Correlation power analysis of LESS and CROSS

Maciej Czuprynko[0009−0007−2433−5285], Anisha Mukherjee[0009−0002−5565−5744], and Sujoy Sinha Roy[0000−0002−9805−5389]

ISEC, Graz University of Technology, Austria
{maciej.czuprynko,anisha.mukherjee,sujoy.sinharoy}@tugraz.at

Abstract. This paper presents a side-channel attack targeting the LESS and CROSS post-quantum digital signature schemes, resulting in full key recovery for both. These schemes have advanced to the second round of NIST's call for additional signatures. By leveraging correlation power analysis and horizontal attacks, we are able to recover the secret key by observing the power consumption during the multiplication of an ephemeral secret vector with a public matrix. The attack path is enabled by the presence of a direct link between the secret key elements and the ephemeral secret, given correct responses. This attack targets version 1.2 of both schemes. In both settings we can recover the secret key in a single trace for the NIST's security level I parameter set. Additionally, we propose improvements to the existing horizontal attack on CROSS, reducing the required rounds that need to be observed by an order of magnitude for the same parameter sets.

Keywords: Side-channel attacks · Post-quantum cryptography · CROSS · LESS

1 Introduction

Current cryptographic primitives predominantly relying on the hardness of discrete logarithm and integer factorization problems face the threat to be rendered broken by Shor's algorithm [22,23], capable of factoring large integers—previously considered super-polynomially hard for classical computers—in polynomial time using quantum computing. In recent times, the potential emergence of large-scale quantum computers has driven extensive research in post-quantum cryptography. In 2016, the National Institute of Standards and Technology (NIST) initiated a process to standardize post-quantum Key Encapsulation Mechanisms (KEMs) and digital signature schemes. Along with theoretical security, NIST also considers practical aspects such as performance and resilience against side-channel attacks (SCA) in its evaluation framework. Until now, three digital signature schemes have been chosen for standardization: Dilithium [13], Falcon and SPHINCS+ [5]. Since two out of the three candidates (Dilithium and Falcon) rely on lattice-based hardness assumptions, in order to improve diversity, NIST announced a separate call for additional digital signatures in 2023.

In response to NIST's call for additional post-quantum digital signatures, many different constructions have been proposed which are based on a diverse

range of mathematical hard problems. NIST has recently [18] announced the list of signatures that have successfully advanced to the second round of evaluation, which includes two novel code-based primitives, LESS [2, 6] and CROSS [3, 4]. LESS is constructed from a zero-knowledge identification protocol based on the hardness of the Linear Code Equivalence problem or LEP. Unlike most code-based cryptographic primitives based on the Syndrome Decoding Problem (SDP), LESS relies on the difficulty of determining the linear isometry between two codes. This allows LESS to choose smaller scheme parameters that minimizes key sizes while maintaining security and allows a well-balanced trade-off between the key/signature sizes and the efficiency of the signing/verification processes. On the other hand, the zero-knowledge identification protocol from which CROSS is constructed is based on the hardness assumption of Restricted-SDP or R-SDP.

An important consideration when implementing cryptographic schemes are side-channel attacks. These attacks exploit implementation-related leakage of the secret key using, for instance, timing variations or the power consumption. In the context of code-based cryptography, an extensive amount of work has been done for the KEMs in the 4th round of NIST [1, 10, 20]. In contrast, code-based signature schemes are relatively new and hence, have received less scrutiny. To the best of our knowledge, the fault injection attack presented in [17] is the only attack against LESS. The recent concurrent work by [21] is the only passive side-channel attack targeting CROSS.

1.1 Our contribution

We summarize our contributions as follows:

Correlation Power Analysis on LESS-1.2: In this work, we introduce the first passive side-channel attack on version-1.2 of LESS [6]. This attack leverages correlation power analysis (CPA) by targeting the scaling of a set of public column vectors $\mathbf{g}_i$ by a set of ephemeral secret elements $\tilde{d}_i$ over the prime field $\mathbb{F}_q$. This is made possible by the fact that $\tilde{d}_i$ is directly related to a single element of the persistent secret key, $d_i \in \mathbb{F}_q$, through a correct signature. As such, a guess on the element d_i can be turned into a guess on a single element of the targeted ephemeral element $\tilde{d}_i$. Furthermore, we use the fact that a column scaling consists of multiple operations involving a single ephemeral secret to fully utilize a single trace. For certain parameter sets, the CPA attack leads to a full secret key recovery in a single trace.

A similar attack has been performed on the KEM Classical McEliece [10]. There, the authors describe a horizontal side-channel attack that targets a matrix-vector multiplication involving a secret vector and a public matrix. This operation is similar to our case because it can be seen as a column (or row) scaling followed by an accumulation. However, the operations in their case are performed over the field $\mathbb{F}_2$. Additionally, the targeted secret is ephemeral, meaning that the attack has to be performed in a single trace. In our case, we do not have this issue, even though we also target an ephemeral secret.

Extending our attack to CROSS: We showcase that our attack is applicable to another code based signature schemes CROSS. The fundamental side-channel attack stays the same. However, the secret key recovery technique from the ephemeral secret elements changes due to the differences in the scheme design. Furthermore, we show how our CPA attack is an extension to the horizontal attack introduced in the concurrent work [21]. In doing so, we report improvements to the existing attack methodology. In particular, we show how we are able to fully utilize the trace and hence, reduce the required number of signatures that have to be observed. Finally, we also present an alternative leakage model. Each of these improvements lead to a reduction in the required observed rounds by an order of magnitude.

2 Background

In this section we first introduce the notations used throughout this paper. We begin by covering the general concepts of code-based cryptography and then we present the scheme specific notions used in LESS and CROSS, respectively. For more details about the schemes as well as code based cryptography, we invite the readers to go to [2] and [3] for LESS and CROSS, respectively.

2.1 Notations

LESS works with elements in the field $\mathbb{F}_q$, while in CROSS, we use the field $\mathbb{F}_p$. Matrices in theses fields are denoted by bold uppercase letter (e.g., $\mathbf{A}$), vectors by bold lowercase letter (e.g., $\mathbf{a}$) and scalars by lowercase letters (e.g., a). For a matrix $\mathbf{A}$, the vector with the subscript $\mathbf{a}_i$ represents either the i-th column or the i-th row, as specified. An element of $\mathbf{A}$ at the i-th row and j-th column is denoted as $a_{i,j}$. For a vector $\mathbf{a}$, the subscript a_i indicates the i-th element. In contexts involving multiple matrices of a similar type, we use the superscript notation $\mathbf{A}^{(i)}$ to indicate the i-th matrix from that collection. Similar notations apply for vectors and scalars.

Furthermore, we denote $\mathbf{A}_J$ the matrix formed by the columns of $\mathbf{A}$ that are indexed by the elements in a set J. In the particular case that $\mathbf{A}$ is a square matrix, we abuse this notation to indicate a square matrix formed by keeping only the elements $a_{i,j}$ of $\mathbf{A}$ that satisfy $i \in J$ and $j \in J$.

Moreover, we typically use the usual implicit multiplication between the aforementioned objects, unless explicitly denoted by "$\cdot$" for clarity. Finally, we denote the element wise multiplication of two vectors using the "$\star$" operator.

2.2 Linear codes

In this section, we introduce the general notions used in linear codes.

An $[n, k]$-linear code $\mathcal{C}$ over the field $\mathbb{F}_q$ is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$. This subspace can be represented by a full rank matrix $\mathbf{G} \in$

$\mathbb{F}_q^{k\times n}$, which is called a generator matrix, where its rows form a basis of the subspace. As such, the elements $\mathbf{c} \in \mathcal{C}$, called code-words, are generated by vectors $\mathbf{u} \in \mathbb{F}_q^k$ such that $\mathbf{c} = \mathbf{uG}$. Another way the linear code can be represented is through the kernel of the parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ where $\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{xH}^\top = \mathbf{0}\}$. However, those representations are not unique since, given an invertible matrix $\mathbf{S}$ and a generator matrix $\mathbf{G}$ of a linear code $\mathcal{C}$, the generator matrix obtained as $\mathbf{SG}$ defines the same code $\mathcal{C}$. This works similarly for the parity check matrix. To have a unique representation, the standard choice is to use the systematic form which is $\mathbf{G} = (\mathbf{I}_k|\mathbf{V})$ for a generator matrix, while the systematic form of the parity-check matrix is given by $\mathbf{H} = (\mathbf{W}|\mathbf{I}_{n-k})$.

2.3 Code-based hard problems underlying LESS and CROSS

The underlying problem in LESS focuses on finding a monomial transformation between two $[n, k]$-linear codes. Let M_n be the set of monomial transformation, then an element of this set can be represented as an $n \times n$ matrix $\mathbf{Q} \in \mathrm{M}_n$ such that $\mathbf{Q} = \mathbf{DP}$, where $\mathbf{D}$ is an $n \times n$ diagonal matrix and its diagonal $\mathbf{d}$ is a vector of non-zero elements in $\mathbb{F}_q^*$. Furthermore, $\mathbf{P}$ is a permutation matrix with the corresponding permutation π of the indices $\{0, \ldots, n-1\}$. As such, a monomial matrix is characterized by the value pair $(\mathbf{d}, \pi)$. We highlight that a right or a left multiplication by $\mathbf{P}$ corresponds to a permutation of the columns or rows, respectively. Moreover, it has the property that its inverse is equal to its transpose, i.e. $\mathbf{P}^{-1} = \mathbf{P}^\top$. As such, we can define the underlying (computational) problem in LESS as:

Definition 1 (Linear Equivalence Problem (LEP) [2]). *Let RREF stand for the process of turning a matrix into its row reduced echelon form. Then, given two (systematic) generator matrices $\mathbf{G}^{(0)}, \mathbf{G}^{(1)} \in \mathbb{F}_q^{k\times n}$, find $\mathbf{Q} \in M_n$ such that $\mathbf{G}^{(1)} = RREF(\mathbf{G}^{(0)}\mathbf{Q})$.*

Here, the row reduced echelon form is used to obtain a unique representation of the generator matrices. In most cases it is equivalent to the systematic form. However, it differs when the first k columns of a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ are not invertible. In those cases, there are some columns that are not part of the identity matrix in-between the identity matrix of the systematic form. Furthermore, since the generator matrix after the RREF defines the same linear code, the RREF can be seen as the application of a change of basis matrix $\mathbf{S}$ such that $\mathrm{RREF}(\mathbf{G}) = \mathbf{SG}$.

On the other hand, CROSS is based on a different hard problem which is defined as follows:

Definition 2 (Restricted Syndrome Decoding Problem (R-SDP) [3]). *Let $\mathbb{E}$ be the cyclic subgroup of $\mathbb{F}_p$ generated by an element $g \in \mathbb{F}_p^*$ of multiplicative order z, such that $\mathbb{E} = \{g^i : i \in \{0, \ldots, z-1\}\}$. Then, given a parity check matrix $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$ and a vector $\mathbf{s} \in \mathbb{F}_p^{n-k}$ (called a syndrome vector), find $\mathbf{e} \in \mathbb{E}^n$ such that $\mathbf{s} = \mathbf{eH}^\top$.*

We note that an element $\mathbf{e} \in \mathbb{E}^n$ can be represented by a vector $\eta \in \mathbb{F}_z^n$[1] such that $\mathbf{e} = (g^{\eta_0}, \ldots, g^{\eta_{n-1}})$, where we use the Greek letters (except π which is used exclusively for permutations) to denote a vector with elements in $\mathbb{F}_z$. A variation of this problem that leads to smaller signature sizes is the Restricted Syndrome Decoding Problem with subgroup $\mathbb{G}$ (R-SDP(G)). In this setting, instead of requiring to find any vector $\mathbf{e} \in \mathbb{E}^n$, or the corresponding vector $\eta \in \mathbb{F}_z^n$, the search space is further restricted so that $\eta \in \mathbb{G}$. In particular, CROSS defines the subgroup $\mathbb{G}$ as a $[m, n]$-linear code over $\mathbb{F}_z$. As such, to sample η, a vector $\zeta \in \mathbb{F}_z^m$ is first uniformly sampled from $\mathbb{F}_z^m$ and then it is multiplied with the generator matrix of the linear code, denoted $\mathbf{M}_G$.

2.4 Correlation power analysis

Correlation power analysis (CPA) based attacks are powerful side-channel attacks that exploit the dependency between secret values and the power consumption during computations that are performed on them. This type of attack was investigated in [7, 11, 16] and is based on the differential power analysis introduced in [14].

A CPA-based attack, as described in [15], targets an intermediate computation, denoted as a function f, of the form $r = f(v, s)$ which is performed in a cryptographic algorithm, where r is the result of the computation, v is a known non-constant value and s is the targeted secret value. The general principle behind CPA is that the power consumption depends on the operation that is performed. A model that is commonly used is the Hamming weight (HW) model, where it assumed that the power consumption is proportional to the number of set bits in the binary representation of the result r, which we denote as $\mathrm{HW}(r)$. This leakage can be used to recover the secret over multiple executions of the algorithm.

To perform a CPA attack, an attacker first observes multiple executions of the algorithm, measuring the power consumption of the targeted device to obtain a power trace for each execution. The attacker then computes the intermediate values for each execution and for each possible secret value. These intermediate values are converted into hypothesized power consumption based on a chosen model. The Pearson correlation coefficient is then computed between the measured traces (at the time-step of the intermediate computation) and the hypothesized leakage for each possible secret value. This step is repeated for all possible secret values. If the selected model is sufficiently correct and enough traces are collected, the secret that leads to the highest correlation coefficient is the correct one. Usually, to obtain an accurate result a large amount of traces need to be collected.

Horizontal attacks The presented CPA attack examines each time-step individually across multiple executions of the targeted algorithm. However, it is

[1] In the case of CROSS, the order z is always chosen to be a prime so that $\mathbb{F}_z$ is a field.

possible that multiple points within a single execution leak the secret. An example of this scenario is the repeated multiplication of a secret coefficient with a set of known values. In such cases, it can be beneficial to include multiple points from the same execution in the analysis. Attacks that utilize information from a single trace are referred to as horizontal attacks, in contrast to vertical attacks, which target the same point across multiple executions. This distinction was first proposed by the the authors of [9]. As such, a CPA-style horizontal attack is a CPA based attack where the points come from the same trace instead of being gathered across multiple executions of the scheme.

Public Data	System parameters $q, n, k \in \mathbb{N}, \mathbf{G}^{(0)} = (\mathbf{I}_k \vert \mathbf{V}), \mathbf{V} \in \mathbb{F}_q^{k \times (n-k)}$ and hash function, 'Hash'.
Private Key	$\mathbf{Q} \in \mathrm{M}_n$.
Public Key	$\mathbf{G}^{(1)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q})$.

PROVER		VERIFIER
$\tilde{\mathbf{Q}} \xleftarrow{\$} \mathrm{M}_n, \tilde{\mathbf{G}} \leftarrow \mathrm{RREF}(\mathbf{G}^{(0)}\tilde{\mathbf{Q}})$		
$\mathrm{cmt} \leftarrow \mathrm{Hash}(\tilde{\mathbf{G}})$	$\xrightarrow{\mathrm{cmt}}$	
	$\xleftarrow{\mathrm{ch}}$	$\mathrm{ch} \xleftarrow{\$} \{0,1\}$
$\mathrm{rsp} \leftarrow (1-\mathrm{ch})\tilde{\mathbf{Q}} + \mathrm{ch} \cdot \mathbf{Q}^{-1}\tilde{\mathbf{Q}}$	$\xrightarrow{\mathrm{rsp}}$	
		Accept if $\mathrm{Hash}(\mathrm{RREF}(\mathbf{G}^{(\mathrm{ch})} \cdot \mathrm{rsp})) = \mathrm{cmt}$

Fig. 1: Sigma protocol at the basis of LESS [6].

3 Correlation Power Analysis-based attack on LESS Sign

We begin with a simplified explanation of our attack technique on the identification scheme underlying LESS [2]. The scheme is recalled in Section 3.1. Our attack is based on a CPA-style horizontal attack which targets the multiplication of a public generator matrix $\mathbf{G}^{(0)} \in \mathbb{F}_q^{(n \times k)}$ in systematic form with an ephemeral secret monomial matrix $\tilde{\mathbf{Q}} \in \mathrm{M}_n$. We show in Section 3.2 that, using side-channel analysis, it is possible to recover the diagonal matrix $\tilde{\mathbf{D}}_J$, where $J = \{k, k+1, \ldots, n-1\}$, which forms the ephemeral monomial $\tilde{\mathbf{Q}} = \tilde{\mathbf{D}}\tilde{\mathbf{P}}$. Then, in Section 3.3, we present a method to convert the knowledge of $\tilde{\mathbf{D}}_J$ into partial knowledge of the secret $\mathbf{Q}$ given multiple responses of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$. Concretely, the part we are able to recover is $\mathbf{Q}_J$. Finally, in Section 3.4, we show that knowing $\mathbf{Q}_J$ is sufficient to find a monomial $\mathbf{Q}' \in \mathrm{M}_n$ such that $\mathbf{G}^{(1)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q}')$. We summarize the attack plan in Figure 2.

In our context, executing the attack on the full scheme follows the same fundamental steps as those used in the identification scheme. The key differences, particularly in the number of observed signatures required for LESS compared to

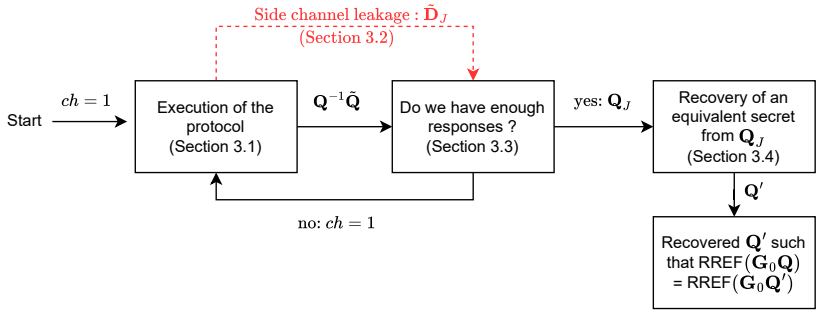


Fig. 2: Diagram representing the attack path

the identification protocol, are discussed in the subsequent Section 3.5. Finally, in Section 3.6, we show how to turn the horizontal attack into a CPA attack. Furthermore, we mention how to deal with some difficulties arsing during the practical attack.

3.1 LESS identification scheme

The LESS signature scheme is based on a sigma protocol, which is a three-step interactive protocol enabling one party (the prover) to demonstrate their identity to another party (the verifier). In this protocol, the prover holds a secret key, while the verifier has access to the corresponding public key. The prover authenticates themself by convincing the verifier of their knowledge of the private key. Concretely, the LESS identification scheme proceeds in the following steps, which are summarized in the Figure 1:

- **Key Generation:** The secret key is a monomial matrix $\mathbf{Q} \in \mathrm{M}_n$. The corresponding public key is formed by a generator matrix in systematic form $\mathbf{G}^{(0)} = (\mathbf{I}_k|\mathbf{V})$ and the generator matrix $\mathbf{G}^{(1)}$ computed as $\mathbf{G}^{(1)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q})$.
- **Commitment:** To prove the knowledge of $\mathbf{Q}$, the prover generates a random monomial matrix $\tilde{\mathbf{Q}} \in \mathrm{M}_n$, applies it on the public generator matrix and computes the RREF obtaining $\tilde{\mathbf{G}} = \mathrm{RREF}(\mathbf{G}^{(0)}\tilde{\mathbf{Q}})$. The prover then commits to it by sending the hash $\mathrm{cmt} = \mathrm{Hash}(\tilde{\mathbf{G}})$ to the verifier.
- **Challenge:** The verify sends a challenge $\mathrm{ch} \in \{0, 1\}$.
- **Response:** The prover responds with $\tilde{\mathbf{Q}}$ or $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ if the challenge is equal to 0 or 1 respectively.
- **Verification:** In this step, the verifier is able to compute $\tilde{\mathbf{G}}$ as $\mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q})$ or $\mathrm{RREF}(\mathbf{G}^{(1)}\mathbf{Q}^{-1}\tilde{\mathbf{Q}})$ depending if the challenge is equal to 0 or 1. Finally, they verify that $\mathrm{cmt} = \mathrm{Hash}(\tilde{\mathbf{G}})$.

To get an intuition on the attack, we notice that if a malicious verifier was able to know the ephemeral monomial $\tilde{\mathbf{Q}}$, they would be able to recover the secret key by sending the challenge equal to 1. Indeed, since they receives $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ they are able to compute the secret key as:

$$\mathbf{Q} = \tilde{\mathbf{Q}} \left(\mathbf{Q}^{-1} \tilde{\mathbf{Q}} \right)^{-1}$$

As such, the target of the attack is the recovery of $\tilde{\mathbf{Q}}$. In the next section, we show how we are able to exploit the leakage during the matrix multiplication $\mathbf{G}^{(0)}\tilde{\mathbf{Q}}$, which is implemented as shown in Algorithm 1, to recover some of the ephemeral monomial. Unfortunately, the part that is learned is not sufficient to recover the secret in the straight forward way discussed in this section. The recovery is thus more involved and is the topic of the rest of the section.

Algorithm 1 $\mathsf{GenMon_Mult}(\mathbf{G}, \mathbf{Q})$

Require: Matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, monomial matrix $\mathbf{Q} \in \mathrm{M}_n$ represented by $(\mathbf{d}, \pi)$
Ensure: $\mathbf{A} = \mathbf{GQ}$
  for $i = 0$ to $n - 1$ **do**
    $\mathbf{t} \leftarrow d_i \cdot \mathbf{g}_i$ ▷ scaling of the i-th column of $\mathbf{G}$
    $\mathbf{a}_{\pi(i)} \leftarrow \mathbf{t}$ ▷ setting the $\pi(i)$-th column of $\mathbf{A}$, equivalent to applying the permutation π
  end for
  return A

3.2 Partial recovery of ephemeral $\tilde{\mathbf{Q}}$ through side-channel attack

In this section we demonstrate how we are able to partially recover the ephemeral secret monomial $\tilde{\mathbf{Q}} = \tilde{\mathbf{D}}\tilde{\mathbf{P}}$.

Recovering elements of ephemeral $\tilde{\mathbf{D}}_J$: To demonstrate the recovery of the ephemeral secret $\tilde{\mathbf{D}}$, we first note from the Algorithm 1 that the matrix multiplication $\mathbf{G}^{(0)}\tilde{\mathbf{Q}}$ can be split into the scaling of the columns of $\mathbf{G}^{(0)}$, represented by the application of the diagonal matrix $\tilde{\mathbf{D}}$, followed by a column permutation using the matrix $\tilde{\mathbf{P}}$, where $\tilde{\mathbf{Q}} = \tilde{\mathbf{D}}\tilde{\mathbf{P}}$.

In the rest of the section, we focus only on the scaling part of the operation, i.e. $\mathbf{G}^{(0)}\tilde{\mathbf{D}}$. Furthermore, taking into account the particular structure of $\mathbf{G}^{(0)}$, we have:

$$\mathbf{G}^{(0)}\tilde{\mathbf{D}} = (\mathbf{I}_k | \mathbf{V})\,\tilde{\mathbf{D}} = \left(\tilde{d}_0 \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \tilde{d}_{k-1} \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}, \tilde{d}_k \begin{bmatrix} \\ \mathbf{v}_0 \\ \\ \end{bmatrix}, \ldots, \tilde{d}_{n-1} \begin{bmatrix} \\ \mathbf{v}_{n-k-1} \\ \\ \end{bmatrix} \right)$$

where $\mathbf{g}_j$ and $\mathbf{v}_j$ are the j-th columns of $\mathbf{G}^{(0)}$ and $\mathbf{V}$ respectively. The element $\tilde{d}_j$ is the j-th element of the diagonal of $\tilde{\mathbf{D}}$. The second part of the computation corresponding to the indices $J = \{k, \ldots, n-1\}$, i.e., the scaling of the columns of the dense matrix $\mathbf{V}$, consists in multiplying a single unknown secret $\tilde{d}_j, j \in J$ with a large number of public elements $v_{i,j-k}$. As such, this computation presents a good target for a CPA since we got an intermediate computation in the desired form that we can target as described in Section 2.4, albeit we are limited in the number of traces that we can use. We show how this issue is solved in Section 3.6. Thus, from the CPA, we are able to retrieve the value of $\tilde{d}_j$. Furthermore, in an implementation without shuffling countermeasures, the attacker also knows in which order each operation is performed, thus also revealing the index j.

The challenge with recovering ephemeral $\tilde{\mathbf{D}}_{J'=\{0,\ldots,n-1\}/J}$: For the rest of the elements of the diagonal matrix (i.e. $\tilde{d}_j, j \in \{0, \ldots, n-1\}/J = \{0, \ldots, k-1\}$), the columns that are scaled are part of an identity matrix. As such, only a single multiplication by a non-zero value is performed. The leakage is thus considerably smaller compared to the leakage that can be observed during the scaling of a dense column. As such, this presents no practical opportunities to gain sufficient leakage to recover the values of $\tilde{d}_j$ for $j \in \{0, \ldots, k-1\}$. However, we show in Sections 3.3 and 3.4 that knowing those elements is not needed for the full key recovery.

3.3 Partial recovery of fixed secret Q

In this section, we move on to the next step and show how the knowledge of a part $\tilde{\mathbf{D}}_J$ of the ephemeral secret $\tilde{\mathbf{D}}$ can be used to recover the corresponding part of the long term secret $\mathbf{Q}_J = \mathbf{D}_J \mathbf{P}_J$.

To achieve this, we first notice that a relationship between $\tilde{\mathbf{Q}}$ and $\mathbf{Q}$ is needed since $\tilde{\mathbf{Q}}$ is randomly generated. In the identification protocol, we get such a thing when the challenge is equal to 1, corresponding to the case when the prover sends $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$. Hereafter, we consider that we get m such responses, which we denote as $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ for $i \in \{0, \ldots, m-1\}$.

To understand the recovery process, we notice that $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ can be represented as:

$$\begin{aligned}
\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)} &= (\mathbf{DP})^{-1}\,\tilde{\mathbf{D}}^{(i)}\tilde{\mathbf{P}}^{(i)} && (1)\\
&= \mathbf{P}^{-1}\mathbf{D}^{-1}\tilde{\mathbf{D}}^{(i)}\tilde{\mathbf{P}}^{(i)} && (2)\\
&= \mathbf{P}^{\top}\mathbf{D}^{-1}\tilde{\mathbf{D}}^{(i)}\tilde{\mathbf{P}}^{(i)} && (3)\\
&= \mathbf{P}^{\top}\mathbf{D}'^{(i)}\tilde{\mathbf{P}}^{(i)} && (4)
\end{aligned}$$

where we use the properties of the inverse of matrices to go from equation 1 to equation 2. Moreover, the multiplication of an inverse diagonal matrix $\mathbf{D}^{-1}$ with a diagonal matrix $\tilde{\mathbf{D}}^{(i)}$ gives another diagonal matrix $\mathbf{D}'^{(i)}$ where the elements on the diagonal are $d'_j = d_j^{-1} \cdot \tilde{d}_j^{(i)}$.

From equation 4, one can see that the multiplication between the matrices $\mathbf{Q}^{-1}$ and $\tilde{\mathbf{Q}}^{(i)}$ translates into two permutations of the diagonal matrix $\mathbf{D}'^{(i)}$. Concretely, the row permutation is given by a persistent and secret matrix $\mathbf{P}^{\top}$ while the column permutation is given by an ephemeral and also secret matrix $\tilde{\mathbf{P}}^{(i)}$. This can be summarized by:

$$q^{(i)}_{i',j'} = \begin{cases} d'_j = d_j^{-1} \cdot \tilde{d}^{(i)}_j & \text{if, } i' = \pi(j) \wedge j' = \tilde{\pi}^{(i)}(j) \\ 0 & \text{otherwise} \end{cases}$$

where $q^{(i)}_{i',j'}$ is the element of $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ on the i'-th row and j'-th column. Moreover, π and $\tilde{\pi}^{(i)}$ are the permutation corresponding to the matrices $\mathbf{P}$ and $\tilde{\mathbf{P}}^{(i)}$ respectively.

A key observation is that the element d_j^{-1} of the secret key is always a factor of the signatures' non-zero element that lies in the i'-th row, regardless of the ephemeral column permutation $\tilde{\pi}^{(i)}$. This leads us to focus only on the rows of the elements $q^{(i)}_{i',j'}$, thus dropping the column index. We denote the obtained column vector as $\mathbf{c}^{(i)}$ such that[2]:

$$c^{(i)}_j = q^{(i)}_{i',j'} \text{ where } i' = \pi(j) \wedge j' = \tilde{\pi}^{(i)}(j).$$

Given this, we can retrieve the coefficients d_j and the permutation $i' = \pi(j)$, corresponding to the secret $\mathbf{Q}_J$, in the following way: we multiply the column vector $\mathbf{c}^{(i)}$ by the inverse of a retrieved element $\tilde{d}^{(i)}_j, j \in \{k, k+1, \ldots, n-1\}$. This gives us,

$$\mathbf{c}^{(i)} \cdot \left(\tilde{d}^{(i)}_j\right)^{-1} = \begin{pmatrix} c^{(i)}_0 \\ \vdots \\ c^{(i)}_{\pi(j)} \\ \vdots \\ c^{(i)}_{n-1} \end{pmatrix} \cdot \left(\tilde{d}^{(i)}_j\right)^{-1} = \begin{pmatrix} d^{-1}_{\pi^{-1}(0)} \tilde{d}^{(i)}_{\pi^{-1}(0)} \cdot \left(\tilde{d}^{(i)}_j\right)^{-1} \\ \vdots \\ d_j^{-1} \tilde{d}^{(i)}_j \cdot \left(\tilde{d}^{(i)}_j\right)^{-1} \\ \vdots \\ d^{-1}_{\pi^{-1}(n-1)} \tilde{d}^{(i)}_{\pi^{-1}(n-1)} \cdot \left(\tilde{d}^{(i)}_j\right)^{-1} \end{pmatrix} = \begin{pmatrix} r^{(i)}_0 \\ \vdots \\ d_j^{-1} \\ \vdots \\ r^{(i)}_{n-1} \end{pmatrix}$$

where $r^{(i)}_{j'} = d^{-1}_{\pi^{-1}(j')} \tilde{d}^{(i)}_{\pi^{-1}(j')} \cdot \left(\tilde{d}^{(i)}_{j'}\right)^{-1}, j' \in \{0, \ldots, n-1\}/\{\pi(j)\}$.

[2] We are guaranteed that each row has only one non zero element since $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ is the (row and column) permutation of a diagonal matrix.

To make a pattern more apparent, we concatenate the products of each of the m responses to obtain,

$$\left(\mathbf{c}^{(0)} \cdot \left(\tilde{d}_j^{(0)}\right)^{-1} \big| \ldots \big| \mathbf{c}^{(m-1)} \cdot \left(\tilde{d}_j^{(m-1)}\right)^{-1}\right) = \begin{pmatrix} r_0^{(0)} & \cdots & r_0^{(m-1)} \\ \vdots & & \vdots \\ d_j^{-1} & \cdots & d_j^{-1} \\ \vdots & & \vdots \\ r_{n-1}^{(0)} & \cdots & r_{n-1}^{(m-1)} \end{pmatrix}.$$

Recovering elements of $\mathbf{D}_J$ and $\mathbf{P}_J$ simultaneously: We notice that the row $j' = \pi(j)$ is a sequence composed only of the element d_j^{-1}, since multiplying $c_j^{(i)} = d_j^{-1} \cdot \tilde{d}_j^{(i)}$ by the inverse of $\tilde{d}_j^{(i)}$ removes the ephemeral factor. On the other hand, given a row where $j' \neq \pi(j)$, the sequence of numbers will be made of random values $r_{j'}^{(i)}$. Indeed, for a given row j', each $r_{j'}^{(i)}$ is the product of a constant value $d_{\pi^{-1}(j')}^{-1}$ and two independent random values: $\tilde{d}_{\pi^{-1}(j')}^{(i)}$ and the inverse of $\tilde{d}_{j'}^{(i)}$.

As such, only the j'-th row contains a unique value, while the others can be made up of any value in $\mathbb{F}_q^*$. This enables us to identify the correct mapping of j by the permutation π, allowing us to deduce $j' = \pi(j)$. However, there exists a possibility that all values in a randomly generated sequence are identical, which stops us from distinguishing the correct result. This happens with probability

$$\mathrm{P}\left(r_{j'}^{(i)} = r_{j'}^{(i')}, \forall i, i' \in \{0, \ldots, m-1\}\right) = \left(1/\left|\mathbb{F}_q^*\right|\right)^{m-1} = (1/126)^{m-1}.$$

As such, we have to gather multiple responses to be certain that we can deduce the correct secret.

In the end, we are able to recover the value d_j^{-1} (and thus, d_j) corresponding to $\mathbf{D}_J$ and the result of $\pi(j) = j'$ corresponding to $\mathbf{P}_J$, where j is known from the CPA and j' is the row in which the element d_j^{-1} is identified. Doing this for each index $j \in \{k, k+1, \ldots, n-1\}$ leads to the knowledge of $\mathbf{Q}_J$.

On the possibility of extending the attack to LESS 2.0: As explained in the aforementioned text, in order to recover an element d_j of $\mathbf{D}_J$ and the permutation $\pi(j) = j'$ in $\mathbf{P}_J$ using the partial matrix $\tilde{\mathbf{D}}_J$ retrieved through CPA, we utilize the response matrices of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ provided over multiple signatures. In particular, we need the pairs of values $(d_i^{-1}\tilde{d}_j^{(i)}, \pi(j))$ where $d_j^{-1}\tilde{d}_j^{(i)} = q_{\pi(j),\tilde{\pi}(j)}^{(i)} \in \mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(i)}$ (note that the index j is not known). However, in LESS 2.0 this attack path is no longer possible since we get less information from the signature. Concretely we only get to know which index is in the set $\pi(\tilde{\pi}^{-1}(\{0, \ldots, k-1\})$[3] and as such we cannot extend it to gain knowledge of $\tilde{d}_k$.

[3] Note that we oversimplify this slightly by assuming that the first k columns of $\mathbf{G}^{(0)}\tilde{\mathbf{Q}}^{(i)}$ are invertible.

The next section covers a way to recover the full secret with this partial information and is the same for both the horizontal and CPA based attacks.

3.4 Full secret recovery

In [19], it was shown that LEP is equivalent to information-set LEP or IS-LEP. In the context of our attack, this means that knowing the action of an information set J (i.e. $\mathbf{Q}_J$) is enough to find the full monomial $\mathbf{Q}$, given $\mathbf{G}^{(0)}$ and $\mathbf{G}^{(1)}$ such that $\mathbf{G}^{(1)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q})$. In this section, we show an algorithm which can be used to find the monomial even if the set that we get is not an information set. Concretely, during the SCA the set that we recover is $J = \{k, k+1, \ldots, n-1\}$ and the action $\mathbf{Q}_J$. This set is not an information set when $\mathbf{G}^{(0)}_J = \mathbf{V}$ is not invertible. To understand why it is possible, notice that we do recover an information set without the action (i.e. we do not get the one to one mapping of the permutation π nor the scaling elements). Indeed, we can recover the sets $\{0, \ldots, n-1\}/J = \{0, \ldots, k-1\}$ and $\pi(\{0, \ldots, k-1\}) = \{0, \ldots, n-1\}/J'$, where the first set corresponds to the column indices of the identity matrix which is always invertible and thus an information set. This knowledge is sufficient to recover $\mathbf{Q}$ as shown by the authors of [8].

To recover the secret $\mathbf{Q}$ using $\mathbf{Q}_J$, first notice that the RREF of $\mathbf{G}^{(0)}\mathbf{Q}$ can always be expressed as a matrix multiplication by an invertible matrix $\mathbf{S}$. As such, we have that $\mathbf{G}^{(1)} = \mathbf{S}\mathbf{G}^{(0)}\mathbf{Q}$. Next, let $J' = \pi(J)$ be the set of the indices after the column permutation and let $\{0, \ldots, n-1\}/J$ and $\{0, \ldots, n-1\}/J'$ be the set of column indices outside J and J', respectively. One can derive the following relations, which are adapted from [19]:

$$\mathbf{G}^{(1)}_{J'} = \mathbf{S}\mathbf{G}^{(0)}_J\mathbf{Q}_J, \quad \mathbf{G}^{(1)}_{\{0,\ldots,n-1\}/J'} = \mathbf{S}\mathbf{G}^{(0)}_{\{0,\ldots,n-1\}/J}\mathbf{Q}_{\{0,\ldots,n-1\}/J},$$

where $\mathbf{Q}_J \in \mathrm{M}_k$ and $\mathbf{Q}_{\{0,\ldots,n-1\}/J} \in \mathrm{M}_{n-k}$. Furthermore, given the parameter set of LESS, we have that $\mathbf{G}^{(1)}_{\{0,\ldots,n-1\}/J'} \in \mathbb{F}^{k\times k}$ is a square matrix and is invertible since it is a composition of invertible matrices. In particular, we always have that $\mathbf{G}^{(0)}_{\{0,\ldots,n-1\}/J} = \mathbf{I}_k$ since $\{0, \ldots, n-1\}/J = \{0, \ldots, k-1\}$ which corresponds to the indices of the identity matrix in $\mathbf{G}^{(0)}$. Thus, we can always derive the following relation:

$$\begin{aligned}
\left(\mathbf{G}^{(1)}_{\{0,\ldots,n-1\}/J'}\right)^{-1}\mathbf{G}^{(1)}_{J'} &= \left(\mathbf{S}\mathbf{G}_{\{0,\ldots,n-1\}/J}\mathbf{Q}_{\{0,\ldots,n-1\}/J}\right)^{-1}(\mathbf{S}\mathbf{G}_J\mathbf{Q}_J) \\
&= \mathbf{Q}^{-1}_{\{0,\ldots,n-1\}/J}\left(\mathbf{G}^{(0)}_{\{0,\ldots,n-1\}/J}\right)^{-1}\mathbf{G}^{(0)}_J\mathbf{Q}_J \\
&= \mathbf{Q}^{-1}_{\{0,\ldots,n-1\}/J}\mathbf{I}_k\mathbf{G}^{(0)}_J\mathbf{Q}_J \\
&= \mathbf{Q}^{-1}_{\{0,\ldots,n-1\}/J}\mathbf{G}^{(0)}_J\mathbf{Q}_J.
\end{aligned}$$

After left multiplying both sides of the equation with $\mathbf{Q}_{\{0,\dots,n-1\}/J}$, the result can be rearranged to obtain:

$$\begin{aligned} & \mathbf{Q}_{\{0,\dots,n-1\}/J} \left(\mathbf{G}^{(1)}_{\{0,\dots,n-1\}/J'}\right)^{-1} \mathbf{G}^{(1)}_{J'} = \mathbf{G}^{(0)}_{J} \mathbf{Q}_J \\ \iff & \mathbf{Q}_{\{0,\dots,n-1\}/J} \mathbf{A} = \mathbf{B} \end{aligned}$$

where $\mathbf{A} = \left(\mathbf{G}^{(0)}_{\{0,\dots,n-1\}/J'}\right)^{-1} \mathbf{G}^{(0)}_{J'}$ and $\mathbf{B} = \mathbf{G}^{(0)}_{J} \mathbf{Q}_J$ are known and the matrix $\mathbf{Q}_{\{0,\dots,n-1\}/J}$ is the only unknown. To solve this system, one way is to right multiply both equation by the inverse of $\mathbf{A}$ which exists when J is an information set. In particular, it depends on the dense part of $\mathbf{G}^{(0)} = (\mathbf{I}_k | \mathbf{V})$ being invertible. For all the parameter sets in LESS, the probability that the matrix $\mathbf{V}$ is non invertible is approximately $1/|\mathbb{F}_q| = 1/126$. To take into account these cases, a more general approach is considered. Furthermore, this approach is also more suited for the practical attack as is shown in Section 3.6.

To consider a more general case, we replace $\mathbf{Q}_{\{0,\dots,n-1\}/J}$ by $\mathbf{M'} \in \mathrm{M}_{n-k}$. Indeed, it might be the case that the recovered monomial is not exactly the same as the one from the secret key. As such, we want to solve the system

$$\mathbf{M'A} = \mathbf{B} \tag{5}$$

To recover the monomial $\mathbf{M'} = \mathbf{D'P'}$, we can notice that each row $\mathbf{a}_{\pi'(i)}$ of the matrix $\mathbf{A}$ is related to a row $\mathbf{b}_i$ of the matrix $\mathbf{B}$ by the relation:

$$d'_{\pi'(i)} \mathbf{a}_{\pi'(i)} = \mathbf{b}_i \tag{6}$$

where d'_j is the j-th element of the diagonal matrix $\mathbf{D'}$ and π' is the permutation associated to the permutation matrix $\mathbf{P'}$. Even though both the permutation and the elements are unknown, this relation allows an efficient brute force search. Indeed, for each i-th row $\mathbf{a}_i$ of $\mathbf{A}$, we can multiply every element in each row of $\mathbf{B}$ by the inverse of the corresponding element in $\mathbf{a}_i$. Then, one of the results will be a vector containing only one constant a since we are guaranteed that for some row $\mathbf{b}_j$ the relation 6 holds. From this, we recover $\pi'(i) = j$ and $d'_j = a$. To complete this procedure, we also need to cover the case where one of the elements of $\mathbf{a}_i$ is equal to zero and thus does not have an inverse. In that case, the corresponding element of the row in $\mathbf{B}$ should also be equal to zero. Another case that has to be considered is when two or more rows of $\mathbf{B}$ are multiples of each other. This leads to multiple choices in vectors which have the same constant at the end of the procedure. However, it is not a problem since choosing either of them will lead to a solution $\mathbf{Q'}$ such that $\mathbf{G}^{(1)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q}) = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q'})$. This solution is not necessarily the same as the secret $\mathbf{Q}$, but can also be used in the identification protocol since both associated public keys are the same.

3.5 Extending the attack from the ID protocol to LESS-1.2

LESS is based on the identification protocol described in Figure 1 and is obtained using the Fiat-Shamir transform. For the purpose of the SCA, there is no difference that fundamentally changes the procedure used for the attack. There are

however differences that influence the expected minimum (without considering noise during the SCA) number of required signatures that need to be observed. The differences are as follows:

- For soundness reasons, the protocol is repeated multiple times. As such, one signature consists of multiple rounds. During these rounds, we are guaranteed to observe W responses of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$, where W is a parameter defined by LESS.
- Depending on the parameter set of LESS, the prover generates from 1 to 7 private keys $\mathbf{Q}^{(i)}$ with the corresponding public key $\mathbf{G}^{(i)} = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q}^{(i)})$, $i \in \{1, \ldots, 7\}$. This is done in order to reduce the signature size, at the expense of the public key size.
- The responses of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ are compressed in such a way that only half of the elements and permutations are sent. Namely the action of $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ on the first (for some ordering) information set of $\tilde{\mathbf{G}} = \mathrm{RREF}(\mathbf{G}^{(0)}\tilde{\mathbf{Q}})$ is sent.

The first point guarantees that we get W matrices of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$ in a single signature, making single-trace attacks possible. On the other hand, points 2 and 3 increase the number of required values of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}$. Indeed, the former requires us to perform the attack on multiple secret keys while the latter reduces the information that is given to us in a signature.

3.6 CPA attack in practice

In this section, we focus on the practical aspect of the attack. First, we explain how to perform the horizontal attack on $\tilde{d}_i\mathbf{v}_{i-k}$. Then we explain how to turn it into a CPA. Finally, we highlight some issues arising from the choice of our power leakage model.

From a horizontal attack to a CPA based attack: In the setting presented before, we supposed that we get $\tilde{\mathbf{D}}_J$ through a CPA-based horizontal attack. There, we directly target the elements $\tilde{d}_i$ during the computation of $\tilde{d}_i\mathbf{v}_{i-k}$. To recover this element, we are limited to a single round since $\tilde{\mathbf{D}}_J$ is generated randomly each time. As such, while performing the correlation, we are limited to the number of elements in the vector $\mathbf{v}_{i-k}$ and thus only a small part of the trace is used. Another disadvantage is that, for a given response $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(j)}$, we have to go through all the possible elements $\tilde{d}_i^{(j)} \in \mathbb{F}_q^*$.

However, this is not optimal because we do not use the fact that, in a sequence of responses, some sequences of elements are not possible since they lead to mutually exclusive secret keys. Thus, rather than attacking the values of $\tilde{d}_i^{(j)}$, we propose to go one step further and attack the values of the secret key directly. In the case of LESS, this is possible since we can compute $\tilde{d}_i^{(j)}$ from the response by making an assumption on a small part of the secret. Concretely, for a given

index $i \in \{k, k+1, \ldots, n-1\}$, we have to find $\pi(i) = i' \in \{0, \ldots, n-1\}$ and the value d_i, thus we have to verify only $n \cdot |\mathbb{F}_q^*|$ values pairs.

To explain how we perform the CPA attack, we focus on the recovery of a single pair of secret values $(d_k, \pi(k))$ corresponding to the index k. The same procedure can be then applied to other indices. As such, we first make a guess on the values of the secret, which we denote (d'_k, k'). Next, we use signatures of the form $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(j)}$ to turn this guess into a sequence of elements from which we can compute a hypothesis on the power trace. To do this, recall from Section 3.3 that if we do not care about the column position, we can represent $\mathbf{Q}^{-1}\tilde{\mathbf{Q}}^{(j)}$ as a vector $\mathbf{c}^{(j)}$, where $c^{(j)}_{\pi(i)} = d_i^{-1}\tilde{d}_i^{(j)}$. Setting those vectors as the columns of a matrix we obtain:

$$\left(\mathbf{c}^{(0)} | \ldots | \mathbf{c}^{(m-1)}\right) = \begin{pmatrix} d^{-1}_{\pi^{-1}(0)}\tilde{d}^{(0)}_{\pi^{-1}(0)} & \cdots & d^{-1}_{\pi^{-1}(0)}\tilde{d}^{(m-1)}_{\pi^{-1}(0)} \\ \vdots & & \vdots \\ d^{-1}_{\pi^{-1}(i)}\tilde{d}^{(0)}_{\pi^{-1}(i)} & \cdots & d^{-1}_{\pi^{-1}(i)}\tilde{d}^{(m-1)}_{\pi^{-1}(i)} \\ \vdots & & \vdots \\ d^{-1}_{\pi^{-1}(k-1)}\tilde{d}^{(0)}_{\pi^{-1}(k-1)} & \cdots & d^{-1}_{\pi^{-1}(k-1)}\tilde{d}^{(m-1)}_{\pi^{-1}(k-1)} \end{pmatrix}$$

where m is the number of collected responses. Given that, we take the k'-th row of this matrix and we multiply it by d'_k. This gives us:

$$d'_k \cdot \left(d^{-1}_{\pi^{-1}(k')}\tilde{d}^{(0)}_{\pi^{-1}(k')}, \ldots, d^{-1}_{\pi^{-1}(k')}\tilde{d}^{(m-1)}_{\pi^{-1}(k')}\right) = \left(\tilde{d}'^{(0)}_k, \ldots, \tilde{d}'^{(m-1)}_k\right)$$

where $\tilde{d}'^{(i)}_k = d'_k \cdot d^{-1}_{\pi^{-1}(k')}\tilde{d}^{(i)}_{\pi^{-1}(k')}$. We thus obtain a vector where the i-th element is a guess on the k-th element of the matrix $\tilde{\mathbf{D}}^{(i)}$. We then compare one by one the series of column scalings $\left(\tilde{d}'^{(0)}_k\mathbf{v}_k, \ldots, \tilde{d}'^{(m-1)}_k\mathbf{v}_k\right)$ with the measured power leakage of the computations $\left(\tilde{d}^{(0)}_k\mathbf{v}_k, \ldots, \tilde{d}^{(m-1)}_k\mathbf{v}_k\right)$, where $\mathbf{v}_k$ is the k-th column of the dense part of the public matrix $\mathbf{G}^{(0)}$. We then have 3 cases:

Case 1: $d'_k \neq d_{\pi^{-1}(k')}$
In that case, the sequence $\left(\tilde{d}'^{(0)}_k, \ldots, \tilde{d}'^{(m-1)}_k\right)$ is random leading us to expect a low correlation between the power consumption and the hypothesized one.

Case 2: $d'_k = d_{\pi^{-1}(k')}$ but $k' \neq \pi(k)$
In that case, the sequence $\left(\tilde{d}'^{(0)}_k, \ldots, \tilde{d}'^{(m-1)}_k\right) = \left(\tilde{d}_{\pi^{-1}(k'),0}, \ldots, \tilde{d}_{\pi^{-1}(k'),m-1}\right)$. However, since $k' \neq \pi(k)$, it is still random compared to $\left(\tilde{d}^{(0)}_k, \ldots, \tilde{d}^{(m-1)}_k\right)$ since each element of $\tilde{\mathbf{D}}$ is independently and randomly generated. Therefore, we also expect to obtain a low correlation.

Case 3: $d'_k = d_{\pi^{-1}(k')}$ and $k' = \pi(k)$
In that case, the sequence $\left(\tilde{d}'^{(0)}_k, \ldots, \tilde{d}'^{(m-1)}_k\right) = \left(\tilde{d}^{(0)}_k, \ldots, \tilde{d}^{(m-1)}_k\right)$ and we expect a high correlation since we have the correct values.

We repeat this procedure for each possible pairs of values (d'_k, k'), keeping only the one which has the highest correlation. Given enough responses, we obtain the correct result. We do the same for the indexes in $\{k+1, \ldots, n-1\}$.

In practice, this only ensures that the function π', which we reconstruct, has an image, i.e. $\forall i \in \{k, k+1, \ldots, n-1\}, \exists j \in \{0, \ldots, n-1\}$ such that $\pi'(i) = j$. But it does not ensure that it is a correct permutation since it might lead to collisions, thus mapping two different indices to the same index j. We resolve those collisions by choosing the mapping that has the highest correlation. For the other one, we take the second best choice and check again for collisions. We do this until the mapping is one to one.

Choice of the power leakage model: To perform the CPA, we found out during experimentation that we get the best results when we target the output of the modular reduction of the multiplication between the secret elements of $\tilde{\mathbf{D}}_J$ and the elements of $\mathbf{G}^{(0)}_J = \mathbf{V}$. In particular, we use the Hamming weight of the output as our model for the power consumption. However, this choice does not let us distinguish between each value. Indeed, notice that for the prime field $q = 2^7 - 1$ used in LESS we have that:

$$\mathrm{HW}(x \mod q) = \mathrm{HW}(2x \mod q)$$

Since the only operations that are used during the computation of the hypothesis are multiplications and permutations, we cannot distinguish two elements if they are related by a power of 2. Thus, we get a set of possible values rather than a particular one[4]. To take this into account, the procedure of obtaining the full secret from Section 3.4 has to be slightly modified. We recall that, to obtain the secret, we had to solve the Equation 5 :

$$\mathbf{M}'\mathbf{A} = \mathbf{B}$$

where $\mathbf{B} = \mathbf{G}^{(0)}_J \mathbf{Q}_J$. In this case, we do not have the exact value of $\mathbf{Q}_J$. Instead, we choose any value from the possible ones in the obtained sets. This gets us

$$\mathbf{Q}'_J = \mathbf{D}''\mathbf{Q}_J$$

where the $\mathbf{D}''$ is a diagonal matrix with elements in the set $\{y : y = 2^x \mod q, x \in \mathbb{N}\}$.

As such, the new procedure involves a preprocessing step where we find the matrix $\mathbf{D}''$ up to a factor of 2. To do this, we begin by doing the same first step

[4] Another disadvantage is that it increases the required number of rounds since the probability that two sequences of guesses lead to the same power consumption is higher.

as in the original procedure. However, in this case we do not expect to find a row $\mathbf{r}$ containing the same element. Instead, we expect that each element is the same up to a power of 2. Concretely, we have that $\mathbf{r} = (c, \ldots, c) \star \mathbf{d}''$ where $c \in \mathbb{F}_q^*$ and $\mathbf{d}''$ are the elements on the diagonal of $\mathbf{D}''$[5]. This lets us recover $d_0''\mathbf{d}''^{-1} = 2^a\mathbf{d}''^{-1} = r_0^{-1}\mathbf{r}$, for some $a \in \mathbb{F}_q$. Then, we form our new matrix $\mathbf{B}' = \mathbf{B}(2^a\mathbf{D}''^{-1})\mathbf{Q}'_J$ and we continue with the normal procedure. At the end of it, we obtain a multiple of the original secret $\mathbf{Q}' = 2^a\mathbf{Q}$. This however is enough since our goal is not to obtain the exact secret but a monomial that matches the public key. Which is the case since $\mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q}') = \mathrm{RREF}(\mathbf{G}^{(0)}2^a\mathbf{Q}) = \mathrm{RREF}(\mathbf{G}^{(0)}\mathbf{Q})$.

4 Extending the Attack to CROSS: Improvement on [21]

In this section, we begin by recalling the identification protocol underlying CROSS. We continue by briefly presenting the idea of the horizontal attack from [21], which targets the computations involving $\mathbf{u}$. The knowledge of this vector can be then used to recover the secret key η. Similarly to LESS, we apply the attack to the identification protocol, which can be generalized for the signature scheme by applying the Fiat-Shamir transform and by considering that multiple rounds of the scheme are executed during one signature.

Next, we show the improvements that can be applied to it. One of them is to turn the horizontal attack into a CPA and the other is a more optimal usage of the traces. Concretely, for the latter part, we will show how to reduce the number of elements of $\mathbf{u}$ that have to be recovered. Then, we propose a way to use all of the signature for the attack. Finally, we also propose a different intermediate computation to exploit the power traces. We note that we explain only the attack on the version of CROSS based on the R-SDP problem since the same attack can be applied on the R-SDP(G) one by considering that the secret key η is generated by a public generator matrix (in systematic form) $\mathbf{M}_G = (\mathbf{V}|\mathbf{I}_m)$[6], where $\mathbf{V} \in \mathbb{F}_z^{m\times(n-m)}$, and a secret vector $\zeta \in \mathbb{F}_z^m$ such that $\eta = \zeta\mathbf{M}_G$. Thus, the underlying secret can be recovered by noticing that $\zeta = (\eta_{n-m}, \ldots, \eta_n)$. Note that the major difference from R-SDP is that, in the R-SDP(G) setting, any vector that is taken from $\mathbb{F}_z^n$ is generated similarly to the secret key.

4.1 CROSS identification protocol

The CROSS signature scheme is based on a 5-pass protocol. Similarly to a sigma protocol, it is an interactive protocol enabling the prover to demonstrate their identity to the verifier. The key difference lies in an additional commitment and challenge phase (thus increasing the number of interactions from 3 to 5).

[5] For the sake of simplicity we omit the case where the row contains any zeros.

[6] Note that we follow the notation from the CROSS specification [3]. As such, we also invert the order of the identity matrix and the dense matrix compared to the standard convention.

We continue by describing each of the steps in the rest of the section and summarize it in the Figure 3. Note that this description of the identification protocol is presented in a different way than in [3] to make computations more explicit. As such, the protocol proceeds as follows:

- **Key generation:** The private key is a random vector $\eta \in \mathbb{F}_z^n$. The public key consists of a random parity check matrix in systematic form $\mathbf{H} = (\mathbf{W}|\mathbf{I}_{n-k})$ and the syndrome vector $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$, where $\mathbf{e} = (g^{\eta_0}, \ldots, g^{\eta_{n-1}})$.
- **First Commitment:** The prover generates random vectors $\eta' \in \mathbb{F}_z^n$ and $\mathbf{u}' \in \mathbb{F}_p^n$. Then, they compute $\tilde{\mathbf{e}} = (g^{\sigma_0}, \ldots, g^{\sigma_{n-1}})$, where $\sigma = \eta - \eta'$. Next, the prover calculates $\mathbf{u} = \tilde{\mathbf{e}} \star \mathbf{u}'$ and the corresponding syndrome vector $\tilde{\mathbf{s}} = \mathbf{u}\mathbf{H}^\top$. The prover commits to theses results by sending the hashes $\mathrm{Hash}(\tilde{\mathbf{s}}, \sigma) = c_0$ and $\mathrm{Hash}(\mathbf{u}', \mathbf{e}') = c_1$, where $\mathbf{e}' = (g^{\eta'_0}, \ldots, g^{\eta'_{n-1}})$.
- **First challenge:** The verifier generates a random element $\beta \in \mathbb{F}_p$ and sends it to the prover.
- **Second Commitment:** The prover computes $\mathbf{y} = \mathbf{u}' + \beta\mathbf{e}'$ and commits to it by sending the hash $h = \mathrm{Hash}(\mathbf{y})$.
- **Second challenge:** The verifier randomly selects $b \in \{0, 1\}$ and sends it to the prover.
- **Response:** The prover reveals either the pair of vectors $(\mathbf{y}, \sigma)$ or $(\mathbf{u}', \mathbf{e}')$, depending on whether the second challenge is 0 or 1, respectively.
- **Verification:** In the case the second challenge is equal to 0, the verifier computes the syndrome $\tilde{\mathbf{s}} = \tilde{\mathbf{y}}\mathbf{H}^\top - \beta\mathbf{s}$, where $\tilde{\mathbf{y}} = \tilde{\mathbf{e}} \star \mathbf{y}$ and $\tilde{\mathbf{e}}$ is obtained from σ. The verifier then checks if $\mathrm{Hash}(\mathbf{y}) = h$ and if $\mathrm{Hash}(\tilde{\mathbf{s}}, \sigma) = c_0$. On the other hand, if the second challenge is equal to 1, the verifier can compute $\mathbf{y} = \mathbf{u}' + \beta\mathbf{e}'$ and verifies that $\mathrm{Hash}(\mathbf{y}) = h$ and $\mathrm{Hash}(\mathbf{u}', \mathbf{e}') = c_1$.

4.2 Horizontal attack on CROSS

Target of the horizontal attack: Similarly to LESS, it is possible to target the multiplication $\mathbf{u}\mathbf{H}^\top$ to recover $\mathbf{u}$ through side-channel analysis. Indeed, comparing Algorithm 1 with Algorithm 2, the later being implemented in CROSS, one can notice that the column scaling of a public matrix is also present there. As such, it presents a potential target for CPA. The remaining part is to show how to recover the secret given that $\mathbf{u}$ is known. While the relation between $\mathbf{u}$ and the secret η is not explicit, there is a straight forward way to compute it given a valid response, regardless of the second challenge. Concretely, there are two ways to achieve it, depending on which second challenge $b \in \{0, 1\}$ was sent.

We show hereafter how to recover the secret from $\mathbf{u}$ element wise. Moreover we remind which data is shared. Let $\log_g : \mathbb{F}_p \to \mathbb{F}_z$ be the discrete logarithm in base g, which can be computed since p is small. We proceed with the recovery as follows:

Public Data System parameters $p, n, k \in \mathbb{N}, g \in \mathbb{F}_p^*$ such that the order of g is z.
Private Key $\eta \in \mathbb{F}_z^n$
Public Key $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}, \mathbf{s} = \mathbf{e}\mathbf{H}^\top \in \mathbb{F}_p^{n-k}$, where $\mathbf{e} = (g^{\eta_0}, \ldots, g^{\eta_{n-1}})$

PROVER		VERIFIER
$\mathbf{u}' \xleftarrow{\$} \mathbb{F}_p^n, \eta' \xleftarrow{\$} \mathbb{F}_z^n$		
$\sigma \leftarrow \eta - \eta', \tilde{\mathbf{e}} \leftarrow (g^{\sigma_0}, \ldots, g^{\sigma_{n-1}})$		
$\mathbf{u} \leftarrow \tilde{\mathbf{e}} \star \mathbf{u}'$		
$\tilde{\mathbf{s}} \leftarrow \mathbf{u}\mathbf{H}^\top$		
$\mathbf{e}' \leftarrow (g^{\eta'_0}, \ldots, g^{\eta'_{n-1}})$		
$c_0, c_1 \leftarrow \text{Hash}(\tilde{\mathbf{s}}, \sigma), \text{Hash}(\mathbf{u}', \mathbf{e}')$	$\xrightarrow{(c_0, c_1)}$	
	$\xleftarrow{\beta}$	$\beta \xleftarrow{\$} \mathbb{F}_p^*$
$\mathbf{y} \leftarrow \mathbf{u}' + \beta\mathbf{e}'$		
$h \leftarrow \text{Hash}(\mathbf{y})$	$\xrightarrow{h}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, 1\}$
If $b = 0, f \leftarrow (\mathbf{y}, \sigma)$		
If $b = 1, f \leftarrow (\mathbf{u}', \mathbf{e}')$	$\xrightarrow{f}$	
		If $b = 0$
		$\tilde{\mathbf{e}} \leftarrow (g^{\sigma_0}, \ldots, g^{\sigma_{n-1}})$
		$\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{e}} \star \mathbf{y}, \tilde{\mathbf{s}} \leftarrow \tilde{\mathbf{y}}\mathbf{H}^\top - \beta\mathbf{s}$
		Accept if
		$\text{Hash}(\mathbf{y}) = h$
		$\text{Hash}(\tilde{\mathbf{s}}, \sigma) = c_0$
		$\sigma \in \mathbb{F}_z^n$
		If $b = 1$
		$\mathbf{y} \leftarrow \mathbf{u}' + \beta\mathbf{e}'$
		Accept if
		$\text{Hash}(\mathbf{y}) = h$
		$\text{Hash}(\mathbf{u}', \mathbf{e}') = c_1$

Fig. 3: 5-pass protocol at the basis of CROSS [4].

Case where $b = 0$: known data β, $\mathbf{y}$, σ.
In this case, we are able to compute $u'_i = g^{-\sigma_i} u_i$ which lets us find $e'_i = \beta^{-1} \cdot (y_i - u'_i)$. From this, we get $\eta'_i = \log_g(e'_i)$ and finally the secret can be recover as $\eta_i = \sigma + \eta'_i$.

Case where $b = 1$: known data $\mathbf{u}'$ and $\mathbf{e}'$.
In this case, we first recover $\sigma_i = \log_g(u_i u_i'^{-1})$ from which we obtain the secret key as $\eta_i = \sigma_i + \log_g(e'_i)$.

General idea of the horizontal attack from [21]: The authors from [21] proceed in three steps, targeting the cases where the second challenge b is equal to 1:

Algorithm 2 VecParity_Mult($\mathbf{u}, \mathbf{H}$)

Require: Matrix $\mathbf{H} = (\mathbf{W}|\mathbf{I}_{n-k})$, with $\mathbf{W} \in \mathbb{F}_q^{(n-k)\times k}$, vector $\mathbf{u} = (u_0, \ldots, u_{n-1}) \in \mathbb{F}_p^n$
Ensure: $\tilde{\mathbf{s}} = \mathbf{u}\mathbf{H}^\top$, where $\tilde{\mathbf{s}} \in \mathbb{F}_p^{n-k}$
  $\mathbf{a}^{(0)} \leftarrow (u_k, \ldots, u_{n-1})$
  for $i = 0$ to $k-1$ **do**
    $\mathbf{t} \leftarrow u_i \cdot \mathbf{w}_i$ ▷ scaling of the i-th column of $\mathbf{W}$
    $\mathbf{a}^{(i+1)} \leftarrow (\mathbf{a}^{(i)} + \mathbf{t}) \mod p$ ▷ here the modular reduction is made explicit
  end for
  $\tilde{\mathbf{s}} \leftarrow \mathbf{a}^{(k)}$
  return $\tilde{\mathbf{s}}$

Step 1: First, a horizontal attack on the computation of $u_i\mathbf{w}_i$ is performed. Thus, recovering each $u_i, i \in \{0, \ldots, k-1\}$.

Step 2: Then, the goal is to recover the rest of the vector $\mathbf{u}$. This is achieved by performing a horizontal attack on the intermediate values of the accumulator,

$$a_i^{(j)} = \left(u_{i+k} + \sum_{l=0}^{j-1} u_l w_{l,i} \right) \mod p$$

which are defined in Algorithm 2. For a given index i, the only unknown is u_{i+k} since the rest of the values where recovered during step 1.

Step 3: Recover the secret η from $\mathbf{u}$ and a valid signature where the challenge $b = 1$ using the procedure introduced in this section.

In the work [21], an optimization is introduced where the search space of $u_i \in \mathbb{F}_p$ is reduced by using the fact that most ephemeral values u_i together with a correct signature do not lead to a valid secret $\eta_i \in \mathbb{F}_z$. Concretely, there exists only one u_i for a given value η_i. Thus, the number of possibilities is reduced from $|\mathbb{F}_p|$ to $|\mathbb{F}_z|$, which is a reduction from 127 or 509 to 7 or 127 for R-SDP and R-SDP(G) versions of CROSS respectively.

Proposed improvements of the attack:

- First, we do not perform the second step of the attack since we can use the public key to recover the rest of the private key given the first k values of η. Indeed, we have that:

$$(e_k, \ldots, e_{n-1}) = \mathbf{s} - (e_0, \ldots, e_{k-1})\mathbf{W}^\top,$$

  where $\mathbf{e} = (g^{\eta_0}, \ldots, g^{\eta_{n-1}})$.
- Another improvement is that we are able to use every round, and thus all of the signature. Indeed, a link between η and $\mathbf{u}$ is shown in the previous part of this section for the cases where $b = 0$ and $b = 1$.

- Moreover, we apply the same idea as in LESS that leads us to the CPA attack, reducing the search space further. Concretely, we do not need to check each possible values of u_i each time for different rounds. Only an initial assumption on η_i is made from which the ephemeral secret u_i is computed.
- Finally, in Section 4.3 we show that the intermediate computation that is used for the attack in step 2 can be used to get better results, which are reported in Section 5.

4.3 Leakage models in the power traces

In this section we present two leakage models. The first one is the same as in Section 4.2, i.e. we target the unreduced multiplication $u_i\mathbf{w}_i$. This can be used to target any round, no matter the value of the second challenge.

However, through testing we found out that there is a better leakage model. The computation that we target is the same as in step 2, i.e. the reduced accumulator $\mathbf{a}^{(j)}$, but rather than recovering $u_i, i \in \{k, k+1, \ldots, n-1\}$ we want to recover the other part of $\mathbf{u}$. One might notice that this cannot be done in a straight forward way because the accumulator is "masked" by the $n-k$ last elements of $\mathbf{u}$. This can be circumvented in the case that we target responses where the challenge is equal to 0. In that case, we are able to compute $\tilde{\mathbf{s}} \leftarrow \tilde{\mathbf{y}}\mathbf{H}^\top - \beta\mathbf{s}$ which is also the last value of the accumulator $\mathbf{a}^{(k)}$. To get the previous values, we can use the recursive formula:

$$\mathbf{a}^{(i)} = (\mathbf{a}^{(i+1)} - u_i\mathbf{w}_i) \mod p$$

One can see that to get the i-th value we need to know $u_j, i \leq j < k$. As such, we proceed by doing a CPA first on the accumulator $\mathbf{a}^{(k-1)}$ to get η_{k-1} to be able to correctly compute u_{k-1} for each response. We then continue with η_{k-2} and so one until we get η_1. We cannot get η_0 by targeting the modular reduction of the accumulator $\mathbf{a}^{(0)}$ since it is not performed. This, however, is not an issue since only one element in $\mathbb{F}_z$ is left unknown, leading to an efficient brute force attack.

5 Results and Discussion

We performed the attack on a ChipWhisperer Nano board equipped with the STM32F030 microcontroller where we used the onboard ADC for trace collection. The code was compiled using `arm-none-eabi` version 10.2.1. To be able to fit the computations in the small ROM and SRAM, we could only implement small parts of the computation. In the case of LESS, we implemented only the inner loop of the matrix multiplication, i.e. the column scaling. Furthermore, for the modular reduction we use the Barrett reduction provided in the reference implementation. As for CROSS, we implement only the vector matrix multiplication with accumulation as in the reference implementation. In this case we use the provided modular reductions.

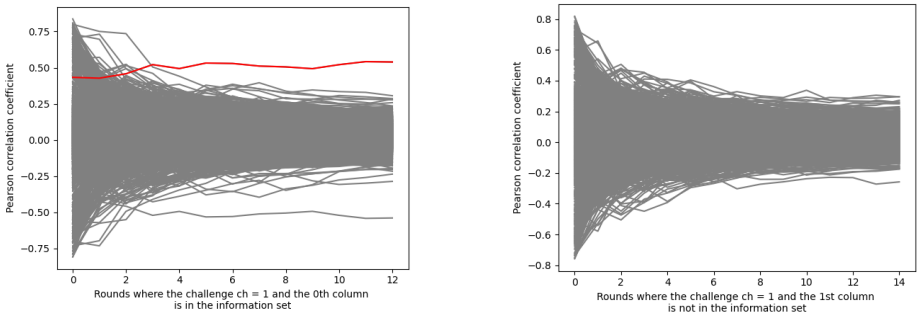
We present the number of rounds required to fully recover the secret key in Table 1 for 10 different key pairs. We target a single parameter set of LESS, CROSS-R-SDP and CROSS-RSDP(G). The parameters sets that we use are the ones for NIST security level 1. In the case of LESS we target the "balanced" instantiation and for both instances of CROSS the "speed" one. Furthermore, for CROSS, we perform the attack assuming two different leakage models described in Section 4.3.

Table 1: Number of required rounds present in signatures for the full secret key recovery. The average is taken over 10 different key pairs.

Scheme	Leakage model	Required rounds			Required challenge value
		Minimum	Average	Maximum	
LESS	$\mathrm{HW}(\tilde{d}_i\mathbf{v}_i)$	15	17.7	24	ch = 1
CROSS-R-SDP	$\mathrm{HW}(\mathbf{a}^{(i)})$	1	1.4	2	b = 0
	$\mathrm{HW}(u_i\mathbf{w}_i)$	15	20.2	28	b = $\{0,1\}$
CROSS-RSDP(G)	$\mathrm{HW}(\mathbf{a}^{(i)})$	4	4.7	6	b = 0
	$\mathrm{HW}(u_i\mathbf{w}_i)$	14	24.6	30	b = $\{0,1\}$

In the case of LESS, we were able to recover the full secret after at most 24 rounds where the challenge ch = 1 for the chosen parameter set, this corresponds to a key recovery in a single trace. We expect similar results for other parameter sets where there is a single secret monomial. On the other hand, when there are multiple secret monomials, the attack has to be performed on each of the keys separately, thus increasing the required rounds that have to be observed. Furthermore, this increase is not linear since the secret key that is used during a round is chosen randomly. As such, the underlying problem in estimating the number of necessary rounds is the coupon collector problem where multiple sets have to be completed [12]. Moreover, we present the evolution of the correlation coefficient for a full signature in Figure 4 where we try to find for a fixed index i the corresponding index j such that $i = \pi(j), j \in \{k, k+1, \ldots, n-1\}$ and the element d_j. We do this for j where there is a solution and another where there is none. We note that the x axis has different maximums and it does not match the number of received rounds with ch = 1. This is due to the fact that we get only half of the elements, as described in Section 3.5, and are thus not always able to use a round.

For CROSS, we tested two leakage models where we assumed either the HW of the accumulator $\mathbf{a}^{(i)}$ leaked or the HW of the unreduced multiplication $u_i\mathbf{w}_i$ leaked. In the former case we were able to recover the key after at most 2 or 6 rounds where the challenge was set to $b = 0$, for the R-SDP and R-SDP(G) variants respectively. In the other case we needed more rounds and we always had the correct key after at most 28 or 30 of them. We present the evolution of the correlation coefficient for both in Figure 5. Note that in Figure 5a the units of the x-axis are different from those in Figure 5b; they represent respectively the number of individual multiplications and the number of rounds. We notice



(a) $i = 0$ (b) $i = 1$

Fig. 4: Evolution of the correlation coefficient for the hypothesis where $\pi(\{k, k+1, \ldots, n-1\}) = i$. For $i = 0$ there is an index that maps to it while for $i = 1$ there is not. The correct result is highlighted in red.

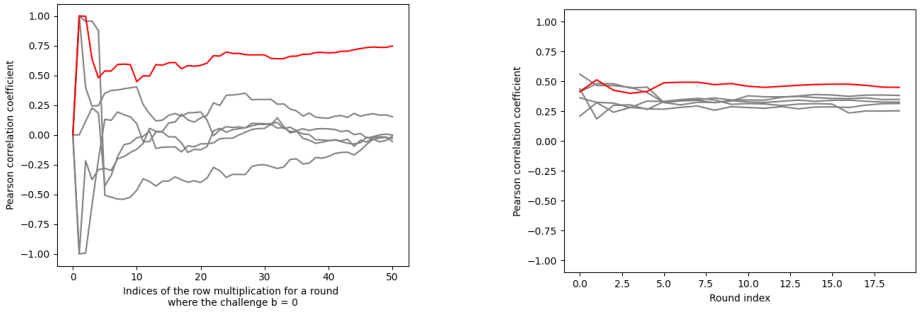
that the first strategy, targeting the reduced accumulator, converges faster, has a higher correlation for the correct value and is more pronounced compared to the incorrect ones. Individually, which strategy is better depends on how much of the signature has as the challenge $b = 0$ and thus how many rounds can be used from a single signature. An alternative idea is to combine both strategies.

Comparing this result with the work in [21] for the same parameter set of CROSS-R-SDP and CROSS-R-SDP(G), we require an order of magnitude less rounds to fully recover the secret key compared to their measurements. Concretely, we reduce the required rounds from 27 and 158 to 2 and 6 for R-SDP and R-SDP(G) respectively. We note that some of this discrepancy might be attributed to the different boards used. In any case, for the simulated noise free attack, we still have a reduction in the required rounds from 30 and 31 for R-SDP and R-SDP(G) respectively.

Comparing the attack on LESS and CROSS we notice that when we target a similar operation, i.e. the result of a computation after the modular reduction, we get noticeably different results even though in LESS we have more elements to work with. Concretely, we get a slower convergence for the correct value. This might come from the fact that different modular reductions are implemented. Furthermore, we require significantly more rounds to recover the secret key. We attribute this discrepancy to the fact that we receive only partial information in the signature, which is noted in 3.5. Another issue is that there is a non negligible probability that an incorrect assumption might be actually present in the trace, which is noted in 3.3, exacerbated by the choice of the leakage model where we are not able to distinguish values if they share the same hamming weight.

Finally, in both LESS and CROSS we notice that, usually, most of correct secret values are recovered early. Thus, the remaining rounds are collected in order to recover only a small part of the secret. If this part is small enough, an

optimization can be investigated where the remaining elements of the secret are recovered using a brute force attack.



(a) Leakage assumption: $\mathrm{HW}(\mathbf{a}^{(i)})$

(b) Leakage assumption: $\mathrm{HW}(u_i \mathbf{w}_i)$

Fig. 5: Comparison of the evolution of the correlation coefficient for two different leakage models. Note that the units in the x axis and the scale of the y axis are different for the two figures. The correct results are highlighted in red.

6 Conclusion

In this work, we present a complete key-recovery attack on the LESS and CROSS post-quantum digital signature schemes using CPA. This is achieved by building on previous horizontal attacks against code based schemes [10, 21]. Our attack exploits the direct dependency of the relatively small parts in the elements of the secret key and with the ephemeral secret. This allows us to mount a CPA-based attack targeting the multiplication of a secret ephemeral vector with a public matrix, leading to the full recovery of the secret key. This attack is applied to the versions 1.2 of both schemes, and leads to the recovery of the secret in a single trace for the parameter sets that are targeted. In the case of LESS the amount of traces that are required is expected to increase in the parameter sets where the number of secret keys is greater than one. On the other hand, for CROSS, the same result is expected for any parameter set. Furthermore, we show optimizations that require by an order of magnitude less rounds compared to existing attacks.

In the new 2.0 versions, it is still effective against CROSS, though it is no longer applicable for LESS. The later is due to the fact that the more recent version of LESS implements a compression technique of the signature which removes the required information for the attack to function.

This work underlines the importance of implementing countermeasures against side-channel attacks. We leave their implementation for future works.

Acknowledgments. This work was partially supported by the FWF grant PAT6402023, and the State Government of Styria, Austria, through the Department Zukunftsfonds Steiermark. Further, this work was partially supported by the SINFONIA project. The SINFONIA project has received funding from the Recovery and Resilience Facility (RRF) as the centrepiece of NextGenerationEU via the Austrian Research Promotion Agency (FFG) and Austria Wirtschaftsservice Gesellschaft mbH (aws) in the frame of the IPCEI ME/CT – Important Project of Common European Interest on Microelectronic and Communication Technologies under FFG project No 917423 and AWS project No P2431566.

References

1. Survey on side-channel attacks on code-based key encapsulation mechanism. Moroccan Journal of Algebra and Geometry with Applications (in press)
2. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.J.O., Santini, P., Wallace, R.: LESS : Linear equivalence signature scheme, version 1.2. National Institute for Standards and Technology (2023), https://www.less-project.com/LESS-2024-02-19.pdf
3. Baldi, M., Barenghi, A., Bitzer, S., Karl, P., Manganiello, F., Pavoni, A., Pelosi, G., Santini, P., Schupp, J., Slaughter, F., Wachter-Zeh, A., Weger, V.: CROSS :code and restricted object signature. National Institute for Standards and Technology (2023), https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/CROSS-spec-web.pdf
4. Baldi, M., Bitzer, S., Pavoni, A., Santini, P., Wachter-Zeh, A., Weger, V.: Zero knowledge protocols and signatures from the restricted syndrome decoding problem. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14602, pp. 243–274. Springer (2024), https://doi.org/10.1007/978-3-031-57722-2_8
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2129–2146. CCS '19, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3319535.3363229
6. Biasse, J., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12174, pp. 45–65. Springer (2020), https://doi.org/10.1007/978-3-030-51938-4_3
7. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29. Springer. https://doi.org/10.1007/978-3-540-28632-5_2
8. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Cryptology ePrint Archive, Paper 2023/1533 (2023), https://eprint.iacr.org/2023/1533

9. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. Cryptology ePrint Archive, Paper 2010/394 (2010), https://eprint.iacr.org/2010/394
10. Colombier, B., Grosso, V., Cayrel, P.L., Drăgoi, V.F.: Horizontal correlation attack on classic McEliece. Cryptology ePrint Archive, Paper 2023/546 (2023), https://eprint.iacr.org/2023/546
11. Coron, J.S., Kocher, P., Naccache, D.: Statistics and secret leakage. In: Frankel, Y. (ed.) Financial Cryptography. pp. 157–173. Springer. https://doi.org/10.1007/3-540-45472-1_12
12. Doumas, A.V., Papanicolaou, V.G.: The coupon collector's problem revisited: Generalizing the double dixie cup problem of newman and shepp (2015), https://arxiv.org/abs/1412.3626
13. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 238–268 (2018), https://doi.org/10.13154/tches.v2018.i1.238-268
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25
15. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
16. Mayer-Sommer, R.: Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2000. pp. 78–92. Springer. https://doi.org/10.1007/3-540-44499-8_6
17. Mondal, P., Adhikary, S., Kundu, S., Karmakar, A.: ZKFault: Fault attack analysis on zero-knowledge based post-quantum digital signature schemes. Cryptology ePrint Archive, Paper 2024/1422 (2024), https://eprint.iacr.org/2024/1422
18. National Institute for Standards and Technology (NIST): Post-quantum cryptography: Additional digital signature schemes, https://https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures
19. Persichetti, E., Santini, P.: A new formulation of the linear equivalence problem and shorter less signatures. Springer-Verlag (2023)
20. Schamberger, T., Renner, J., Sigl, G., Wachter-Zeh, A.: A power side-channel attack on the CCA2-secure HQC KEM. Cryptology ePrint Archive, Paper 2020/910 (2020), https://eprint.iacr.org/2020/910
21. Schupp, J., Sigl, G.: A horizontal attack on the codes and restricted objects signature scheme (CROSS). Cryptology ePrint Archive, Paper 2025/116 (2025), https://eprint.iacr.org/2025/116
22. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
23. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (1994). https://doi.org/10.1109/SFCS.1994.365700