_i \neq e_{i+n/2}\right) = \mathbb{P}\left((e_V)_i = 1\right) \approx \frac{\mathbb{E}\left(|\mathbf{e}_V|\right)}{n/2}. \tag{4}$$

On the other hand, if we take non-matched pairs (i, j), then we expect (signatures have Hamming weight $\approx w$),

$$\mathbb{P}\left(e_i \neq e_j\right) \approx \frac{2\binom{n-2}{w-1}}{\binom{n}{w}}. \tag{5}$$

If we suppose that our approximations are good enough, then we expect to be able to distinguish between pairs of positions which are matched or not by making naive empirical statistics. In Figure 1 we compare these approximations with the empirical probabilities of $\mathbb{P}(e_i \neq e_j)$ whether (i, j) are matched or not. To draw this figure we used signatures output by the signing algorithm from the reference implementation (which can be found at `https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures`) for some known permutation $\mathbf{P}$ (we also used these iterations to estimate $\mathbb{E}(|\mathbf{e}_V|)$).

As we can appreciate, we can easily differentiate between matched or not pairs, i.e., images by the permutation $\mathbf{P}$ of pairs of the form $(i, i+n/2)$, $(i+n/2, i)$ or not. In other words, given genuine Enhanced pqsigRM signatures we can recover matched pairs. It explains why signatures leak information about the secret key. In the following section we explain how to exploit this knowledge.

5. Recovering the code structure

The attack begins with the extraction of matched pairs from the signatures using standard correlation statistics. After this step, the obtained matched pairs are unordered: each post-permutation matched pair corresponds either to $(i, i+n/2)$ or $(i+n/2, i)$ pre-permutation for some $i \in [\![1, n/2]\!]$. Now if one manages to distinguish these two cases, then the $(U, U+V)$-structure of the code is exposed. This is achieved by applying linear algebra to the generator matrix which is derived from the public-key, using the matched pairs information. The final stage of the attack involves systematically deconstructing the $(U, U+V)$-recursive structure layer by layer until classic Reed-Muller attacks [CB14, MS07] apply to recover the full structure.

5.1. Recovering the Matched Pairs.

In order to recover the matched pairs, we propose to compute the Pearson correlation coefficients $\rho_{i,j}$ between $(-1)^{s_i}$ and $(-1)^{s_j}$ for each pair of signature bits s_i and s_j with a dataset of N samples. These coefficients form the weighted adjacency matrix $\mathbf{P}$ of a graph where each vertex is a signature bit. By solving the maximum weight matching problem on this graph, we efficiently identify the pairs exhibiting the strongest correlation. This technique is particularly effective with a sample size $N = 10^5$.

5.2. Ordering the Matched Pairs Without Appended Rows.

We now need to turn our attention on the generator matrix of the Enhanced pqsigRM public-code. For the sake of simplicity we will first proceed as if there were no appended rows in the key generation (blocks $\mathbf{A}$ and $\mathbf{B}$ in Equation (3)). Let us suppose that a generator matrix of the public code has the following form $\mathbf{G} \stackrel{def}{=} \mathbf{S} \cdot \begin{bmatrix} \mathbf{U} & \mathbf{U} \\ \mathbf{0} & \mathbf{V} \end{bmatrix} \mathbf{P}$ for some non-singular matrix $\mathbf{S}$ and a permutation matrix $\mathbf{P}$.

Recovering Permuted U and V Codes. First, one can rearrange the columns of the public generator matrix such that the matched pairs are $n/2$ places apart. After this step, we obtain a matrix that follows the form:

$$\mathbf{S} \cdot \begin{bmatrix} \mathbf{U}\mathbf{P}' & \mathbf{U}\mathbf{P}' \\ \mathbf{V}'\mathbf{P}' & \mathbf{V}''\mathbf{P}' \end{bmatrix}. \tag{6}$$

Here, $\mathbf{V}'$ and $\mathbf{V}''$ partition the columns of $\mathbf{V}$ into two parts so their sum is $\mathbf{V}$, and $\mathbf{P}'$ is a permutation matrix. Now, multiplying (6) by the matrix $\mathbf{J} \stackrel{def}{=} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$, we obtain

$$\mathbf{S} \cdot \begin{bmatrix} \mathbf{0} & \mathbf{U}\mathbf{P}' \\ \mathbf{V}\mathbf{P}' & \mathbf{V}'\mathbf{P}' \end{bmatrix}. \tag{7}$$

Using straightforward Gaussian elimination, we identify $\mathbf{U}\mathbf{P}'$ and $\mathbf{V}\mathbf{P}'$ up to multiplication on the left by non-singular matrices. But at this point, we have no information about what should be on the left or on the right.

What is on the Left, What is on the Right. We would like to find a permutation $\mathbf{Q}$ that right-multiplied to (6) gives:

$$\mathbf{S} \cdot \begin{bmatrix} \mathbf{UP'} & \mathbf{UP'} \\ \mathbf{0} & \mathbf{VP'} \end{bmatrix}. \tag{8}$$

Then by multiplying by $\mathbf{J}$, we will obtain,

$$\mathbf{S} \cdot \begin{bmatrix} \mathbf{0} & \mathbf{UP'} \\ \mathbf{VP'} & \mathbf{0} \end{bmatrix}.$$

So the right permutation $\mathbf{Q}$ renders $\mathbf{V}'$ as zero. Our task is now to find the right combination of swaps among the $n/2$ possibilities that achieves the form given in Equation (8).

The main idea comes from realising that those swaps constitute linear operations. Indeed, swapping the two columns at positions i and $i + n/2$ in $\mathbf{G}$ has the same effect as multiplying on the right by the elementary matrix $\mathbf{T_{i,i+n/2}}$. But,

$$\mathbf{T_{i,i+n/2}} \cdot \mathbf{J} = \mathbf{J} \cdot \mathbf{L_{i,i+n/2}}^{2}.$$

So swapping matched columns then multiplying by $\mathbf{J}$ has the same effect as multiplying by $\mathbf{J}$ then adding the ith column to the $i + n/2$th column. For this reason, we will now directly work on the matrix $\mathbf{G}' \overset{def}{=} \mathbf{G} \cdot \mathbf{J}$.

Let us now focus on the actual situation that we obtain after Gaussian elimination on $\mathbf{G}'$:

$$\begin{bmatrix} \mathbf{S_1VP'} & \mathbf{S_1V'P'} \\ \mathbf{0} & \mathbf{S_2UP'} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{S_V} & \mathbf{0} & \mathbf{S_{V'}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{S_U} \end{bmatrix}. \tag{9}$$

A matched columns swap of $\mathbf{G}$ followed by a Gaussian elimination of $\mathbf{G}'$ only changes $\mathbf{S_{V'}}$. Essentially, there are two possibilities: if the column at position $i + n/2$, (i) contains a pivot for $\mathbf{S_2UP'}$ then the row corresponding to this pivot is added to every row in the support of that column, otherwise (ii) the column at position i is directly added to the column at position $i+n/2$. We should observe that this process is a purely linear operation with a dimension of $n/2$ when considering all possible swaps. We can indeed derive a matrix $\mathbf{E}$ with dimensions $n/2 \times (\dim V \cdot n/2)$ such that, after swapping columns i and $i+n/2$ for each i in the support of some vector $\mathbf{x}$ within matrix $\mathbf{G}$, the matrix $\mathbf{G}'$ transforms to, after Gaussian elimination:

$$\begin{bmatrix} \mathbf{I} & \mathbf{S_V} & \mathbf{0} & \mathbf{S_{V'}} + (\mathbf{I}_{\dim V} \otimes \mathbf{x}) \cdot \mathbf{E}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{S_U} \end{bmatrix}. \tag{10}$$

Consequently, a permutation that zeroes out $\mathbf{S_{V'}}$ can be identified by solving a large system of linear equations. However, progressively zeroing rows of $\mathbf{S_{V'}}$ as in Algorithm 2 is more fitting than addressing a fairly large system because it can easily be adapted to handle the appended rows scenario.

5.3. **Ordering Matched Pairs in the General Case.** In §5.2, for the sake of clarity, we only explain the method of ordering matched pairs using the generator matrix in the simpler scenario where no additional rows are appended. In practice, the actual attack begins with the scenario that includes appended rows. Upon completing this phase, the version with no appended rows can be considered, as the further knowledge of the structure of the matrix allows extracting the $\mathbf{U}$ and $\mathbf{V}$ blocks, thus removing the appended rows.

The main idea of the algorithm remains consistent in both scenarios. However, the presence of appended rows in the initial phase introduces additional complexity

---

[2] $(T_{i,j})_{m,n} = \begin{cases} \delta_{m,n} & \text{if } m \neq i, m \neq j \\ \delta_{m,j} & \text{if } m = i \\ \delta_{m,i} & \text{if } m = j \end{cases}$ and $(L_{i,j})_{m,n} = \begin{cases} 1 & \text{if } (m,n) = (i,j) \\ \delta_{m,n} & \text{otherwise} \end{cases}$

Algorithm 2 Column swap algorithm to nullify $\mathbf{S_{V'}}$.

```
Require: Initial permutation $\mathbf{P}$ and matrix $\mathbf{G'}$ as in Equation (9).
Ensure: Updated permutation $\mathbf{P}$ for $\mathbf{G}$ such that $\mathbf{S_{V'}} = \mathbf{0}$.
1: Compute the matrix $\mathbf{E}$ as in Equation (10).
2: while $\mathbf{S_{V'}} \neq \mathbf{0}$ do
3:     for all rows $\mathbf{R}_i$ in $\mathbf{S_{V'}}$ do
4:         Solve $\mathbf{x} \cdot \mathbf{E_i}^\top = \mathbf{R_i}$ for $\mathbf{x}$.        ▷ Compute a swap vector
5:         for all $j$ in the support of $\mathbf{x}$ do
6:             Swap column $P_j$ with $P_{j+n/2}$.
7:             $G'_{j+n/2} \leftarrow G'_{j+n/2} + G'_j$.
8:             Echelonize $\mathbf{G'}$.
   return $\mathbf{P}$
```

unnecessary for the understanding. We detail here an approach that has proven effective in the Enhanced pqsigRM case.

We consider the full Enhanced pqsigRM generator matrix, with the appended rows, (9) thus becomes,

$$\begin{bmatrix} \mathbf{A_1} & \mathbf{A_2} & \mathbf{B_1} & \mathbf{B_2} \\ \mathbf{I} & \mathbf{S_V} + \mathbf{S_A} & \mathbf{0} & \mathbf{S_{V'}} + \mathbf{S_{B_2}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{S_U} + \mathbf{S'_{B_2}} \end{bmatrix}. \tag{11}$$

In this scenario, we lack direct access to $\mathbf{S_{V'}}$; instead, linear combinations of $\mathbf{B_2}$ are added in the block under consideration. To address this, we follow the heuristic approach detailed in Algorithm 3.

Our strategy involves identifying rows that potentially contain components of $\mathbf{B_2}$, aggregating them into a set denoted as $\mathcal{A}$. Rather than attempting to cancel each individual row, our objective is now cancelling out the $\mathbf{V'}$ component, leaving the $\mathbf{B_2}$ component intact. This is achieved by incorporating the rows from $\mathcal{A}$ into our equation-solving process, while deliberately discarding the solution bits corresponding to these rows that extend the matrix $\mathbf{E}$. Moreover, should the initial iteration fail to fully uncovering the structure, we iterate the process, after applying any column-swap that reduces the rank of $\mathbf{S_{B_2}}$.

5.4. Finishing the Attack—Recovering an Equivalent Private Key.

Our attack effectively finds a $(U, U+V)$-decomposition into two codes, namely codes U and V admitting as generator matrices $\mathbf{U'} = \mathbf{UP'}$ and $\mathbf{V'} = \mathbf{VP'}$. These are equivalent to the original codes generated by $\mathbf{U}$ and $\mathbf{V}$ with the same permutation. Therefore, identifying a permutation that further decomposes the code $\mathbf{U'}$ will also decompose the code $\mathbf{V'}$.

The technique for deconstructing the generator matrix can be successively applied to progress along recursive paths of the code construction. To this aim it is necessary during the process to use signatures to obtain the matched pairs corresponding to each recursive level. It can be done because the signature bits can be reordered according to the matched pairs, giving vectors of the form $(\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$. Applying a XOR operation to these two components isolates $\mathbf{e}_V$. Conversely, an AND operation yields a subset of $\mathbf{e}_U$.[3].

Once the recursive $(U, U+V)$-structure down to two levels has been recovered, an equivalent global permutation for $\mathbf{P}$ can be recovered using [CB14, MS07]. Indeed, by applying these techniques, we can easily recover a permutation that exposes the Reed-Muller code $\mathbf{G}(5, 11)$ from which we derive a global permutation $\mathbf{Q}$. This

[3]We use an AND operation as the support of $\mathbf{e}_V$ is treated like erasures by the U-decoder We cancel these positions to avoid adding correlation noise.

Algorithm 3 Column swap algorithm to nullify $\mathbf{S}_{\mathbf{V}'}$ with appended rows.

```
Require: Initial permutation $\mathbf{P}$ and matrix $\mathbf{G}'$ as in Equation (11).
Ensure: Updated permutation $\mathbf{P}$ for $\mathbf{G}$ such that $\mathbf{S}_{\mathbf{V}'} = \mathbf{0}$.
1: Calculate the matrix $\mathbf{E}$ as in Equation (10).
2: while $\mathbf{S}_{\mathbf{V}'} \neq \mathbf{0}$ do
3:     $\mathcal{A} \leftarrow \emptyset$
4:     for all rows $\mathbf{R}_i$ in $\mathbf{S}_{V'}$ do
5:         $\mathbf{E}'_\mathbf{i} \leftarrow \mathbf{E}_\mathbf{i}$ extended with $\mathcal{A}$-indexed rows of $\mathbf{S}_{\mathbf{V}'}$.
6:         Solve $\mathbf{x} \cdot {\mathbf{E}'_\mathbf{i}}^\top = \mathbf{R}_\mathbf{i}$ for $\mathbf{x}$.          ▷ Compute a swap vector
7:         if it fails then
8:             $\mathcal{A} \leftarrow \mathcal{A} \cup \{i\}$
9:         for all $j$ in the support of $\mathbf{x}$ do
10:            if $j < n/2$ then          ▷ Ignore components from $\mathcal{A}$
11:                Swap column $P_j$ and $P_{j+n/2}$.
12:                $G'_{j+n/2} \leftarrow G'_{j+n/2} + G'_j$.
13:                Echelonize $\mathbf{G}'$.
14:     for $k = 0, \ldots, n/2 - 1$ do
15:         Swap columns $P_k$ and $P_{k+n/2}$ provisionally.
16:         $G'_{k+n/2} \leftarrow G'_{k+n/2} + G'_k$ provisionally.
17:         if rank$(\mathbf{S}_{\mathbf{V}'} + \mathbf{S}_{\mathbf{B_2}})$ decreases then
18:             Confirm the provisional operations.
19:             Exit for loop.
20:         else
21:             Undo the provisional operations.
```

approach can similarly be applied to $\mathbf{G}(4, 11)^{\sigma_2}$, enabling the derivation of a corresponding permutation, σ'_2. Regarding $\mathbf{G}(6, 11)^{\sigma_1}$, [CB14, MS07] are not directly applicable, as $\mathbf{G}(6, 6)$ has been replaced by a random code. However, by descending one additional recursive level, we can target the Reed-Muller code $\mathbf{G}(5, 10)$.

6. Conclusion

We have presented our attack against [CNL+23] which has been submitted to the NIST call for standardizing signatures. Some comments can be made.

How to Avoid our Attack. Our crucial remark has been to notice that the distribution of $(e_i \neq e_j)$ depends on (i, j) being matched or not as illustrated by Equations (4), (5) and Figure 1. A way to circumvent our attack would be to compute the $\mathbf{e}_V$'s to equalize both approximations. However doing so would not remove biases of order 2, i.e., standard deviations would not a priori be the same. It is necessary to ensure that at this step, distributions of the $(e_i \neq e_j)$'s are identically distributed by performing an appropriate rejection sampling as done in [DST17b] which also uses $(U, U + V)$-codes. However, [CNL+23] involves many recursive layers of $(U, U + V)$ which makes it very difficult to design (even one level of recursion makes that task difficult as illustrated by the rather involved analysis from [DST17b, §5]).

Another Attack. Our attack presented all along this paper has been announced on the NIST forum [DLV23]. Few months later another attack has also been announced [BBRST23]. It differs from ours because it does not require access to signatures, it only exploits the structure of the public-key. However, beyond the merits of [BBRST23] which uses less information than us, we believe that our attack is particularly relevant. The design rationale behind Enhanced pqsigRM (and its ancestor pqsigRM) has been to avoid such structural attacks by tweaking the code's structure as illustrated by the two submissions at the first and second NIST calls. It is likely that some other tweaks would avoid [BBRST23]. Our attack demonstrates that [CNL+23] approach suffers from the same issue given its design (our attack also applies similarly to pqsigRM): signatures coming from a $(U, U + V)$-decoder are biased unless a proper rejection sampling is performed.

Acknowledgments

The work of Thomas Debris-Alazard and Pierre Loisel was funded by the French Agence Nationale de la Recherche through ANR JCJC COLA (ANR-21-CE39-0011) and was supported by the PEPR quantique France 2030 programme (ANR-22-PETQ-0008).

References

[ABB$^+$23a] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[ABB$^+$23b] Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. MIRITH. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[ABB$^+$23c] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesus-Javier Chi-Dominguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Mathhieu Rivain, and Jean-Pierre Tillich. MIRA. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[ABB$^+$23d] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[AMFG$^+$23] Carlos Aguilar Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[ASP18] Jacob Alperin-Sheriff and Ray Perlner. Official NIST comments made for pqsigRM, 2018. Official NIST comments made for pqsigRM.

[BBB$^+$23a] Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Santini Paolo, and Wallace Robert. LESS. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[BBB$^+$23b] Marco Baldi, Alessandro Barenghi, Sebastian Bitzer, Patrick Karl, Alessio Manganiello Felice and, Pavoni, Gerardo Pelosi, Paolo Santini, Jonas Schupp, Freeman Slaughter, Antonia Wachter-Zeh, and Violetta Weger. CROSS. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[BBRST23] Pierre Briaud, Maxime Bros, Perlner Ray, and Daniel Smith-Tone. Official NIST comments made for Enhanced pqsigRM, 2023. Official NIST comments made for Enhanced pqsigRM.

[BCC$^+$23] Gustavo Banegas, Kévin Carrier, André Chailloux, Alain Couvreur, Thomas Debris-Alazard, Philippe Gaborit, Pierre Karpman, Johanna Loyer, Ruben Niederhagen, Nicolas Sendrier, Benjamin Smith, and Jean-Pierre Tillich. WAVE. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[BCD$^+$16] Magali Bardet, Julia Chaulet, Vlad Dragoi, Ayoub Otmani, and Jean-Pierre Tillich. Cryptanalysis of the McEliece public key cryptosystem based on polar codes. In *Post-Quantum Cryptography 2016*, LNCS, pages 118–143, Fukuoka, Japan, February 2016.

[CB14] Ivan V. Chizhov and Mikhail A. Borodin. Effective attack on the McEliece cryptosystem based on Reed–Muller codes. *Discrete Math. Appl.*, 24(5):273–280, 2014.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174, Gold Coast, Australia, 2001. Springer.

[CMT23] Alain Couvreur, Rocco Mora, and Jean-Pierre Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part IV*, volume 14441 of *LNCS*, pages 3–38. Springer, 2023.

[CNL$^+$23] Jinkyu Cho, Jong-Seon No, Yongwoo Lee, Young-Sik Kim, and Zahyun Koo. Enhanced pqsigRM. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[CNP$^+$23] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Samardjiska Simona, and Trimoska Monika. MEDS. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[DLV23] Thomas Debris-Alazard, Pierre Loisel, and Valentin Vasseur. Official NIST comments made for Enhanced pqsigRM, 2023. Official NIST comments made for Enhanced pqsigRM.

[DST17a] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. The problem with the surf scheme. preprint, November 2017. arXiv:1706.08065.

[DST17b] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Surf: a new code-based signature scheme. preprint, September 2017. arXiv:1706.08065v3.

[DT18] Thomas Debris-Alazard and Jean-Pierre Tillich. Two attacks on rank metric code-based schemes: Ranksign and an identity-based-encryption scheme. In *Advances in Cryptology - ASIACRYPT 2018*, volume 11272 of *LNCS*, pages 62–92, Brisbane, Australia, December 2018. Springer.

[Dum04] Ilya Dumer. Recursive decoding and its performance for low-rate Reed-Muller codes. *IEEE Trans. Inform. Theory*, 50(5):811–823, 2004.

[FGO$^+$11] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. In *Proc. IEEE Inf. Theory Workshop- ITW 2011*, pages 282–286, Paraty, Brasil, October 2011.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[HBPL18] Andreas Huelsing, Daniel J. Bernstein, Lorenz Panny, and Tanja Lange. Official NIST comments made for RaCoSS, 2018. Official NIST comments made for RaCoSS.

[LJ12] Carl Löndahl and Thomas Johansson. A new version of McEliece PKC based on convolutional codes. In *Information and Communications Security, ICICS*, volume 7168 of *LNCS*, pages 461–470. Springer, 2012.

[LKLN17] Wijik Lee, Young-Sik Kim, Yong-Woo Lee, and Jong-Seon No. Post quantum signature scheme based on modified Reed-Muller code pqsigRM. First round submission to the NIST post-quantum cryptography call, November 2017.

[MS07] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 347–360, Barcelona, Spain, 2007.

[NCL$^+$22] Jong-Seon No, Jinkyu Cho, Yongwoo Lee, ZaHyun Koo, and Young-Sik Kim. Enhanced pqsigrm: Code-based digital signature scheme with short signature and fast verification for post-quantum cryptography. *IACR Cryptol. ePrint Arch.*, page 1493, 2022.

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

[PT16] Aurélie Phesso and Jean-Pierre Tillich. An efficient attack on a code-based signature scheme. In *Post-Quantum Cryptography 2016*, volume 9606 of *LNCS*, pages 86–103, Fukuoka, Japan, February 2016. Springer.

[RBK$^+$23] Stefan Ritterhoff, Sebastian Bitzer, Patrick Karl, Georg Maringer, Thomas Schamberger, Jonas Schupp, Georg Sigl, Antonia Wachter-Zeh, and Violetta Weger. FuLeeca. NIST Round 1 submission to the Additional Call for Signature Schemes, 2023.

[Sid94] Vladimir Michilovich Sidelnikov. A public-key cryptosytem based on Reed-Muller codes. *Discrete Math. Appl.*, 4(3):191–207, 1994.