Key-Recovery by Side-Channel Information on the Matrix-Vector Product in Code-Based Cryptosystems

Boly Seck, Pierre-Louis Cayrel, Idy Diop, Vlad-Florin Dragoi, Kalen Couzon, Brice Colombier, Vincent Grosso

▶ **To cite this version:**

Boly Seck, Pierre-Louis Cayrel, Idy Diop, Vlad-Florin Dragoi, Kalen Couzon, et al.. Key-Recovery by Side-Channel Information on the Matrix-Vector Product in Code-Based Cryptosystems. 25th International Conference on Information Security and Cryptology (ICISC 2022), Nov 2022, Séoul, South Korea. pp.219-234, 10.1007/978-3-031-29371-9_11 . hal-04059124

HAL Id: hal-04059124
https://hal.science/hal-04059124v1

Submitted on 18 Oct 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 4.0 International License

Key-Recovery by Side-Channel Information on the Matrix-Vector Product in Code-Based Cryptosystems

Boly Seck[1,2][0000−0002−2362−601X], Pierre-Louis Cayrel[2][0000−0002−6708−868X], Idy Diop[1][0000−0002−9143−196X], Vlad-Florin Dragoi[4][0000−0002−8673−9097], Kalen Couzon[3], Brice Colombier[2][0000−0002−6028−3028], and Vincent Grosso[2][0000−0002−3874−7527]

[1] ESP, Laboratoire d'imagerie médicale et de Bio-informatique, Dakar, Sénégal
seck.boly@ugb.edu.sn
[2] Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France
pierre.louis.cayrel@univ-st-etienne.fr
[3] Univ de Versailles Saint-Quentin, Yvelines, France
kalen.couzon@ens.uvsq.fr
[4] Faculty of Exact Sciences, Aurel Vlaicu University, Arad, Romania
vlad.dragoi@uav.ro

Abstract. The modern security protocols in most of our systems rely primarily on three basic functions of asymmetric cryptography: public key encryption, digital signature, and key exchange. Today, we only do key exchange (TLS 1.3) with the ECDH protocol. The confidentiality is persistent because the session keys are discarded at the end and to certify this key exchange, we sign it with RSA or ECDSA. However, these cryptosystems are at least theoretically attackable in a quantum computer model. Thus the NIST PQC standardization process has given significant momentum to research on code-based public-key cryptosystems specifically. Their security is based on the hardness of the syndrome decoding problem. In this article, we first propose a new formalism of the matrix-vector product in based-code cryptography. Second, we present a chosen-ciphertext attack on the first step of Niederreiter decryption by solving the matrix-vector product problem with side-channel information. Finally, we put this result to recover secret information in code-based cryptosystems including some candidates for the extension of the NIST PQC normalization process.

Keywords: · Code-Based Cryptography · Side-Channel Attack · Matrix-Vector Product Problem · NIST PQC Standardization

1 Introduction

In recent years, a lot of research has been done on quantum computers [17, 32, 14]. These are computers that exploit the phenomena of quantum mechanics

to solve difficult mathematical problems in number theory, such as the Integer Factorization Problem or the Discrete Logarithm Problem. Shor [28] proved that if large-scale quantum computers are built, they will be able to break most of the current asymmetric cryptography like RSA, ECDSA, and ECDH schemes. This would seriously compromise the confidentiality and integrity of all digital communications.

Since then, cryptographic community proposed alternative solutions which remain safe in the quantum era. These schemes are called post-quantum resistant. In 2016, the National Institute of Standards and Technology (NIST) made a call to the community to propose post-quantum secure solutions for standardization. The process consists of several rounds, and only some of the candidates in each round are chosen to enter the next round. The most popular approaches are those based on the search for low-weight words for lattice, the problem of decoding random codes, solving multivariate polynomial systems, isogenies, and hash functions [7, 5]. Lattice-based cryptography has the reputation of being very efficient. Code-based cryptography using some codes is often considered to be already more mature and reliable such as McEliece [23] and Niederreiter [25] cryptosystems.

The majority of code-based post-quantum cryptosystems base their security on the classic hard problem in code-based cryptography: the binary Syndrome Decoding Problem (SDP). Informally, for a binary linear code $\mathcal{C}$ of length n and dimension k, having a parity-check matrix $\boldsymbol{H}$, the SDP is defined as follows: given $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$, find a binary vector $\boldsymbol{x}$ having less than t values equal to one, such that $\boldsymbol{Hx} = \boldsymbol{s}$. The best algorithm to solve this problem in this original version is the Information Set Decoding (ISD) proposed by Prange [27]. The ISD techniques are considered the best strategy for message recovery. It consists, in randomly permuting the matrix $\boldsymbol{H}$ (denote $\boldsymbol{P}$ such a permutation) until the support of the permuted $\boldsymbol{x}$ is included in the set $\{0, \ldots, n-k-1\}$, *i.e.*, the set where the $\boldsymbol{HP}$ is in upper triangular form. It has been considerably improved since then [3, 18, 19, 21, 22, 29], although the complexity remains exponential in t.

A recent possible solution to solve the syndrome decoding problem is to use Integer Linear Programming (ILP). The idea was first proposed by Feldman [11] and later improved by Feldman *et al.* [12]. Since the initial problem is nonlinear, some relaxation was proposed in order to decrease the complexity. Most recently, Cayrel *et al.* [9] showed that the SDP becomes considerably easier to solve if the syndrome is computed in $\mathbb{N}$. They perform a laser fault injection attack on the matrix-vector product when computing the syndrome in the encapsulation of *Classic McEliece*. This allows them by corrupting some specific instructions during this operation to have a syndrome in $\mathbb{N}$. To solve the syndrome decoding problem in $\mathbb{N}$, they propose to define the SDP as an ILP inspired by the ideas of Tanatmis *et al.* [33]. The complexity of recovering the secret message from the faulty syndrome is polynomial $\mathcal{O}(n^2)$ with an optimized version of their algorithm.

Afterwards, Colombier *et al.* [10] proposed to perform a message-recovery attack in *Classic McEliece* that relies on side-channel information only instead

of laser fault injection in the previous work[9]. The latter depends on the very strong attacker model and does not apply to optimized implementations of the algorithm that make optimal usage of the machine words capacity. Improvements include the power consumption analysis that is sufficient to obtain an integer syndrome using machine learning techniques. To recover the secret message they use the computationally-efficient score function and known information-set decoding methods.

Contribution: In this work, a key-recovery chosen-ciphertext attack against code-based cryptosystems is performed. We analyze in particular the secret operation of matrix-vector multiplication in Niederreiter decryption using a physical attack. First, we will introduce a new formalism in code-based cryptography. Informally, for $\boldsymbol{z}$ in $\mathbb{F}_2^{n-k}$ of any weight, the Matrix-Vector Product Problem (MVPP) is defined as follows: given $\boldsymbol{z}^*$ in $\mathbb{N}^{n-k}$, find $\boldsymbol{S} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ such that $\boldsymbol{S}\boldsymbol{z}^T = \boldsymbol{z}^*$. To get $\boldsymbol{z}^*$ in $\mathbb{N}^{n-k}$, we will use the same method of the power analysis attack in [10]. This method is based on side-channel analysis using random forests to recover $\boldsymbol{z}^*$ from the Hamming weight information obtained from the matrix-vector product in the first step of Niederreiter decryption. Second, we show that if we can construct the matrix $\boldsymbol{Z}^* = (\boldsymbol{z}_1^*, \cdots, \boldsymbol{z}_{n-k}^*)$ correctly, we can directly find the secret of the cryptosystem. We obtain directly the secret without solving the syndrome decoding problem unlike in [9, 10] and this is applicable for most of the code-based cryptosystems.

Organization: The paper is organized as follows. In Section 2, we focus on code-based cryptosystems, and in particular on the results of the NIST PQC competition. Section 3 defines the new formalism in code-based cryptography, the Matrix-Vector Product Problem (MVPP). In Section 4, we present our attack on the matrix-vector product in Niederreiter decryption using a side-channel attack. Finally, we conclude this paper in Section 5.

2 Code-Based Cryptosystems

2.1 Encoding Theory

Notations The following conventions and notations are used. A finite field is denoted by $\mathbb{F}$, and the ring of integers by $\mathbb{N}$. Vectors (column vectors) and matrices are written in bold, *e.g.*, a binary vector of length n is $\boldsymbol{x} \in \{0,1\}^n$, an $m \times n$ integer matrix is $\boldsymbol{A} = (a_{i,j})_{\substack{0\le i\le m-1 \\ 0\le j\le n-1}} \in \mathcal{M}_{m,n}(\mathbb{N})$. $\boldsymbol{A}[i]$ denotes the i-th line of $\boldsymbol{A}$ and a row sub-matrix of $\boldsymbol{A}$ indexed by a set $I \subseteq \{0,\ldots,m-1\}$ is denoted by $\boldsymbol{A}_{I,} = (a_{i,j})_{\substack{i\in I \\ 0\le j\le n-1}}$. The same applies to column vectors, *i.e.*, $\boldsymbol{x}_I$ is the sub-vector induced by the set I on $\boldsymbol{x}$.

Error-Correcting Codes An $[n,k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a vector subspace of $\mathbb{F}_q^n$, where k, n are positive integers with $k < n$. The elements of $\mathcal{C}$ are called

codewords. The support of a codeword $\mathsf{Supp}(\boldsymbol{c})$ is the set of non-zero positions of $\boldsymbol{c}$. We will represent a code either by its generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$ such that its lines form a basis of the vector space $\mathcal{C}$, or by its parity-check matrix, $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, where $\boldsymbol{H}\boldsymbol{G}^T = \boldsymbol{0}$ holds. One of the key elements of decoding is the use of metrics. In the Hamming metric, we consider codes with coefficients in $\mathbb{F}_q$ (generally, $\mathbb{F}_2$).

Definition 1 (Hamming metric). *Let $\boldsymbol{x} \in \mathbb{F}_2$, the Hamming weight $\mathsf{wt}(\boldsymbol{x})$ is the number of non null coordinates in $\boldsymbol{x}$, and the distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is the number of non null coordinates in $\mathsf{wt}(\boldsymbol{x}-\boldsymbol{y})$.*

The hardness of general decoding for a linear code is an $\mathcal{NP}$-complete problem in coding theory [4]. This is the Syndrome Decoding Problem (SDP), which is the hard problem in code-based cryptography.

Definition 2 (Binary-SDP). *Let $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$, a vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \in \mathbb{N}$. The syndrome decoding problem is to find $\boldsymbol{x} \in \mathbb{F}_2^n$ such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}$ and $\mathsf{wt}(\boldsymbol{x}) \leq t$.*

The best-known algorithms for solving this problem are all exponential in t. Except if the syndrome is computed in $\mathbb{N}$ instead of $\mathbb{F}_2$ [9].

Definition 3 ($\mathbb{N}$-SDP). *Let $\boldsymbol{H} \in \mathcal{M}_{n-k,n}(\mathbb{N})$, with $h_{i,j} \in \{0,1\}$ for all i,j, a vector $\boldsymbol{s} \in \mathbb{N}^{n-k}$ and $t \in \mathbb{N}^*$. The syndrome decoding problem in $\mathbb{N}$ is to find $\boldsymbol{x} \in \mathbb{N}^n$ with $x_i \in \{0,1\}$ such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}$ and $\mathsf{wt}(\boldsymbol{x}) \leq t$.*

$\boldsymbol{H}$ and $\boldsymbol{x}$ are sampled in the same way as for the binary SDP, only the matrix-vector multiplication operation changes, and thus its result $\boldsymbol{s}$.

Thus we define the new problem on the matrix-vector product as follows,

Definition 4 (Binay-Matrix-Vector Product Problem (MVPP)). *Let $\boldsymbol{z} \in \mathbb{F}_2^{n-k}$ of any weight, a vector $\boldsymbol{z}^* \in \mathbb{N}^{n-k}$. The matrix-vector product problem is to find $\boldsymbol{S} \in \mathbb{F}^{(n-k)\times(n-k)}$ such that $\boldsymbol{S}\boldsymbol{z}^T = \boldsymbol{z}^*$.*

We can find $\boldsymbol{z}^*$ for side-channel information with power consumption analysis and then with a chosen-ciphertext attack we find $\boldsymbol{S}$.

2.2 NIST PQC Standardization and Result

On July 5, 2022, NIST released the first four winning algorithms from a campaign launched in 2016 to standardize post-quantum cryptographic algorithms. These future standards will be default options for selecting post-quantum algorithms in the majority of security products. Provided that these post-quantum algorithms are also combined with proven classical algorithms through hybrid mechanisms. The main goal of the process started by NIST is to replace three standards that are considered the most vulnerable to quantum attacks, *i.e.*, FIPS 186-4[5] (for digital signatures), NIST SP 800-56A[6], and NIST SP 800-56B[7](both for keys

[5] `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`

[6] `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf`

[7] `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br1.pdf`

establishment in public-key cryptography). For the first round of this competition, 69 candidates met the minimum criteria and the requirements imposed by NIST. 26 out of 69 were announced on January 30, 2019, for moving to the second round. Of these, 17 are public-key encryption and/or key-establishment schemes and 9 are digital signature schemes. In July 2020, NIST started the third round of this process where only seven finalists were admitted (four PKE/KEM and three signature schemes). In addition to the finalists, eight alternate candidates were selected.

The first four algorithms selected are a key establishment algorithm named CRYSTALS-Kyber; and three digital signature algorithms named CRYSTALS-Dilithium, FALCON, and SPHINCS+. The first three of these algorithms are based on structured lattices; the last one, SPHINCS+ is a hash-based signature scheme. These four algorithms will therefore be used as the basis for writing U.S. federal standards. The scope of the NIST announcement is international with strong involvement of the cryptography research community, which will make the future US standards also used as de facto international industry standards. Beside the four winners, an extension of the NIST PQC standardization campaign (4th round) is planned for four key establishment algorithms: BIKE [1], HQC [24], *Classic McEliece* [2] (all three based on error-correcting codes), and SIKE [16] (isogeny graphs-based). *Classic McEliece* was the first selected finalist as a key encapsulation mechanism, while BIKE and HQC were alternative candidates. The latter two use special codes to reduce the size of the public key, which is considered the main drawback of code-based cryptosystems.

Classic McEliece is a code-based scheme using binary Goppa codes, the same codes that McEliece originally proposed in [23]. During Round 2 the scheme merged with NTS-KEM, which was using the same codes. The Classic McEliece scheme uses the dual of McEliece's scheme, as proposed by Niederreiter [25], and tightly turns this OW-CPA PKE into an IND-CCA2 KEM.

BIKE (Bit Flipping Key Encapsulation) is a key encapsulation mechanism (KEM) based on quasi-cyclic codes with moderate density parity check matrices. The code structure in BIKE is public and allows to reduce the size of the public key. Bit flipping corrects errors by repeatedly flipping the input bits that, given the secret moderate-density parity checks, seem most likely to be errors.

HQC (Hamming Quasi-Cyclic) uses error-correcting codes built from Reed-Muller and Reed-Solomon. The public key includes a random $\boldsymbol{h}$ and $\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y}$ are secretes and small Hamming weights. The ciphertext includes $\boldsymbol{u} = \boldsymbol{r}_1 + \boldsymbol{r}_2 \cdot \boldsymbol{h}$ and $\boldsymbol{v} = \mathbf{M} + \boldsymbol{s} \cdot \boldsymbol{r}_2 + \boldsymbol{e}$, where $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{e}$ are small Hamming weights and $\mathbf{M}$ is a message encoded using an error-correcting code. The receiver computes $\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{y} = \mathbf{M} + \boldsymbol{s} \cdot \boldsymbol{r}_2 + \boldsymbol{e} - \boldsymbol{u} \cdot \boldsymbol{y}$, which is close to $\mathbf{M}$ since $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{e}$ are small, and decodes the error-correcting code to recover $\mathbf{M}$.

SIKE (Supersingular Isogeny Key Encapsulation) is a key encapsulation mechanism based on the hard problem of pseudo-random walks in supersingular isogeny graphs. SIKE is a relatively new problem in the cryptographic arena and currently undergoing several attacks like its instantiation SIDH (Supersingular Isogeny Diffie–Hellman key exchange protocol) [8, 13, 20, 34, 35]. These are key

recovery attacks, reduces of the level security, side-channel attacks, and fault injection.

Some of these algorithms could therefore later join the same standardization process as the four algorithms already selected. The final objective of NIST is indeed to be able to standardize a varied range of algorithms to cover a majority of use cases. Most of these constructions based on error-correcting codes use matrix-vector products in the decryption, as in Niederreiter's scheme (Table 1).

The goal of our attack is to find the secrete matrix $\boldsymbol{Q}$. But first, let's assume that we already have the result of the product $\boldsymbol{Q}^{-1}\boldsymbol{z}$ in $\mathbb{N}$ using the same technique as in [10].

Table 1: Niederreiter PKE scheme

$\mathsf{KeyGen}(n,k,t) = (\mathsf{pk},\mathsf{sk})$
$\boldsymbol{H}$-parity-check of $\mathcal{C}$ that corrects t errors
An $n \times n$ permutation matrix $\boldsymbol{P}$
An $(n-k) \times (n-k)$ invertible matrix $\boldsymbol{Q}$
Compute $\boldsymbol{H}_{\text{pub}} = \boldsymbol{QHP}$
$\mathsf{pk} = (\boldsymbol{H}_{\text{pub}}, t)$
$\mathsf{sk} = (\boldsymbol{Q}, \boldsymbol{H}, \boldsymbol{P})$

$\mathsf{Encrypt}(\boldsymbol{m},\mathsf{pk}) = \boldsymbol{z}$
Encode $\boldsymbol{m} \rightarrow \boldsymbol{e}$ \\ error vector of $\mathsf{wt}(\boldsymbol{e}) = t$
$\boldsymbol{z} = \boldsymbol{H}_{\text{pub}}\boldsymbol{e}$

$\mathsf{Decrypt}(\boldsymbol{z},\mathsf{sk}) = \boldsymbol{m}$
Compute $\boldsymbol{z}^* = \boldsymbol{Q}^{-1}\boldsymbol{z} = \boldsymbol{S}\boldsymbol{z}$ \\ target of our attack
$\boldsymbol{z}^* = \boldsymbol{HPe}$
$\boldsymbol{e}^* = \mathcal{D}ecode(\boldsymbol{z}^*, \boldsymbol{H})$
Retrieve $\boldsymbol{m}$ from $\boldsymbol{P}^{-1}\boldsymbol{e}^*$

3 Our Attack and Result

3.1 On the Decryption of Niederreiter

Our attack on Niederreiter's decryption is now described. It consists in directly finding the secret matrix $\boldsymbol{Q}$. In the following, we note $\boldsymbol{S} = \boldsymbol{Q}^{-1} \in \mathbb{F}_2^{(n-k)\times(n-k)}$. We assume that, we can recover the result of the matrix-vector product $\boldsymbol{Sz}$ in $\mathbb{N}$ ($\boldsymbol{z}^* \in \mathbb{N}^{n-k}$) with side-channel information as in [10].

Profiled side-channel measurements. We performed side-channel measurements during the computation of the product $\boldsymbol{Sz}$ in the Niederreiter decryption implementation. The vector $\boldsymbol{z}^*$ is computed as the matrix-vector product $\boldsymbol{Sz} = \boldsymbol{z}^*$. We have recorded a single trace that will be sufficient to form the training set for the profiled attack. This trace is composed of n samples and stored as a row vector. We chose the ciphertexts $\boldsymbol{z}_i$ in $\mathbb{F}_2^{(n-k)}$ linearly independent as the inputs of the matrix-vector product algorithm. In addition, we stored a second trace, used as a test set when training the classifier. For both traces, we also stored the Hamming weights of the intermediate value resulting from the matrix-vector product $\boldsymbol{Sz}_i$.

After the traces acquisition, we performed an adequate preprocessing for reducing the dimension (eight dimensions since there are nine possible values for the Hamming weight of a byte) of the data by Linear Discriminant Analysis (LDA) to make it easier to handle by the classifier. We chose the random forest algorithm, used previously for side-channel analysis with good results [15], to recover the Hamming weight of $\boldsymbol{z}^*$.

We obtained the Hamming weight of the intermediate value of the product $\boldsymbol{Sz}_i$, we derived the entries of $\boldsymbol{z}^*$ in $\mathbb{N}$ with 98.65% accuracy. Indeed, the Hamming distance between two consecutive intermediate values is exactly the number of 1's found in the bitwise AND between the row of the matrix $\boldsymbol{S}$ and the byte of the ciphertext $\boldsymbol{z}$. Computing the value of the integer $\boldsymbol{z}^*$ entry is equivalent to counting those ones, which in turn is equivalent to summing the Hamming distances between consecutive intermediate values (the absolute value of the difference of their Hamming weights). In our implementation ($n = 6,960$, $k = 5,413$ and $t = 119$), the Hamming distance between two consecutive intermediate values is small and satisfies the condition in [10, Equation (3)] to recover the entries of $\boldsymbol{z}^*$ in $\mathbb{N}$ with good accuracy (82% for Hamming distance).

Course of the attack. We propose a chosen-ciphertext attack that essentially consists of 4 steps:

Step 1. We choose the ciphertexts or vectors $\boldsymbol{z}_i$ in $\mathbb{F}_2^{(n-k)}$ linearly independent. We therefore define a matrix $\boldsymbol{Z} = (\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{n-k})$ which is invertible.

Step 2. For each vector $\boldsymbol{z}_i$ thanks to the side-channel attack, we have the vector $\boldsymbol{z}_i^* = \boldsymbol{Sz}_i$ in $\mathbb{N}$ (in reality we get the Hamming weight for each component). This gives us a new matrix $\boldsymbol{Z}^* = (\boldsymbol{z}_1^*, \cdots, \boldsymbol{z}_{n-k}^*)$.

Step 3. We solve a matrix system $\boldsymbol{SZ} = \boldsymbol{Z}^*$ with $\boldsymbol{S}$ the unknown matrix in $\mathbb{F}_2^{(n-k)\times(n-k)}$. Since $\boldsymbol{Z}$ is invertible, we multiply on the right-hand side by its inverse and we obtain

$$\boldsymbol{S} = \boldsymbol{Z}^* \boldsymbol{Z}^{-1}. \tag{1}$$

Then we just have to read the entries of the right matrix to get the values of $\boldsymbol{S}$ and thus the secret matrix $\boldsymbol{Q}$. A toy example is discribed in the Appendix A.

The attack as presented above allows to find the secret matrix directly. However, in Step 2, we can raise two questions:

1. Can we know if the matrix $\boldsymbol{Z}^*$ is not correct?

2. If so, how can we correct the errors and find the secret matrix $\boldsymbol{S}$?

We will discuss question 2 in Section 3.2. For the first point, we assume that in Step 2 we obtain the matrix $\boldsymbol{Z}^{**}$ instead of $\boldsymbol{Z}^{*}$. So we have

$$\boldsymbol{S}' = \boldsymbol{Z}^{**}\boldsymbol{Z}^{-1} \tag{2}$$

and

$$\boldsymbol{Z}^{**} = \boldsymbol{Z}^{*} + \boldsymbol{E}_r \tag{3}$$

where $\boldsymbol{E}_r$ is an error matrix.

How to distinguish $\boldsymbol{S}$ from $\boldsymbol{S}'$?
We know that $\boldsymbol{S}$ is an invertible matrix in $\mathbb{F}_2$. Thus it's enough to look directly at its coefficients and compute its determinant.

We have shown that our attack allows us to directly find the secret matrix $\boldsymbol{S}$ in the case where there is no error in Step 2. Otherwise, we know how to detect it. We have two levels of optimization of this attack either minimize the risk of errors when recovering the matrix $\boldsymbol{Z}^{*}$ or reduce its coefficients modulo an integer number to speed up the computations. We can choose judiciously the $\boldsymbol{z}_i$ at step 1, for example, taking $\boldsymbol{z}_i$ of low Hamming weights allows having regular words with a "1" for each $\boldsymbol{z}_i$. This can considerably reduce the risk of errors during the acquisition of the traces. Moreover, in this case, we would have $\boldsymbol{Z} = \boldsymbol{I}_{n-k}$ and we obtain the secrete matrix $\boldsymbol{S}$ directly without computing $\boldsymbol{Z}^{-1}$. We can also suppose that the victim does not accept to decrypt $n-k$ ciphertexts for example, but with the choice of ciphertexts with low regular weights we avoid this problem.

We will now see how to correct the errors in the matrix $\boldsymbol{Z}^{*}$ and find the correct matrix $\boldsymbol{S}$.

3.2 Error Correction

In this section, we will provide an answer to question 2 and show that we can indeed find the matrix $\boldsymbol{S}$ in some cases. We consider the case where we have $\boldsymbol{E}_r$ in the matrix $\boldsymbol{Z}^{*}$ at Step 2, equations (2) and (3). We consider two assumptions *h1* et *h2* about $\boldsymbol{E}_r$:

1. The matrix $\boldsymbol{E}_r$ has coefficients 0 or 1, (*h1*).
2. The matrix $\boldsymbol{E}_r$ has, at most, a 1 on each row, (*h2*).

These two assumptions are not restrictive, we will see that we can deduce the general case and we assume that the error can be controlled to some extent, i.e., $\boldsymbol{Z}^{**}$ does not differ "too much" from $\boldsymbol{Z}^{*}$.

According to the above assumptions, there exist two finite sets I and J such that:

$$\boldsymbol{E}_r = \sum_{(i,j)\in I\times J} \boldsymbol{E}_{i,j} \tag{4}$$

with $\boldsymbol{E}_{i,j}$ the square matrix of order $n-k$ where all coefficients are zero, except the one of row i and column j which is 1.

We will need the following lemma:

Lemma 1. *Let $1 \leq a, b \leq n$. Let $\boldsymbol{M} = (m_{i,j})$ be a square matrix of order $n-k$. then :*

$$\boldsymbol{E}_{a,b}\boldsymbol{M} = \begin{pmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ m_{b,1} & \cdots & m_{b,n-k} \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{pmatrix} \leftarrow a\text{-th row}$$

In other words: $[\boldsymbol{E}_{a,b}\boldsymbol{M}]_{i,j} = \begin{cases} 0 & \text{if } i \neq a \\ m_{b,j} & \text{if } i = a \end{cases}$

To find $\boldsymbol{S}$ despite the error in $\boldsymbol{Z}^*$. We have

$$\boldsymbol{S}\boldsymbol{Z} = \boldsymbol{Z}^* + \boldsymbol{E}_r = \boldsymbol{Z}^* + \sum_{(i,j)\in I\times J} \boldsymbol{E}_{i,j}$$

Hence

$$\boldsymbol{S} = \boldsymbol{Z}^*\boldsymbol{Z}^{-1} + \boldsymbol{E}_r\boldsymbol{Z}^{-1} = \boldsymbol{Z}^*\boldsymbol{Z}^{-1} + \sum_{(i,j)\in I\times J} (\boldsymbol{E}_{i,j}\boldsymbol{Z}^{-1})$$

From the above we deduce the following theorem:

Theorem 1. *For any $i \in [1, n-k]$, there exists $j \in [1, n-k]$ and $\varepsilon \in \{0, 1\}$ such that $\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] - \varepsilon\boldsymbol{Z}^{-1}[j]$ is binary and $\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] - \varepsilon\boldsymbol{Z}^{-1}[j] = \boldsymbol{S}[i]$.*

Proof. Let us suppose $|I \times J| = 1$, let $I \times J = (a, b)$. Then we have

$$\boldsymbol{S} = \boldsymbol{Z}^*\boldsymbol{Z}^{-1} + \boldsymbol{E}_{a,b}\boldsymbol{Z}^{-1}.$$

According to Lemma 1, only the line a of $\boldsymbol{E}_{a,b}\boldsymbol{Z}^{-1}$ is nonzero.

We deduce that for all $i \neq a$,

$$\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] = \boldsymbol{S}[i].$$

It is ,therefore, sufficient to take $\varepsilon = 0$ and any j.

According to Lemma 1,

$$\boldsymbol{E}_{a,b}\boldsymbol{Z}^{-1}[a] = \boldsymbol{Z}^{-1}[b]$$

and so it suffices to take $\varepsilon = 1$ and $j = b$.

Let $(a, b) \in I \times J$, $(c, d) \in (I \times J) \setminus (a, b)$ and if $|I \times J| \geq 2$, the hypothesis *(h2)* implies that $c \neq a$.

According to Lemma 1, we have

$$\boldsymbol{E}_{c,d}\boldsymbol{Z}^{-1}[a] = 0.$$

thus

$$\boldsymbol{Z}^*\boldsymbol{Z}^{-1}[a] + \sum_{(i,j)\in I\times J} (\boldsymbol{E}_{i,j}\boldsymbol{Z}^{-1}[a]) = \boldsymbol{Z}^*\boldsymbol{Z}^{-1}[a] + \boldsymbol{E}_{a,b}\boldsymbol{Z}^{-1}[a].$$

We thus have, on each line, at most one contribution. We deduce then

1. if $i \notin I$, then $\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] = \boldsymbol{S}[i]$, it is enough to take $\varepsilon = 0$ and any j.
2. if $i \in I$, then there exists j such that $(i,j) \in I \times J$, we have $\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] = \boldsymbol{S}[i] + \boldsymbol{Z}^{-1}[j]$, so it is enough to take the j given previously and $\varepsilon = 1$.

We deduce that in some cases, it is possible to find S by the following algorithm 1.

Algorithm 1 Finding $\boldsymbol{S}$ with errors in $\boldsymbol{Z}^*$

1: We assume that we can determine the erroneous line(s) of $\boldsymbol{S}$ (for example, $\boldsymbol{S}$ is not binary).
2: Let $\boldsymbol{L}_i$ be an erroneous row of $\boldsymbol{S}$, we subtract from $\boldsymbol{L}_i$ the rows of the matrix $\boldsymbol{Z}^{-1}$ and keep those that are binary.
3: We thus obtain a list of possible candidates for $\boldsymbol{S}$ and for each candidate matrix, we compute its determinant to check its invertibility in $\mathbb{F}_2$.
4: In particular, if this list is reduced to one element, we obtain $\boldsymbol{S}$.

We will now lighten the assumptions by deleting (*h2*). According to Theorem 1, we can deduce the general case of our approach.

Corollary 1. *Let $i \in [1, n-k]$, let r_i be the Hamming weight of the vector $\boldsymbol{E}_r[i]$, there exists a sequence $(j_k)_{1\leq k\leq r_i}$ of distinct pairwise elements and a sequence $(\varepsilon_k)_{1\leq k\leq r_i}$ such that*

$$\boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] - \sum \boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] - \sum_{k=1}^{r_i} \varepsilon_k \boldsymbol{Z}^{-1}[j_k] \text{ is binary and } \boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[i] - \sum_{k=1}^{r_i} \varepsilon_k \boldsymbol{Z}^{-1}[j_k] = \boldsymbol{S}[i].$$

Proof. We use the same notations as before. We suppose that $|I \times J| \geq 2$ (if not, we are in the previous case). Let $(a,b) \in I \times J$.

We try to count the couples (a,t) with t in J. There are as many as the amount of "1" on the row $Er[a]$, in other words, there are r_a couples (a,t).

Let $G_a = \{\,(a,t) \mid t \in J\} = \{(a,j_1),(a,j_2),\cdots,(a,j_{r_a})\}$.

By Lemma 1 , for all $c \neq a$ and $d \in J$, the a-th row of $\boldsymbol{E}_{c,d}\boldsymbol{Z}^{-1}$ is zero. We deduce that

$$\boldsymbol{S}[a] = \boldsymbol{Z}^{**}\boldsymbol{Z}^{-1}[a] = \boldsymbol{Z}^*\boldsymbol{Z}^{-1}[a] + \boldsymbol{E}_r\boldsymbol{Z}^{-1}[a] = \boldsymbol{Z}^*\boldsymbol{Z}^{-1}[a] + \sum_{(i,j)\in I\times J} (\boldsymbol{E}_{i,j}\boldsymbol{Z}^{-1})[a]$$

$$= \boldsymbol{Z}^* \boldsymbol{Z}^{-1}[a] + \sum_{k=1}^{r_a} \boldsymbol{Z}^{-1}[j_k].$$

Table 2: McEliece PKE scheme

$\mathsf{KeyGen}(n, k, t) = (\mathsf{pk}, \mathsf{sk})$
$\boldsymbol{G}$-generator matrix of $\mathcal{C}$ that corrects t errors
An $n \times n$ permutation matrix $\boldsymbol{P}$
An $k \times k$ invertible matrix $\boldsymbol{S}$
Compute $\boldsymbol{G}_{\text{pub}} = \boldsymbol{SGP}$
$\mathsf{pk} = (\boldsymbol{G}_{\text{pub}}, t)$
$\mathsf{sk} = (\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P})$

$\mathsf{Encrypt}(\boldsymbol{m}, \mathsf{pk}) = \boldsymbol{z}$
Encode $\boldsymbol{m} \to \boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}}$
$\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$ \\ $\boldsymbol{e}$ is an error vector of $\mathsf{wt}(\boldsymbol{e}) = t$

$\mathsf{Decrypt}(\boldsymbol{z}, \mathsf{sk}) = \boldsymbol{m}$
Compute $\boldsymbol{z}^* = \boldsymbol{z}\boldsymbol{P}^{-1}$
$\boldsymbol{z}^* = \boldsymbol{m}\boldsymbol{S}\boldsymbol{G} + \boldsymbol{e}\boldsymbol{P}^{-1}$
$\boldsymbol{m}^* = \mathcal{D}ecode(\boldsymbol{z}^*, \boldsymbol{G})$
Retrieve $\boldsymbol{m}$ from $\boldsymbol{m}^*\boldsymbol{S}^{-1}$

Thus we have the following result :

1. If $i \notin I$, then the sequence (ε_k) is null and it is enough to take any (j_k) and two by two distinct.
2. If $i \in I$, considering the set G_i defined above and according to what precedes, it is enough to take the sequence (ε_k) constant equal to 1 and the (j_k) given by G_i.

In the case where the matrix $\boldsymbol{E}_r$ has negative coefficients or is not binary, we can adapt Theorem 1. It is sufficient to allow the sequence ε_k to take the value -1 and to be able to subtract (or add) the same row several times.

We notice that, unlike in the previous case, we cannot give an algorithm to determine a list of possible candidates for the matrix $\boldsymbol{S}$. A similar approach as the one presented above would be too expensive. We do not know, a priori, the number of rows to remove from each erroneous row. Although correcting the error in the matrix $\boldsymbol{Z}^*$ at Step 2 is theoretically possible, it may be difficult in practice.

3.3 Comparison to other Attacks

Recall that the goal of our attack is to find the secret matrix $\boldsymbol{Q}$ in the Niederreiter scheme. It has been shown in various previous works that it is possible to obtain secret information about the decryption in the McEliece scheme (Table 2). Falko Strenzke proposed in [30, 31] two attacks on each of the two calls of the Extended Euclidean Algorithm (EEA) in McEliece decoding with the parameters $n = 2,048$ and $t = 50$. This vulnerability in the Patterson algorithm [26] in the error correction phase allows an attacker to gather information about the secret $n \times n$ permutation matrix $\boldsymbol{P}$ through a timing side channel.

The first attack [30] targets the second call of the EEA in the Patterson algorithm to determine the polynomials forming the error locator polynomial $\sigma(x)$. The polynomial of $deg(\sigma(x)) \leq t$ consists of two polynomials $a(x)$ and $b(x)$ whose degrees have a direct impact on the number of iterations of the EEA. This variation in the number of iterations implies a difference in the execution times and makes possible a timing attack. The attacker creates ciphertext (chosen-ciphertext attack) using random error vectors with Hamming weight $\mathsf{wt} = 4$ and then lets the decryption routine decrypt the ciphertext. It evaluates whether zero or one iteration occurred in the EEA. If an iteration has occurred, $deg(b(x)) = 1$, nothing is done, and if there are no iterations, $deg(b(x)) = 0$, the error vector is added as a new row of a matrix over $\mathbb{F}_2$ having n columns. Each time a row is added, a Gauss-Jordan elimination is performed and the rank is determined. Once the maximum rank is reached (here 2,036), the attack is completed with 7,848,229 ciphertexts. However, such an approach only recovers the secret permutation matrix $\boldsymbol{P}$ when the Hamming weight of error $\boldsymbol{e}$ is small (2 or 4).

The second attack [31] is based on the vulnerabilities that are present in the inversion of the error syndrome through the extended euclidean algorithm (first call in the Patterson algorithm). Strenzke showed the existence of a timing side channel vulnerability in the syndrome inversion that allows the attacker to gain knowledge of the zero-element of the secret support. It is based on the analysis of the key equation to deduce the relations between the degrees of the polynomials involved in it. As in the first attack, this approach only works for Hamming weights of 2, 4, or 6 of the error vector $\boldsymbol{e}$ to recover the secret permutation matrix $\boldsymbol{P}$. Despite the improvements in [6], the main problem with these two previous attacks is the number of cases (depending on the Hamming weight of $\boldsymbol{e}$) that can be exploited to find the secret. In our attack, we have no constraints on the Hamming weights of the ciphertext (or error vector $\boldsymbol{e}$) to find the secret matrix $\boldsymbol{Q}$ in the case that we correctly construct the matrix $\boldsymbol{Z}^*$ with a random forest. Moreover, we attack the least complex step of Niederreiter decoding (first step). The Niederreiter PKE is slightly different from the McEliece PKE (Table 1 and Table 2). However, here too, an error vector is chosen during the encryption and decryption features, and since these features are the prerequisites of our attack, it is also applicable to McEliece's PKE. Unlike the attack of Strenzke in [30] we only need 2,048 ciphertexts instead of 7,848,229 to find the secret $n \times n$

Table 3: Attacks to recover the secret matrix $\boldsymbol{P}$

Attack	Hamming weight	Number of ciphertexts	Target
Timing attacks of Strenzke	2, 4 or 6	7,848,229 in [30]	EEA in Patterson algorithm
Our attack against MVPP	no constraints	2,048	First step in McEliece's decryption

permutation matrix $\boldsymbol{P}$. The Table 3 give more details on the comparison with Strenzke's attacks to find the secret matrix $\boldsymbol{P}$.

4 Conclusion

This article presents a key-recovery attack against the Matrix-Vector Product Problem, which is a new formalism that we introduce in based-code cryptography. We have also shown that with a side-channel attack on this operation, we can recover secret information on based-code cryptosystems without solving the hardness of the binary syndrome decoding problem. In addition to the side-channel information, we performed a chosen-ciphertext attack, which, with careful choice of the ciphertexts, can find the secret matrix without errors. When noise (errors) is present during the attack, we have shown that in some cases it is possible to find the secret matrix. Our attack can be applied to code-based cryptosystems with matrix-vector product operations.

References

[1] C Aguilar Melchor, N Aragon, P Barreto, S Bettaieb, L Bidoux, O Blazy, J. Deneuville, P Gaborit, S Ghosh, S Gueron, et al. "BIKE: Bit Flipping Key Encapsulation". In: *NIST Post-Quantum Cryptography Standardization Project (Round 3)* (2020).

[2] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, et al. "Classic McEliece". In: *NIST Post-Quantum Cryptography Standardization Project (Round 3), https://classic. mceliece. org* (2020).

[3] A. Becker, A. Joux, A. May, and A. Meurer. "Decoding Random Binary Linear Codes in $2^{n/20}$: How 1 + 1 = 0 Improves Information Set Decoding". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Apr. 2012, pp. 520–536.

[4] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[5] D. J. Bernstein and T. Lange. "Post-quantum cryptography". In: *Nature* 549.7671 (2017), pp. 188–194.

[6] D. Bucerzan, P.-L. Cayrel, V. Dragoi, and T. Richmond. "Improved timing attacks against the secret permutation in the McEliece PKC". In: *International Journal of Computers Communications & Control* 12.1 (2016), pp. 7–25.

[7] J. A. Buchmann, D. Butin, F. Göpfert, and A. Petzoldt. "Post-quantum cryptography: state of the art". In: *The new codebreakers* (2016), pp. 88–108.

[8] W. Castryck and T. Decru. *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. `https://eprint.iacr.org/2022/975`. 2022. URL: `https://eprint.iacr.org/2022/975`.

[9] P.-L. Cayrel, B. Colombier, V.-F. Drăgoi, A. Menu, and L. Bossuet. "Message-recovery laser fault injection attack on the classic McEliece cryptosystem". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021, pp. 438–467.

[10] B. Colombier, V.-F. Drăgoi, P.-L. Cayrel, and V. Grosso. "Profiled Side-channel Attack on Cryptosystems based on the Binary Syndrome Decoding Problem". In: *IEEE Transactions on Information Forensics and Security* (2022), pp. 3407–3420. DOI: `10.1109/TIFS.2022.3198277`.

[11] J. Feldman. "Decoding error-correcting codes via linear programming". PhD thesis. Massachusetts Institute of Technology, Cambridge, MA USA, 2003.

[12] J. Feldman, M. J. Wainwright, and D. R. Karger. "Using linear programming to Decode Binary linear codes". In: *IEEE Transactions on Information Theory* 51.3 (2005), pp. 954–972.

[13] T. B. Fouotsa and C. Petit. "A new adaptive attack on SIDH". In: *Cryptographers' Track at the RSA Conference*. Springer. 2022, pp. 322–344.

[14] L. Gyongyosi and S. Imre. "A survey on quantum computing technology". In: *Computer Science Review* 31 (2019), pp. 51–71.

[15] B. Hettwer, S. Gehrer, and T. Güneysu. "Applications of machine learning techniques in side-channel attacks: a survey". In: *Journal of Cryptographic Engineering* 10.2 (2020), pp. 135–162.

[16] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, et al. "SIKE-Supersingular isogeny key encapsulation". In: *NIST Round* 3 (2020).

[17] M. V. Larsen, X. Guo, C. R. Breum, J. S. Neergaard-Nielsen, and U. L. Andersen. "Deterministic multi-mode gates on a scalable photonic quantum computing platform". In: *Nature Physics* 17.9 (2021), pp. 1018–1023.

[18] P. J. Lee and E. F. Brickell. "An Observation on the Security of McEliece's Public-Key Cryptosystem". In: *Workshop on the Theory and Application*

of of Cryptographic Techniques. Ed. by C. G. Günther. Vol. 330. Lecture Notes in Computer Science. Davos, Switzerland: Springer, May 1988, pp. 275–280.

[19] J. S. Leon. "A probabilistic algorithm for computing minimum weights of large error-correcting codes". In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 1354–1359.

[20] L. Maino and C. Martindale. "An attack on SIDH with arbitrary starting curve". In: *Cryptology ePrint Archive* (2022).

[21] A. May, A. Meurer, and E. Thomae. "Decoding Random Linear Codes in $\mathcal{O}(2^{0.054n})$". In: *International Conference on the Theory and Application of Cryptology and Information Securitys*. Ed. by D. H. Lee and X. Wang. Vol. 7073. Lecture Notes in Computer Science. Seoul, South Korea: Springer, Dec. 2011, pp. 107–124.

[22] A. May and I. Ozerov. "On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer, Apr. 2015, pp. 203–228.

[23] R. J. McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[24] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. Bos, J.-C. Deneuville, A. Dion, P. Gaborit, J. Lacan, et al. *Hamming Quasi-Cyclic (HQC). NIST Post-Quantum Cryptography Standardization Project (Round 3), 2020*.

[25] H. Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory". In: *Prob. Contr. Inform. Theory* 15.2 (1986), pp. 157–166.

[26] N. Patterson. "The algebraic decoding of Goppa codes". In: *IEEE Transactions on Information Theory* 21.2 (1975), pp. 203–207.

[27] E. Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9.

[28] P. W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[29] J. Stern. "A method for finding codewords of small weight". In: *International Colloquium on Coding Theory and Applications*. Ed. by G. D. Cohen and J. Wolfmann. Vol. 388. Lecture Notes in Computer Science. Toulon, France: Springer, Nov. 1988, pp. 106–113.

[30] F. Strenzke. "A timing attack against the secret permutation in the McEliece PKC". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2010, pp. 95–107.

[31] F. Strenzke. "Timing attacks against the syndrome inversion in code-based cryptosystems". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2013, pp. 217–230.

[32] S Takeda and A Furusawa. "Toward large-scale fault-tolerant universal photonic quantum computing". In: *APL Photonics* 4.6 (2019), p. 060902.

[33] A. Tanatmis, S. Ruzika, H. W. Hamacher, M. Punekar, F. Kienle, and N. Wehn. "A separation algorithm for improved LP-decoding of linear block codes". In: *IEEE Transactions on Information Theory* 56.7 (2010), pp. 3277–3289.

[34] É. Tasso, L. De Feo, N. El Mrabet, and S. Pontié. "Resistance of isogeny-based cryptographic implementations to a fault attack". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2021, pp. 255–276.

[35] F. Zhang, B. Yang, X. Dong, S. Guilley, Z. Liu, W. He, F. Zhang, and K. Ren. "Side-channel analysis and countermeasure design on ARM-based quantum-resistant SIKE". In: *IEEE Transactions on Computers* 69.11 (2020), pp. 1681–1693.

A Simple Version of Our Attack

Case 1: no errors in $\boldsymbol{Z}^*$
Let the matrices $\boldsymbol{Z}$ invertible in $\mathbb{F}_2$ and $\boldsymbol{Z}^*$ in $\mathbb{N}$ respectively constructed in Step 1 and recovered in Step 2 of our attack. One chooses here $n-k=3$, we have for instance

$$\boldsymbol{Z} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

its inverse

$$\boldsymbol{Z}^{-1} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

and

$$\boldsymbol{Z}^* = \begin{pmatrix} 2 & 2 & 1 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \end{pmatrix}.$$

From the equation 1, $\boldsymbol{S} = \boldsymbol{Z}^*\boldsymbol{Z}^{-1}$, we find

$$\boldsymbol{S} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Case 2: few errors in $\boldsymbol{Z}^*$
We keep the same $\boldsymbol{Z}$, its inverse and $\boldsymbol{Z}^*$ matrices in **case 1**.
According to equation 3, $\boldsymbol{Z}^{**} = \boldsymbol{Z}^* + \boldsymbol{E}_r$, we have

$$\boldsymbol{E}_r = \begin{pmatrix} 1\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0 \end{pmatrix}.$$

By performing the same operation as before, we obtain the following $\boldsymbol{S}'$ matrix according to equation 2

$$\boldsymbol{S}' = \begin{pmatrix} 0\,1\,2 \\ 0\,1\,0 \\ 1\,1\,0 \end{pmatrix}.$$

This matrix $\boldsymbol{S}'$ is not binary, so we deduce that the first row contains a fault. We then apply the algorithm 1 of Section 3.2 to determine the list of possible candidates for the matrix $\boldsymbol{S}$

$$\begin{aligned} \boldsymbol{S}'[1] - \boldsymbol{Z}^{-1}[1] &= [1\ \ 0\ \ 1] \\ \boldsymbol{S}'[1] - \boldsymbol{Z}^{-1}[2] &= [-1\ \ 2\ \ 2] \\ \boldsymbol{S}'[1] - \boldsymbol{Z}^{-1}[3] &= [-1\ \ 1\ \ 3]. \end{aligned}$$

Only the first case gives a binary vector, we deduce that $\boldsymbol{S}'[1] = \boldsymbol{S}[1] + \boldsymbol{Z}^{-1}[1]$ and we have directly $\boldsymbol{S}$.

Case 3: errors in $\boldsymbol{Z}^*$
We consider the error matrix

$$\boldsymbol{E}_r = \begin{pmatrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0 \end{pmatrix}.$$

We then obtain the following matrix

$$\boldsymbol{S}' = \begin{pmatrix} 1\,0\,1 \\ 1\,0\,0 \\ 1\,1\,0 \end{pmatrix}.$$

Here the matrix is binary, moreover its determinant $det(\boldsymbol{S}') = 1$. In this case, we cannot detect that the matrix is not correct.