A Little LESS Secure

Side-Channel Attacks Exploiting Randomness Leakage

Dina Hesse[1], Elisabeth Krahmer[1,✉], Yi-Fu Lai[2], and Jonas Meers[1]

[1] Ruhr-University Bochum, Germany, firstname.lastname@rub.de
[2] Shanghai Jiao Tong University, China, yifu.lai@sjtu.edu.cn

Abstract. Schnorr and (EC)DSA signatures famously become completely insecure once a few bits of the random nonce are revealed to an attacker. In this work, we explore whether the Fiat-Shamir based post-quantum signature scheme LESS is vulnerable to analogous attacks. In particular, we investigate the impact of partial leakage of the commitment randomness – a scenario that falls under the broader class of Hidden Number Problems – on the security of the secret key.

We present an efficient attack on LESS that requires knowledge of a *single bit* of the randomness with less than 1200 signatures to fully recover the secret key. Our attack leverages the observation that knowledge of one bit is sufficient to distinguish secret key entries from random candidates. In addition, we describe a variant of this attack that requires one-bit leakage of multiple randomness values, but succeeds with only two signatures.

To demonstrate the practicality of our attacks, we identify and exploit two different side-channels that are present in the reference implementation: One timing-based attack and one exploiting the power side-channel leakage. Both show that the assumptions regarding the required single-bit leakage can be obtained in practice and that our attack poses a realistic threat to the current implementation of LESS. To our knowledge, these are the first practically verified side-channel attacks on LESS.

Keywords: Hidden Number Problem, Fiat-Shamir, LESS, Post-Quantum, Side-Channel Attacks

1 Introduction

The Fiat-Shamir paradigm [14] stands as a cornerstone in modern cryptography, providing an efficient and elegant approach to the construction of secure signature algorithms from identification protocols. One of the famous examples is the Schnorr signature scheme [30], which uses a group of prime order q and works as follows: To sign a message m with secret key (x, g^x), one first picks some random nonce $r \in \mathbb{Z}_q^*$ (also called the *commitment randomness*) and computes the hash value $e = H(g^r||m)$. The signature then becomes $\sigma = (s, e)$ where $s = r + xe \mod q$. For the security of Schnorr signatures it is essential that the randomness r is never reused. Otherwise, the secret key can be recovered via $x = (s' - s)/(e' - e)$ where $\sigma' = (s', e')$ is a second signature using the same r.

In fact, the randomness is even more sensitive and needs to be strictly sampled from the uniform distribution. A line of research initiated by Bleichenbacher [8] as well as Howgrave-Graham and Smart [17], developed sophisticated attacks that recover the secret key of Fiat-Shamir based conventional signature schemes even if only a small fraction of the randomness is exposed or exhibits a mild bias [26,2]. For instance, in the case of Schnorr and (EC)DSA signatures the task of recovering the secret key reduces to the famous *Hidden Number Problem* (HNP) of Boneh and Venkatesan [9]. The latter can be solved efficiently either by mapping it to (an easy instance of) the Closest Vector Problem (as originally done by [9]) or by using Bleichenbacher's approach based on Fourier analysis. These results were largely theoretical at first, however many of them led to practical attacks that obtained bits of the randomness via timing attacks [24], oversights in the implementation [13] or weak pseudorandom number generators [11].

In the context of post-quantum cryptography, the Fiat-Shamir paradigm remains influential, with the recently standardized Dilithium scheme (a.k.a. ML-DSA) [3] being the best known representative. Furthermore, 16 out of 40 submissions to the NIST call for additional digital signature schemes follow the Fiat-Shamir paradigm [10]. It is therefore a natural question whether the commitment randomness in the latest post-quantum candidates is just as sensitive and whether it lends itself to similar attacks.

LESS [4] is a code-based signature scheme and one of the Fiat-Shamir based contestants in the NIST call that advanced to the second round. It relies on the hardness of the Linear Equivalence Problem (LEP), which asks to recover an isometric mapping between two linear codes. Furthermore, its commitment randomness is an isometric mapping too. As it is customary for Fiat-Shamir based constructions, revealing the whole commitment randomness easily gives rise to a key recovery attack. Nevertheless, it is unclear whether only a fraction of the commitment randomness suffices similar to the Schnorr signature scheme.

Additionally, assumptions about any partial leakage of the randomness in LESS have so far remained purely theoretical. In practice, information like this is usually acquired by side-channel attacks. Side-channel attacks are non-invasive methods that help gaining knowledge about intermediate values by measuring the physical behavior of the target system, such as variation in timing or power consumption. Many successful attacks on pre- and post-quantum schemes have been realized, e.g. [31,32,15]. However, to our knowledge, to date, no side-channel attack on LESS has been practically demonstrated.

1.1 Our Contribution

In this paper, we investigate the combination of both worlds; We propose a novel key recovery attack on the latest version 2.0 of LESS and realize the attack in practice via timing and power side-channel analysis. Concretely:

– In Section 3, we provide the theoretical justification for the success of the attack. We define and analyze the LESS Hidden Number Problem, where

Security Level	LESS Parameter Set (worst case)	#Signatures
NIST-I	LESS-252-45	1 120
NIST-III	LESS-400-102	457
NIST-V	LESS-548-137	516

Table 1: Overview of the approximate number of (leaky) signatures necessary for our attack. The attack assumes one known bit per execution of the underlying ID protocol.

we first assume leakage of one bit of every entry in the commitment randomness. Our attack is efficient and requires less than three signatures on average to recover the secret keys at all security levels. We then generalize our algorithm to only take one *single* bit of leakage and successfully simulate the key recovery (cf. Table 1).

– In a second step, we verify that the assumptions on the leaked information are indeed reasonable. We identify two different points of interest in the official reference implementation of LESS [5] that leak the required bits of the commitment randomness via a timing, respectively a power side-channel. Both target operations are attacked successfully and, hence, show the relevance of our attack in practice. We close by suggesting countermeasures to protect these sensible operations.

1.2 Technical Details

To study the security of the signature scheme, it suffices to study the security of the underlying identification (ID) protocol. Essentially, the Fiat-Shamir transform simply replaces the verifier with a hash function (and, if necessary, runs the ID protocol several times in parallel) to convert the ID protocol to a signature scheme. In other words, the security of the signature scheme reduces to the security of the ID protocol. Therefore, we define a variant of the Hidden Number Problem (HNP) for the ID protocol underlying LESS (illustrated in Figure 1). We then leverage knowledge of bits of the commitment randomness $\tilde{\mu}$ to present a key-recovery attack on the secret key sk using the protocol's transcripts $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$.

Solving LESS HNP. In the LESS signature scheme, both the secret key and the commitment randomness consist of *monomial matrices*. Every monomial matrix can be compactly represented by two n-dimensional arrays (π, Δ), where the first component describes a permutation π over $[n] = \{1, 2, \ldots, n\}$, and the second is a vector from $\left(\mathbb{F}_q^\times\right)^n$. (Suggested parameter sets use $n \in \{252, 400, 548\}$ and $q = 127$.) For simplicity, we first study the LESS Hidden Number Problem in a slightly simplified model, in which we assume knowledge of the MSBs of *all* n entries of π in the commitment randomness. For this case, we introduce a novel

distinguisher that uses this knowledge of the MSBs to efficiently distinguish secret key entries from random ones. In our simplified model, we show that the distinguisher allows us to efficiently recover the secret key from expected $\lceil \log(n) \rceil$ executions of the ID protocol. Building upon these results, we then obtain our attack that requires knowledge of only a *single* bit per run of the ID protocol. Naturally, we require slightly more signatures in this setting. However, the amount increases only by a small polynomial factor $\widetilde{\mathcal{O}}(n)$.

Practicability of the Attack. We turn our theoretical results into a practical attack using side-channel measurements to obtain information on the commitment randomness $\tilde{\mu}$ via its permutation matrix. Several operations in LESS's *sign* function use $\tilde{\mu}$, each a potential target of a side-channel attack. We conduct attacks on two points of interest: The implementation of the systematic form transformation allows for a simple, but easily fixable, timing side-channel, described in Section 5.1. On the other hand, when $\tilde{\mu}$ is multiplied to the public key generator matrix $\mathbf{G}$ during signature generation, we achieve the necessary leakage for the attack by a power side-channel attack (cf. Section 5.2). Our code is available at

https://github.com/Chair-for-Security-Engineering/a-little-LESS-secure.

2 Preliminaries

2.1 Notation

The notation $[\![\cdot]\!]$ denotes a boolean test. Let $\mathbb{N}$ denote the natural numbers, and $\mathbb{F}_q$ denote the finite field of size q. For $n \in \mathbb{N}$, let $[n] := \{1, \cdots, n\} \subset \mathbb{N}$. For a ring R, we let R^n and $R^{k \times n}$ denote an array of dimension n and the k by n matrix space of R, respectively. Let $\mathbf{e}_i$ represent the i-th standard basis vector, e.g., $\mathbf{e}_1 = (1, 0, \ldots, 0)$. Any natural number n can be represented in binary form $n = \sum_{i=0}^{x} n_i \cdot 2^i$ with $n_i \in \{0, 1\}$ and $x = \min\{y \in \mathbb{N} | 2^y \geq n\}$. We call n_0 the LSB whereas n_x is the MSB.

Isometries. The concept of LESS is based on proving knowledge of an isometry between two $[n, k]$-linear codes, that is, knowledge of a Hamming weight preserving map. Three types of isometries are used in the scheme: permutations, partial permutations, and monomial maps.

Permutations. The set of all length n permutations is denoted by $\mathcal{S}_n$, also called the *symmetric group*. Each permutation $\pi \in \mathcal{S}_n$ can be represented by a matrix $\mathbf{P}_\pi \in R^{n \times n}$ with exactly one 1 entry in each column and row. For a matrix $\mathbf{G}$ with n columns we write

$$\pi(\mathbf{G}) = \mathbf{G} \cdot \mathbf{P}_\pi$$

to describe the permutation of the columns of $\mathbf{G}$ according to π.

Partial permutations $\mathcal{S}_{k,n}$ are a subset of $\mathcal{S}_n$ consisting of the permutations that reorder all elements of a k-subset $J \subset [n]$ to the first k positions in the output, formally

$$\mathcal{S}_{k,n} = \{\pi \in \mathcal{S}_n | \pi^{-1}(1) < \pi^{-1}(2) < ... < \pi^{-1}(k), \\ \text{and } \pi^{-1}(k+1) < \pi^{-1}(k+2) < \pi^{-1}(n)\}.$$

Monomials. The set of all length n monomial maps is denoted by $\mathcal{M}_n(q)$. Each monomial map $\mu \in \mathcal{M}_n(q)$ can be represented by a matrix $\mathbf{Q}_\mu$ of the form $\mathbf{Q}_\mu = \mathbf{DP}$, where $\mathbf{P} \in \mathbb{F}_q^{n\times n}$ is a permutation matrix, and $\mathbf{D} \in \mathbb{F}_q^{n\times n}$ is a non-singular diagonal-matrix. In other words, all diagonal elements of $\mathbf{D}$ belong to $\mathbb{F}_q^\times := \mathbb{F}_q \setminus \{0\}$. With slight abuse of notation we use $\mathcal{M}_n(q)$ for the set of monomial maps as well as for the set of monomial matrices. Analogously to above, we write

$$\mu(\mathbf{G}) = \mathbf{G} \cdot \mathbf{Q}_\mu$$

for a matrix $\mathbf{G}$ with n columns.

Since a monomial matrix $\mathbf{Q}_\mu = \mathbf{DP} \in \mathcal{M}_n(q)$ has only n non-zero entries, one can compactly store $\mathbf{Q}_\mu$ using only $\mathcal{O}(n(\log(q)+\log(n)))$ bits: By identifying $\mathbf{P}$ with an element π of the symmetric group $\mathcal{S}_n$, $\mathbf{P}$ can be equivalently written as $\pi = (\pi_1, \ldots, \pi_n) \in \mathcal{S}_n$, where $\pi_i = \pi(i) \in [n]$. [1] Similarly, $\mathbf{D}$ can be represented by an n-dimensional array $\Delta = (\Delta_1, \ldots, \Delta_n)$, where Δ_i is the i-th diagonal entry of $\mathbf{D}$. The two arrays (π, Δ) together then yield a compact description of $\mathbf{Q}_\mu$. We write

$$\mu \simeq \mathbf{Q}_\mu \simeq (\pi, \Delta)$$

to denote that (π, Δ) is the compact representation of $\mathbf{Q}_\mu$. We call π the *permutation part* of μ, and Δ the *diagonal part* of μ.

We further define a subgroup $F \subset \mathcal{M}_n(q)$ of the monomial maps with the following property: Any map $\varphi \in F$ can be uniquly represented by a pair $(\varphi_r, \varphi_c) \in \mathcal{M}_k(q) \times \mathcal{M}_{n-k}(q)$ s.t. φ_r is a monomial map on the first k coordinates and φ_c is a monomial map on the last $n-k$ coordinates. Any monomial in $\mathcal{M}_n(q)$ can then be written as a concatenation of an element in F and a partial permutation from $\mathcal{S}_{k,n}$. With the help of F we define the *canonical form* CF of a matrix: For a matrix $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$, the function CF returns either $\perp$ or $\mathbf{A}' = \mathbf{Q}_{\varphi_r} \cdot \mathbf{A} \cdot \mathbf{Q}_{\varphi_c}$ if $\mathbf{A}$ and $\mathbf{A}'$ are mapped by an isomentry (φ_r, φ_c) in F. This function fulfills $\mathsf{CF}(\mathbf{A}) = \mathsf{CF}(\mathbf{Q}_{\varphi_r} \cdot \mathbf{A} \cdot \mathbf{Q}_{\varphi_c})$ for all $\varphi \in F$.

Codes. An $[n, k]$-linear code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$. The standardized LESS parameter sets use codes of *rate* $1/2$, i.e., codes with $n = 2k$.

[1] LESS orders the elements of π differently. While in our notation, the i-th entry is the *image* of i, in LESS it is the *preimage* of i. This is, however, only a small syntactical change, which we chose for notational convenience. It does not affect the attack in any way.

Every $[n, k]$-linear code $\mathcal{C}$ can be represented by a *generator matrix* $\mathbf{G} \in \mathbb{F}^{k \times n}$ with $\mathcal{C} = \{\mathbf{vG} \mid \mathbf{v} \in \mathbb{F}_q^k\}$. Any generator matrix $\mathbf{G}$ can be transformed into its reduced row-echelon form (RREF) through elementary row operations; when additionally allowing a permutation of the columns π, this yields

$$\mathsf{SF}(\mathbf{G}) = \mathsf{RREF}^*(\mathbf{G}) = \mathsf{RREF}(\pi(\mathbf{G})) = (\mathbf{I}_k \mid \mathbf{A})$$

which is known as *systematic form* of $\mathbf{G}$. Here, RREF^* describes the concatenation of a suitable column permutation and the subsequent transformation into the RREF, yielding the identity matrix in the k first columns of the output. It additionally returns the used permutation π.

The systematic form of a generator matrix $\mathbf{G}$ can further be turned into the canonical form by replacing the non-systematic part $\mathbf{A}$ by its canonical form $\mathsf{CF}(\mathbf{A})$:

$$\mathsf{CF}(\mathbf{G}) = (\mathbf{I}_k \mid \mathsf{CF}(\mathbf{A}))$$

2.2 Side-Channel Analysis

Computing secrets requires devices with physical properties. They need time, they require power, they produce heat or electrical emanation. In some form, these properties are always affected by the processed (secret) values. The observation and the analysis of such effects to determine whether they reveal secrets is known as Side-Channel Analysis (SCA). In the late nineties, Kocher et al. [21,20] proposed the first attacks exploiting timing, resp. power side-channel to compromise the security of cryptographic implementations. In a timing SCA, the attacker exploits the secret-dependent execution time of an algorithm, whereas in a power SCA, they leverage the secret-dependent variations in the power consumption of the target device.

In practice, however, concurrent processing of unrelated data and environmental influences introduce disturbances, commonly referred to as measurement noise. This noise is typically modeled as additive and Gaussian, so that the attacker observes $f(x) + \mathcal{N}(0, \sigma)$ [23]. In a (power) side-channel setting, the noise level is often quantified by the Signal-to-Noise Ratio (SNR), defined as the ratio between the variance of the signal $f(x)$ and the variance of the noise σ^2. A high SNR indicates that the secret-dependent variations in the power consumption are more easily distinguishable.

Over time, a variety of analysis strategies have been developed, each with different strengths and limitations. For example, in Differential Power Analysis (DPA) [20], an attacker collects many side-channel observations, often called traces, of the secret-dependent computation and groups them according to a key (byte) guess. If the statistical difference between the clustered traces is maximal, the corresponding hypothesis is likely to be correct.

Simple Power Analysis. Conversely, Simple Power Analysis (SPA) exploits the side-channel information from only a small number of traces [27]. This is particularly relevant when the cryptographic operation is executed only once, for

example during key generation. If the leakage is sufficiently strong, the secret values may be directly observable in the traces. A way to increase the SNR, even in a single-trace setting, is to average out the noise. This is achieved by computing the mean of multiple noisy observations corresponding to the same intermediate value and is known as a *horizontal attack*. Battistello et al. [6] showed that this approach reduces the noise by a factor of $\sqrt{n}$, with n denoting the number of observations. If this method is not possible, or the SNR remains too low, more advanced attack strategies, such as machine learning or *template attacks*, are needed.

Template attacks are a subclass of SPA, in which the attacker first profiles the target device, or a similar device under their control [23]. In a first step (also called *offline phase*), the adversary builds templates, using traces recorded for known secret-dependent values. During the attack (*online*) phase, only one trace is recorded and classified using these templates. To conduct a template attack, it is necessary to record a large number of traces for the offline phase, however, only a single trace from the actual attacked target is needed. For that reason, template attacks are commonly referred to as single-trace attacks. The classification is typically performed using a distance metric (e.g., Euclidean), and the key hypothesis corresponding to the smallest distance is selected.

Countermeasures. With the growing number of side-channel attacks, various countermeasures have been developed to protect cryptographic implementations. Roughly, we can categorize side-channel countermeasures as *masking* and *hiding*. In masking, secret values are split into independent shares that are processed independently and only recombined only at the end of a sensitive computation [18]. The goal is to de-correlate the intermediate values and computations from the underlying secrets. Thus, masking schemes can often provide formal security guarantees. The most widely used variant is Boolean masking, based on splitting secrets additively modulo 2. Boolean masking is straightforward for linear operations, but it becomes significantly more complex for non-linear ones, requiring additional computations and fresh randomness. Consequently, masking typically introduces non-negligible performance and implementation overhead.

In contrast, *hiding* affects the side-channel signal itself rather than the processed values [23]. It aims to reduce the dependency of the signal on the secret by equalizing or randomizing the power consumption or execution time. Methods such as inserting random delays or shuffling the execution order increase resistance to SCA in the time dimension. In the power domain, the SNR can be reduced, for example, by adding noise or by lowering the intrinsic leakage at the hardware or cell level. In practice, combining hiding and masking is often recommended: hiding increases the noise level, which in turn strengthens the effectiveness of masking and leads to more robust overall protection.

2.3 LESS

LESS is a post-quantum signature scheme relying on the Linear Equivalence Problem and is currently evaluated in the second round of the NIST's call for

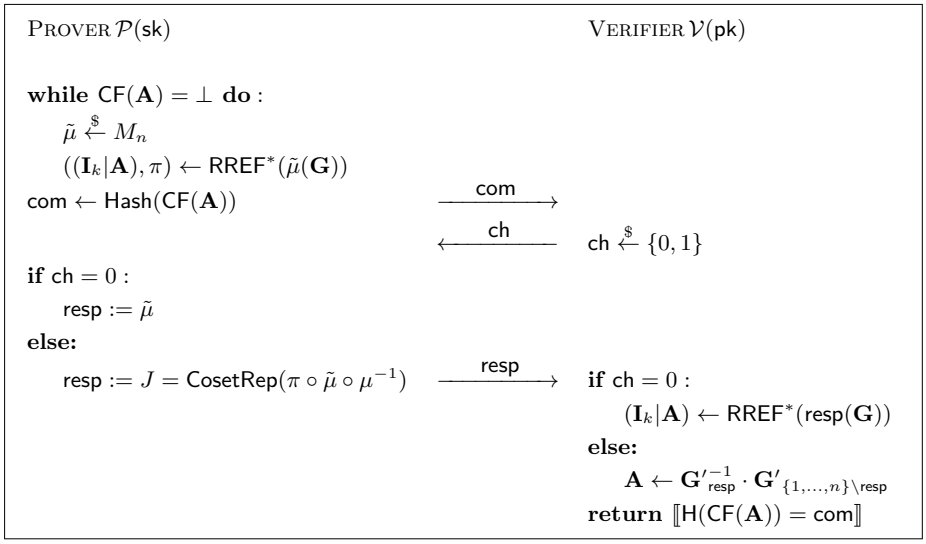


Fig. 1: The LESS ID protocol $\Sigma = (\mathcal{P}, \mathcal{V})$, taken from [4].

additional signature schemes [10]. Our work is based on the latest version at the moment of submission, namely 2.0. As signature scheme, LESS consists of algorithms for *key generation*, *signature generation*, and *signature verification*, however, only the second is relevant for our attack and hence we skip the details of the remaining two.

The LESS ID Protocol. As LESS is using the Fiat-Shamir transformation, it is based on a zero-knowledge identification protocol (called Σ) which is transformed into a signature scheme by replacing the verifier with a hash function. This protocol is depicted in Figure 1. The (public) parameters are n, k, q and a cryptographic hash function $\mathsf{H}(\cdot)$. Furthermore, the secret key sk is a monomial matrix $\mu \in \mathcal{M}_n(q)$, and the public key pk consists of the generator matrices $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ and $\mathbf{G}' = \mathsf{RREF}(\mu(\mathbf{G}))$ for two codes $\mathcal{C}$ and $\mathcal{C}'$ that are linked by the secret key isometry. The elements of the ID protocol are as follows:

- The commitment randomness of the ID protocol is a monomial map $\tilde{\mu} \in \mathcal{M}_n(q)$.
- For the commitment com, the prover computes the canonical form of $\tilde{\mu}(\mathbf{G})$ and applies the hash function H. The result is then sent to the verifier.
- The verifier's challenge ch is a single bit.
- Given the challenge $\mathsf{ch} \in \{0, 1\}$, the response resp is either the commitment randomness $\tilde{\mu}$ or a representative J for the coset of $\pi \circ \tilde{\mu} \circ \mu^{-1}$ regarding F, which is the first k entries of the permutation component. Here, π, roughly speaking, is an adjustment ensuring that $\pi \circ \tilde{\mu}(\mathbf{G})$ is in the systematic form.

Depending on the parameter set, this ID protocol is repeated in LESS's signature generation up to 345 times to ensure a small soundness error. At the same time,

the number of repetitions with $\mathsf{ch} = 1$ is fixed. Also, by specification, LESS utilizes the multi public key mechanism [7] to achieve smaller signature sizes. As detailed in Section 3.5, our results can easily be adapted to these refinements. An overview of the parameters relevant for our attack is given in Table 2.

Parameter set	Security Level	n	# Rep. of Σ per Signature	of which with $\mathsf{ch} = 1$	# Secret keys
LESS-252-192	NIST-I	252	192	36	1
LESS-252-68	NIST-I	252	68	42	3
LESS-252-45	NIST-I	252	45	34	7
LESS-400-220	NIST-III	400	220	68	1
LESS-400-102	NIST-III	400	102	61	3
LESS-548-345	NIST-V	548	345	75	1
LESS-548-137	NIST-V	548	137	79	3

Table 2: Relevant elements of the LESS parameter sets as specified in [4].

2.4 Partial Secret Key Recovery in LESS

The secret key structure in LESS allows for partial key recovery. If as much as half of all entries in the permutation π of the secret key μ is known, an attacker can reconstruct all missing values of the secret monomial. This partial key attack property is implicitly used in the literature for the construction [12, Prop 10]. We develop the approach from a different perspective in Section A.

3 Efficiently Solving the LESS Hidden Number Problem

In this section, we investigate the LESS variant of the Hidden Number Problem (LESS HNP). We begin by formally defining the LESS HNP and then introduce our efficient attack for solving it in Section 3.2. For simplicity and notational convenience, we first study our Hidden Number Problem for a slightly simplified variant of LESS with more side-channel information than needed at minimum, but show in the later sections how to adapt our results to more general and realistic scenarios.

3.1 The LESS HNP

We define the LESS HNP via the game $\mathrm{G}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A})$ as shown in Figure 2. The task is to recover the secret key of a LESS instance when given the transcripts

of the ID protocol $\mathsf{trans} = (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ along with some leakage of the commitment randomness $\tilde{\mu}$. As described in the preliminaries, $\tilde{\mu}$ is represented by a tuple of its permutation and its diagonal matrix: $\tilde{\mu} \simeq (\tilde{\pi}, \tilde{\Delta})$. There are four notes to make regarding the definition of the game.

Game $\mathrm{G}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A})$	$\underline{\mathsf{LeakLESS}(\mathsf{pk}, \mathsf{sk})}$
00 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LESSKeyGen}(\lambda)$	04 **parse** sk **as** μ
01 **parse** sk **as** $\mu = (\pi, \Delta)$	05 **parse** pk **as** $(\mathbf{G}, \mathbf{G}')$
02 $\pi' \leftarrow \mathcal{A}^{\mathsf{LeakLESS}(\mathsf{pk},\mathsf{sk})}(\mathsf{pk})$	06 $\tilde{\mu} \leftarrow \mathcal{M}_n(q)$
03 **return** $[\![\pi' = \pi]\!]$	07 $\mathsf{trans} = (\mathsf{com}, \mathsf{resp}) \leftarrow (\mathsf{H}(\mathsf{CF}(\mathbf{A})), J)$
	08 **parse** $\tilde{\mu}$ **as** $(\tilde{\pi}, \tilde{\Delta})$
	09 **return** $(\mathsf{trans}, \mathsf{MSB}(\tilde{\pi}))$

Fig. 2: LESS Hidden Number Problem G^{HNPLESS}. Here, $\mathsf{MSB}(\tilde{\pi})$ denotes the n-dimensional array containing the MSBs of the n entries of $\tilde{\pi}$.

1. We consider only the case where the leakage occurs in the permutation part $\tilde{\pi}$ of the commitment randomness.
2. We require the adversary to compute only the permutation part π of the secret key $\mu \simeq (\pi, \Delta)$. This is, however, no limitation since recovering the diagonal part Δ from π can be done efficiently: Let $\mathbf{P}$ and $\mathbf{D}$ be the two matrices corresponding to π and Δ, such that $\mathsf{RREF}(\mathbf{GDP}) = \mathbf{G}'$. Given π (and thereby $\mathbf{P}$), we compute a parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ for $\mathbf{G}'$, i.e., $\mathbf{G}'\mathbf{H}^T = 0$, to obtain the equation $\mathbf{GDPH}^T = 0$ in the unknown $\mathbf{D}$. This defines an efficiently solvable linear system of equations with $k(n-k)$ equations and n unknowns, in which the n diagonal entries of $\mathbf{D}$ are (with high probability) the unique solution.
3. Additionally, the adversary $\mathcal{A}^{\mathsf{LeakLESS}}$ is given only transcripts for protocol executions where $\mathsf{ch} = 1$, meaning the response is of the form $\mathsf{resp} := J = \mathsf{CosetRep}(\pi \circ \tilde{\mu} \circ \mu^{-1})$. Responses corresponding to $\mathsf{ch} = 0$, namely those consisting of the commitment randomness $\mathsf{resp} = \tilde{\pi}$ do not depend on the secret key – and thus cannot provide any useful leakage to the adversary. Again, this restriction is reasonable, since, by specification, LESS fixes the number of ID protocol runs with $\mathsf{ch} = 1$ per signature (see Table 2).
4. For the simplicity of presentation, we assume that the MSBs of *all* entries of $\tilde{\pi}$ leak simultaneously. We will later explain in Section 3.5, all our results on the LESS HNP can be extended to the more general setting, where only the MSB of a *single* random entry of $\tilde{\pi}$ leaks per execution of the ID protocol.

Definition 3.1 (LESS Hidden Number Problem). *Let $\lambda \in \mathbb{N}$ be a security parameter and let the game $\mathrm{G}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A})$ be defined as in Figure 2. We define the advantage of an adversary $\mathcal{A}$ winning the game as*

$$\mathsf{Adv}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A}) := \Pr[\mathrm{G}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A}) \Rightarrow 1].$$

3.2 An Efficient Attacker

In the following, we describe our attack on $\mathrm{G}^{\mathsf{HNPLESS}}$.

Theorem 3.2. *There is a polynomial time adversary $\mathcal{A}$ against the LESS Hidden Number Problem $\mathrm{G}^{\mathsf{HNPLESS}}$ making $\kappa \in \mathbb{N}$ queries to* LeakLESS *such that*

$$\mathsf{Adv}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A}) > 1 - 2^{-\kappa} \cdot n^2.$$

In particular, for $\kappa \geq 2\log(n) + 1$, $\mathcal{A}$ has success probability at least $1/2$.

$\mathcal{A}^{\mathsf{LeakLESS}(\mathsf{pk},\mathsf{sk})}(\mathsf{pk})$

10 $\mathsf{MEAS} \leftarrow \emptyset$
11 **repeat** κ times
12 $\quad(\mathsf{trans}, \mathsf{MSB}(\tilde{\pi})) \leftarrow \mathsf{LeakLESS}(\mathsf{pk}, \mathsf{sk})$
13 $\quad$**parse** trans **as** (hash, J)
14 $\quad\mathsf{MEAS} \leftarrow \mathsf{MEAS} \cup \{(J, \mathsf{MSB}(\tilde{\pi}))\}$
15 **for** $i \in [n]$
16 $\quad j \leftarrow 1$
17 $\quad$**while** $\mathcal{B}_{\mathsf{MEAS}}(i, j) = False$
18 $\quad\quad j \leftarrow j + 1$
19 $\quad\pi'_i \leftarrow j$
20 **return** $(\pi'_1, \ldots, \pi'_n)$

$\mathcal{B}_{\mathsf{MEAS}}(i, j)$

21 **for** $(J, \mathsf{MSB}(\tilde{\pi})) \in \mathsf{MEAS}$
22 $\quad$**if** $[\![\mathsf{MSB}(\tilde{\pi})_i = 0]\!] \neq [\![j \in J]\!]$
23 $\quad\quad$**return** $False$
24 **return** $True$

Fig. 3: Adversary $\mathcal{A}^{\mathsf{LeakLESS}(\mathsf{pk},\mathsf{sk})}$ from Theorem 3.2, using distinguisher $\mathcal{B}_{\mathsf{MEAS}}$ as a sub-routine.

Proof. Let pk be a LESS public key with corresponding secret key $\mathsf{sk} = \mu \simeq (\pi, \Delta)$, where $\pi = (\pi_1, \ldots, \pi_n) \in S_n$. As described in Figure 3, our adversary $\mathcal{A}$ first collects κ independent leakage transcripts from LeakLESS, i.e., $\mathsf{MEAS} = \{(J^{(t)}, \mathsf{MSB}(\tilde{\pi})^{(t)})\}_{t \in [\kappa]}$, and then uses the predicate $\mathcal{B}_{\mathsf{MEAS}}$ to test candidates $(i, j) \in [n]^2$. To help following the proof, we provide an illustration of the concept behind $\mathcal{B}$ in Figure 4.

Correct pairs always pass. Fix $(i, j) \in [n]^2$ with $\pi_i = j$. For any single measurement $(J, \mathsf{MSB}(\tilde{\pi})) \in \mathsf{MEAS}$, by construction of the LESS transcript for challenge $\mathsf{ch} = 1$, the membership of $j = \pi_i$ in J is determined by whether column i is mapped by $\tilde{\mu}$ into the information set (of $\tilde{\mu}(\mathbf{G})$), which in our leakage model is captured by $\mathsf{MSB}(\tilde{\pi})_i$. Concretely, the following equivalence holds deterministically for the correct image index $j = \pi_i$:

$$[\![\mathsf{MSB}(\tilde{\pi})_i = 0]\!] \;=\; [\![j \in J]\!].$$

Hence $\mathcal{B}_{\mathsf{MEAS}}(i, j)$ never aborts, and therefore

$$\Pr\left[\mathcal{B}_{\mathsf{MEAS}}(i, j) = True \mid \pi_i = j\right] = 1.$$

Incorrect pairs pass with probability about $2^{-\kappa}$. Fix $(i,j) \in [n]^2$ with $\pi_i \neq j$. Consider a single measurement $(J, \mathsf{MSB}(\tilde{\pi}))$. Since LESS uses parameters with $n = 2k$, the set J has fixed size $|J| = n/2$. Moreover, $\tilde{\mu}$ is sampled freshly and independently in each invocation, and thus the induced set J behaves (for our purposes) like a uniformly random size-$n/2$ subset of $[n]$. In particular, for a fixed incorrect index $j \neq \pi_i$ we have $\Pr[j \in J] \approx \frac{|J|}{n} = \frac{1}{2}$ and $\Pr[j \notin J] \approx \frac{1}{2}$. Therefore, conditioning on the leaked bit $\mathsf{MSB}(\tilde{\pi})_i$, the equality $[\![\mathsf{MSB}(\tilde{\pi})_i = 0]\!] = [\![j \in J]\!]$ holds with probability of approximately $1/2$, i.e.,

$$\Pr\left[\mathcal{B}_{\mathsf{SINGLE}}(i,j) = True \mid \pi_i \neq j\right] \approx \frac{1}{2},$$

where $\mathcal{B}_{\mathsf{SINGLE}}$ denotes $\mathcal{B}_{\mathsf{MEAS}}$ for $|\mathsf{MEAS}| = 1$. Since the κ measurements are independent, $\mathcal{B}_{\mathsf{MEAS}}(i,j)$ requires the above condition to hold for all $t \in [\kappa]$, and thus

$$\Pr\left[\mathcal{B}_{\mathsf{MEAS}}(i,j) = True \mid \pi_i \neq j\right] \approx \left(\frac{1}{2}\right)^{\kappa} = 2^{-\kappa}.$$

From the distinguisher to key recovery. The adversary $\mathcal{A}$ outputs, for each $i \in [n]$, the first $j \in [n]$ such that $\mathcal{B}_{\mathsf{MEAS}}(i,j) = True$. An error occurs only if there exists an incorrect $j \neq \pi_i$ that passes $\mathcal{B}_{\mathsf{MEAS}}(i,j)$. By a union bound over all $i \in [n]$ and all incorrect $j \in [n] \setminus \{\pi_i\}$, we obtain

$$\begin{aligned}
\mathsf{Adv}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A}) &\geq 1 - \sum_{i \in [n]} \sum_{j \in [n] \setminus \{\pi_i\}} \Pr\left[\mathcal{B}_{\mathsf{MEAS}}(i,j) = True \mid \pi_i \neq j\right] \\
&> 1 - n(n-1) \cdot 2^{-\kappa} \\
&> 1 - n^2 \cdot 2^{-\kappa},
\end{aligned}$$

as claimed. In particular, for $\kappa \geq 2\log(n) + 1$ we have $n^2 \cdot 2^{-\kappa} \leq 1/2$ and hence $\mathsf{Adv}_{\lambda}^{\mathsf{HNPLESS}}(\mathcal{A}) \geq 1/2$. □

3.3 Even Less Leakage: Extension to Single-Bit Leakage

As discussed above, our game $\mathrm{G}^{\mathsf{HNPLESS}}$ describes a slightly simplified model: Figure 3 assumes that $\mathsf{MSB}(\tilde{\pi})_i$ is known for each i in every measurement. To adapt the adversary to the case where only the MSB of a single entry is known, we add an intermediate step to the distinguisher that separates our measurements regarding the leaking index. The modified algorithm is given in Figure 5.

Obviously, one needs more queries to the leakage function than just a factor of n, as we now require κ measurements *per index*. By the (generalized) coupon collector problem [25], the expected number of queries is increased by a factor of $(n\kappa + \Theta(n\sqrt{\kappa \log(n)} + O(n\log(n))))/\kappa$. However, recall from Section 2.4 that we can reconstruct the full secret key from only half of the secret key permutation. Using the partial key recovery approach, the attack can be successfully conducted in the single bit setting with significantly less signatures.

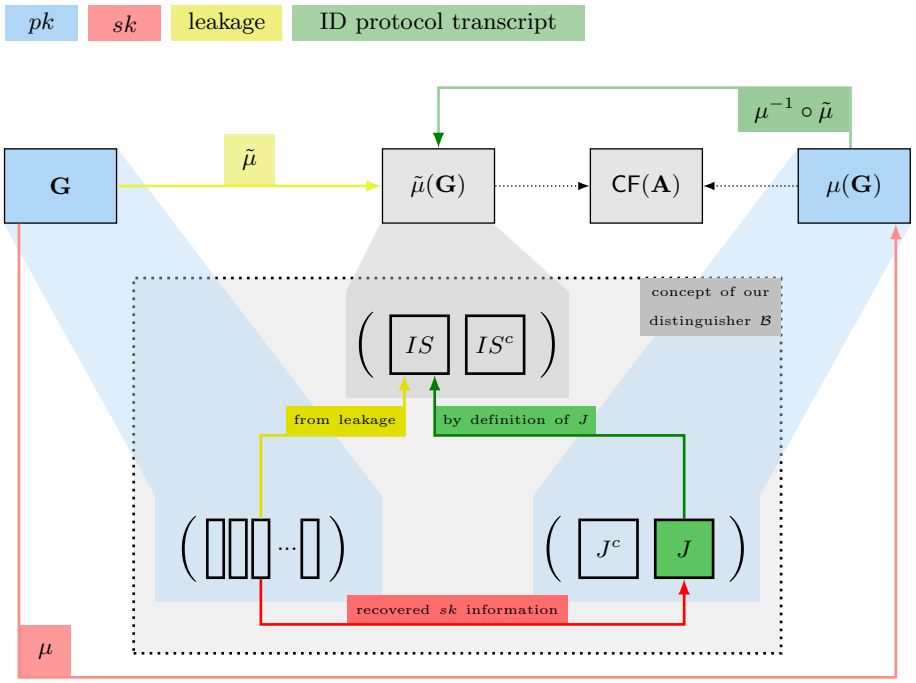


Fig. 4: Illustration of the concept of our distinguisher $\mathcal{B}$ in the context of the LESS proof of knowledge (cf. [4, Fig. 3]).

$\mathcal{A}^{\mathsf{LeakLESSsingle(pk,sk)}}(\mathsf{pk})$

for $i \in [n]$
  $\mathsf{MEAS}_i \leftarrow \emptyset$
while $\exists i : |\mathsf{MEAS}_i| < \kappa$
  $(i, \mathsf{trans}, \mathsf{MSB}(\tilde{\pi}_i)) \leftarrow \mathsf{LeakLESSsingle}(\mathsf{pk}, \mathsf{sk})$
  parse trans **as** (hash, J)
  $\mathsf{MEAS}_i \leftarrow \mathsf{MEAS}_i \cup \{(J, \mathsf{MSB}(\tilde{\pi}_i))\}$
for $i \in [n]$
  $j \leftarrow 1$
  while $\mathcal{B}_{\mathsf{MEAS}}(i, j) = False$
    $j \leftarrow j + 1$
  $\pi'_i \leftarrow j$
return $(\pi'_1, \ldots, \pi'_n)$

$\mathsf{LeakLESSsingle}(\mathsf{pk}, \mathsf{sk})$

$i \leftarrow [n]$
parse sk **as** μ
parse pk **as** $(\mathbf{G}, \mathbf{G}')$
$\tilde{\mu} \leftarrow \mathcal{M}_n(q)$
$\mathsf{trans} = (\mathsf{com}, \mathsf{resp}) \leftarrow (\mathsf{H}(\mathsf{CF}(\mathbf{A})), J)$
parse $\tilde{\mu}$ **as** $(\tilde{\pi}, \tilde{\Delta})$
return $(i, \mathsf{trans}, \mathsf{MSB}(\tilde{\pi}_i))$

$\mathcal{B}_{\mathsf{MEAS}}(i, j)$

for $(J, \mathsf{MSB}(\tilde{\pi}_i)) \in \mathsf{MEAS}_i$
  if $[\![\mathsf{MSB}(\tilde{\pi})_i = 0]\!] \neq [\![j \in J]\!]$
    return $False$
return $True$

Fig. 5: Adversary $\mathcal{A}^{\mathsf{LeakLESSsingle(pk,sk)}}$, adapted to the single bit leakage scenario.

3.4 The Other Extreme: A Note on Full Randomness Leakage

As defined in the previous Section 3.1, our LESS HNP assumes knowledge of only single bits of the randomness permutation (even in the simplified model). We showed that this is already sufficient to recover the secret key. However,

in our practical attack realization (Section 5) we achieve even more leakage: With the timing side-channel that we identified, we are able to recover *all* of the permutation $\tilde{\mu}$. Obviously, this enables our MSB-based attack, but we assume that it might open the door for a simpler and more efficient attack (which is out of scope of this work).

3.5 Further Subtleties

There are two more notes to make regarding the practicality of the attack from the theoretical viewpoint: So far we ignored the multi public key mechanism of LESS, and we always assumed *perfect leakage*, meaning that if we get a bit of side-channel information, we can be sure it is correct.

Multi Public Key Mechanism. In practice, LESS uses $s \in \{1, 3, 7\}$ secret keys at once to reduce the number of repetitions of the ID protocol. In order to recover all of the s secret keys, it is necessary to increase the number of queries by a factor of $\approx s$. However, recovering one key alone is already sufficient to break existential unforgeability.

Non-Perfect Leakage. In practice, the leakage of the MSB of an entry in $\tilde{\pi}$ will not always be perfect. As our experimental results show in Section 5, an error of 5% in the leaked bit is a realistic assumption, which leads to about 95% correctly recovered entries in π. Recall that π is a permutation and hence all entries from 1 to n appear exactly once. Assuming that, when the recovery fails, the erroneous entry in the recovered π is uniformly distributed, we can identify it as a repeatedly appearing number[2], meaning we end up with successfully knowing 90% of π. Again, using the partial key recovery (cf. Section 2.4) we know that 50% of π is sufficient for recovering the full key, hence the attack does not require perfect leakage.

4 Expected and Simulated Results

To better estimate the amount of signatures required in practice, we implemented our attack in SageMath with simulated perfect leakage and ran it against the standardized parameter sets. This was done for the single bit leakage as well as for the simplified case, where the MSB of all entries in $\tilde{\pi}$ leak, but without taking the partial key attack from [12] resp. Section A into account. The results are presented in Table 3 and compared to the expected numbers from the previous theoretical analysis.

Before starting the attack, every number between 0 and $N - 1$ is possible for every key byte. On average, each execution of the ID protocol reduces this

[2] This assumption is backed by all our experiments. In more than 100 tests, the complementary case, where two (or more) erroneous entries in the permutation swapped, happened only once. In that rare case, the error cannot be identified, the attack fails and needs to be repeated.

Parameter set	$\#\Sigma$ runs simplified		#Signatures simplified		$\#\Sigma$ runs single bit	#Signatures single bit
	exp.	sim.	exp.	sim.	sim.	sim.
LESS-252-192	8	20	1	1	5 440	151
LESS-252-68	8	20	1	2	5 440	388
LESS-252-45	8	20	2	4	5 440	1 120
LESS-400-220	9	21	1	1	9 293	137
LESS-400-102	9	21	1	1	9 293	457
LESS-548-345	10	27	1	1	13 588	182
LESS-548-137	10	27	1	1	13 588	516

Table 3: Expected number of (leaky) ID protocol runs resp. signatures necessary for our attack compared to the simulations. Simulation results are averaged over 1000 runs.

set of candidates for each key byte by half. Thus, we need expected $\log(N)$ executions to reduce the list from N to 1 candidate. For the tested parameter sets this results in $\lceil\log(252)\rceil = 8$, $\lceil\log(400)\rceil = 9$, resp. $\lceil\log(548)\rceil = 10$ runs of the ID protocol. Each signature contains between 34 and 79 executions of the ID protocol with $\mathsf{ch} = 1$, but also between 1 and 7 independent secret keys (depending on the chosen parameter set).

Consequently, per signature, there are on average $36/1 = 36$, $42/3 = 14$, $34/7 \approx 4.8$, $68/1 = 68$, $61/3 \approx 20.3$, $75/1 = 75$, resp. $79/3 \approx 26.3$ executions per secret key and thus, in theory, the total number of required signatures is bounded from below by $\left\lceil \frac{\lceil\log(252)\rceil}{34/7} \right\rceil = 2$ for LESS-252-45, and $\lceil \frac{8}{36} \rceil = \lceil \frac{8}{14} \rceil = \lceil \frac{9}{68} \rceil = \lceil \frac{9}{20} \rceil = \lceil \frac{10}{75} \rceil = \lceil \frac{10}{26} \rceil = 1$ for all other parameter sets. In the simulations, we see slightly higher values than theoretically expected as sometimes less than half of the key candidates are eliminated by one execution.

For the generalized case, in which only on single bit per execution of Σ leaks, those numbers have to be multiplied with the expectation value from the coupon collector problem as described in Section 3.3.

Additionally, we simulated the attack with varying degrees of the leakage quality. Figure 6 illustrates the relation between the number of needed (leaky) signatures for recovering the full secret key and the error rate of the leaked bit. As expected, the number of required executions grows exponentially with the deterioration of the leakage information. Our experiments showed an error rate of around 5%, however, for larger rates could still leverage the partial key recovery attack from Section 2.4 to minimize the required number of signatures.

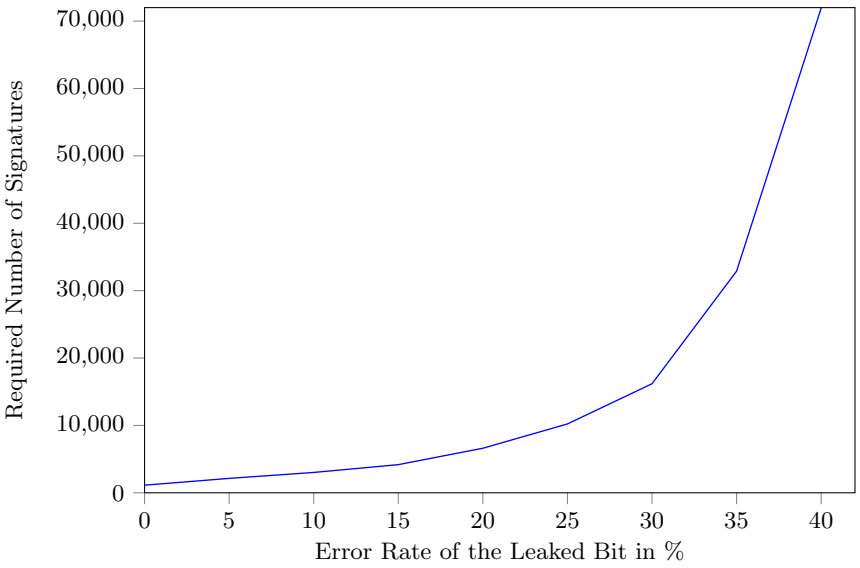


Fig. 6: Attack simulation with varying quality of the side-channel information for a full secret key recovery.

5 Practical Results

Recall that the practical attacks presented in this section require leakage of (one of) the MSBs of the commitment randomness permutation $\tilde{\pi}$. The commitment randomness is drawn randomly with each invocation of the ID protocol. Thus, we must recover the MSB of $\tilde{\pi}$ from one execution, meaning our attack needs to be a single-trace attack.

For this, we identified two points of interest in the computation of the commitment (as it is done in the reference implementation) that provide the required leakage: In Section 5.2, we mount a successful power side-channel attack when $\tilde{\mu}(\mathbf{G})$ is calculated. In addition, we exploit a timing side-channel during the computation of the systematic form $\mathsf{RREF}^*(\tilde{\mu}(\mathbf{G}))$ in Section 5.1.

Note that there is a (seemingly obvious) third possibly exploitable operation: At the start of each ID protocol invocation, the commitment randomness is newly generated. In the reference implementation, this is done using the Fisher-Yates (FY) algorithm. The vulnerabilities of sampling random vectors are not LESS-specific and have been investigated in several works already, e.g. [19,22,28]. We therefore refrain from discussing this potentially leaking operation in further detail.

Set-Up. For both approaches, we compiled and executed the reference implementation with the `LESS-252-45` parameter set[3] on the ChipWhisperer ARM Target

[3] The attack can likewise be run on any of the parameter sets without substantial differences.

Algorithm 1: Non-Constant-Time Code Fragment in `LESS.c`

```
1  piv_idx = 0;
2  for col_idx := 0 to N do
3  |   row_idx := 0;
4  |   for t := 0 to N do
5  |   |   if μ̃.permutation[t] == col_idx then
6  |   |   |   row_idx = t;
7  |   |   |   break;
8  |   |   end
9  |   end
10 |   ...
11 end
```

Board [1], which deploys an STM32F303RCT6 microcontroller. The chip runs at a clock frequency of 7.384615 MHz, and the connected PicoScope oscilloscope samples the power consumption over an amplifier.

Naming Convention. The reference implementation defines a structure to handle monomials. Hence, a monomial $\tilde{\mu}$ in the code consists of two variables, namely $\tilde{\mu}.permutation$ and $\tilde{\mu}.coefficients$. To allow better consistency with the reference implementation, we use those designations for the rest of this section, in particular we write $\tilde{\mu}.permutation$ for the commitment randomness permutation $\tilde{\pi}$. Additionally, we take over the capital letters N and K for the LESS parameters n and k.

5.1 Exploiting Non-Constant-Time-ness

Algorithm 1 shows the part of the implementation of RREF^* in the function `LESS_sign()` that is susceptible to a timing attack.

Attack. Due to the **break** in line 7 of Algorithm 1, the execution time of the inner **for**-loop depends directly on the content of $\tilde{\mu}.permutation$. Therefore, measuring the execution time of all N iterations of the outer **for**-loop reveals the exact values of all entries $\tilde{\mu}.permutation[t]$ in the permutation. As it is common practice in side-channel security research, we used a trigger signal to precisely determine the start and end of each iteration of the outer loop. This strong signal allowed us to sample at a relatively slow rate of 26.04 MHz. Nonetheless, obtaining the relevant time frames is also possible without an explicit trigger signal. With a higher sampling rate, the start and end of the loop could be identified by using signal processing methods such as correlation, making the attack a realistic threat.

Results. By applying this strategy, we correctly recover 100% of $\tilde{\mu}.permutation$ in all our experiments, resulting in perfect leakage of the information required for our attack[4]. This flawless recovery rate can be explained with the help of

[4] and even more, see note in Section 3.4

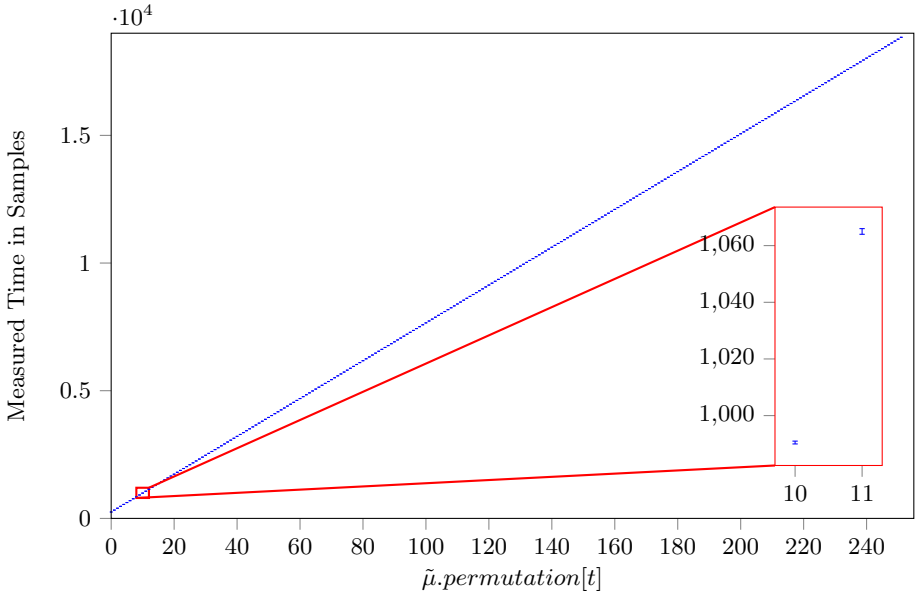


Fig. 7: Measured execution time and variance of all possible values for $\tilde{\mu}.permutation[t]$, averaged over 1500 runs of the ID protocol. The zoomed-in detail clearly shows that there is no overlap in measurements between two values for $\tilde{\mu}.permutation$.

Algorithm 2: Revised Algorithm 1 Code Snippet, Now Constant-Time

```
1 piv_idx = 0;
2 for col_idx := 0 to N do
3 |   row_idx := 0;
4 |   for t := 0 to N do
5 |   |   c = verify(μ̃.permutation[t], col_idx, len=1);
6 |   |   cmove(row_idx, t, c);
7 |   end
8 |   ...
9 end
```

Figure 7: It shows the ranges of the measured loop execution time of all possible values for $\tilde{\mu}.permutation$ over 1500 runs of the ID protocol. One can see that the difference between execution times of distinct values (x-axis) is much larger than the variation of measurements within one value. Therefore, distinguishing the values by the measured execution time is trivial.

Using this side-channel information, we are able to recover the complete secret key with the attack described in Section 3. Unsurprisingly, our results perfectly match the simulations evaluated in Section 4.

Mitigation. In the following, we propose a set of changes to the implementation to achieve cryptographically constant-time for Algorithm 1. A first change to equalize the execution time of the outer **for**-loop is to remove the **break** in line 7. Since the permutation array of $\tilde{\mu}$ contains all numbers from 0 to $N-1$ in random order, it can be removed without altering the functionality of the code. This modification results in a runtime of $N \cdot N$ iterations of the inner **for**-loop, doubling the average runtime of the initial solution.

Unfortunately, the resulting implementation would still not be constant-time in the cryptographic sense, due to the **if**-statement behaving differently depending on the secret. To avoid that, it needs to be replaced with a constant-time comparison for equality, followed by a conditional move. This particular comparison between two bit-arrays (in this case $\tilde{\mu}.permutation[t]$ and col_idx) already exists in the LESS reference codebase. It is implemented in the `verify()` function that can be found in `utils.c`. The bytes of the inputs are combined using a bit-wise exclusive or:

$$c := [a == b] \quad \Longleftrightarrow \quad a \oplus b = 0$$

The result of this comparison c must then be used in a conditional move `cmov`. According to [22], this can be implemented in constant-time by a sequence of Boolean instructions, e.g., $dest = (dest \cdot \overline{\mathbf{c}}) + (source \cdot \mathbf{c})$ (with $\mathbf{c}$ being a repetition of the bit c). The code fragment including the presented mitigations is shown in Algorithm 2.

Cautionary Note. The modified implementation is cryptographically constant-time, however, some of the adjustments may lead to a new power side-channel vulnerability: The mask $\mathbf{c}$ is either a bitstring of only zeros or a bitstring of only ones. In [16], the authors show that such a distribution of values is quite vulnerable to power side-channel attacks. Protecting against that kind of attacks would require more sophisticated hiding or masking countermeasures, as discussed in, e.g., [22].

5.2 Advanced Power Side-Channel Attack

In order to demonstrate that our attack is successful even if the previously presented timing vulnerability is prevented, we now examine the second identified point of interest in the reference implementation.

Analyzing the Target Operation. In the function `generator_monomial_mul` (found in `codes.c`), $\tilde{\mu}$ is read-out in order to compute the generator matrices for the commitments. This code part is illustrated in Algorithm 3.

The outer **for**-loop iterates over all indices of $\tilde{\mu}$ and thus, loads all entries in ascending order. Due to the inner **for**-loop, line 3 of Algorithm 3 is repeated K times for the same `src_col` index of $\tilde{\mu}$. The subsequent multiplication and modular operations are not relevant for revealing $\tilde{\mu}.permutation$. Consequently, observing the power consumption of reading $\tilde{\mu}.permutation[index]$ in line 3 can

Algorithm 3: generator_monomial_mul from codes.c

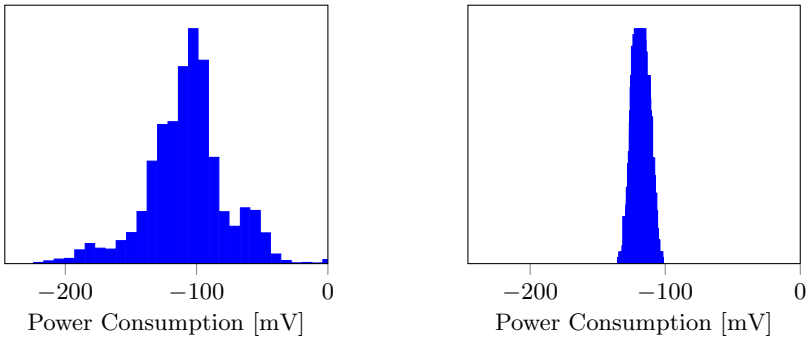
```
1 for src_col := 0 to N do
2 |   for row := 0 to K do
3 |   |   target_idx = μ̃.permutation[src_col];
4 |   |   res[row][target_idx] = G[row_idx][src_col] · μ̃.coefficients[src_col]
  |   |     mod q;
5 |   end
6 end
```

reveal information on the content of the permutation—the side-channel leakage required for our attack. To achieve high precision measured data, we sample the power consumption with 1.25 GHz sample rate.

A common observation in side-channel attacks is the Hamming Weight (HW) leakage, meaning that the observable leakage reflects the HW of the processed value. Our attack exploits the fact that a HW of zero for $\tilde{\mu}.permutation[t]$ implies that the MSB is also zero. Therefore, if we can distinguish the entry in $\tilde{\mu}.permutation$ with HW 0, we identified an entry with $\mathsf{MSB} = 0$.

Using a Horizontal Attack. In reality, the leaked HW value is obscured by noise, which is mostly additive and Gaussian distributed. This is also the case for our measurements: Figure 8a shows the distribution of the observed leakage at a fixed index, a fixed sample point, and a fixed HW. The overall distribution closely resembles a Gaussian distribution.

Using those measurements would result in a higher error rate and a weaker attack. However, reducing the noise is possible by applying a horizontal attack. In this attack, the average is taken over all K measurements of the iterations of the inner loop, where the same $\tilde{\mu}$ is processed. This reduces the variance of the noise by a factor of $\sqrt{K} = \sqrt{126} \approx 11$ as shown by [6]. Figure 8 illustrates this effect.



(a) Individual Traces. (b) Averaged Traces.

Fig. 8: Distribution of the Measured Power Consumption at Fixed Index [15] and Fixed Sample Point 550.

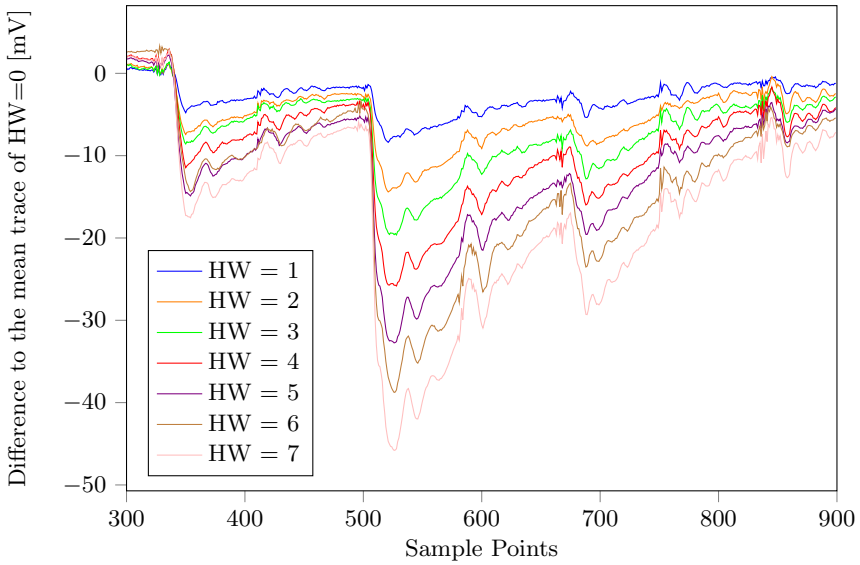


Fig. 9: Mean traces of the different possible values of HW($\tilde{\mu}.permutation[t]$).

Applying the Template Approach. The approximately symmetric noise distribution enables the use of mean templates for our single-trace attack. Since only a single measurement is available per execution of the identification protocol, the classification is based on the distance to precomputed mean template traces. To construct these templates, a profiling phase is performed in which a sufficiently large number of training traces is recorded. In our experiments, we use 15 000 profiling traces (corresponding to ID protocol runs) to obtain stable and low-noise mean traces. The templates are then generated by averaging all traces corresponding to the same HW of $\tilde{\mu}.permutation[t]$. The resulting mean traces (templates) are shown in Figure 9. Their analysis confirms the reasonability of the assumed HW leakage model, as the power consumption of the different mean traces decreases linearly with the HW.

Selecting the index for which the distance between the measured trace and the mean template for HW=0 is minimal should identify an index with MSB equal to zero. Figure 10 again shows the mean traces for the HWs zero and one, together with the variances of their distributions. Unfortunately, the figure indicates that the measurement variance is too large to reliably and consistently distinguish between HWs zero and one – the variance corridors overlap.

Nevertheless, since our goal is to recover the MSB, misclassifying indices with HW one as entry zero is acceptable in most cases. Assuming that we can always identify an index such that HW($\tilde{\mu}.permutation[index]$) ≤ 1, the only incorrectly classified value is 128, as it is the only value with HW one whose MSB equals one.

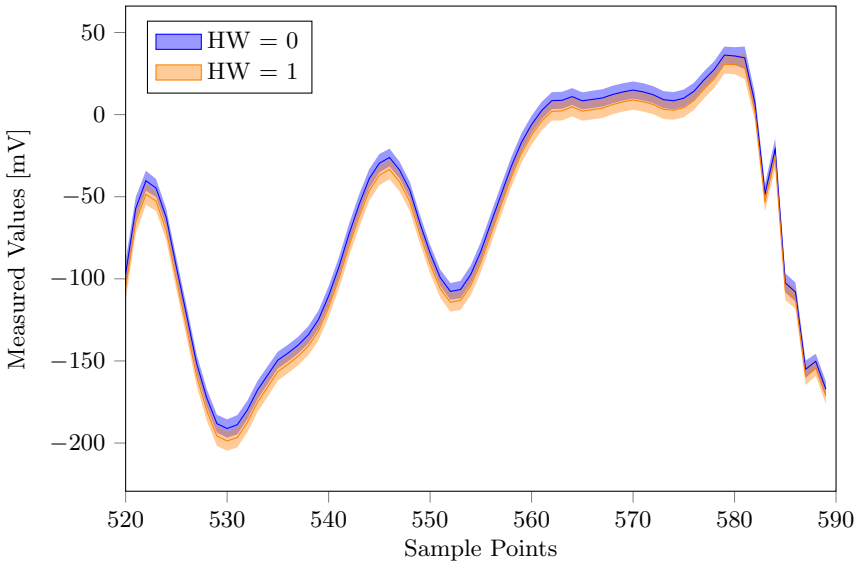


Fig. 10: Template Traces and Variances.

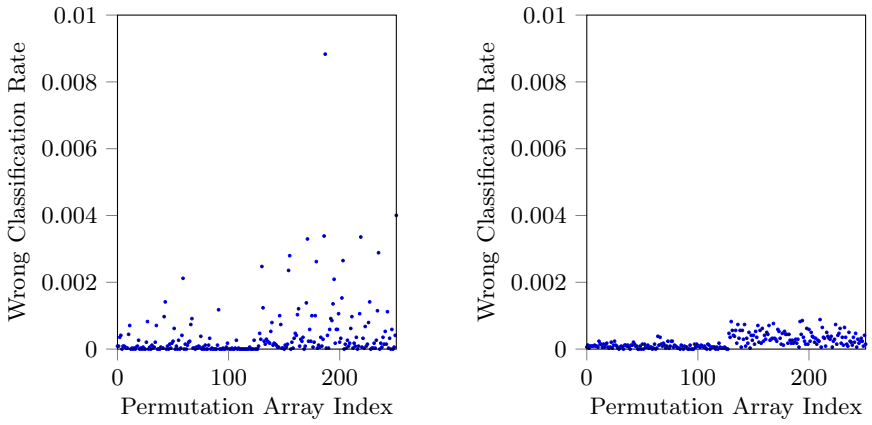
Improving with Counters. To further mitigate inaccuracies arising from the noisy measurements, our approach maintains a two-dimensional counter table. Algorithm 4 summarizes this method. Each row of the counter table corresponds to a secret-key position, and each column corresponds to a possible image under the secret permutation for that position. Whenever the classification of the side-channel information indicates that a certain secret-key image is inconsistent with the ID protocol transcripts (obtained by the resulting signature), the corresponding candidate is penalized by incrementing its counter. As a result, correct mappings are statistically penalized less often than incorrect ones. After processing a sufficient number of ID protocol runs and their traces, the attack selects, for each position, the candidate with the minimum counter, and the resulting permutation is finally inverted to recover the secret key.

Refining the Templates. The success rate of this attack method strongly depends on the recovery accuracy of the individual values of $\tilde{\mu}.permutation[t]$. Recall that Figure 6 illustrates the simulated number of signatures required for a successful attack as a function of the failure rate in recovering $\tilde{\mu}.permutation[t]$. For example, a failure rate of 10% implies that approximately 3000 signatures are required for the attack to succeed. When applying the HW templates to the side-channel traces, we observe a failure rate of less than 7%.

Unfortunately, our experimental results with one template for each possible HW showed that some secret key positions did not converge to a single candidate value. The underlying reason is illustrated in Figure 11a. We see, that the mis-

Algorithm 4: Permutation Recovery with Counter-Based Strategy

```
for secret-key column i := 0 to N − 1 do
    for possible image j := 0 to N − 1 do
        counter[i][j] = 0;
        Row i: secret input position
        Column j: candidate output position under the permutation
    end
end
for s := 0 to S − 1 do
    obtain side-channel trace for sample s;
    LowHWSet = classify_low_HW(trace);
    for secret-key column i := 0 to N − 1 do
        for possible image j := 0 to N − 1 do
            if j ∉ LowHWSet then
                counter[i][j] = counter[i][j] + 1;
            end
        end
    end
end
for i := 0 to N − 1 do
    perm_guess[i] = argmin_j(counter[i][j]);
end
secret_key_perm = invert(perm_guess);
```



(a) Applying One Template per Hamming Weight.

(b) Applying One Template per Hamming Weight *and* Index.

Fig. 11: Distribution of the Wrong Classification Rate for Different Template Sets.

classification errors are not uniformly distributed across all indices. Instead, some indices show significantly higher failure rates. This behavior can be explained by the fact that the side-channel leakage depends not only on $\tilde{\mu}.permutation[t]$ but also on the index t itself. As a consequence, the leakage represents a superposition of value-dependent and index-dependent leakage, which biases the template matching for specific indices. This non-uniform error distribution poisons the convergence of the counter-based recovery.

To counter this effect, we built individual HW templates for each index value between 0 and N. This requires more training data but results in a significantly more uniformly distributed misclassification rate of less than 5%. Figure 11b shows the failure rate of the classification using index-specific templates.

Applying these templates, we are able to recover more than 95% of the key bytes using fewer than 2 000 signatures. Our attack is successful, as the remaining missing bytes can be recovered as described in Section 2.4.

Mitigation. The question arises whether this side-channel vulnerability can be mitigated as easily as the timing attack described in Section 5.1. The most robust countermeasure is masking *all* secret-dependent values, thereby eliminating the dependency between the side-channel leakage and the secret. Techniques for securely masking constant-time comparison, conditional move, and the Fisher–Yates shuffle are discussed in [22]. However, masking `generator_monomial_mul` (Algorithm 3) remains an open and non-trivial problem. In particular, $\tilde{\mu}.permutation$ cannot simply be masked and used afterwards, since it serves as an index for the result array. As with all masking schemes, a secure solution is expected to incur a considerable implementation overhead.

A more simple, but also less secure implementation change to counter our attack would be to shift the index range by a constant from $[0, \ldots, N-1]$ to $[0+c, \ldots, N-1+c]$. This modification eliminates zero from the set of potentially observable indices. If $c \geq 5$, the probability that the MSB equals zero is reduced. Consequently, learning the HW reveals significantly less information about the MSB than in the current implementation.

A similar effect can be achieved by randomizing the currently unused bits in the register. In the current implementation, only 8 bits are used to store and process an index. However, most modern micro-controllers employ at least a 32-bit architecture, leaving 24 bits unused for random switching. This increases the switching noise and makes it substantially more difficult to distinguish between HW zero and non-zero.

Finally, randomizing the execution order of `generator_monomial_mul` strengthens the side-channel resistance, as it randomizes the index of the outer **for**-loop. The index then increases randomly, rather than linear, preventing an attacker from easily associating leakage with the currently processed value of $\tilde{\mu}.permutation[t]$. Although the loop index itself may still leak (due to the discussed index-dependence) and could, in principle, be exploited, perfectly recovering its value in practice is unlikely due to the leakage model and the presence of noise. Therefore, shuffling the outer **for**-loop significantly increases the attack complexity and thus improves the side-channel resistance.

Overall, our results demonstrate that the design of LESS and it's reference implementation exhibit exploitable single-trace leakage through the commitment randomness. While the lightweight countermeasures discussed above can reduce the information available to an attacker, achieving strong protection requires dedicated masking schemes. Our findings highlight the importance of carefully identifying all secret-dependent values in a cryptographic scheme and considering side-channel leakage as a potential information source.

Acknowledgements

The work described in this paper has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and by the German Federal Ministry of Education and Research BMBF through the projects POST (16ME2157) and DI-OCDCpro (16ME0942). Yi-Fu Lai is partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT – No. 101020788), by the Research Council KU Leuven grant C14/24/099 and by CyberSecurity Research Flanders with reference number VOEWICS02.

References

1. Cw303 arm target, https://chipwhisperer.readthedocs.io/en/latest/Targets/CW303%20Arm.html
2. Aranha, D.F., Novaes, F.R., Takahashi, A., Tibouchi, M., Yarom, Y.: LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 225–242. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417268
3. Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: Algorithm specifications and supporting documentation v3.1, https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf
4. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., O.Saarinen, M.J., Santini, P., Wallace, R.: Less: Linear equivalence signature scheme, https://www.less-project.com/LESS-2025-02-07.pdf
5. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., O.Saarinen, M.J., Santini, P., Wallace, R.: Less reference implementation, https://github.com/less-sig/LESS, https://github.com/less-sig/LESS
6. Battistello, A., Coron, J.S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 23–39. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53140-2_2
7. Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 464–492. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_16

8. Bleichenbacher, D.: On the generation of one-time keys in dl signature schemes. Presentation at IEEE P1363 Working Group Meeting (November 2000)
9. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 129–142. Springer, Berlin, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_11
10. Call for additional digital signature schemes for the post-quantum cryptography standardization process. https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures, accessed: 30.04.2024
11. Checkoway, S., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H., Fredrikson, M.: On the practical exploitability of dual EC in TLS implementations. In: Fu, K., Jung, J. (eds.) USENIX Security 2014. pp. 319–335. USENIX Association (Aug 2014), https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/checkoway
12. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Cryptology ePrint Archive, Report 2023/1533 (2023), https://eprint.iacr.org/2023/1533
13. Cve-2024-31497. https://nvd.nist.gov/vuln/detail/CVE-2024-31497, accessed: 30.04.2024
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
15. Hermelink, J., Ning, K.C., Petri, R.: Finding and protecting the weakest link: On side-channel attacks on y in masked ml-dsa. In: Annual International Cryptology Conference. pp. 3–37. Springer (2025)
16. Hesse, D., Feldtkeller, J., Güneysu, T., Hermelink, J., Land, G., Krausz, M., Richter-Brockmann, J.: t-probing (in-)security - pitfalls on noise assumptions. Cryptology ePrint Archive, Report 2025/1202 (2025), https://eprint.iacr.org/2025/1202
17. Howgrave-Graham, N., Smart, N.P.: Lattice attacks on digital signature schemes. DCC **23**(3), 283–290 (2001). https://doi.org/10.1023/A:1011214926272
18. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_27
19. Karabulut, E., Alkim, E., Aysu, A.: Single-trace side-channel attacks on ω-small polynomial sampling: With applications to ntru, ntru prime, and crystals-dilithium. In: 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 35–45. IEEE (2021)
20. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual international cryptology conference. pp. 388–397. Springer (1999)
21. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Annual international cryptology conference. pp. 104–113. Springer (1996)
22. Krausz, M., Land, G., Richter-Brockmann, J., Güneysu, T.: A holistic approach towards side-channel secure fixed-weight polynomial sampling. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 94–124. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31371-4_4

23. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
24. Merget, R., Brinkmann, M., Aviram, N., Somorovsky, J., Mittmann, J., Schwenk, J.: Raccoon attack: Finding and exploiting most-significant-bit-oracles in TLS-DH(E). In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 213–230. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/merget
25. Newman, D.J.: The double dixie cup problem. American Mathematical Monthly **67**, 58 (1960)
26. Nguyen, P.Q., Shparlinski, I.E.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. DCC **30**(2), 201–217 (2003). https://doi.org/10.1023/A:1025436905711
27. Oswald, E.: Enhancing simple power-analysis attacks on elliptic curve cryptosystems. In: Kaliski Jr., B.S., Koç, Çetin Kaya., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 82–97. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/3-540-36400-5_8
28. Puškáč, L., Benovič, M., Breier, J., Hou, X.: Make shuffling great again: A side-channel-resistant fisher–yates algorithm for protecting neural networks. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2025)
29. Saeed, M.A.: Algebraic approach for code equivalence. Ph.D. thesis, Normandie Université; University of Khartoum (2017)
30. Schnorr, C.P.: Efficient identification and signatures for smart cards (abstract) (rump session). In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89. LNCS, vol. 434, pp. 688–689. Springer, Berlin, Heidelberg (Apr 1990). https://doi.org/10.1007/3-540-46885-4_68
31. Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on AES: Combining side channel- and differential-attack. In: Joye, M., Quisquater, J.J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28632-5_12
32. Steffen, H.M., Land, G., Kogelheide, L.J., Güneysu, T.: Breaking and protecting the crystal: Side-channel analysis of Dilithium in hardware. In: Johansson, T., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023. pp. 688–711. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-40003-2_25

– Supplementary Material –

A Partial Secret Key Recovery

We show how to efficiently obtain the full monomial matrix when only partial information are given.

Let $(\mathbf{G}_0, \mathbf{G}_1)$ be the LESS public key where $\mathbf{G}_0 = (\mathbf{I}_k|\mathbf{G}'_0)$ and with a uniformly-random-drawn secret key $\mathbf{Q}_s$ such that $\mathbf{U}\mathbf{G}_0\mathbf{Q}_s = \mathbf{G}_1$ for $\mathbf{U} \in \mathsf{GL}_k(\mathbb{F}_q)$. We recall the secret key of LESS is a monomial $\mathbf{Q}_s = \mathbf{D}_s\mathbf{P}_s$ represented as (π, Δ) with $\pi = (\pi_1, ..., \pi_n)$ being the permutation part of s and $\Delta = (\Delta_1, ..., \Delta_n)$ being the diagonal part of s. For simplicity of the presentation, we assume $n = 2k$ and $\mathbf{G}_1 = (\mathbf{I}_k|\mathbf{G}'_1)$.

In this section, we present an argument showing that knowing each of the following partial information about the secret key is sufficient to recover the entire secret key of LESS with an overwhelming probability:

1. Arbitrary k entries of π and Δ.
2. Arbitrary $k+1$ entries of π.

We start by introducing three essential propositions of the Kronecker product for cryptanalysis [29].

Proposition A.1. *For a matrix multiplication* $\mathbf{A}_1\mathbf{G}\mathbf{A}_2^T = \mathbf{C}$*, we have* $(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{vec}(\mathbf{G}) = \mathbf{vec}(\mathbf{C})$ *where* $\mathbf{vec}(\cdot)$ *is the row-vectorization of a matrix and* $\otimes$ *is the Kronecker tensor product.*

Proposition A.2. $\mathsf{rank}(\mathbf{A}_1 \otimes \mathbf{A}_2) = \mathsf{rank}(\mathbf{A}_1) \cdot \mathsf{rank}(\mathbf{A}_2)$.

Proposition A.3. $(\mathbf{A}_1 \otimes \mathbf{A}_2)(\mathbf{B}_1 \otimes \mathbf{B}_2) = (\mathbf{A}_1\mathbf{B}_1 \otimes \mathbf{A}_2\mathbf{B}_2)$.

Regarding the key of LESS, since $\left(-\mathbf{G}_1'^T \middle| \mathbf{I}_k\right)$ is the parity check matrix, we know that:

$$\mathbf{G}_0\mathbf{Q}_s \left(-\mathbf{G}_1'|\mathbf{I}_k\right)^T = \left(\mathbf{I}_k|\mathbf{G}_0'\right)\mathbf{Q}_s \left(-\mathbf{G}_1'|\mathbf{I}_k\right)^T = \mathbf{0}.$$

Hence, we have:

$$\left(\mathbf{I}_k|\mathbf{G}_0'\right) \otimes \left(-\mathbf{G}_1'|\mathbf{I}_k\right)\mathbf{vec}(\mathbf{Q}_s) = \mathbf{0}, \tag{1}$$

where $\mathbf{0}$ represents the zero vector. We use this tool to show the partial key attacks on LESS. Each equation represents a system of $4k^2$ linear equations with rank k^2.

Intuitively, since a monomial matrix has only one non-zero coefficient in each row and column, knowing k and $k+1$ entries of π is equivalent to knowing $3k^2-k$ and $3k^2+k-2$ entries of $\mathbf{vec}(\mathbf{Q}_s)$, respectively. Thus, knowing k entries of both π and Δ reveals $3k^2$ entries of $\mathbf{vec}(\mathbf{Q}_s)$.

It suffices to show that after the substitution of the partial solution above into Equation (1), the remaining linear equations are still of sufficient rank to recover the secret key using simple linear algebra.

In our analysis, we assume the following heuristic assumption, which holds with a high chance in the experiments. **Heuristic Assumption.** Let $\mathbf{G}_1$ be a random generated public key as specified in LESS of rank k. For $J \subset [2k]$, we have $\mathsf{rank}(\mathbf{G}_0^J) = \mathsf{rank}(\mathbf{G}_1^J) = \min\{k, |J|\}$.

Theorem A.4. *Let* $\mathbf{G}_1$ *be a LESS public key for some unknown secret key* $sk = (\pi, \Delta)$*. Under the heuristic assumption stated above, given* k *entries of* π *and the corresponding* k *entries in* Δ*, we can recover* sk *with a good chance in polynomial-time*$(k, \log(q))$*. Moreover, the result holds the same when given* $k+1$ *entries of* π*.*

Proof. Let $J \subseteq [2k]$. Given the entries of index $j \in J$ in π and the corresponding diagonal entries in Δ, we can simplify the linear equation in Equation (1) by considering its submatrix. Specifically, we can represent the undetermined part of $\mathbf{vec}(\mathbf{Q}_s)$ as $\left(\sum_{i\in\bar{J}} \mathbf{E}_{i,i}\right) \otimes \left(\sum_{i\in\bar{J}} \mathbf{E}_{i,i}\right) \mathbf{vec}(\mathbf{Q}_s)$, where $\bar{J}$ is the complement of J over $[2k]$ and $\mathbf{E}_{i,j}$ denotes the elementary matrix of size $2k \times 2k$.

Thus, we can rewrite the matrix from Equation (1) as:

$$\begin{aligned} &((\mathbf{I}_k \mid \mathbf{G}'_0) \otimes (-\mathbf{G}'_1 \mid \mathbf{I}_k)) \left(\left(\sum_{i\in\bar{J}} \mathbf{E}_{i,i} \right) \otimes \left(\sum_{i\in\bar{J}} \mathbf{E}_{i,i} \right) \right) \\ = \; &\left((\mathbf{I}_k \mid \mathbf{G}'_0)^{\bar{J}} \otimes (-\mathbf{G}'_1 \mid \mathbf{I}_k)^{\bar{J}} \right). \end{aligned}$$

For the case where $|J| = k$, under the heuristic assumption, since $|J| = |\bar{J}| = k$, we can assume $\mathsf{rank}((\mathbf{I}_k \mid \mathbf{G}'_0)^{\bar{J}}) = \mathsf{rank}((-\mathbf{G}'_1 \mid \mathbf{I}_k)^{\bar{J}}) = k$. This allows us to recover the entire secret key by solving the remaining k^2 variables with k^2 linearly independent equations.

In the case where $|J| = k + 1$ but without the diagonal entries, we assume the resulting submatrix has rank $(k-1)^2$, as per the heuristic assumption. If the diagonal coefficients in Δ are not provided, we extend the submatrix by adding the corresponding columns. Each row associated with an undetermined diagonal coefficient contributes an independent pivot, resulting in a matrix of rank $(k-1)^2 + k + 1$. Given that there are also $(k-1)^2 + k + 1$ variables, we can solve the linear equations using linear algebra to recover the secret key. □

Remark A.5. Clearly, the partial key attack described above can be further improved. For example, even if only k permutation entries are known without the diagonal entries, the attack can still be applied by guessing an additional permutation entry, running the algorithm, and then verifying the correctness using the public key information.