Improving Skipping Fault Correction Attacks on Randomized Dilithium via MILP

Haobo Ouyang[1,2,3], Chaoran Wang[1,2,3], Guowei Liu[1,2,3], Lixuan Wu[1,2,3], Meiqin Wang[1,2,3], and Yanhong Fan[1,2,3,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
{lixuanwu,mqwang,yanhongfan}@sdu.edu.cn
{hbouyang,chaoranwang,guoweiliu}@mail.sdu.edu.cn
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao 266237, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[4] Quan Cheng Shandong Laboratory, Jinan, China

Abstract. Dilithium, as a quantum-secure digital signature standard in FIPS 204, has received widespread attention for its physical implementation security. NIST selected Dilithium's randomized signing mode as the default, which can mitigate the severe physical attacks that exploit the deterministic signing mode. However, the physical attack resilience of randomized signing mode is currently an open question. In 2024, Krahmer et al. demonstrated a key-recovery attack against randomized Dilithium by exploiting fault injection. The skipping fault correction attack in Krahmer et al.'s work benefits from operational simplicity in practical settings. Nevertheless, it uses a full-rank collection strategy, requiring several effective faults equal to the number of key coefficients.

By developing an optimized skipping fault correction attack, we prove the non-necessity of the full-rank collection strategy. Theoretically, we derive the minimum number of faults $M_{\min}$ and a key-dependent trend for reliable key recovery. For the $M_{\min}$, we obtain it by analyzing secret coefficient coverage and unique bounded solution probability under the random-row model. Lemma 1 guarantees high-probability coverage of all coefficient positions, while Lemmas 2– 4 and Theorem 1 prove sufficient uniqueness conditions. Furthermore, we instantiate $M_{\min}$ of Dilithium for security levels L2, L3, and L5 with a high key recovery probability. For the key-dependent trend, Remark 2 reveals that the secret key with a larger number of ± 2 in the coefficients (denoted as abs2) tends to be easier to recover, offering a feature-based insight into recovery efficiency. We formulate the key recovery of Dilithium as an MILP problem and propose an MILP model. Based on the MILP model, we design algorithms of adaptive skipping fault correction attacks for plain and shuffling settings to recover the key with fewer faults.

In our experiments, we explored factors influencing key recovery success, confirming theoretical predictions that success rates increase with more faults and larger abs2. Compared to Krahmer et al.'s work, our improved attack reduces the required number of faults for recovering the private

key. Specifically, for the plain setting, fault reductions were 25.9% (L2), 16.2% (L3), and 25.6% (L5). Similarly, for the shuffling setting, reductions were 25.6% (L2), 13.5% (L3), and 26.3% (L5).

Keywords: Lattice-based signatures · Dilithium · Fault Injection Attacks · Mixed Integer Linear Programming · Correction Fault Attacks

1 Introduction

Currently, emerging quantum technologies pose a potential long-term threat to widely deployed public-key cryptography, as Shor's algorithm could efficiently break foundational schemes based on integer factorization and discrete logarithms [26,18]. In response, NIST initiated the Post-Quantum Cryptography (PQC) project, aiming to prepare digital security for future challenges [21]. In 2022, NIST announced the first batch of algorithms selected for standardization during the PQC project [19]. Among the selected signature algorithms, CRYSTALS-Dilithium, standardized as ML-DSA in FIPS 204, features an optimal balance among provable security, comprehensive performance, implementation robustness, and deployment feasibility [20].

Dilithium's signing algorithm follows the Fiat–Shamir with aborts paradigm. It samples a nonce $\boldsymbol{y}$, forms the response $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$, and applies rejection sampling. As a result, the accepted signatures are close to being secret-independent in the random oracle model. Dilithium provides two signing modes [2], namely deterministic signing and randomized (or hedged) signing. The deterministic mode prevents nonce reuse attacks from poor randomness but is extremely vulnerable to differential fault attacks and has worse side-channel properties. Conversely, the randomized signing mode introduces fresh randomness in each signing operation, greatly enhancing robustness against physical attacks like fault and side-channel attacks.

In prior work, side-channel attacks on Dilithium can be broadly categorized into two main types based on their analysis method. Profiled Simple Power Analysis (SPA) attacks typically target intermediate values such as the nonce $\boldsymbol{y}$ [29], the public challenge c [14] and the rejection sampling procedure involving $\boldsymbol{z}$ [4,11,31]. In contrast, Correlation Power Analysis (CPA) attacks often focus on secret-dependent polynomial multiplications, particularly the NTT operations computing $c\boldsymbol{s}_1$, $c\boldsymbol{s}_2$ [6,27,30], or other arithmetic algorithms [22]. Both classes of attacks demonstrate that physical leakage can be leveraged to recover sensitive key information. Beyond side-channel attacks, the attacker can also perform key recovery on Dilithium through fault attacks. Current literature covers differential fault attacks on both deterministic and randomized Dilithium [5,8], skipping faults targeting the computation $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$ [23], NTT-based faults that intentionally corrupt the polynomial transform process [24], loop abort faults that sparsify $\boldsymbol{y}$ [28], and correction fault attacks that leverage public verification as a correction oracle [12,15]. In this work, we focus on fault attacks against Dilithium.

Some early fault attacks on Dilithium primarily target the deterministic mode. A key vulnerability lies in the deterministic generation of nonces: effective nonce reuse can compromise the security of Fiat-Shamir with aborts signatures [5,23]. A related implementation-oriented direction targets the concrete computation of $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$ via skipping faults. By skipping one addition at a chosen coefficient position, the attacker can obtain a faulted response $\boldsymbol{z}'$ whose affected coefficient may coincide with the corresponding coefficient of $\boldsymbol{y}$ or of $(c\boldsymbol{s}_1)$. If the fault exposes $(c\boldsymbol{s}_1)$, one can directly accumulate linear equations in $\boldsymbol{s}_1$. If the fault instead exposes $\boldsymbol{y}$, eliminating $\boldsymbol{y}$ typically requires additional structure, such as a clean reference in [23].

Although the above approaches are not directly applicable to randomized signing, the randomized mode does not fully preclude fault-based key recovery. Islam et al. [12] propose a fault injection attack applicable to both deterministic and randomized Dilithium, where bit flips in the secret key induced by Rowhammer are exploited and corrected using the public signature verification procedure. Krahmer et al. [15] extend this correction-oracle perspective to faults affecting intermediate computations in randomized Dilithium. Their central idea is that public verification can act as a correction oracle, in the spirit of prior signature-correction based fault attacks [12]. If a faulted signature can be corrected to make verification succeed, the correction yields a secret-dependent intermediate, which in turn implies a linear equation about $\boldsymbol{s}_1$.

In [15], the authors propose two variants, a skipping fault correction attack targeting $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$, and a correction attack injecting faults into the public matrix $\boldsymbol{A}$. Compared to the correction attack with a fault in $\boldsymbol{A}$, the skipping fault correction attack offers several practical advantages [15]. First, the correction attack with a fault in $\boldsymbol{A}$ requires reliably hitting many distinct coefficient positions, up to $\ell \cdot N$ targets in the worst case, while the skipping fault correction attack can repeatedly inject at a fixed position within each component and thus only needs ℓ stable injection targets. Moreover, the online correction in the fault-in-$\boldsymbol{A}$ setting relies on an exhaustive search over a space of size q, which is on the order of 2^{23}, whereas the skipping attack only enumerates the correction candidate $\alpha \in [-\beta, \beta]$ in a small bounded range with $2\beta < 400$. As a result, failed injections are easier to detect and filter in the skipping attack, which makes it more robust in practice. Due to the advantages of the skipping fault correction attack, our work focuses on this attack.

For the skipping fault correction attack targeting $\boldsymbol{z} = \boldsymbol{y}+c\boldsymbol{s}_1$, the work in [15] uses a full-rank collection strategy, specifically where $\ell \cdot N$ faults are injected to generate $\ell \cdot N$ linear equations for the ℓ components of $\boldsymbol{s}_1$. We propose an improved skipping fault correction attack, which demonstrates that the full-rank collection strategy is not necessary.

1.1 Contributions

We propose improved skipping fault correction attacks on randomized Dilithium based on MILP, which can further reduce the number of faults needed in [15]. Our main contributions are summarized as follows.

Theoretical Analysis. The theoretical analysis derives the minimum number of faults M_{min} by analyzing coverage of secret coefficients and the probability of a unique bounded solution under the random-row model. We first establish Lemma 1, showing that all coefficient positions are covered with high probability. We then prove sufficient uniqueness conditions in Lemmas 2–4 and Theorem 1, and instantiate $(M_{\min})_{\mathrm{L2,L3,L5}} = (122, 137, 114)$ required to achieve key recovery with probability at least 99% for each security level. Finally, Remark 2 shows a key-dependent trend, namely that secrets with larger abs2[5] tend to be easier to recover.

MILP attack model. For the MILP attack model, we frame key recovery as an MILP problem that combines collected linear equations, a coefficient-wise bound $\boldsymbol{s} \in [-\eta, \eta]^N$, and an explicit objective function. These jointly limit the search to a small-integer domain and let the solver find a consistent key in an underdetermined system. We develop adaptive correction fault attacks for both the plain and the shuffling settings in Algorithms 3–4, which contain online and offline phases. The online phase collects equations via fault injection and correction, and the offline phase invokes the MILP solver to recover the key coefficients of each secret component.

Experimental evaluation. Based on the experimental results of the objective functions, we have selected one from the five as the solver configuration for subsequent experiments. To explore factors affecting the key recovery success rate (e.g., the number of faults and a private key feature), we conducted two experiments. Both experiments confirm theoretical predictions: key recovery success rates increase with a larger number of faults and higher abs2 values. Compared to Krahmer et al. [15], our improved attacks reduce the number of required faults in both the plain and the shuffling settings. Specifically, for L2, L3, and L5, we achieve fault reductions of 25.9%, 16.2%, and 25.6% in the plain setting, and 25.6%, 13.5%, and 26.3% in the shuffling setting. Our implementation of the proposed adaptive attacks (including code to construct the MILP instances and run the solver) in both the plain and the shuffling settings is available at `https://anonymous.4open.science/r/sfcafDvM-H73B/`.

1.2 Outline

In Section 2, we introduce the basic notations and background knowledge necessary for understanding this work. In Section 3, we present a theoretical analysis of the offline recovery problem modeled as a bounded integer equation system. Section 4 describes the MILP-based key-recovery model used in the offline phase of the attack, covering both the plain and the shuffling settings. In Section 5, we report important experimental results of the key recovery. Section 6 discusses the relationship between the theoretical analysis and the experimental observations. Finally, we conclude our work in Section 7.

[5] which is the number of ± 2 in the coefficients of a vector.

2 Preliminaries

2.1 Notations

We follow standard conventions similar to those in the Dilithium specification. For integers $a \le b$ we write $[a, b] = \{a, a+1, \ldots, b\} \subset \mathbb{Z}$. For a modulus $q \in \mathbb{N}$ we set $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ and always interpret $x \in \mathbb{Z}_q$ via its centered representative in $[-(q-1)/2, (q-1)/2]$, which we denote by $x \bmod \pm q$. The polynomial ring is $R_q \coloneqq \mathbb{Z}_q[X]/(X^N+1)$ and $R \coloneqq \mathbb{Z}[X]/(X^N+1)$, where N is the ring dimension. A polynomial $a \in R_q$ is written $a = a_0 + a_1 X + \cdots + a_{N-1}X^{N-1}$ and is identified with its coefficient vector $(a_0, \ldots, a_{N-1}) \in \mathbb{Z}_q^N$. We denote polynomials by regular lowercase letters (e.g., a), vectors of polynomials by bold lowercase letters (e.g., $\boldsymbol{z}$), and matrices of polynomials by bold uppercase letters (e.g., $\boldsymbol{A}$). Bold lowercase letters over $\mathbb{Z}$ (such as $\boldsymbol{s}$) are used for integer vectors, and plain uppercase letters (such as C) for integer matrices. For a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ over $\mathbb{Z}$ or $\mathbb{Z}_q$ we write $\|\boldsymbol{v}\|_\infty = \max_i |v_i|$. For a polynomial $a \in R_q$ we set $\|a\|_\infty = \max_{0 \le i < N} |a_i|$. The secret distribution is modeled by $S_\eta \coloneqq \{s \in R_q : \|s\|_\infty \le \eta\}$, and we write $s \leftarrow S_\eta$ for a uniform sampling of polynomials. Throughout this work, we identify each polynomial $s \in S_\eta$ with its coefficient vector $\boldsymbol{s} \in \mathbb{Z}^N$. The set of challenge polynomials with exactly τ coefficients in $\{\pm 1\}$ and all others zero is denoted $B_\tau \coloneqq \{c \in R : \|c\|_\infty \le 1,\ \|c\|_0 = \tau\}$. Throughout, N denotes the number of coefficients in one polynomial component, η the secret bound, and M the number of collected equations for one component (as well as the number of faults injected). Given an integer system $C\boldsymbol{s} = x$ with $C \in [-1, 1]^{M \times N}$ and $x \in \mathbb{Z}^M$, we write $\mathcal{S} \coloneqq \{\, \boldsymbol{s} \in \mathbb{Z}^N : C\boldsymbol{s} = x,\ \|\boldsymbol{s}\|_\infty \le \eta \,\}$ for its set of bounded solutions, and $\mathcal{D} \coloneqq \{\, \boldsymbol{d} \in \mathbb{Z}^N : \boldsymbol{d} \ne 0,\ \|\boldsymbol{d}\|_\infty \le 2\eta \,\}$ for the corresponding set of admissible difference vectors. Random variables are written in uppercase (e.g., X). We denote expectation and variance by $\mathbb{E}[\cdot]$ and $\mathrm{Var}[\cdot]$, respectively. By $\mathcal{N}(0, \sigma^2)$ we denote a normal distribution of mean 0 and variance σ^2. By log we mean the base-2 logarithm. For a coefficient vector $\boldsymbol{s} = (s_0, \ldots, s_{N-1})$, the number of ± 2 in the coefficients of $\boldsymbol{s}$ is represented by $\mathrm{abs2} \coloneqq \#\{i : |s_i| = 2\}$.

In the remainder of the paper, we fix a single polynomial component $(\boldsymbol{s}_1)_j$ and identify it as a coefficient vector $\boldsymbol{s} \in S_\eta \subset \mathbb{Z}^N$, thereby dropping the component index for clarity. The fault correction procedure then yields an integer system $C\boldsymbol{s} = x$ with $C \in [-1, 1]^{M \times N}$ and $\|x\|_\infty \le \beta$ for Dilithium.

2.2 CRYSTALS-Dilithium

CRYSTALS-Dilithium is a module-lattice-based signature scheme following the Fiat–Shamir with aborts paradigm [16,17]. All computations are carried out over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^N+1)$ with $N = 256$ and $q = 8380417$, and vectors and matrices over R_q are interpreted component-wise.

Key generation. This algorithm samples a public seed ρ and expands a uniformly distributed matrix $\boldsymbol{A} \in R_q^{k \times \ell}$. It then samples short secret vectors $(\boldsymbol{s}_1, \boldsymbol{s}_2) \in S_\eta^\ell \times S_\eta^k$, and forms the public vector $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2$. After a rounding

and decomposition step for $\boldsymbol{t}$, the public key is $(\rho, \boldsymbol{t}_1)$, while the secret key contains $(\rho, K, \mathrm{tr}, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_0)$.

Signing and rejection sampling. To sign a message m, the signer derives a message digest μ and samples a short masking vector $\boldsymbol{y}$. It computes $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y}$, extracts its high bits $\boldsymbol{w}_1$, and derives a challenge seed $\tilde{c}$ as a hash of $(\mu, \boldsymbol{w}_1)$. The challenge polynomial is then obtained as $c = \mathsf{SampleInBall}(\tilde{c})$. Under the standard XOF idealization in the specification, this procedure is close to uniform over B_τ. The response is then computed as $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$. Dilithium applies rejection sampling [3,20]. Concretely, the signer aborts and restarts if any of the following prescribed bounds is violated, namely if $\|\boldsymbol{z}\|_\infty \geq \gamma_1 - \beta$, or if $\|\boldsymbol{r}_0\|_\infty \geq \gamma_2 - \beta$ for the low-bits vector $\boldsymbol{r}_0$ computed from $\boldsymbol{w} - c\boldsymbol{s}_2$, or if $\|c\boldsymbol{t}_0\|_\infty \geq \gamma_2$, or if $\mathrm{wt}(h) > \omega$, where h is a hint that helps reconstruct the high bits. This ensures that the distribution of accepted signatures is statistically close to being secret-independent in the random oracle model. The resulting signature contains $(\tilde{c}, \boldsymbol{z}, h)$.

Scheme / security level	(k, ℓ)	η	τ	$\beta(= \eta \cdot \tau)$
Dilithium-L2	$(4, 4)$	2	39	78
Dilithium-L3	$(6, 5)$	4	49	196
Dilithium-L5	$(8, 7)$	2	60	120

Table 1: Relevant Dilithium parameter choices [20,2]

Verification. Given a signature $(\tilde{c}, \boldsymbol{z}, h)$ on message m, the verifier recomputes the digest μ, derives $c = \mathsf{SampleInBall}(\tilde{c})$, reconstructs the corresponding high bits $\boldsymbol{w}_1$ from $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}_1 \cdot 2^d$ using h, and checks that (i) $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta$, (ii) $\tilde{c} = H(\mu, \boldsymbol{w}_1)$, and (iii) $\mathrm{wt}(h) \leq \omega$. The verifier accepts the signature if and only if all the above checks hold.

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \coloneqq c\boldsymbol{s}_1 = \begin{bmatrix} c_0 & -c_{N-1} & \cdots & -c_2 & -c_1 \\ c_1 & c_0 & \cdots & -c_3 & -c_2 \\ c_2 & c_1 & \cdots & -c_4 & -c_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-2} & c_{N-3} & \cdots & c_0 & -c_{N-1} \\ c_{N-1} & c_{N-2} & \cdots & c_1 & c_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_{1\,0} \\ \boldsymbol{s}_{1\,1} \\ \boldsymbol{s}_{1\,2} \\ \vdots \\ \boldsymbol{s}_{1\,N-2} \\ \boldsymbol{s}_{1\,N-1} \end{bmatrix} \tag{1}$$

Dilithium admits a deterministic variant and a randomized variant. They differ only in how the seed used to generate $\boldsymbol{y}$ is obtained. In deterministic signing, the seed is deterministically derived by hashing the secret key together with the message, whereas in randomized signing it is derived using fresh randomness sampled for each signature. In this work, we consider the randomized signing mode. Dilithium has three security levels, i.e., Dilithium-L2/L3/L5 [2], target-

ing the ML-DSA security levels ML-DSA-44/65/87 respectively [20]. Table 1 summarizes the parameter choices for each security level.

As shown in Algorithm Sign [20], Dilithium computes $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$, where $\boldsymbol{y}$ is the mask vector. The challenge polynomial c has exactly τ nonzero coefficients in $\{-1, 1\}$, and the secret key polynomial $\boldsymbol{s}_1$ satisfies $\|\boldsymbol{s}_1\|_\infty \leq \eta$ with coefficients sampled uniformly from $\{-\eta, \ldots, \eta\}$. The i-th coefficient of the product $c\boldsymbol{s}_1$ admits the explicit form $x_i = \sum_{j=0}^{i} c_{i-j}\, \boldsymbol{s}_{1\,j} - \sum_{j=i+1}^{N-1} c_{N+i-j}\, \boldsymbol{s}_{1\,j}$, which reflects the negacyclic convolution modulo $X^N + 1$. Equivalently, set $c' = \mathrm{rot}(c, i) \coloneqq c \cdot X^i \bmod (X^N + 1)$ so that c' is the i-th row of the matrix in Equation 1. Then $x_i = \sum_j c'_j\, \boldsymbol{s}_{1\,j} = \langle c_i, \boldsymbol{s}_1 \rangle$, where c_i denotes the coefficient vector of the i-th rotation of c.

Each coefficient x_i is therefore a signed sum of exactly τ secret coefficients, so $\|x\|_\infty \leq \eta \cdot \tau = \beta$. Under the randomness assumptions on c and $\boldsymbol{s}_1$, the Central Limit Theorem suggests that the distribution of each x_i is well approximated by a normal distribution $\mathcal{N}(0, \sigma_s^2)$ with variance $\sigma_s^2 = \frac{\tau \cdot \eta \cdot (\eta+1)}{3}$ [31].

2.3 Skipping Fault Correction Attack

The skipping fault correction attack of [15] is formulated for the randomized (or hedged) Dilithium. The adversary has physical access to a device implementing the Dilithium signing algorithm with an unknown secret but known public key. The adversary can trigger signing and gather the (not necessarily controllable) signed message as well as the resulting signature.

For each signing attempt, the adversary can inject a single instruction-skipping fault during the computation $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$ that skips one coefficient addition in one secret key component. Concretely, if the targeted coefficient is position i in component j, the faulty signature contains $\boldsymbol{z}'_j[i] = \boldsymbol{y}_j[i]$ instead of $\boldsymbol{z}_j[i] = \boldsymbol{y}_j[i] + (c\boldsymbol{s}_1)_j[i]$, while all other coefficients are computed normally.

For the theoretical presentation of the skipping fault correction attack, it is assumed that the fault injection always succeeds. Moreover, since this operation is executed inside Dilithium's rejection-sampling loop, it assumes that the injected fault can be made to affect the final successful iteration, for example, by repeating fault injections until a faulty but verifying signature is observed.

Given a faulty signature $\sigma' = (\tilde{c}, \boldsymbol{z}', h)$, the adversary runs the public verification algorithm as a correction oracle. The adversary generates the challenge polynomial c by $\mathsf{SampleInBall}(\tilde{c})$. For the chosen pair $(j, i) \in [0, \ell-1] \times [0, N-1]$, the adversary successively tests candidate correction values $\alpha \in \mathbb{Z}$ by replacing $\boldsymbol{z}'_j[i]$ with $\boldsymbol{z}'_j[i] + \alpha$ and re-running verification on the modified signature. Under the skipping-fault model of [15], verification accepts exactly when $\alpha = (c\boldsymbol{s}_1)_j[i]$. Therefore, each successful correction yields one linear equation between the public challenge polynomial c and the secret component $(\boldsymbol{s}_1)_j$, which can be viewed as a single coefficient of the product $(c\boldsymbol{s}_1)_j$. In what follows, we model faulted signing as an oracle $\mathcal{O}(m, \mathcal{T})$ that, on input a message m and a candidate target coefficient position set $\mathcal{T} \subseteq [0, \ell-1] \times [0, N-1]$, returns a single faulty signature $\sigma' = (\tilde{c}, \boldsymbol{z}', h)$ whose fault location lies in $\mathcal{T}$.

Algorithm 1 EqFromSig

Input: sign-with-fault oracle $\mathcal{O}$, public key pk, message m, candidate target set $\mathcal{T}$, β
Output: linear equation (j, i, eq, eq_sol)

1: $(\tilde{c}, \boldsymbol{z}', h) := \mathcal{O}(m, \mathcal{T})$
2: **for** $\alpha = 0, 1, -1, 2, -2, \ldots, \beta, -\beta$ **do**
3: **for** $(j, i) \in \mathcal{T}$ **do**
4: $\boldsymbol{z}_{\mathsf{temp}} := \boldsymbol{z}'$
5: $(\boldsymbol{z}_{\mathsf{temp}})_j[i] := (\boldsymbol{z}')_j[i] + \alpha$
6: **if** $\mathsf{verify}((\tilde{c}, \boldsymbol{z}_{\mathsf{temp}}, h), \mathsf{pk}) = \mathsf{accept}$ **then**
7: $eq_sol := \alpha$
8: $c := \mathsf{SampleInBall}(\tilde{c})$
9: $eq := \mathsf{rotmult}(c, i)$
10: **return** (j, i, eq, eq_sol) ▷ where $(c\boldsymbol{s}_1)_j[i] = \alpha$
11: **return** $\perp$

Algorithm 1 formalizes this extraction procedure as EqFromSig. It takes as input the signing oracle $\mathcal{O}$ to inject faults, the public key pk, a message m, a candidate target set $\mathcal{T}$, and the bound β. It first obtains a faulty signature (Line 1). By applying bounded correction (Line 4-5) and public verification (Line 6), it succeeds in extracting a single linear equation (Line 7-10) on $(\boldsymbol{s}_1)_j$ as output.

Throughout the paper, we focus on recovering one polynomial component $\boldsymbol{s} = (\boldsymbol{s}_1)_j \in S_\eta$ at a time. The remaining components can then be recovered independently by targeting their coefficients in the same way.

As described in [7,20], Dilithium's rejection sampling mechanism can reveal information about faulted computations through its accept/reject behavior. When an adversary injects a skipping fault and then corrects the resulting faulty signature until it verifies, the successful correction yields a linear equation on the secret key. In the correction-fault setting considered here, our goal is to reconstruct the secret key component $\boldsymbol{s}$ from a collection of such equations and the known public challenges c.

2.4 ILP and MILP

An integer linear programming (ILP) problem optimizes a linear objective subject to linear constraints and integrality requirements on all decision variables. In its standard form, it seeks $\boldsymbol{x} \in \mathbb{Z}^n$ that optimizes a linear function while satisfying $A\boldsymbol{x} \leq \boldsymbol{b}$ and $\boldsymbol{x} \geq \boldsymbol{0}$. A mixed integer linear programming (MILP) problem generalizes ILP by allowing a subset of variables to be continuous, namely $\boldsymbol{x} \in \mathbb{Z}^p \times \mathbb{R}^q$. When $q = 0$, MILP reduces to a pure ILP instance, hence ILP is a special case of MILP. Although our recovery problem is a pure ILP, we follow common practice and present it in an MILP framework. This choice is motivated by two practical considerations: (i) MILP solvers natively support the pure integer case with specialized algorithms, and (ii) the MILP formulation is general and facilitates extensions and tool integration.

From a complexity-theoretic point of view, the MILP problem is NP-hard, while the ILP problem is NP-*complete* [9,10]. When the number of integer variables is treated as a fixed constant, Lenstra [13] showed that such problems can be solved in time $2^{O(n_{\mathrm{I}}^3)} \cdot (m \log V)^{O(1)}$, where V denotes an upper bound on the absolute values of coefficients. More recently, Reis et al. [25, Theorem 41] improved this bound to $(\log n_{\mathrm{I}})^{O(n_{\mathrm{I}})} \cdot (m \log V)^{O(1)}$.

In practice, modern MILP solvers such as Gurobi and CPLEX rely on branch-and-bound and branch-and-cut algorithms combined with heuristics tailored to integer domains, which have become standard tools for large-scale ILP/MILP problems. In this setting, while the objective function does not affect feasibility, it can significantly impact solver performance by guiding the search, particularly for instances with small integer domains.

3 Improved Skipping Fault Correction Attack

Krahmer et al. [15] proposed two correction fault attacks, i.e., a skipping fault correction attack and a correction attack with a fault in the public matrix $\boldsymbol{A}$. Our work focuses on the skipping fault correction attack. To recover the secret key $\boldsymbol{s}_1$, [15] requires $\ell \cdot N$ ideal fault injections to generate $\ell \cdot N$ linear equations for the ℓ components of $\boldsymbol{s}_1$. We propose an improved skipping fault correction attack model for Dilithium. Our analysis shows that even if the number of faults is less than $\ell \cdot N$, $\boldsymbol{s}_1$ can still be recovered by exploiting the bounded-domain constraint.

3.1 Improved Attack Model

Fix a single N-coefficient component of $\boldsymbol{s}_1$ and write it as $\boldsymbol{s} \in \mathbb{Z}^N$ with $\|\boldsymbol{s}\|_\infty \leq \eta$. Each corrected faulty signature produces one equation $\langle c_i, \boldsymbol{s} \rangle = x_i$, where $c_i \in [-1, 1]^N$ has exactly τ nonzero elements and $x_i \in \mathbb{Z}$ is a small signed value determined by the correction procedure ($|x_i| \leq \beta$ as defined) [15]. After collecting M corrected samples, we obtain an integer system $C\boldsymbol{s} = x$, $C \in [-1, 1]^{M \times N}$, $\|x\|_\infty \leq \beta$, together with the bounded-domain constraint $\boldsymbol{s} \in [-\eta, \eta]^N$.

To capture the dominant structure induced by the challenge sampling via SampleInBall, we adopt the standard random-row model: each row of C selects τ positions uniformly at random and assigns independent unbiased signs in $\{\pm 1\}$ to these positions. In particular, the sign bits produced by SampleInBall are uniform, so for any row j we have $\mathbb{E}[c_j] = 0$ and $\mathrm{Var}(c_j) = \Pr[c_j \neq 0] = \tau/N$ under this idealized model. We use the random-row model as an analytic proxy for the matrix C obtained by stacking the rotated challenge vectors returned by EqFromSig, for the ensuing coverage and collision analysis.

Based on this random-row model, we propose an improved attack model. The theoretical description focuses on a uniqueness question for bounded solutions. After collecting M corrected samples via EqFromSig (Algorithm 1), we stack the resulting rotated challenge vectors to form $C \in [-1, 1]^{M \times N}$ and the

corresponding correction values to form $x \in \mathbb{Z}^M$ (with $|x_i| \leq \beta$ by construction). We then study how large M must be so that, with high probability over the randomness of SampleInBall and the signing process, the bounded feasible set $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{Z}^N : C\boldsymbol{s} = x,\ \|\boldsymbol{s}\|_\infty \leq \eta\}$ contains *exactly one* element, i.e., the true secret component $\boldsymbol{s}$.[6] More specifically, we seek an explicit threshold $M_{\min} = \mathcal{L}(\eta, \tau, N, \lambda)$ such that for $M \geq M_{\min}$ the probability of successful key recovery is at least $1 - 2^{-\lambda}$. The following analyses derive $M_{\min}$ by analyzing coverage of secret coefficients and the probability of a unique bounded solution under the random-row model.

3.2 Coverage of Secret Coefficients in the Equation System

A successful bounded key recovery requires that the collected equations involve all N coefficients of the target secret vector $\boldsymbol{s}$. Each equation involves only τ coefficients. To fully recover $\boldsymbol{s}$, all the N coefficients of $\boldsymbol{s}$ must be covered by the M collected equations.

Let H_j denote the number of equations in which coefficient position $j \in [0, N-1]$ appears. If the coefficient of position j never appears in any equation, meaning $H_j = 0$, then the equation system imposes no constraint on $(\boldsymbol{s})_j$. Consequently, any alternative bounded vector $\boldsymbol{s}'$ satisfies all collected equations if it differs from $\boldsymbol{s}$ only at position j. So uniqueness is impossible regardless of the value of M. Therefore, coverage of all coefficient positions is necessary for uniqueness. Under Dilithium's random challenge sampling, full coverage occurs with high probability for sufficiently large M as shown by the following lemma.

Based on Lemma 1, full coverage of the secret coefficients is a necessary condition for unique key recovery. If a coefficient does not appear in any collected equation, its value remains unconstrained and the bounded solution cannot be unique. However, full coverage only ensures that every coefficient appears in the equation system and does not exclude the existence of multiple bounded solutions. Section 3.3 therefore analyzes nonzero difference vectors in the kernel of C. Combining the coverage result in Lemma 1 with the uniqueness condition in Theorem 1 yields the concrete threshold $M_{\min}$ in Section 3.4.

Lemma 1 (Full coverage with high probability). *Let H_j be the number of equations in which coefficient position $j \in [0, N-1]$ appears. We say that* full coverage *holds if $H_j \geq 1$ for all j. Under the random-row model for C, the full-coverage event occurs with probability at least $1 - N(1-\tau/N)^M$.*

Proof. Under the random-row model, each row of C involves τ positions uniformly at random. In one row, the probability that position j does not appear is $1 - \tau/N$. Independence across the M rows gives $\Pr[H_j = 0] = (1-\tau/N)^M$. By the union bound, $\Pr[\exists j :\ H_j = 0] \leq \sum_{j=0}^{N-1} \Pr[H_j = 0] = N(1-\tau/N)^M$, hence $\Pr[\forall j,\ H_j \geq 1] = 1 - \Pr[\exists j :\ H_j = 0] \geq 1 - N(1-\tau/N)^M$.

[6] $\mathcal{S} \neq \varnothing$ since $C\boldsymbol{s} = x$.

For Dilithium-L2, we have $N = 256$ and $\tau = 39$. For $M = N/2 = 128$, Lemma 1 yields $\Pr[\exists j : H_j = 0] \leq 256 \times (1 - 39/256)^{128} \approx 1.7 \times 10^{-7}$, which is extremely small in practice. Thus, every coefficient of $\boldsymbol{s}$ is covered by the equation system with high probability.

Based on Lemma 1, full coverage of secret coefficients is a necessary prerequisite for our uniqueness analysis. We next analyze the probability that the bounded solution is unique and derive a sufficient condition guaranteeing uniqueness.

3.3 Uniqueness of Bounded Solutions

We consider the equation system $C\boldsymbol{s} = x$ with $\boldsymbol{s} \in [-\eta, \eta]^N$. Such a system may admit a unique solution or multiple solutions. To study the probability of uniqueness, we first analyze the probability of non-uniqueness (there exist two distinct bounded solutions), and then subtract it from 1 to obtain the probability of uniqueness.

Lemma 2 (Difference characterization of non-uniqueness). *Let* $\mathcal{S} = \{\, \boldsymbol{s} \in \mathbb{Z}^N : C\boldsymbol{s} = x,\ \|\boldsymbol{s}\|_\infty \leq \eta \,\}$ *be the set of bounded solutions and let* $\mathcal{D} = \{\boldsymbol{d} \in \mathbb{Z}^N : \boldsymbol{d} \neq 0,\ \|\boldsymbol{d}\|_\infty \leq 2\eta\}$ *be the set of admissible difference vectors. If* $\mathcal{S}$ *contains two distinct solutions* $\boldsymbol{s}, \boldsymbol{s}'$ *in* $[-\eta, \eta]^N$*, then their difference* $\boldsymbol{d} = \boldsymbol{s}' - \boldsymbol{s}$ *lies in* $\mathcal{D}$ *and satisfies* $C\boldsymbol{d} = 0$*. Therefore, a solution in* $[-\eta, \eta]^N$ *is non-unique if and only if there exists a nonzero* $\boldsymbol{d} \in \mathcal{D}$ *with* $C\boldsymbol{d} = 0$ *such that* $\boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$ *for* $\boldsymbol{s} \in \mathcal{S}$*.*

Proof. If $C\boldsymbol{s} = C\boldsymbol{s}'$ then $C(\boldsymbol{s}' - \boldsymbol{s}) = 0$ by linearity, so $\boldsymbol{d}$ lies in the kernel of C. If $\|\boldsymbol{s}\|_\infty \leq \eta$ and $\|\boldsymbol{s}'\|_\infty \leq \eta$, then $\|\boldsymbol{s}' - \boldsymbol{s}\|_\infty \leq 2\eta$, so $\boldsymbol{d} \in \mathcal{D}$. Conversely, for $\boldsymbol{s} \in \mathcal{S}$, there exists a nonzero $\boldsymbol{d} \in \mathcal{D}$ such that $C\boldsymbol{d} = 0$. Assume $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$. Then $C\boldsymbol{s}' = C(\boldsymbol{s} + \boldsymbol{d}) = C\boldsymbol{s} + 0 = x$, so $\boldsymbol{s}' \in \mathcal{S}$. Finally, $\mathcal{S}$ contains at least two distinct bounded solutions.

Lemma 3 (The number of admissible $\boldsymbol{d} \in \mathcal{D}$ for Lemma 2). *Let* $\boldsymbol{s} \in [-\eta, \eta]^N$ *and* $\mathcal{D} = \{\boldsymbol{d} \in [-2\eta, 2\eta]^N : \boldsymbol{d} \neq 0,\ \boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N\}$*. The number of* $\boldsymbol{d} \in \mathcal{D}$ *such that* $\boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$ *equals* $(2\eta + 1)^N - 1$*. In particular, this count does not depend on the choice of* $\boldsymbol{s}$*.*

Proof. The condition $\boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$ holds if and only if $-\eta \leq s_j + d_j \leq \eta$ for every coefficient position j. Equivalently, $d_j \in [-\eta - s_j,\ \eta - s_j]$ for all j. Since $s_j \in [-\eta, \eta]$, the interval $[-\eta - s_j,\ \eta - s_j]$ is always contained in $[-2\eta, 2\eta]$, so it is compatible with the constraint $\|\boldsymbol{d}\|_\infty \leq 2\eta$.

For each fixed j, the interval $[-\eta - s_j,\ \eta - s_j]$ contains exactly $2\eta + 1$ integers. Therefore the number of integer vectors $\boldsymbol{d}$ satisfying $\boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$ is $(2\eta + 1)^N$. Removing the case $\boldsymbol{d} = \boldsymbol{0}$ leaves $(2\eta + 1)^N - 1$ admissible nonzero $\boldsymbol{d}$ in $\mathcal{D}$.

Lemma 4 (Single-row collision probability). *Fix a nonzero* $\boldsymbol{d} \in \mathcal{D}$ *and let* $Z_i = C_i\boldsymbol{d}$ *be the inner product with the* i*-th row of* C*. Under the random-row model for* C*, the random variables* $Z_1, \ldots, Z_M$ *are independent and identically distributed, and the single-row collision (i.e.,* $Z_i = 0$*) probability is well approximated by* $p_{\text{gauss}} \coloneqq \frac{1}{\sqrt{2\pi}\,\sigma_d}$*, with* $\sigma_d^2 = 2\tau\eta(\eta + 1)/3$*.*

Proof. Each row C_i has exactly τ nonzero entries in $\{\pm 1\}$ at uniformly random positions, independent of the other rows. Let $\boldsymbol{s}, \boldsymbol{s}'$ be two independent secrets s, s' sampled from S_η, and identify them with coefficient vectors $\boldsymbol{s}, \boldsymbol{s}'$. Write $\boldsymbol{d} = \boldsymbol{s} - \boldsymbol{s}'$ and $\boldsymbol{d} = (d_1, \ldots, d_N)$. We have $Y_i = \langle c_i, \boldsymbol{s} \rangle$ and $Y_i' = \langle c_i, \boldsymbol{s}' \rangle$, where the coefficients of $\boldsymbol{s}$ and $\boldsymbol{s}'$ are independent and uniform on $[-\eta, \eta]$. By the Central Limit Theorem, for moderate τ the sums Y_i and Y_i' are well approximated by independent normal random variables with distribution $\mathcal{N}(0, \sigma_s^2)$, where $\sigma_s^2 = \tau\eta(\eta+1)/3$. Their difference satisfies $Z_i = Y_i - Y_i'$, so under this approximation Z_i is distributed as $\mathcal{N}(0, \sigma_d^2)$ with variance $\sigma_d^2 = 2\sigma_s^2 = 2\tau\eta(\eta+1)/3$. Different rows of C are sampled independently, hence the random variables $Z_1, \ldots, Z_M$ are independent and identically distributed.

The underlying equation system is defined over integers, and both the entries of C and the secrets $\boldsymbol{s}, \boldsymbol{s}'$ are integers. Hence Y_i, Y_i' and Z_i take values in $\mathbb{Z}$ and the event $Z_i = 0$ is equivalent to $Z_i \in (-\frac{1}{2}, \frac{1}{2})$. Using the Gaussian approximation $Z_i/\sigma_d \approx \mathcal{N}(0,1)$ in the sense of the Lindeberg–Levy central limit theorem, the standardized variable Z_i/σ_d has (approximately) the standard normal density $\varphi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$.

We therefore obtain

$$\begin{aligned}\Pr[Z_i = 0] = \Pr\left[|Z_i| \le \tfrac{1}{2}\right] &= \Pr\left[\left|\frac{Z_i}{\sigma_d}\right| \le \frac{1}{2\sigma_d}\right] \\ &\approx \int_{-1/(2\sigma_d)}^{1/(2\sigma_d)} \varphi(u)\, du = \frac{1}{\sqrt{2\pi}} \int_{-1/(2\sigma_d)}^{1/(2\sigma_d)} e^{-u^2/2}\, du.\end{aligned}$$

For $|u| \le 1/(2\sigma_d)$ we have $e^{-\frac{1}{8\sigma_d^2}} \le e^{-u^2/2} \le 1$, and hence $e^{-\frac{1}{8\sigma_d^2}}/(\sqrt{2\pi}\,\sigma_d) \le \Pr[Z_i = 0] \le 1/(\sqrt{2\pi}\,\sigma_d)$. In particular, we take the upper bound

$$p_{\text{gauss}} \coloneqq \frac{1}{\sqrt{2\pi}\,\sigma_d},$$

so that $\Pr[Z_i = 0] \le p_{\text{gauss}}$. Moreover, since $\Pr[Z_i = 0] \ge e^{-\frac{1}{8\sigma_d^2}}/(\sqrt{2\pi}\,\sigma_d)$ and $1 - e^{-x} \le x$ for $x > 0$, we have

$$0 \le p_{\text{gauss}} - \Pr[Z_i = 0] \le \frac{1 - e^{-\frac{1}{8\sigma_d^2}}}{\sqrt{2\pi}\,\sigma_d} \le \frac{1}{8\sqrt{2\pi}\,\sigma_d^3},$$

which is small enough [7] and thus we use p_{gauss} as an approximation for it.

Theorem 1 (Minimum number of faults for uniqueness). *Let $s \leftarrow S_\eta$ be a random secret (identified with coefficient vector $\boldsymbol{s} \in \mathbb{Z}^N$) and sample $C \in [-1,1]^{M \times N}$ according to the random-row model. Let $C\boldsymbol{s} = x$, and let p_{gauss} be as in Lemma 4. Then*

$$\Pr\left[\exists\, \boldsymbol{s}' \in [-\eta, \eta]^N,\ \boldsymbol{s}' \neq \boldsymbol{s} \text{ with } C\boldsymbol{s}' = x\right] \le \left((2\eta+1)^N - 1\right) p_{\text{gauss}}^M.$$

[7] less than 7.2×10^{-5} for σ_d at Dilithium-L2.

In particular, let RHS $\leq 2^{-\lambda}$ for $\lambda \geq 0$, it suffices to choose

$$M \geq \frac{\log((2\eta+1)^N - 1) + \lambda}{-\log p_{\text{gauss}}} = \mathcal{L}(\eta, \tau, N, \lambda) \tag{2}$$

to ensure that the linear system admits a unique solution in $[-\eta, \eta]^N$ with probability at least $1 - 2^{-\lambda}$.

Proof. Since $C\boldsymbol{s} = x$, the vector $\boldsymbol{s}$ is a bounded solution. By Lemma 2, the existence of a second bounded solution $\boldsymbol{s}' \neq \boldsymbol{s}$ is equivalent to the existence of a nonzero $\boldsymbol{d} \in [-2\eta, 2\eta]^N$ such that $C\boldsymbol{d} = 0$ and $\boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N$. Define

$$\mathcal{D}_{\boldsymbol{s}} = \{\, \boldsymbol{d} \in [-2\eta, 2\eta]^N : \boldsymbol{d} \neq 0,\ \boldsymbol{s} + \boldsymbol{d} \in [-\eta, \eta]^N \,\}.$$

By Lemma 3, we have $|\mathcal{D}_{\boldsymbol{s}}| = (2\eta+1)^N - 1$.

For each $\boldsymbol{d} \in \mathcal{D}_{\boldsymbol{s}}$ define the indicator variable $I_{\boldsymbol{d}} = \mathbb{1}_{\{C\boldsymbol{d}=0\}}$, and let $\mathcal{N}_{\boldsymbol{s}} = \sum_{\boldsymbol{d}\in\mathcal{D}_{\boldsymbol{s}}} I_{\boldsymbol{d}}$. Then $\Pr[\mathcal{N}_{\boldsymbol{s}} \geq 1]$ is exactly the non-uniqueness probability. By linearity of expectation, $\mathbb{E}[\mathcal{N}_{\boldsymbol{s}}] = \sum_{\boldsymbol{d}\in\mathcal{D}_{\boldsymbol{s}}} \Pr[C\boldsymbol{d} = 0]$. Under the row model, the rows of C are independent, so for fixed $\boldsymbol{d}$ we have

$$\Pr[C\boldsymbol{d} = 0] = \Pr[Z_1 = 0, \ldots, Z_M = 0] = \Pr[Z_1 = 0]^M \leq p_{\text{gauss}}^M$$

by Lemma 4. Hence

$$\mathbb{E}[\mathcal{N}_{\boldsymbol{s}}] \leq |\mathcal{D}_{\boldsymbol{s}}|\, p_{\text{gauss}}^M = \left((2\eta+1)^N - 1\right) p_{\text{gauss}}^M.$$

Markov's inequality gives $\Pr[\mathcal{N}_{\boldsymbol{s}} \geq 1] \leq \mathbb{E}[\mathcal{N}_{\boldsymbol{s}}]$, which gives the stated upper bound on the probability that there exists a nonzero $\boldsymbol{d} \in \mathcal{D}$ with $C\boldsymbol{d} = 0$. By Lemma 2, this event is equivalent to the existence of a second bounded solution in $[-\eta, \eta]^N$. Solving $\left((2\eta+1)^N - 1\right)p_{\text{gauss}}^M \leq 2^{-\lambda}$ for M gives (2). Indeed, taking logarithms on both sides yields $\log\left((2\eta+1)^N - 1\right) + M \log p_{\text{gauss}} \leq -\lambda$. Since $p_{\text{gauss}} \in (0, 1)$, we have $\log p_{\text{gauss}} < 0$ and thus

$$M \geq \frac{\log\left((2\eta+1)^N - 1\right) + \lambda}{-\log p_{\text{gauss}}}.$$

Remark 1 (Theoretical analysis of $M_{\min}$). Theorem 1 is stated under the sampling assumptions that the secret key $\boldsymbol{s}$ has independent coefficients, each uniform over $\{-\eta, \ldots, \eta\}$, and that each row of C is sampled uniformly by SampleInBall. The proof uses a first moment argument, which applies a union bound over all admissible difference vectors $\boldsymbol{d} \in \mathcal{D}$. This approach yields a clean sufficient condition and a closed-form threshold $M_{\min}$, but it is not intended to be tight. In particular, the union bound treats the events $\{C\boldsymbol{d} = 0\}$ independently, and the resulting threshold can be conservative.

Remark 2 (Effect of the secret key feature on uniqueness). Fix $\boldsymbol{s}$ and let $Z = \langle c, \boldsymbol{s} - \boldsymbol{s}' \rangle$, where c is sampled by SampleInBall and $\boldsymbol{s}'$ has i.i.d. coefficients uniform in $\{-\eta, \ldots, \eta\}$. Then $\mathbb{E}[Z \mid \boldsymbol{s}] = 0$ and $\mathrm{Var}(Z \mid \boldsymbol{s}) = \frac{\tau}{N}\|\boldsymbol{s}\|_2^2 + \tau \cdot \frac{\eta(\eta+1)}{3}$. Under the local-CLT heuristic used in Lemma 4, a larger $\|\boldsymbol{s}\|_2^2$ decreases the single-row collision probability $\Pr[Z = 0 \mid \boldsymbol{s}]$.[8]

[8] For Dilithium (L2/L5) secrets, a larger value of abs2 typically implies a larger $\|\boldsymbol{s}\|_2^2$, which provides an explanation for key-dependent collision rates.

Proof. Write $Z = \sum_{j=1}^{N} c_j(s_j - s'_j)$. For uniform $s'_j \in \{-\eta, \ldots, \eta\}$, one has $\mathbb{E}[s'_j] = 0$ and $\mathbb{E}[(s'_j)^2] = \eta(\eta+1)/3$. Under the random-row model, $\mathbb{E}[c_j] = 0$ and $\mathbb{E}[c_j^2] = \Pr[c_j \neq 0] = \tau/N$. Using that c_j is independent of s'_j, we obtain

$$\mathrm{Var}\big(c_j(s_j - s'_j) \mid s_j\big) = \mathbb{E}[c_j^2] \cdot \mathbb{E}\big[(s_j - s'_j)^2 \mid s_j\big] = \frac{\tau}{N} \cdot \Big(s_j^2 + \frac{\eta(\eta+1)}{3}\Big).$$

Moreover, the cross terms vanish since the random signs in SampleInBall are uniform, so $\mathrm{Var}(Z \mid \boldsymbol{s}) = \sum_{j=1}^{N} \mathrm{Var}(c_j(s_j - s'_j) \mid s_j)$. Summing over j yields $\mathrm{Var}(Z \mid \boldsymbol{s}) = \frac{\tau}{N}\|\boldsymbol{s}\|_2^2 + \tau \cdot \frac{\eta(\eta+1)}{3}$. If $\boldsymbol{s}$ is modeled with i.i.d. coefficients uniform in $\{-\eta, \ldots, \eta\}$, then $\mathbb{E}\|\boldsymbol{s}\|_2^2 = N\eta(\eta+1)/3$, and hence $\mathbb{E}_{\boldsymbol{s}}[\mathrm{Var}(Z \mid \boldsymbol{s})] = 2\tau\eta(\eta+1)/3$, which matches σ_d^2 in Lemma 4. Finally, since $\mathbb{E}[Z \mid \boldsymbol{s}] = 0$, the local-CLT approximation gives $\Pr[Z = 0 \mid \boldsymbol{s}] \approx 1/\sqrt{2\pi\,\mathrm{Var}(Z \mid \boldsymbol{s})}$.

3.4 Minimum number of faults for Dilithium

For the concrete instantiation, we evaluate the minimum number of faults from the three Dilithium parameter sets at NIST levels L2, L3, and L5. In all cases $N = 256$ and the specific parameters are $(\eta, \tau) \in \{(2, 39), (4, 49), (2, 60)\}$ for all three levels [20]. Using Lemma 4 we obtain the Gaussian upper bound p_{gauss} on the single-row collision probability. Finally, applying Theorem 1, we compute the theoretical threshold $M_{\min}$ (for a probability of 99%, i.e., $\lambda = 7$) as the smallest number of faults. The resulting values of p_{gauss} and $M_{\min}$ for the three security levels are summarized in Table 2.

Table 2: Instantiation of theoretical parameters for Dilithium

Scheme / security level	η	τ	p_{gauss}	$M_{\min}$(99% probability)
Dilithium-L2	2	39	3.194×10^{-2}	122
Dilithium-L3	4	49	1.561×10^{-2}	137
Dilithium-L5	2	60	2.575×10^{-2}	114

By Lemma 1, once M effective equations have been collected, the resulting equation system is fully covered with high probability in the sense that all $N = 256$ coefficients of $\boldsymbol{s}$ appear in the constraints and hence no variable remains unconstrained. Conditioned on this event, Theorem 1 implies that for $M \geq M_{\min}$ the bounded solution set over $[-\eta, \eta]^N$ only contains $\boldsymbol{s}$. Therefore, at $M = M_{\min}$ key recovery reduces to solving a bounded instance with N variables and M constraints, which can be done with a time complexity of $(\log N)^{O(N)} \cdot (M \log \beta)^{O(1)}$ via an MILP model for the ILP instance by [25, Theorem 41], yielding the unique solution $\boldsymbol{s}$.

$$
\begin{array}{lll}
\text{objective function} & f(\boldsymbol{s}) & \\
\text{subject to} & C_i \boldsymbol{s} = x_i \quad & \forall\, i \in [0, M-1] \qquad (1) \\
& -\eta \le \boldsymbol{s}_j \le \eta \quad & \forall\, j \in [0, N-1] \qquad (2)
\end{array}
$$

Fig. 1: MILP formulation with bounded integer variables and equality constraints.

4 Improved Key-Recovery Method via an MILP Model

This section presents our key-recovery pipeline from a mixed integer linear programming viewpoint. We first specify the MILP formulation used in the offline recovery stage. We then describe two adaptive online equation collection strategies and offline MILP solving, corresponding to the plain skipping-fault setting and the shuffling countermeasure.

4.1 MILP Model

We model the key recovery procedure as an integer feasibility problem over a bounded domain. After collecting faulty signatures and running the correction procedure, we obtain a linear system $C\boldsymbol{s} = x$ with $C \in [-1, 1]^{M \times N}$ and $\|x\|_\infty \le \beta$. The unknown vector $\boldsymbol{s}$ represents one polynomial component of secret $\boldsymbol{s}_1$ with N coefficients. We constrain the coefficients by the bound $\|\boldsymbol{s}\|_\infty \le \eta$. In parameter settings where the feasible solution is unique with high probability, the choice of objective does not affect correctness but can significantly influence solver performance. For notational convenience, we denote the objective function by $f(\boldsymbol{s})$. Figure 1 gives the MILP formulation for key recovery.

Algorithm 2 SolveMILP

Input: $C \in \mathbb{Z}^{M \times N}$, $x \in \mathbb{Z}^M$, bound η, *time_limit*
Output: $(solved, sol)$

$\boldsymbol{s}[0..N-1] \leftarrow$ Instantiate integer vars with domain $[-\eta, \eta]$
for $i \leftarrow 0$ **to** $M-1$ **do**
    AddConstr$(C_i \boldsymbol{s} = x[i])$
SetObjective$(f(\boldsymbol{s}))$
$status \leftarrow$ Optimize()
if $status \in \{$OPTIMAL$\}$ **then**
    for $j \leftarrow 0$ **to** $N-1$ **do**
        $sol[j] \leftarrow \boldsymbol{s}[j]$
    return (true, sol)
else if Failed to solve within *time_limit* **then**
    return (false, $\perp$)

To clearly describe how to utilize an MILP solver for determining the secret key coefficients, we present Algorithm 2 SolveMILP. SolveMILP instantiates integer variables for $\boldsymbol{s}$ with $-\eta \leq \boldsymbol{s}_j \leq \eta$ (Line 1). Based on the linear equations obtained in the online phase, lines 2-3 of Algorithm 2 construct M equality constraints $C_i\boldsymbol{s} = x_i$. Line 4 sets the objective function and describes it, improving the efficiency of the MILP solver. Based on the above objective function and constraint conditions, the solver is initiated to find the optimal solution using the underlying optimization algorithm (Line 5). Once the solver finds an optimal solution, it reports success and the corresponding solution vector $\boldsymbol{s}$ is extracted (Line 6-9). If the solver fails to find a solution within the given limited time, it terminates and reports failure (Line 10-11).

4.2 Adaptive MILP Attack in the Plain Skipping-Fault Setting

In the plain skipping-fault attack, our target is the partial private key $\boldsymbol{s}_1$ of Dilithium with no countermeasures. The recovery of one component $(\boldsymbol{s}_1)_j$ involves two phases, i.e., an online phase and an offline phase. During the online phase, equations are collected, and in the offline phase, they are solved with an MILP solver to recover $\boldsymbol{s}_1$.

Online phase. During the online phase, the attacker targets component j. Then, the attacker repeatedly triggers signatures with chosen messages, injecting an instruction-skipping fault at a chosen coefficient position (e.g., $(j, 0)$) when computing $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$. The resulting faulty signature is processed through a bounded correction search, where the candidate coefficient α is restricted to the range $[-\beta, \beta]$. This search works by iteratively modifying the signature and rerunning the public verification process until a valid correction is found. Only the chosen position (e.g., $(j, 0)$) whose correction makes verification pass is accepted. Each successful correction yields one equation for the targeted component, which is added to the equation set S.

Offline phase. Once at least M equations have been collected for a component, we utilize an MILP solver for key recovery based on the model in Figure 1. The candidate solution is checked by reconstructing signatures and running the public verification algorithm. If the check fails, the online phase continues collecting additional equations and the MILP is invoked again.

More specifically, Algorithm 3 presents the entire adaptive MILP attack in the plain skipping-fault setting. In the attack, we select an arbitrary message m (Line 1). For each component index j, we initialize the equation set S, the counter n of collected equations, and the current target coefficient position set $\mathcal{T}$ (Lines 2–5). In the online phase, we repeatedly inject faults targeting $(j, 0)$ and invoke EqFromSig to obtain correction equations, which are added to S (Lines 6–12). Once $n \geq M$, we invoke $\mathsf{SolveMILP}(S)$ to solve for $(\boldsymbol{s}_1)_j$ in the offline phase (Lines 13–17). After all components are processed, the algorithm returns the reconstructed $\boldsymbol{s}_1$ (Lines 18–19).

Algorithm 3 Adaptive MILP attack in the plain skipping-fault setting

Input: Public key pk, faulty signing oracle $\mathcal{O}_{\text{skip}}$, threshold M, β
Output: Recovered secret vector $\boldsymbol{s}_1$

select an arbitrary message as m
for $j \leftarrow 0$ **to** $\ell - 1$ **do**
  $S \leftarrow \emptyset$ ▷ Initialize equation set for component $(\boldsymbol{s}_1)_j$
  $n \leftarrow 0$
  $\mathcal{T} \leftarrow \{(j, 0)\}$ ▷ Chosen coefficient position for fault injection
  while $n < N$ **do**
    $(j, i, eq, eq_sol) \leftarrow \textsc{EqFromSig}(\mathcal{O}_{\text{skip}}, \mathsf{pk}, m, \mathcal{T}, \beta)$ ▷ $(c\boldsymbol{s}_1)_j[i] = \alpha \in [-\beta, \beta]$
    if $(j, i, eq, eq_sol) = \perp$ **then**
      continue ▷ Online phase
    $S.\mathsf{append}((eq, eq_sol))$
    $n \leftarrow n + 1$
    if $n \geq M$ **then**
      $(solved, \boldsymbol{s}) \leftarrow \mathsf{SolveMILP}(S)$ ▷ MILP solution in the offline phase
      if *solved* **then**
        $(\boldsymbol{s}_1)_j \leftarrow \boldsymbol{s}$
        break
$\boldsymbol{s}_1 \leftarrow ((\boldsymbol{s}_1)_0, (\boldsymbol{s}_1)_1, \ldots, (\boldsymbol{s}_1)_{\ell-1})$
return $\boldsymbol{s}_1$

4.3 Adaptive MILP Attack under Shuffling

A natural countermeasure against skipping faults is to randomize the order in which coefficient-wise additions for computing $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$ are performed during signing. Under fine-grained shuffling, a fault attempt no longer deterministically identifies a preselected component and coefficient, since the skipped addition may occur at an unpredictable position. However, such countermeasures merely increase the complexity of fault injection rather than fundamentally preventing attacks. The attacker can still adapt their strategies to overcome shuffling and extract sensitive information through a combination of online fault injection and offline data analysis.

Online phase. During the online phase, the attacker repeatedly triggers signatures with chosen messages and injects instruction-skipping faults during the shuffled computation of $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}_1$. Due to shuffling, the exact location of a successful fault injection is not known in advance. For each faulty signature, the attacker therefore performs a correction search over all possible component and coefficient positions, where the candidate coefficient α is restricted to the range $[-\beta, \beta]$. Each successful correction resolves the fault location and yields one linear equation, which is added to the corresponding equation set.

Offline phase. For each component, once the number of collected equations reaches the given threshold M, the attacker invokes an MILP solver to attempt recovery. If the solver succeeds, the recovered component is marked as solved and is no longer subjected to further solving attempts. Otherwise, the attacker

Algorithm 4 Adaptive MILP attack under shuffling

Input: Public key pk, sign-with-fault oracle $\mathcal{O}_{\mathsf{skipShuff}}$, threshold M, β
Output: Recovered secret vector $\boldsymbol{s}_1$

1: select an arbitrary message as m
2: $\mathcal{T} \leftarrow \{0, \ldots, \ell - 1\} \times \{0, \ldots, N - 1\}$
3: **for** $j \leftarrow 0$ **to** $\ell - 1$ **do**
4: $\quad S_j \leftarrow \emptyset$ ▷ Initialize equation set for component $(\boldsymbol{s}_1)_j$
5: $\quad n_j \leftarrow 0$
6: $\quad \mathsf{solved}_j \leftarrow \mathsf{false}$
7: **for** $k \leftarrow 0$ **to** $\ell \cdot N - 1$ **do**
8: $\quad (j, i, eq, eq_sol) \leftarrow \textsc{EqFromSig}(\mathcal{O}_{\mathsf{skipShuff}}, \mathsf{pk}, m, \mathcal{T}, \beta)$ ▷ $(c\boldsymbol{s}_1)_j[i] = \alpha \in [-\beta, \beta]$
9: $\quad$ **if** $(j, i, eq, eq_sol) = \perp$ **then**
10: $\quad\quad$ **continue** ▷ Online phase
11: $\quad S_j.\mathsf{append}((eq, eq_sol))$
12: $\quad n_j \leftarrow n_j + 1$
13: $\quad$ **if** $\mathsf{solved}_j = \mathsf{false}$ **and** $n_j \geq M$ **then**
14: $\quad\quad (solved, \boldsymbol{s}) \leftarrow \mathsf{SolveMILP}(S_j)$ ▷ MILP solution in the offline phase
15: $\quad\quad$ **if** *solved* **then**
16: $\quad\quad\quad (\boldsymbol{s}_1)_j \leftarrow \boldsymbol{s}$
17: $\quad\quad\quad \mathsf{solved}_j \leftarrow \mathsf{true}$
18: $\quad$ **if** $\forall j \in [0, \ell - 1]$, $\mathsf{solved}_j = \mathsf{true}$ **then**
19: $\quad\quad$ **break**
20: $\boldsymbol{s}_1 \leftarrow ((\boldsymbol{s}_1)_0, (\boldsymbol{s}_1)_1, \ldots, (\boldsymbol{s}_1)_{\ell-1})$
21: **return** $\boldsymbol{s}_1$

returns to the online phase to collect additional equations. This process continues adaptively until all components of $\boldsymbol{s}_1$ have been successfully recovered.

Algorithm 4 describes the adaptive MILP attack in the shuffling setting. We select an arbitrary message m and define the target coefficient position set $\mathcal{T}$ (Lines 1–2). For each component j we initialize an equation set S_j, a counter n_j of collected equations, and a boolean flag solved_j indicating whether the corresponding secret component has been recovered (Lines 3–6). We then repeatedly invoke EqFromSig to inject faults and extract correction equations (Lines 7–12). Due to shuffling, each successful invocation returns an equation associated with a component index j, which would be controllable in the plain setting but is random under shuffling. The extracted equation is appended to S_j and the corresponding counter n_j is increased. Once $n_j \geq M$ and the j-th component has not been solved yet, we invoke $\mathsf{SolveMILP}(S_j)$ to recover $(\boldsymbol{s}_1)_j$ (Lines 13–14). If the recovery succeeds, we mark this component as solved. The procedure continues until all components are solved, after which the secret vector $\boldsymbol{s}_1$ is reconstructed (Lines 15–21).

5 Experimental Results

To evaluate the improved skipping fault correction attacks in Section 4, we implemented the improved key recovery procedures of the attack, based on the classic skipping fault correction attack implementation on Dilithium [15]. Faults were simulated by programmatically injecting them into the `crypto_sign_signature` function in the Dilithium implementation [1]. All experiments were performed on a desktop equipped with an 11th Gen Intel i7 2.5 GHz CPU, and the results are reported below.

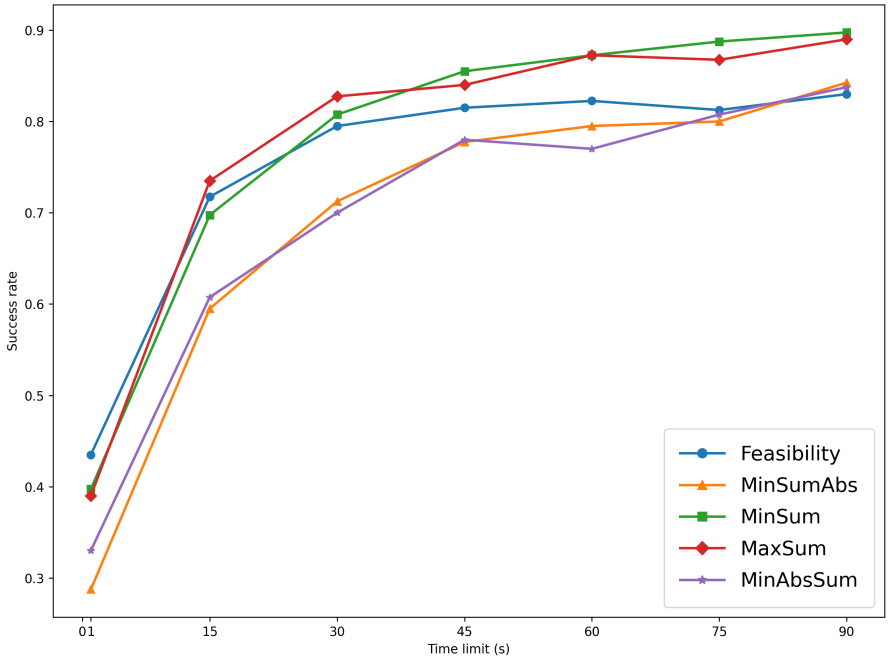


Fig. 2: Success rate vs. time limit for different objective functions.

5.1 Objective Function Determination for the MILP Solver

When solving our bounded integer programs with an MILP solver (Gurobi), the choice of the objective function can have a significant impact on practical solver efficiency. To maximize the probability of recovering the key $\boldsymbol{s}_1$ within a limited runtime, we benchmark five objective options and select a default for subsequent experiments. Let $\boldsymbol{s} \in \mathbb{Z}^N$ denote the unknown coefficient vector of a single target component. We evaluate five objective functions: (i) Feasibility, meaning a constant objective and the solver is asked to find any feasible integer point, (ii) MinSumAbs, minimizing the L1-norm $\sum_{i=1}^{N} |s_i|$, (iii) MinSum, minimizing

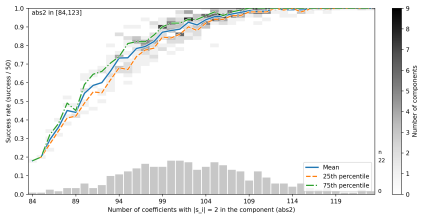
the linear sum $\sum_{i=1}^{N} s_i$, (iv) MaxSum, maximizing the linear sum $\sum_{i=1}^{N} s_i$, (v) MinAbsSum, minimizing the absolute sum $\left|\sum_{i=1}^{N} s_i\right|$.

For each instance, we collect exactly 193 usable equations via the fault injection procedure to recover the private key $\boldsymbol{s}_1$ using the Gurobi solver with time limits $T \in \{1, 15, 30, 45, 60, 75, 90\}$ seconds. For each time limit T, we perform 100 independent experiments with distinct private keys, totaling 400 polynomial components.[9] Each objective function is evaluated under this identical setup and linear system. We record the number of successful recoveries and calculate the success rate. The full curves for five objective functions are shown in Figure 2.

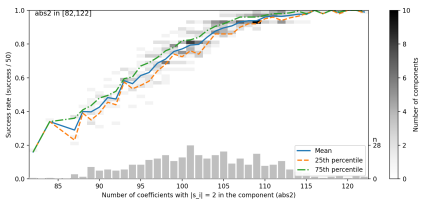
As shown in Figure 2, for any objective function, success rates increase with longer time limits before flattening out after 60 seconds. The Feasibility case performs best when time limits are very short. Once the time limit exceeds 30 seconds, both the MinSum and MaxSum approaches outperform the others. MinSum and MaxSum achieve nearly identical success rates at $T = 60$ seconds. When the time limit is extended, MinSum is slightly better than MaxSum. Accordingly, we adopted MinSum as the default objective function and a 60-second time limit ($T = 60$s) in subsequent experiments.

5.2 Impact of Private Key Feature on Recovery Success Rate

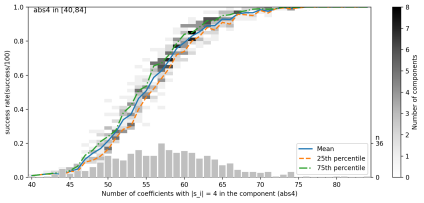
Under identical attack conditions, the MILP solver yielded varying recovery success rates for different private keys of Dilithium. To this end, we conducted experiments to explore which properties of private keys make them more susceptible to recovery.



(a) Success rate vs. abs2 for L2.



(b) abs2 for L2 with Feasibility.



(c) abs4 for L3 with MinSum.

[9] Each Dilithium-L2 key contains $\ell = 4$ components.

Setting the number of faults $M = 193$, we sample 100 independent private keys and evaluate each key under the same attack conditions (Gurobi, MinSum objective, 60s time limit). For each private key, we repeat the attack 50 times, and report the corresponding success rate. From these tests, we observe that the success rate correlates with the number of coefficients in the component whose absolute value is 2. For ease of description, we define this number as $\text{abs2} := \#\{i : \ |s_i| = 2\}$. Figure 3a presents the relationship and distribution diagram between the key-recovery success rate and abs2 for the private key feature corresponding to each component of Dilithium-L2.

In Figure 3a, the heatmap encodes the number of components that achieve a particular success rate (with step size 0.02, based on 50 repeated trials) for the fixed setting $M = 193$ and a 60 s time limit. Curves overlaid on the heatmap indicate the mean and the 25/75 percentiles for each abs2 value. The bottom bar chart shows the distribution of sampled components across different abs2 values. The right axis ($n \in [0, 22]$) indicates the number of components with the same abs2 value.

Over the full range, the mean success rate increases sharply from low values at abs2 ≈ 84 to near-perfect recovery for abs2 $\gtrsim 110$. Most components fall in the range abs2 $\in [92, 112]$, where the success rate steadily climbs and approaches 1 when abs2 gets to 112. The heatmap and interquartile range in this range further indicate that this trend is not driven by a few outliers, but persists across many components.

The trend curve of success rate in Figure 3a-3c follows Remark 2. As abs2 or abs4 (L3) increases, the conditional variance $\text{Var}(Y \mid \boldsymbol{s})$ grows, which reduces the single-row collision probability and makes the system more constrained. Consequently, components with larger abs2 are easier for the MILP solver to recover.

5.3 Impact of the Number of Faults on Key Recovery Success

To further explore factors affecting the key-recovery success rate, we conducted experiments on the impact of the number of faults on the success rate. To obtain a more universal conclusion, we select private keys with different abs2 values during the experiments.

We focus on abs2 $\in [92, 112]$ because the distribution observed in Section 5.2 is most concentrated in this interval. For each abs2 $\in \{92, 97, 102, 107, 112\}$, we randomly sample 80 components with the prescribed abs2 value. For every sampled component, we run the online phase and keep collecting faulty signatures until we obtain exactly M usable equations. We then solve $\boldsymbol{s}_1$ with an MILP solver under the following conditions (Gurobi, MinSum objective, 60s time limit from Section 5.1).

Figure 4 plots the success rate as a function of $M \in \{183, 185, ..., 191, 193\}$, with one curve for each abs2. The curve in Figure 4 shows that with abs2 fixed, the success rate increases as the number of faults increases. In parallel, with the number of faults fixed, the success rate increases as abs2 increases. Specifically, at abs2 = 92, the success rate rises from 0% (at $M = 183$) to 62.5% ($M = 193$).

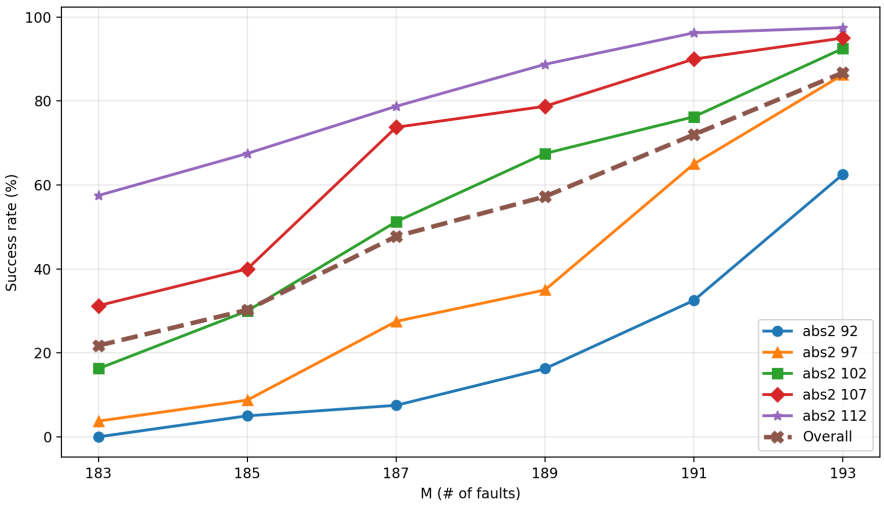


Fig. 4: Success rate vs. the number of faults M

Meanwhile, at $M = 193$, it increases from 62.5% (abs2 = 92) to 97.5% (abs2 = 112).

Overall, these measurements validate two key observations: (1) increasing the number of faults improves recovery performance, and (2) secret components with larger abs2 are easier for the MILP solver to recover under the same conditions. This is consistent with the analysis in Section 3.3 that larger M makes the system more constrained, while components with larger abs2 tend to produce equations that are easier for the MILP solver to resolve.

5.4 Effectiveness of the Improved Attack Model

We then evaluate the effectiveness of our improved attack model and MILP based recovery strategy. We use the correction fault attack of Krahmer et al. [15] as a baseline for comparison[10]. We keep the attacker capability[11] unchanged, using the same fault primitive as in the baseline, and vary only the recovery strategy and the handling of extracted constraints. We consider two attack scenarios. In the plain setting without countermeasures, each extracted equation can be attributed to its fault location. In the shuffling setting, shuffling hides the fault location, so extracted equations cannot be directly localized.

Adaptive MILP Attack in the Plain Skipping-Fault Setting. To get a fair comparison, we evaluate our plain adaptive attack under two complementary experimental conditions, namely, the improved plain attack I with low time

[10] We reproduced the experiments of [15] on our machine.
[11] Every fault injection is effective.

Table 3: **Effectiveness of the improved attack model.** Average end-to-end time and average number of injected faults for plain and shuffling settings. Plain results are averaged over 200 trials and shuffling results over 10 trials.

Attack model	avg. time	avg. # of faults	Ref
Dilithium-L2			
Plain attack	1.5 sec	1024	[15]
Shuffling adapted attack	17 min	1157	[15]
Improved plain attack I	3.3 sec	805	This work
Improved plain attack II	8.7 min	759	This work
Improved shuffling adapted attack	19 min	861	This work
Dilithium-L3			
Plain attack	4.7 sec	1280	[15]
Shuffling adapted attack	1 h 16 min	1374	[15]
Improved plain attack I	8.6 sec	1113	This work
Improved plain attack II	16.2 min	1073	This work
Improved shuffling adapted attack	1 h 27 min	1189	This work
Dilithium-L5			
Plain attack	8.3 sec	1792	[15]
Shuffling adapted attack	2 h 27 min	2075	[15]
Improved plain attack I	9.2 sec	1412	This work
Improved plain attack II	14.1 min	1334	This work
Improved shuffling adapted attack	2 h 14 min	1529	This work

complexity and the attack II with a small number of faults in Table 3. First, in attack I we use a short solver time limit $T = 1$ s and set $M_{\text{try}} = (200, 220, 200)$ at $(\text{L2}, \text{L3}, \text{L5})$, for which recovery is almost always achieved in a single solver call. We run 200 independent trials with fresh private keys, and whenever an MILP attempt fails we collect four additional usable equations and retry. Under this setting, our average time is comparable to [15] while the average number of faults is reduced by $(\text{L2}, \text{L3}, \text{L5}) = (21.4\%, 13.0\%, 21.2\%)$.

Second, in attack II we use the default solver configuration from Section 5.1 with $T = 60$ s and start from a smaller budget $M_{\text{try}} = (184, 204, 184)$ at $(\text{L2}, \text{L3}, \text{L5})$, motivated by the observation that components with larger abs2 already exhibit at least 60% recovery in Section 5.3. We apply the same retry rule by adding four equations after each failed attempt. This setting further reduces the required faults by $(\text{L2}, \text{L3}, \text{L5}) = (25.9\%, 16.2\%, 25.6\%)$, at the cost of increased runtime due to potentially multiple solver invocations per component.

Adaptive MILP Attack under Shuffling. We next evaluate the effectiveness under the shuffling countermeasure. Following Algorithm 4 in Section 4.3 and using the setting from attack II above, we run the same adaptive recovery procedure. Whenever an MILP attempt fails, we collect four additional usable equa-

tions and retry. Overall, this strategy reduces the required number of injected faults in Table 3, with fault reductions of $(\mathrm{L2}, \mathrm{L3}, \mathrm{L5}) = (25.6\%, 13.5\%, 26.3\%)$.

The shuffling results are averaged over 10 trials, while the results for the plain setting are averaged over 200 trials. This difference is caused by the higher computational cost of each shuffling trial. Under shuffling, the fault location is unknown, so extracting one equation requires a correction search over all $\ell \times N$ coefficient positions for each faulty signature. In the plain setting, the target position is known and only that position needs to be tested. As shown in Table 3, one shuffling trial therefore takes several hours, while one plain trial takes only seconds. Krahmer et al. [15] also report shuffling results over 10 trials because of the same computational cost.

6 Discussion

As the number of faults increases, the success rate of key recovery also increases. This consistency is demonstrated both theoretically and experimentally. Equation 2 in Section 3.3 can be derived to show that a larger number of faults M leads to a higher probability of obtaining a unique key solution through solving the system of linear equations. Simultaneously, Section 5.3 provides experimental results illustrating the trend of success rate growth with increasing the number of faults. The experimental results show that under the condition abs2 = 112, the success rate increases from 57.5% at 183 faults to 97.5% at 193 faults.

When the key recovery success rate is relatively high, there exists a certain gap between the theoretical and experimental values for the minimum number of faults M_{min}. Taking Dilithium-L2 as an example, in Table 2 of Section 3.4, where the key recovery success rate approaches 100%, the given theoretical minimum for Dilithium-L2 is $M_{min} = 122$. Whereas, the test results presented in Section 5.3 show that under the condition abs2 = 112, with $M = 193$, the key recovery success rate is 97.5%. This discrepancy is expected, as the theoretical analysis is based on an idealized model and does not account for several practical factors arising in real attacks. We will next discuss which factors affect the experimental results in real-world key recovery attacks.

The efficiency of generic MILP solving. The experimental recovery rate is further limited by the efficiency of generic MILP solving. Even when the bounded solution is unique, MILP solvers rely on a branch-and-bound style search whose runtime varies significantly across instances and depends on available computational resources. In particular, limited memory or a small number of cores can prevent the solver from exploring enough branches within the given time, so timeouts or solver failure may occur despite uniqueness.

The limited time for solving. Our experiments are designed to collect large-scale statistics, so we impose relatively short per-instance solver time limits and sample private keys uniformly at random. In Section 5.1, when fixing $M = 193$ and using the time limit $T = 1$ second, the MILP solver often cannot explore enough search nodes. The success rate increases steadily as the running time grows and tends to flatten out around $T \approx 60$ seconds.

The private key feature. The recovery performance exhibits a clear key-dependent trend reflected in the private key feature abs2, defined as the number of coefficients satisfying $|s_i| = 2$. As shown in Figure 3a in Section 5.2, with the attack parameters fixed (including $M = 193$ and a 60-second time limit), the success rate correlates strongly with abs2, and components with larger abs2 approach near-perfect recovery.

Furthermore, to illustrate the impact of time and private key feature on success rate, we ran an additional targeted experiment at Dilithium-L2. In contrast to the short per-instance solution time, we fix $M = 164$ and allow a much larger solver time limit of $T = 60$ minutes per instance for 30 trials, which leads to fewer timeouts and thus higher success rates. At the same time, to examine the effect of the private key distribution, we select secret components with abs2 $= 140$ (which occur on average once every 4×10^5 private keys), thereby focusing on high-abs2 instances. Under these conditions, the recovery succeeds in 36.7% of the trials, indicating that the private key can be recovered with a smaller number of faults within a practical time budget, especially for high-abs2 instances.

Remarks on additional countermeasures. Our work focuses on shuffling because it directly hides the fault location used in the plain attack. **Sign-then-verify** and **double computation** [15] protect the implementation in a different way by preventing faulty signatures from being released. Sign-then-verify runs verification inside the signing device and does not return a signature that fails the internal verification. The attacker therefore cannot obtain the faulty signature required by EqFromSig and cannot use public verification as a correction oracle. Double computation performs the signing computation twice and compares the results before releasing the signature. A successful attack would therefore require compatible faults in both executions, typically at the same coefficient position. This requirement is more difficult to satisfy than the single skipping fault considered in the plain setting. We do not evaluate our attack under these countermeasures and leave their quantitative analysis for future work.

7 Conclusion

In this work, we study skipping fault correction attacks against randomized Dilithium by modeling the offline recovery problem as a bounded integer equation system. We show that collecting a full-rank system of equations is not required for key recovery. By exploiting the bounded small-integer structure of the secret coefficients, we derive sufficient conditions under which an underdetermined system admits a unique bounded solution with high probability, yielding explicit fault thresholds for all Dilithium parameter sets. Based on this analysis, we propose adaptive MILP-based correction attacks for both the plain and the shuffling settings, which contain online fault injection and offline key recovery phases. Experimental results show a reduction in the number of required fault injections compared to prior work, while remaining consistent with the theoretical analysis.

Our analysis and experiments reveal a dependence on a concrete feature of the secret key instance. In particular, secret components with higher abs2 values are observed to be easier to recover under identical attack conditions. This observation suggests that, from an implementation perspective, private keys with unusually large abs2 values could be avoided during key generation.

Finally, there remains a gap between the sufficient minimum number of faults derived in theory and the number of faults observed in practice. Despite the various factors discussed in Section 6, future work will explore more effective methods to further reduce the number of required fault injections and improve fault efficiency.

References

1. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Reference implementation of Dilithium. `https://github.com/pq-crystals/dilithium` (2023), gitHub repository, accessed Dec 31, 2025
2. Bai, S., Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: Algorithm specifications and supporting documentation (version 3.1). `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf` (2021), homepage
3. Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: Cryptographers' Track at the RSA Conference. pp. 28–47. Springer (2014)
4. Bronchain, O., Azouaoui, M., ElGhamrawy, M., Renes, J., Schneider, T.: Exploiting small-norm polynomial multiplication with physical attacks application to crystals-dilithium. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 359–383 (2024). `https://doi.org/10.46586/TCHES.V2024.I2.359-383`, `https://doi.org/10.46586/tches.v2024.i2.359-383`
5. Bruinderink, L.G., Pessl, P.: Differential fault attacks on deterministic lattice signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 21–43 (2018)
6. Chen, Z., Karabulut, E., Aysu, A., Ma, Y., Jing, J.: An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature. In: 2021 IEEE 39th International Conference on Computer Design (ICCD). pp. 583–590. IEEE (2021)
7. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 238–268 (2018)
8. ElGhamrawy, M., Azouaoui, M., Bronchain, O., Renes, J., Schneider, T., Schönauer, M., Seker, O., van Vredendaal, C.: From mlwe to rlwe: A differential fault attack on randomized & deterministic dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 262–286 (2023)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
10. von zur Gathen, J., Sieveking, M.: A bound on solutions of linear integer equalities and inequalities. Proceedings of the American Mathematical Society **72**(1), 155–158 (1978)

11. 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)
12. Islam, S., Mus, K., Singh, R., Schaumont, P., Sunar, B.: Signature correction attack on dilithium signature scheme. In: 2022 IEEE 7th European symposium on security and privacy (euroS&p). pp. 647–663. IEEE (2022)
13. Jr., H.W.L.: Integer programming with a fixed number of variables. Mathematics of Operations Research **8**(4), 538–548 (1983)
14. 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)
15. Krahmer, E., Pessl, P., Land, G., Güneysu, T.: Correction fault attacks on randomized crystals-dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 174–199 (2024)
16. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International conference on the theory and application of cryptology and information security. pp. 598–616. Springer (2009)
17. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 738–755. Springer (2012)
18. Mosca, M.: Cybersecurity in an era with quantum computers: Will we be ready? IEEE Security & Privacy **16**(5), 38–41 (2018)
19. NIST: PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. `https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4` (2022), accessed: Dec 31, 2025
20. NIST: FIPS 204, module-lattice-based digital signature standard (FIPS 204). Tech. Rep. FIPS 204, National Institute of Standards and Technology (2024), available at `https://doi.org/10.6028/NIST.FIPS.204`
21. NIST Information Technology Laboratory Computer Security Resource Center: Post-quantum cryptography standardization. `https://csrc.nist.gov/projects/post-quantum-cryptography` (2017), accessed: 31 December 2025
22. Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Side-channel assisted existential forgery attack on dilithium-a nist pqc candidate. Cryptology ePrint Archive (2018)
23. Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Exploiting determinism in lattice-based signatures: practical fault attacks on pqm4 implementations of nist candidates. In: Proceedings of the 2019 ACM Asia conference on computer and communications security. pp. 427–440 (2019)
24. Ravi, P., Yang, B., Bhasin, S., Zhang, F., Chattopadhyay, A.: Fiddling the twiddle constants - fault injection analysis of the number theoretic transform. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(2), 447–481 (2023). `https://doi.org/10.46586/TCHES.V2023.I2.447-481`, `https://doi.org/10.46586/tches.v2023.i2.447-481`
25. Reis, V., Rothvoss, T.: The subspace flatness conjecture and faster integer programming. In: 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2023). pp. 974–988. IEEE, Santa Cruz, CA, USA (2023)
26. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)

27. Steffen, H., Land, G., Kogelheide, L., Güneysu, T.: Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware. In: International Conference on Post-Quantum Cryptography. pp. 688–711. Springer (2023)
28. Ulitzsch, V.Q., Marzougui, S., Bagia, A., Tibouchi, M., Seifert, J.P.: Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ilp. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 367–392 (2023)
29. Ulitzsch, V.Q., Marzougui, S., Tibouchi, M., Seifert, J.: Profiling side-channel attacks on dilithium - A small bit-fiddling leak breaks it all. In: Smith, B., Wu, H. (eds.) Selected Areas in Cryptography - 29th International Conference, SAC 2022, Windsor, ON, Canada, August 24-26, 2022, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13742, pp. 3–32. Springer (2022). `https://doi.org/10.1007/978-3-031-58411-4_1`, `https://doi.org/10.1007/978-3-031-58411-4_1`
30. Wang, H., Gao, Y., Liu, Y., Zhang, Q., Zhou, Y.: In-depth correlation power analysis attacks on a hardware implementation of crystals-dilithium. Cybersecurity **7**(1), 21 (2024)
31. Zhou, Y., Wang, W., Sun, Y., Yu, Y.: Rejected signatures' challenges pose new challenges: Key recovery of crystals-dilithium via side-channel attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(4), 817–847 (2025)