Keep Track of Your Errors
Solving ILWE and Improving Attacks on ML-DSA

Mohamed ElGhamrawy[1] Thomas Eisenbarth[1,2] Julius Hermelink[3] Anja Rabich[1]
Florian Sieck[1] Silvan Streit[4,5] Jonas Thietke[6] Zhiyuan Zhang[3]

[1]*Institute of IT Security, University of Luebeck* [2]*DFKI* [3]*Max Planck Institute for Security and Privacy*
[4]*Fraunhofer AISEC* [5]*Technical University of Munich (TUM)* [6]*Ruhr-University Bochum*

{mohamed.elghamrawy, thomas.eisenbarth, a.rabich, florian.sieck}@uni-luebeck.de
{julius.hermelink, zhiyuan.zhang}@mpi-sp.org *silvan.streit@aisec.fraunhofer.de* *jonas.thietke@rub.de*

Abstract

With the widespread adoption of NIST's new signature standard ML-DSA imminent, understanding its vulnerability to side-channel attacks is increasingly important. Current approaches that are based on deriving (Concealed) Integer Learning with Errors (ILWE) samples from side information require hundreds of thousands of signatures even in noise-free conditions, limiting their practicality.

In this work, we show that the number of signatures required in these attacks has been drastically overestimated. Keeping track of the error probability distribution when deriving ILWE samples allows for a soft-analytic approach to solving ILWE. Concretely, we show that distribution hints (Eurocrypt 2025) may be derived from ILWE samples and the corresponding belief propagation-based solver can recover the secret key efficiently. We then conceptually compare previous solvers to our soft-analytic approach. Furthermore, we provide an information-theoretic analysis and answer an open question—on how to filter out relations—posed in a different line of attacks against ML-DSA (Crypto 2025); thereby, we can propose improvements in the application of these solvers.

We evaluate our approach across various ILWE and Concealed ILWE parameter sets. In addition, we analyze a recently discovered timing leakage in ML-DSA and show how to solve for the secret key with far fewer signatures. In a noise-free attack setting, our approach reduces the average number of signatures required by a factor of 62—from 139 million to 2.25 million—compared with linear regression, as originally proposed, with larger improvements under noise. Thus, we show that attacks that derive ILWE and Concealed ILWE instances require far fewer signatures than previously believed, and their impact has been underestimated.

1 Introduction

The National Institute of Standards and Technology (NIST) has standardized the lattice-based signature scheme Dilithium [5] under the name Module-Lattice-based Digital Signature Algorithm (ML-DSA) [65]; it is intended as primary post-quantum replacement for currently deployed primitives [1]. The post-quantum standard for Key Encapsulation Mechanisms (KEMs), called Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM), is already being deployed in practice, e.g., in Signal [48], iMessage [26], available in AWS [42], Firefox [29], and Chrome [3], and as of August 2026, 59% of Cloudflare's requests use ML-KEM [16]. While ML-DSA is not yet widely deployed in public TLS certificates, larger cloud service providers support ML-DSA in their key management services [2,28] or in active directory [60]. With wider deployment being imminent, understanding the real-world security of cryptosystems relying on ML-DSA is crucial, and several side-channel attacks—most prominently power and timing leakage-based—have been presented, e.g., [9, 13, 17, 20–22, 38, 53, 54, 72, 73, 78, 81, 86].

Several of these recent attacks [20–22, 53, 72, 81] target ML-DSA by deriving so-called Integer Learning with Errors (ILWE) [11] or Concealed Integer Learning with Errors (CILWE) [20] samples from side-channel leakage. The secret key is then recovered by solving the (C)ILWE problem using regression—Ordinary Least Squares (OLS), Huber [40], or Cauchy regression [62]. While prior work already demonstrates the practicality of this approach, these attacks require a very large number of samples, typically a six to seven figure number [81], when relying on these methods.

A different approach was taken in [13] and [38]. These power side-channel attacks approach leakage from a soft-analytic [83] perspective: the adversary derives probability distributions on intermediate variables and solves for the secret key by "combining" the leakage[1] using Belief Propagation (BP). This approach has been proven successful *for power side-channel attacks* in many settings such as, e.g., symmetric schemes [14, 44, 83], lattice-based KEMs [7, 35, 37, 69, 71], symmetric primitives in lattice-based KEMs [57], lattice-based signature schemes [13, 38], code-based schemes [6, 32, 34], or, MPC in the Head (MPCitH) [31, 57], with analysis in, e.g., [64] and libraries [15]. So far, re-

[1]Technically: computing the marginals of the joint probability distributions of all variables conditions on the leakage.

search into soft-analytical attacks is mostly focused on power side-channel attacks[2] and, except for works such as [39], it is tailored to concrete attack settings that do not generalize.

Our contributions. In this work, we show that soft-analytic techniques may also be applied to generically solve ILWE and CILWE problems and greatly improve upon timing side-channel attacks. We conceptually and empirically compare different approaches and demonstrate that previous attacks that derive ILWE problems are substantially more threatening than previously believed. In addition, we answer an open question posed in a recent work at CRYPTO 2025 [38] regarding how to filter information fed into the BP instantiation. We then employ the proposed techniques to analyze a recent timing leak [43] in PQClean [45]. Concretely, our contributions are the following:

- We propose a soft-analytic approach to solving ILWE and CILWE using the solver of [39] and provide a conceptual comparison of BP and OLS where we show that our approach and OLS asymptotically arrive at the same result under Gaussian assumptions.
- Using information theory, we solve the problem of filtering information BP first discussed in [38].
- Our evaluations show a reduction in required traces/samples by 7.5–51.5x for ILWE and 1.78–1.92x for CILWE; thereby, we show that previously proposed attacks on ML-DSA were substantially underestimated.
- We use the developed techniques to analyze and simulate a concrete recently discovered timing leakage; our techniques drastically reduce the required traces and remain effective under bit error rates of over 0.45.
- We demonstrate the timing attack against hedged ML-DSA executing inside an SGX enclave and estimate the collection time needed for key recovery. While the single-stepping slowdown of our current setup yields a collection time that is not necessarily realistic, we discuss improvements a real-world attacker could make.

More generally, we draw the conclusion that keeping track of noise, leakage/error distributions enables improvements in solving (C)ILWE and allows for more powerful attacks.

Overview of results. Table 1 provides a summary of the results of this work compared against prior work, namely [20–22]. Baseline results for ILWE and the timing attack are reported using scaled OLS as described in [22], while baseline results for CILWE are reported using Cauchy regression as described in [20]. Notably, our BP solver significantly reduces the number of relations/signatures required for key recovery from noisy bit leakage when compared to that of [22]. Moreover, for CILWE, we substantially decrease the number of samples required and can tolerate concealment of 0.99.

Setting	Baseline	Our work	Improvement
ILWE ($\chi_e = \mathcal{U}(196)$)			
$B_a = 1$	514,500	10,000	51.5x
$B_a = 5$	46,000	2,750	16.7x
CILWE ($q = 0.90$)			
ML-DSA-44	15,780	8,886	1.78x
Timing Attack (Section 6)			
ML-DSA-44	1.39×10^8	2.25×10^6	62x

Table 1: Summary of main results.

Concurrent and recent related work. Four recent preprints also improve upon solving for the secret key in ML-DSA from obtained relations. The work of [85], concurrent to ours, also applies several solvers to ILWE and CILWE problems including BP. Their work evaluates further solvers, such as, e.g., the greedy solver of [39], and focuses on an empirical comparison. Their work also improves upon the hill-climbing algorithm presented in [76], which in-turn also improves upon [21, 22]. A different approach is taken by [8]: the authors build a statistical model of (bit leakage in) ML-DSA and use a SAT solver to recover the secret key from noisy single-bit leakage. In [67], the authors observe that relations itself give information on the correctness probability of observed bits; they use this to derive an improved regression-based recovery algorithm and for a pre-processing step that flips bits based on their derived correctness probability.

In contrast to [85], we provide a detailed analysis of how to apply [39] with arbitrary distributions. We also provide detailed conceptual comparisons between linear regression and BP, improvements on the BP, comparison of threat models of previous attacks, and analyze a recent timing side-channel leakage [43]. The authors of [85] instead focus on evaluating further known and new solvers. Where the evaluations of [85] and ours overlap, they seem[3] to largely agree, except for that our implementation also improves upon the case of concealment rates > 0.8 where their BP degrades. Comparing BP-based approaches to [8] could potentially be seen as an extension of works such as [33, 79]. The work of [76] explicitly mentions the question of applying BP to their setting which we answer in this work. The pre-processing step of [67] is implicitly considered in the BP without the requirement for an explicit step simply by keeping track of the correct error distributions. Another main contribution of [67] is a method to estimate the probability of bit flips given relations. We

[2] Though, e.g., [37] technically applies more generally.

[3] To the best of our knowledge, no artifact for [85] is available yet.

assume this probability to be known (e.g., from a profiling device); their method can likely be applied to improve upon the practicality of our attacks.

2 Background

Notation. We denote the i-th coefficient of a polynomial p as $p_{[i]}$, the i-th coefficient of the j-th polynomial in a vector $\mathbf{a}$ as $\mathbf{a}_{[j,i]}$. When the lowest index is omitted, we refer to the complete j-th polynomial $\mathbf{a}_{[j]}$. We write $x \sim \chi$ to denote that the random variable x is drawn from the distribution χ. We write $P_\chi(x)$ to denote the probability mass function of χ at x; the subscript is omitted when the distribution is clear from the context. For $a, b \in \mathbb{Z}$ with $a \leq b$, let $\mathcal{U}(a,b)$ be the discrete uniform distribution over $\{a, \ldots, b\}$. We abbreviate $\mathcal{U}(-B,B)$ as $\mathcal{U}(B)$. For $\mu \in \mathbb{R}$ and $\sigma > 0$, let $\mathcal{N}(\mu, \sigma^2)$ denote the normal distribution with mean μ and variance σ^2. We write $D_{\mathbb{Z},\sigma}$ for the centered discrete Gaussian over $\mathbb{Z}$ with parameter σ.

2.1 ML-DSA

ML-DSA [65] is the standardized variant of the Dilithium [5, 25] signature scheme. ML-DSA defines three security levels, its security is based on the Module Learning with Errors (MLWE) [12, 50] and the Module Short Integer Solution (MSIS) problem [47, 50] , it operates over $R = \mathbb{F}_q[x]/(x^n+1)$, where $n = 256$ and $q = 2^{23} - 2^{13} + 1$.

Algorithm 1 ML-DSA.KeyGen Simplified

Input: Randomness seed ζ
Output: Public key $\mathtt{pk}$, secret key $\mathtt{sk}$

$\rho, \rho', K \leftarrow H(\zeta)$
$\mathbf{A} \in R^{k \times \ell} \xleftarrow{S} ExpandA(\rho)$
$(\mathbf{s}_1, \mathbf{s}_2) \in R^\ell \times R^k \xleftarrow{S} H(\rho')$
$\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
$(\mathbf{t}_1, \mathbf{t}_0) \leftarrow Power2Round(\mathbf{t})$
$tr \leftarrow H(\rho, \mathbf{t}_1)$
return $\mathtt{pk} = (\mathbf{t}_1, \rho), \mathtt{sk} = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

Algorithm 2 ML-DSA.Verify Simplified

Input: Public key $\mathtt{pk}$, message $\mathtt{m}$, Signature $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$
Output: Boolean

if $\mathbf{h} = \perp$ **then return** false
end if
$\mathbf{A} \in R^{k \times \ell} \xleftarrow{S} ExpandA(\rho)$
$\mu \leftarrow H(H(\mathtt{pk}), \mathtt{m})$
$c \leftarrow SampleInBall(\tilde{c})$
$\mathbf{w}'_{Approx} \leftarrow \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d$
$\mathbf{w}'_1 \leftarrow UseHint(\mathbf{h}, \mathbf{w}'_{Approx})$
$\tilde{c}' \leftarrow H(\mu, \mathbf{w}'_1)$
return $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = \tilde{c}'$

Algorithm 3 ML-DSA.Sign Simplified

Input: Secret key $\mathtt{sk} = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$, message $\mathtt{m}$, randomness rnd
Output: Signature $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

$\mathbf{A} \in R^{k \times \ell} \leftarrow ExpandA(\rho)$
$\mu \leftarrow H(H(\mathtt{pk}), \mathtt{m})$
$\rho'' \leftarrow H(K, rnd, \mu)$
$\kappa \leftarrow 0$
$(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
  $\mathbf{y} \in R^\ell \leftarrow ExpandMask(\rho'', \kappa)$
  $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$
  $\mathbf{w}_1 \leftarrow HighBits(\mathbf{w})$
  $\tilde{c} \leftarrow H(\mu, \mathbf{w}_1)$
  $c \leftarrow SampleInBall(\tilde{c})$
  $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$
  $\mathbf{r}_0 \leftarrow LowBits(\mathbf{w} - c\mathbf{s}_2)$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(\mathbf{z}, \mathbf{h}) := \perp$
  else
    $\mathbf{h} := MakeHint(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0)$
    if $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω **then**
      $(\mathbf{z}, \mathbf{h}) := \perp$
    end if
  end if
  $\kappa \leftarrow \kappa + \ell$
end while
return $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

Figure 1: Simplified version of ML-DSA [65].

During key generation Algorithm 1, the coefficients of the secret key components $\mathbf{s}_1 \in R^\ell$ and $\mathbf{s}_2 \in R^k$ are small—chosen from $\mathcal{U}(\eta)$—and the public matrix $\mathbf{A} \in R^{k \times \ell}$ is chosen uniformly at random over R. To obtain the public key, the signer computes the MLWE sample $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ and drops the $d = 13$ Least Significant Bits (LSBs) bits of $\mathbf{t}$ resulting in higher bits $\mathbf{t}_1$ and the lower bits $\mathbf{t}_0$. The signer then only publicly releases $\mathbf{t}_1$ and stores $\mathbf{t}_0$ as part of the secret key.

The signing procedure Algorithm 3 follows the "Fiat-Shamir with Aborts" heuristic [55]: To sign a message, a vector $\mathbf{y} \in R^\ell$ with uniformly random coefficients in $\{-\gamma_1 + 1, \ldots, \gamma_1 - 1\}$ is sampled; the signer computes $\mathbf{w} = \mathbf{A}\mathbf{y}$, decomposes it into a higher ($\mathbf{w}_1$) and a lower part ($\mathbf{w}_0$) such that $\mathbf{w} = 2\gamma_2\mathbf{w}_1 + \mathbf{w}_0$, and a challenge $c \in R$ is computed from the hash of the message and $\mathbf{w}_1$. The challenge polynomial has τ non-zero entries with coefficients in $\{-1, 1\}$. Now, $\mathbf{z}$ and $\mathbf{r}_0$ determine whether the signature is rejected. In the latter case we go back to sampling $\mathbf{y}$; in the former case, the signature is given by $(\tilde{c}, \mathbf{z}, \mathbf{h})$. Here, $\tilde{c}$ allows computing c, and $\mathbf{h}$ is a vector of 1-bit "hints" allowing the other party to compute the higher bits of $\mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0$ without knowing $\mathbf{t}_0$ (analogs to carry bits). The signature is rejected if either

$$\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta \qquad \text{or} \qquad \|r_0\|_\infty \geq \gamma_2 - \beta, \tag{1}$$

where $\beta = \tau\eta$ is the maximum coefficient size of $c\mathbf{s}_1$ and $c\mathbf{s}_2$.

To verify a signature Algorithm 2, the hints are used to compute $\mathbf{w}'_1$ from $\mathbf{A}\mathbf{z} - c\mathbf{t}_1$. This is possible due to the second condition in Equation (1) and

$$\mathbf{A}\mathbf{z} - c\mathbf{t} = \mathbf{A}\mathbf{z} - c\mathbf{A}\mathbf{s}_1 - c\mathbf{s}_2 = \mathbf{w} - c\mathbf{s}_2. \tag{2}$$

Using the hint, the verifier may compute the $2\gamma_2$-high bits of $\mathbf{w} - c\mathbf{s}_2$. The signature is accepted if the acceptance bound on $\mathbf{z}$ holds, $\tilde{c}$ matches the hash of the message and $\mathbf{w}'_1$, and the number of 1-bits in $\mathbf{h}$ is at most ω. Both conditions in Equation (1) are necessary to ensure the zero-knowledge property of the signature on $\mathbf{s}_1$ and $\mathbf{s}_2$. In both cases, β is the maximal absolute value of $c\mathbf{s}$ for $\mathbf{s} \in \{\mathbf{s}_1, \mathbf{s}_2\}$.

If the first condition in Equation (1) is omitted, $\mathbf{z}$ leaks information on $\mathbf{s}_1$ through the relation $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$. On the other hand, if the second condition in Equation (1) is omitted, $\tilde{\mathbf{r}}$ leaks information on $\mathbf{s}_2$ through the relation $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$ because $\tilde{\mathbf{r}}$ can be recomputed from $\tilde{\mathbf{r}} = \mathbf{A}\mathbf{z} - c\mathbf{t} - 2\gamma_2\mathbf{w}_1$ as described in [4]. An honest verifier only has access to $\mathbf{t}_1$ and, thus, the high bits of $\mathbf{w} - c\mathbf{s}_2$; however, every signature contains information on $\mathbf{t}_0$ and Oliveira et al. [66] showed that $\mathbf{t}_0$ can be recovered from 2×10^5–5×10^5 signatures. Given $\mathbf{s}_2$ ($\mathbf{s}_1$), $\mathbf{s}_1$ ($\mathbf{s}_2$) can be computed from the public key equation as soon as $\mathbf{t}$ has been recovered.

Side-channel attacks targeting y and w. The authors of [81] identify zero coefficients of $\mathbf{y}$ from power traces. Thereby, they obtain—potentially noisy—equations on $\mathbf{s}_1$, which they solve using Integer Linear Programming (ILP). This approach requires a large number (several 100k) of traces as only zero coefficients are being targeted; in addition, ILP seems to be suboptimal when it comes to solving noisy equations. In fact, [20] shows that a similar attack relying on Cauchy regression requires far fewer traces; we discuss the details of their approach in the next section. On the other hand, the recent work of [8] shows that ILP may come close to the information theoretic boundary, at least in non-noisy

problems. More generic approaches dealing with leakage on both $\mathbf{y}$ and $\mathbf{s}_1$ have been presented in [13] and [38]. The former work presents a generic framework for leakage on these two variables; the latter presents and analyzes attacks against masked implementations using the framework of [39].

The work of [53] shows that bit leakage on $\mathbf{y}$ results in a *solvable* ILWE instance (see next section). This may be solved using linear regression. However, for more significant bits, their attack quickly becomes unfeasible. In [72], the authors propose a practical template attack based on the method of [53]. To overcome the limitations of [53], the authors of [21] first provide a thorough analysis and then propose a transform that results in an ILWE problem with improved properties. With their method, the key can be recovered from more significant bits as well, and the number of required relations is reduced in all cases. The authors present their method in targeting $\mathbf{y}$, however, it applies to targeting $\mathbf{w}_0$ with almost no modification.

2.2 Integer Learning with Error

ILWE denotes the Learning with Errors (LWE) problem without reductions. This problem was—to the best of our knowledge—first formally defined in [11] in the context of Side-Channel Attacks (SCAs) [27] on BLISS [24]. Using linear regression allows the authors of [11] to solve all instances of the attack of [27]. Subsequently, linear regression on ILWE instances has been used, e.g., in [21, 52], with improvements in [20].

ILWE is often stated without explicitly denoting the error distributions, we follow [11] and make the error distributions explicit; this will be important for our improved approach.

Definition 1. *Let χ_s be a probability distribution over $\mathbb{Z}^n$ with support S, $\mathbf{s} \sim \chi_s$, and χ_a, χ_e probability distributions over $\mathbb{Z}$; an ILWE sample is given by*

$$(\mathbf{a}, z) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \text{ for } a \sim \chi_a^n, e \sim \chi_e. \tag{3}$$

The ILWE problem asks to recover $\mathbf{s}$ from given ILWE samples.

An ILWE sample is called zero-knowledge if it holds no information on $\mathbf{s}$. This is the case for a sufficiently large uniform error distribution χ_e combined with limits on z for the ILWE samples, i.e.

$$z > \eta_{e,-} + \beta_+ \text{ and } z < \eta_{e,+} + \beta_-, \tag{4}$$

where $\eta_{e,-} = \min(\mathrm{supp}(\chi_e))$, $\eta_{e,+} = \max(\mathrm{supp}(\chi_e))$

$$\beta_+ = \max\left(\langle \mathbf{a}, \mathbf{s}' \rangle \mid \mathbf{s}' \in S\right), \text{ and } \beta_- = \min\left(\langle \mathbf{a}, \mathbf{s}' \rangle \mid \mathbf{s}' \in S\right). \tag{5}$$

The CILWE problem was introduced in [20]; we here restate it in a slightly more formal way with explicit distributions:

Definition 2. *Let $q \in (0,1)$, and let χ_a, χ_e be such that the ILWE problem parameterized by (n, χ_a, χ_e) is zero-knowledge. Then, the CILWE problem is defined as outputting an error-free ILWE sample, i.e., a $(n, \chi_a, 0)$ sample, with probability q and a zero-knowledge (n, χ_a, χ_e) sample otherwise.*

As follows from Definition 2, the CILWE problem could also be constructed as a ILWE problem with a non-uniform error distribution χ_e, by adding the probability weight of error-free ILWE samples to the original error distribution at value zero. Note, this construction would still need to include the rejection criteria on z. This allows interpreting the ILWE problem as a generalization of the CILWE problem.

In ML-DSA, $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ and $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$ directly give zero-knowledge ILWE samples. However, if the adversary learns bits of $\mathbf{y}$ or $\mathbf{w}_0$, the error distributions of $\mathbf{y}$ or $\mathbf{w}_0$ can be narrowed down, and the adversary may derive an ILWE equation that is not zero-knowledge. This is the basic principle of, e.g., [20, 21, 81].

Solving ILWE and CILWE. [11] first proposed using OLS for ILWE instances with subgaussian χ_e. The improvements of [21] are orthogonal to solving ILWE—the ILWE problem obtained by [21] has a smaller error. The authors of [20] obtained a different χ_e from their side-channel and coin this distinct setting CILWE. They model the error distribution as heavy-tailed and therefore analyze different regression algorithms, which are able to handle these. The authors propose to use Cauchy regression [62] in practice. The authors of [11] also provide an information-theoretic analysis based on asymptotic statistical distances. They show that, under conditions satisfied in their attack, with standard deviations of χ_a, χ_e, σ_a, σ_e, OLS may recover the secret key with $C \log(n) \frac{\sigma_e^2}{\sigma_a^2}$ samples for some constant C.

2.3 Generic Soft-Analytic Solvers

At Eurocrypt 2025, Hermelink et al. [39] introduced a framework to deal with (noisy) side-channel information in lattice-based schemes. Instead of using algebraic methods, the authors rely on several soft-analytic approaches [83], including BP [30]. Their work formalizes several types of information encountered in previous attacks, and generalizes definitions from primal attack-based frameworks such as [19, 59]. The authors of [39] define distribution hints as follows:

Definition 3. *For a vector $\mathbf{a} \in \mathbb{R}^n$ and secret $\mathbf{s}$, the tuple $(\mathbf{a}, \mathcal{D})$ is called a distribution hint if $\langle \mathbf{a}, \mathbf{s} \rangle \sim \mathcal{D}$.*

The authors present two different solvers for distribution hints: a BP-based solver and a greedy solver; the latter is inspired by [74]. The BP-based solver updates probability distributions for the secret key coefficients based on the distribution hints. Initially starting with the prior distribution, each *factor node* computes an update based on the current

state; the *variable nodes* combine the updates from the factor nodes for the individual coefficients, i.e., the updates based on the distribution hints. In contrast, the greedy solvers updates guesses on the secret key; the scores that determine which updates are performed are based on distribution hints, where each distribution hint provides a "local" score.

Given a distribution hint $(\mathbf{a}, \mathcal{D})$, the BP-based solver updates the probability distribution for the key coefficients as follows: Let $(S_{i,t})_{i\in\{1,\dots,n\}}$ denote the current "state", i.e., the probability distributions for the secret key after t iterations. Let $(S_{i,t+1})_{i\in\{1,\dots,n\}}$ denote the updated local probability, i.e., the probability distribution computed for the distribution hint $(\mathbf{a}, \mathcal{D})$. For $k \in \{1,\dots,n\}$, let $L_{k,t}$ be the random variable representing $\sum_{i=1, i\neq k}^{n} a_i S_{i,t}$ at step t. The updated probability distribution $p(v)$ is computed as

$$P(S_{k,t+1} = v) = \sum_{b\in \operatorname{supp}\mathcal{D}} P_{\mathcal{D}}(b) P(L_{k,t} = b - a_k v). \quad (6)$$

The updates—for all key coefficients and for all distribution hints—are sent to the respective variable node, combined (by computing the normalized product), and give the state $(S_{i,t+1})_{i\in\{1,\dots,n\}}$ for the next step.

Subsequent lattice reduction. The work of [36] shows how to combine BP with lattice reduction in LWE-based schemes if information on the secret key is being obtained. The authors of [39] propose to apply this method to their recovery method.

2.4 Information Theory

To discard samples that hold little information, we rely on information theoretic concepts in later sections. We require the definitions of (Shannon) entropy [77], Kullback-Leibler (KL) divergence [49], and Mutual Information (MI) [77]; for an introduction to information theory, see, e.g., [56, 70].

In the following, X, Y denote random variables with support $\operatorname{supp} X = \mathcal{X}$ and $\operatorname{supp} Y = \mathcal{Y}$. If X is discrete, the entropy is defined as $H(X) = -\sum_{x\in\mathcal{X}} P(X=x)\log_2 P(X=x)$. If X is continuous, the differential entropy is defined as $H(X) = -\int_{x\in\mathcal{X}} P(X=x)\log_2 P(X=x)$. In the discrete case, the MI may be defined as

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) \quad (7)$$

where $H(X \mid Y)$ is the conditional entropy, i.e., the entropy of X conditioned on Y,

$$H(X \mid Y) = -\sum_{x,y\in\mathcal{X}\times\mathcal{Y}} P(X=x, Y=y)\log_2 \frac{P(X=x, Y=y)}{P(Y=y)}. \quad (8)$$

The KL divergence [49] between two distributions P and Q is defined as

$$\mathrm{D}_{\mathrm{KL}}(P \,\|\, Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}, \quad (9)$$

with the conventions $0\log\frac{0}{q} = 0$ and $\mathrm{D}_{\mathrm{KL}}(P\,\|\,Q) = +\infty$ if $P(x) > 0 = Q(x)$ for some x. KL divergence measures how much information is gained about X when a prior $p(x)$ is updated to the posterior $p(x \mid y)$ upon observing $Y = y$; this information gain is $\mathrm{D}_{\mathrm{KL}}\big(p(x \mid y) \,\|\, p(x)\big)$, see, e.g., [70]. The MI is the average information gain, $I(X;Y) = \mathbb{E}_{p(y)}\big[\mathrm{D}_{\mathrm{KL}}(p(x \mid y) \,\|\, p(x))\big]$ [70, Def. 2.12, Thm. 3.2]. Alternatively, MI is commonly *defined* as $\mathrm{D}_{\mathrm{KL}}(p(x,y) \,\|\, p(x)p(y))$ [70, Def. 3.1].

3 A Soft-Analytic Approach to ILWE

Both ILWE and distribution hints involve information on a linear intermediate in the secret key. We show that, in fact, distribution hints generalize ILWE (and thereby also CILWE). More importantly, the solvers of [39] apply to ILWE and CILWE under the same conditions as outlined in [39]: for the BP-based solver, the coefficients of $\mathbf{a}$ need to be "small"—we will be more precise on this condition later on.

3.1 Modeling ILWE using Distribution Hints

First, we show that distribution hints are a *strict* generalization of ILWE. Intuitively, an ILWE sample $(\mathbf{a}, z)$ with error distribution χ_e can be seen as distribution hints by centering χ_e around z; defining this distribution as $\mathcal{D}$ allows us to write $\langle \mathbf{a}, \mathbf{s}\rangle \sim \mathcal{D}$—which is a distribution hint. However, in the other direction, ILWE fixes the error distribution, whereas distribution hints allow for an individual distribution for every sample. A concrete—and practically relevant—counterexample would, for example, be Hamming weight (HW) leakage on z; i.e., we sample $\langle \mathbf{a}, \mathbf{s}\rangle = z$, but only output $\mathrm{HW}(z)$. Now, for $z = 0$, the value is leaked directly, and the error is always 0; but for most values (depending on the distribution of $\mathbf{a}$), the error is non-zero. Therefore, we get the following statement:

Proposition 1. *Solving distribution hints, the* distribution hint problem*, is a strict generalization of the ILWE problem.*

This statement *on its own* is of little value; in fact, the same is true for LWE. However, our evaluations show that—in the case of ILWE—the distribution hint solvers also perform better in most cases with improvements of a factor of up to 51.5 fewer relations needed.

3.2 Solving ILWE

Concretely, given a ILWE instance, $(\mathbf{A}, \mathbf{z}) \in \mathbb{Z}^{m\times n} \times \mathbb{Z}^m$, i.e., m samples, with error distribution χ_e, we construct the following distribution hints as follows: For each row of $\mathbf{A}$, $\mathbf{a}_j$ and each entry of $\mathbf{z}$, z, for $j \in \{1,\dots,m\}$, we compute $\mathcal{D}_j$

$$P_{\mathcal{D}_j}(x = v) = P_{\chi_e}(z - v). \quad (10)$$

We then get $\langle \mathbf{a}_j, \mathbf{s}\rangle \sim \mathcal{D}_j$ for all $j \in \{1,\dots,m\}$ as the ILWE samples represent the equation

$$\langle \mathbf{a}_j, \mathbf{s}\rangle + e_j = z_j \quad (11)$$

and therefore

$$e_j = z_j - \langle \mathbf{a}_j, \mathbf{s} \rangle . \tag{12}$$

Thus, the probability for $\langle \mathbf{a}_j, \mathbf{s} \rangle$ being v for known z_j, is given by the shifted and inverted distribution of e_j, which is just χ_e.

The solvers are now instantiated as described in [39]. For the BP-based solver, this works as follows: Every key coefficient is modelled by a *variable node*. The variable nodes hold the current "belief" in the respective key coefficient, i.e., a probability distribution representing the current state. The initial belief is set to the prior, i.e., the distribution that the coefficients of $\mathbf{s}$ are sampled from or a uniform distribution. For every distribution hint, i.e., every $(\mathbf{a}_j, \mathcal{D}_j)$ constructed above, we instantiate a *factor node*. The factor nodes compute updates for each key coefficient based on the distribution hint; the update for key coefficient i involves the beliefs in all variables $i' \neq i$ (see Section 2.3, (6)).

3.3 Solving CILWE

In the CILWE problem, the error is not independent of $\mathbf{z}$ as already noted in [20]: for a large coefficient of $\mathbf{z}$, the respective sample must have been a zero-knowledge sample; while for a smaller value, we obtain a spike in the probability mass function for that respective coefficient of $\mathbf{z}$. As distribution hints do not assume the error distribution to be independent of $\mathbf{z}$, a similar construction as above works for CILWE as well. Damm et al. [20] condition their samples to mimic rejection sampling in ML-DSA: in the concealed branch, they repeatedly draw $e \sim \mathcal{U}(2\beta)$ until $z_j \in \{-\tau, \dots, \tau\}$, while error-free samples outside this interval are rejected. Since $\langle \mathbf{a}_j, \mathbf{s} \rangle \in \{-\beta, \dots, \beta\}$, an accepted concealed observation is uniform on $\{-\tau, \dots, \tau\}$ with probability mass $U = (2\tau+1)^{-1}$. Applying the construction of distribution hints above to these CILWE samples for $q \in (0,1]$ gives

$$P_{\mathcal{D}_j}(v) = \frac{1}{Z_{\mathcal{D}}} \begin{cases} (1-q) + qU & v = z_j, \\ qU & v \in \{-\tau, \dots, \tau\} \setminus \{z_j\}, \\ U & v \in \{-\beta, \dots, \beta\} \setminus \{-\tau, \dots, \tau\} \end{cases} \tag{13}$$

where $Z_{\mathcal{D}} = 1 + 2(\beta - \tau)U$. The last case accounts for candidate inner products outside $\{-\tau, \dots, \tau\}$, which can yield an accepted sample only through the concealed branch.

3.4 Filtering Distribution Hints

Distribution hints allow for several practical ways to estimate the information obtained in an attack. In some attacks, estimating the information that can be learned from individual hints is crucial as the BP may fail with too many low-information hints [38]. The authors of [38] propose to compute the entropy of $\mathcal{D}$ and discard hints that are close to uniform (for a hint $\mathcal{H} = (\mathbf{a}, \mathcal{D})$). Thereby, they avoid failing BP instantiations.

However, as the entropy of $\mathcal{D}$ holds no information on the prior, it is clear that this method can at best be an approximation to how much information a hint provides. The reasoning in [38] is that the average of $H(\mathbf{s}) - H(\mathbf{s} \mid \mathcal{H})$ is the mutual information of the random variable targeted in the attack. This is correct; however, we can improve upon that: The KL divergence [49] between the posterior and prior ($\mathrm{D}_{\mathrm{KL}}(p_{\mathrm{post}} \mid\mid p_{\mathrm{prior}})$) represents the information we may obtain from an individual hint. The average *also* yields the MI [70]. It should be noted that previous works mostly focus on estimating the MI and filtering out hints with very low information content to avoid numerical problems; this can already be achieved by their method.

The authors of [38] also note that the KL-divergence between prior and hint could be used, but did not see an advantage over using the entropy; they left improvements in filtering for future work. It is not surprising that the KL-divergence between hint and prior did not result in improvements: the KL-divergence has to be applied to posterior and prior.

We first look into the general problem of estimating the information gain given by a distribution hint. In the next section, we show that we can do better in concrete attacks against ML-DSA. Both [38] and [37] already consider filtering improvements; however, these are so-far heuristic.

Given a hint, $\mathcal{H} = (\mathbf{a}, \mathcal{D})$, one might be tempted to compute the posterior distribution by applying Bayes' formula

$$P(\mathbf{s} = \mathbf{x} \mid \mathcal{H}) \propto P(\mathcal{H} \mid \mathbf{s} = \mathbf{x}) P(\mathbf{s} = \mathbf{x}). \tag{14}$$

However, it is unclear what $P(\mathcal{H} \mid \mathbf{s} = \mathbf{x})$ is supposed to denote, i.e., what the measure-theoretic interpretation would be. In fact, by the way that the authors of [39] construct $\mathcal{D}$, it is already a (normalized) likelihood. Thus, the correct application of Bayes' formula is to simply multiply $P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle)$ by the prior for $\mathbf{x}$. Consequently, we define the above abuse of notation to mean

$$P(\mathbf{s} = \mathbf{x} \mid \mathcal{H}) \propto P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) P(\mathbf{s} = \mathbf{x}). \tag{15}$$

Hence, the posterior is computed by multiplying the likelihood, i.e., $\mathcal{D}$, with the prior and then normalizing. The BP proposed in [39] does this correctly; the comment in [39] on not including the prior in $\mathcal{D}$ stems from the same reason.

The previous filtering method, proposed in [38] and discussed in [37], relied solely on filtering by entropy of the likelihood $\mathcal{D}$. However, as this has no relation to the prior or $\mathbf{a}$, it should be clear that this can only be an approximation to a correct comparison between samples. The authors of [38] also discuss using the KL-divergence [49] between likelihood and prior; however, this gave no improvements. In fact, the information gain is given by the KL-divergence, which may be used to quantify the information in Bayesian updating [51].

Lemma 1. *The information gain* IG *of a hint* $\mathcal{H} = (\mathbf{a}, \mathcal{D})$ *is*

given by

$$\mathrm{IG}(\mathcal{H}) = \sum_{\xi} \frac{P(\langle \mathbf{a}, \mathbf{s} \rangle = \xi)}{Z} P_{\mathcal{D}}(\xi) \log_2 P_{\mathcal{D}}(\xi) - \log_2 Z \quad (16)$$

where

$$Z = \sum_{\gamma} P_D(\gamma) P(\langle \mathbf{a}, \mathbf{s} \rangle = \gamma),$$

and the sum goes over the support of $P(\langle \mathbf{a}, \mathbf{s} \rangle = \xi)$, i.e. all possible scalar outputs of the inner product of the vector $\mathbf{a}$ *with the random variable* $\mathbf{s}$*.*

Proof. The proof follows from the observation above that $\mathcal{D}$ is a normalized likelihood. The rest is standard information-theoretic computations. Let M be the random variable $\mathcal{H}$ arose from. For $m \in \operatorname{supp} M$, have that

$$P(\mathbf{s} = \mathbf{x} \mid M = m) = \frac{P(M = m \mid \mathbf{s} = \mathbf{x}) P(\mathbf{s} = \mathbf{x})}{P(M = m)} \quad (17)$$

$$= \frac{P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) P(\mathbf{s} = \mathbf{x})}{\sum_{\mathbf{y}} P_D(\langle \mathbf{a}, \mathbf{y} \rangle) P(\mathbf{s} = \mathbf{y})} \quad (18)$$

where the denominator for normalization equals the above defined Z. Let p_{post} and p_{prior} denote posterior and prior, i.e., $p_{\text{post}} = \frac{P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) p_{\text{prior}}}{Z}$ and $p_{\text{prior}} = P(\mathbf{s} = \mathbf{x})$. Then the KL divergence between posterior and prior (i.e. the information gain of a hint) is

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}(p_{\text{post}} \mid\mid p_{\text{prior}}) = \sum_{\mathbf{x}} p_{\text{post}}(\mathbf{x}) \log_2 \frac{p_{\text{post}}(\mathbf{x})}{p_{\text{prior}}(\mathbf{x})} \\
&= \sum_{\mathbf{x}} \frac{P(\mathbf{s} = \mathbf{x})}{Z} P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) \log_2 P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) \\
&- \frac{\log_2 Z}{Z} \sum_{\mathbf{x}} P_{\mathcal{D}}(\langle \mathbf{a}, \mathbf{x} \rangle) P(\mathbf{s} = \mathbf{x}) \\
&= \sum_{\xi} \frac{P(\langle \mathbf{a}, \mathbf{s} \rangle = \xi)}{Z} P_{\mathcal{D}}(\xi) \log_2 P_{\mathcal{D}}(\xi) - \log_2 Z \quad (19)
\end{aligned}$$

□

Note, that $P(\langle \mathbf{a}, \mathbf{s} \rangle = \xi)$ will tend towards a discrete Gaussian distribution, assuming coefficients of $\mathbf{s}$ are independent and identically distributed, with higher dimensions of $\mathbf{a}$ by the central limit theorem [18].

The first term in Equation (16) is the weighted average over $\log_2 P_{\mathcal{D}}(\xi)$, where the weights are given by the posterior distribution; i.e., it is the posterior expected log-likelihood capturing how well the posterior fits the observed data. Its minimum is $\log_2 Z$ by Jensen's inequality [18], and its maximum is zero (as we normalized the likelihood); the second term—the log-evidence— expresses how likely the observation was under the prior penalizing more surprising observations.

In the case of solving ILWE, this confirms the intuition: more extreme samples, i.e., with larger absolute value of $\mathbf{a}$ give more information; in addition, of course, an error distribution with lower entropy is beneficial to the attacker. More importantly, it provides an improved approach to filtering in attacks that rely on [39].

Note that in the case of solving ILWE and CILWE, we did not encounter settings where filtering was necessary due to oscillations or numerical issues. Nevertheless, filtering provides a way to drastically reduce the runtime of the solving process; this may enable, e.g., online attacks.

4 Understanding and Comparing Solvers

Several previous works rely on OLS while the power SCAs of [13,38] use BP. In power SCA, the general approach of [13, 38] has the clear advantage that they do not discard non-zero coefficients of y and, therefore, efficiently utilize information extracted from traces in various leakage models. In contrast, deriving ILWE samples as proposed in [21] is a valid attack strategy in case of bit leakage. This leads to the question of how the OLS- and BP-based solvers compare conceptually and empirically; we give an answer to the former question in this section and to the latter in the following section.

4.1 Implicit Assumptions

In this section we provide a comparison of BP and OLS in terms of their theoretical derivation taken from [10]. Both algorithms can be derived from statistics, however OLS is a Maximum Likelihood Estimator (MLE), which tries to find a fixed but unknown parameter $\mathbf{s}$ that explains the observed samples $(\mathbf{a}_i, z_i)$ best under some assumptions of normality, while BP is a Bayesian method that treats the secret as a random variable with a prior distribution.

In belief propagation, we try to compute a posterior probability $p(\mathbf{s} | \mathcal{H}_1, \ldots, \mathcal{H}_m)$ by iteratively multiplying the probability of observing a certain sample given the current belief about the distribution of the secret with the current belief (the current prior) of the secret according to Equation (14). Finally, we obtain the estimate of the secret $\mathbf{s}$ by choosing the most likely value from the posterior, which is the Maximum A Posteriori (MAP) estimate. It is computed element-wise for each coefficient s_j assuming independence between them. Note however, that general BP only approximates this as it is only guaranteed to compute the exact posterior distribution in a factor graph without loops, whereas the factor graph induced by (C)ILWE problems is fully connected.

4.1.1 Assumptions in OLS

OLS starts with the assumption that the samples are generated according to Equation (11) with error of each independently from a centred normal distribution with mean zero and some variance σ_e^2, i.e. $\chi_e = \mathcal{N}(0, \sigma_e^2)$. Hence, every observation

$z_i|\mathbf{a}_i, \sigma_e^2$ is distributed as $\mathcal{N}(\mathbf{a}_i^T\mathbf{s}, \sigma_e^2)$ and the likelihood for all the collected samples is

$$p(\mathbf{z}|\mathbf{A}, \mathbf{s}, \sigma_e^2) = \prod_{i=1}^{m} p(z_i|\mathbf{a}_i^T\mathbf{s}, \sigma_e^2) = \prod_{i=1}^{m} \mathcal{N}(\mathbf{a}_i^T\mathbf{s}, \sigma_e^2). \quad (20)$$

Least squares finds a maximum of this likelihood for the samples observed. Its formula can be derived from the logarithm of the likelihood as:

$$\begin{aligned} \ln p(\mathbf{z}|\mathbf{A}, \mathbf{s}, \sigma_e^2) &= \sum_{i=1}^{m} \ln \mathcal{N}(\mathbf{a}_i^T\mathbf{s}, \sigma_e^2) \\ &= \frac{m}{2} \ln \frac{1}{2\pi\sigma_e^2} - \frac{1}{2\sigma_e^2} \sum_{i=1}^{m} (z_i - \mathbf{a}_i^T\mathbf{s})^2. \end{aligned} \quad (21)$$

Ignoring constants and taking the gradient with respect to $\mathbf{s}$, we obtain $\nabla_{\mathbf{s}} \ln p(\mathbf{z}|\mathbf{A}, \mathbf{s}, \sigma_e^2) = \sum_{i=1}^{m} (z_i - \mathbf{a}_i^T\mathbf{s})\mathbf{a}_i^T$. Setting this to zero and rearranging for $\mathbf{s}$ gives

$$\hat{\mathbf{s}}_{ML} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{z}, \quad (22)$$

which is both the maximum likelihood estimate and the value that minimizes the sum of squares in the former equation. It is worth to note that computing the sum of squares (and rescaling with the amount of samples) is computing an estimator of expectation of the likelihood.

A product of multiple Gaussian distributions is Gaussian as well, so the final OLS estimate is distributed as $\mathcal{N}(\mathbf{s}, \mathbf{S})$. The covariance matrix is $\mathbf{S} = \sigma_e^2(\mathbf{A}^T\mathbf{A})^{-1}$, from which a sample bound can be derived. [11] essentially showed that this also holds if χ_e is only subgaussian, where the variance proxy is an approximation of the true variance of the errors, while [21] used a CLT instead. [20] considered deviations from normality. For a Huber regression [40], normality is only assumed for a subset of the samples with rather small error variance, whereas Cauchy regression [62] considers χ_e to be a Cauchy distribution with scale parameter $\nu = 1$. Note that none of these assumptions actually match the error distribution of (C)ILWE samples obtained from leakage in practice.

In order to bridge the derivation of BP and OLS, we sketch how to explicitly derive the formula of OLS with Bayesian statistics. We again assume that samples $(\mathbf{a}_i, z_i)$ are generated by Equation (11) and the error distribution is $\chi_e = \mathcal{N}(0, \sigma_e^2)$ with known variance. The error is again independent of both $\mathbf{a}$ and $\mathbf{s}$. However, in Bayesian statistics, we no longer treat $\mathbf{s}$ as a universal constant, but as a random variable from some prior distribution. In order to achieve a Gaussian posterior, we use a Gaussian prior with $p(\mathbf{s}) = \mathcal{N}(\mu_0, \mathbf{S}_0)$. Using Bayesian updates as

$$p(\mathbf{s}|\mathbf{a}, \mathbf{z}, \sigma_e^2) \propto p(\mathbf{z}|\mathbf{s}, \mathbf{a}, e)p(e)p(\mathbf{s}), \quad (23)$$

we obtain that

$$p(\mathbf{s}|\mathbf{z}, \mathbf{A}) = \mathcal{N}(\mu_m, \mathbf{S}_m) \quad (24)$$

with $\mu_m = \mathbf{S}_m(\mathbf{S}_0^{-1}\mu_0 + \frac{1}{\sigma_e^2}\mathbf{A}^T\mathbf{z})$ and $\mathbf{S}_m^{-1} = \mathbf{S}_0^{-1} + \frac{1}{\sigma_e^2}\mathbf{A}^T\mathbf{A}$.

If we do not want to bias our estimate, we need to use a prior that is Gaussian but does not provide any information. We choose as an *uninformative* prior an infinitely broad Gaussian $p(\mathbf{s}|\sigma_{e0}^2) = \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I}_n)$ with $\sigma_0^2 \to \infty$. Taking the limit, we obtain a variance $\mathbf{S}_m = \sigma_e^2(\mathbf{A}^T\mathbf{A})^{-1}$ and a mean $\mu_m = \frac{1}{\sigma_e^2}\mathbf{S}_m\mathbf{A}^T\mathbf{z}$. The MAP estimate $\mathbf{s}_{MAP}$ is the most likely value of the posterior distribution, which for Gaussians is its mean μ_m, which coincides with the MLE of Equation (22). $\mathbf{S}_m$ also matches the covariance of OLS. The same behavior can be achieved for $m \to \infty$, which also rules out the influence of the prior.

Using this deviation of OLS, we gather two fundamental differences to BP. Firstly, OLS does not assume any information on the secret $\mathbf{s}$. However, in the case of (C)ILWE instances, the distribution of secret is known beforehand and can be used by Bayesian methods. This can be the range of the coefficients of $\mathbf{s}$ as in Definition 1, or possibly even a distribution on them. The prior information is especially influential if only a few samples are present. This makes it more suitable for cryptanalysis, where the goal is to derive an estimate of the secret based on as few samples as possible. Secondly, OLS is only optimal if the error distribution is Gaussian. In this case, it is a valid method and BP should find a similar solution with a similar amount of samples, as it approximates Bayesian regression in this case. As long as the error distribution is subgaussian, OLS still converges to correct solution, but requires more samples [11]. Huber and Cauchy regression are also MLEs for their respective distributions, for which similar results hold likewise [41].

4.1.2 Assumptions in BP

Our theoretical observations all consider BP as an approximation of Bayesian Regression. BP is guaranteed to provide the correct posterior (and therefore the correct estimate for $\mathbf{s}$) for acyclic graphical models [68]. For graphs with cycles, BP has been shown to often provide good approximations [63].

The reason for not using Bayesian Regression is that it is hard to compute if the priors are not *conjugate*. A conjugate prior is when the posterior is of the same distribution as the prior after each update and has a closed formula, such that its MAP can be determined analytically. For priors such as a uniform distribution in ML-DSA in combination with the distribution of each sample (which does not have a closed form distribution), this does not exist. Hence, there are no efficient means of computing the exact posterior distribution $p(\mathbf{s}|\mathbf{A}, \mathbf{z}, \sigma_e^2)$. BP is a reasonable heuristic for this inference in two ways. Firstly, it only heuristically computes a posterior. Secondly, this posterior is computed per coefficient s_j instead of an exact posterior for all coefficients in $\mathbf{s}$ jointly. The latter can be justified as the variables s_j are sampled i.i.d. and therefore do not depend on each other directly. Further, their interactions become smaller the more samples are used,

which follows from the fact that the covariance matrix of OLS converges to a diagonal matrix [22, Lemma 11].

If all distributions (priors and posteriors) are Gaussian, BP behaves like a Bayesian Regression with Gaussian distributions as outlined above. This was proven by [58]. Using a Gaussian distribution only for the messages and not for the prior gives the solver of [23].

4.2 Acyclic Graphs and Gaussian Priors

Recall that BP computes the marginals of the joint probability distribution for acyclic graphical models [68]. For graphs that contain cycles, BP has been shown to often provide good approximations [63]; this is particularly true in Soft Analytical Side-Channel Attack (SASCA) [83] and may lead to new attacks, see, e.g., [6, 13, 31–34, 57, 64, 69, 71][4]. In the case of our instantiation, for hints $\mathcal{H}_i$, we compute or approximate the marginals of $P(\mathbf{s} \mid \mathcal{H}_i)$, with the abuse of notations discussed in Section 3.4.

For a CILWE or ILWE problem, $\mathbf{As}+\mathbf{e}=\mathbf{z}$, the resulting graph can only be acyclic if no two rows share more than one entry. Thus, we can in general not expect the graph to be acyclic. However, for an acyclic graph, we would have the following result.

Proposition 2. *Given a (continuous) ILWE problem $(\mathbf{A}, \mathbf{z}, \mathcal{N}(0, \sigma_e^2))$ for $\sigma_e^2 \in \mathbb{R}_{>0}$, such that the matrix $\mathbf{A}$ results in an acyclic graph under the construction of Section 3.1, and A has full column rank, OLS and the BP-based solver of Section 3.2 arrive at the same result.*

Proof. For $\mathbf{As}+\mathbf{e}=\mathbf{z}$ with Gaussian error $\mathbf{e}$ (coefficient-wise) and no prior on $\mathbf{s}$, OLS is the maximum likelihood estimator [10]. At the same time, our instantiation of the solver of [39] computes the marginals of $P(\mathbf{s} \mid \mathbf{As}+\mathbf{e}=\mathbf{z})$. As the graph is acyclic by assumption, BP is exact [68], and computes the marginals of the joint probability distribution of all the secret key conditioned on the ILWE samples. Marginals of a Gaussian are Gaussian, and the mode of a univariate Gaussian is its mean; thus, the vector of coefficient-wise most likely values with respect to the marginal distributions computed by the BP, i.e., the output of the BP-based recovery algorithm, matches the maximum likelihood key, i.e., the output of the OLS-based key recovery algorithm. □

As mentioned before, we cannot expect our graph to be acyclic; in fact, the average row weight needs to be less than 2, which is extremely unlikely.However, there is another set of assumptions under which BP may converge. If we consider BP on a graphical model in which every node has a continuous Gaussian distribution, the posterior distribution becomes Gaussian as well. [58] show that in this case, loopy BP converges to the correct posterior distribution regardless of the summation order under some conditions. We use this to show the following:

Proposition 3. *Let χ_a be a univariate distribution with mean zero and variance $\sigma_a^2 \in \mathbb{R}_{>0}$. Let $\mathbf{A}$ consist of $m \times n$ i.i.d. samples according to χ_a. Given a (continuous) ILWE problem $(\mathbf{A}, \mathbf{z}, \mathcal{N}(0, \sigma_e^2))$ for $\sigma_e^2 \in \mathbb{R}_{>0}$, OLS and the BP-based solver of Section 3.2 converge to the same result asymptotically in m.*

Proof. We assume that $\mathbf{A}$ is full column rank with high probability and therefore $\mathbf{A}^T\mathbf{A}$ is invertible. As before, for $\mathbf{As}+\mathbf{e}=\mathbf{z}$ with Gaussian error $\mathbf{e}$ (coefficient-wise) and flat prior on $\mathbf{s}$, OLS is the maximum likelihood estimator [10].

Further, if we initialise BP with the same flat Gaussian prior, all involved distributions are Gaussian. Therefore, the posterior distribution of the underlying graph is also Gaussian. It can be derived as $p(\mathbf{s}|\mathbf{A}, \mathbf{z}, \sigma_e^2) \propto \exp\left(-\frac{1}{2}\mathbf{s}^T\mathbf{C}\mathbf{s}+\mathbf{h}^T\mathbf{s}\right)$ with $\mathbf{C}=\frac{1}{\sigma_e^2}\mathbf{A}^T\mathbf{A}$ and $\mathbf{h}=\frac{1}{\sigma_e^2}\mathbf{A}^T\mathbf{z}$ [58], which agrees with Equation (23). It remains to show that BP converges to this solution: [58] shows convergence of BP under different properties of $\mathbf{C}$. We use [58][Cor. 22], which states that BP converges to the correct posterior mean if $\mathbf{C}$ is diagonally dominant, i.e. $\forall i : |\mathbf{c}_{i,i}| \geq \sum_{j \neq i} |\mathbf{c}_{i,j}|$.

We compute the expectation of the on- and off-diagonal entries of a normalised matrix $\mathbf{G}=\frac{\sigma_e^2}{m}\mathbf{C}=\frac{1}{m}\mathbf{A}^T\mathbf{A}$, as diagonally dominance is stable under positive scalar multiplication. Let $\mathbf{a}_{:i}$ denote the i-th column of $\mathbf{A}$. Then, $\mathbf{g}_{i,j}=\frac{\mathbf{a}_{:i}^T\mathbf{a}_{:j}}{m}$. Furthermore, $\mathbb{E}[g_{i,i}]=\mathbb{E}[\frac{1}{m}\mathbf{a}_{:i}^T\mathbf{a}_{:i}]=\frac{1}{m}\mathbb{E}[\sum_{k=1}^{m} a_{ki}^2]=\sigma_a^2$ and $\mathbb{E}[g_{i,j}]=\mathbb{E}[\frac{1}{m}\sum_{k=1}^{m} a_{ki}a_{kj}]=\mathbb{E}[a_{ki}]\mathbb{E}[a_{kj}]=0$ for $i \neq j$.

By the weak law of large numbers, all entries converge to their respective mean point-wise in probability. As n is fixed, also the overall maximum deviations for both on- and off-diagonal elements converge to zero in probability. Hence, for $\delta > 0$, there exists m_0 such that for $m > m_0$ all on-diagonal values are close to σ_a^2 and the sum of all off-diagonal $|g_{i,j}|$ is small enough, such that $P(\forall i : |g_{i,i}| > \sum_{j \neq i} |g_{i,j}|) > 1-\delta$. As $\mathbf{C}$ is only a positive scalar multiple of $\mathbf{G}$, $\mathbf{C}$ is diagonally dominant as well. BP converges to the same mean as Bayesian Regression. It was shown in Section 4.1.1 that under these priors, Bayesian Regression and OLS are equivalent because the mode of Gaussian distribution is its mean. □

Note that we introduced further assumptions on χ_a, which can be slightly relaxed to independent rows with uncorrelated entries. These assumptions are in line with [11] and also fulfilled by ILWE instances from bit leakage in ML-DSA [53]. In fact, if we assume the other extreme case, i.e. a dimension of χ_a is linearly dependent on others, then $\mathbf{A}$ does not have a full rank and the OLS estimate cannot be computed. If it has strong correlation, convergence of OLS is difficult [10].

We have shown that BP will converge to the result of OLS *if the error is Gaussian and the prior on* $\mathbf{s}$ *is flat*. However, in

[4] An overview on the conditions under which BP provides good approximations in SASCA can be found in [64]. It should be noted that [79] find that Algebraic Side-Channel Attack (ASCA) may be advantageous in some settings.

practice, our error distribution is not continuous and the prior on $\mathbf{s}$ will be restricted to a small, discrete set. Therefore, while the usage of OLS is rooted in well-established statistical theory and provides optimal results under the above-mentioned conditions, these conditions differ greatly from the information derived on $\mathbf{s}_1$ and $\mathbf{s}_2$ from SCA as discussed in prior works [20]. The conditions for optimality for the BP are also not fulfilled in most cases; however, BP has proven to give reasonable approximations outside of those by the use of prior knowledge [63] (in the context of SCA, see, e.g., [64]) and will converge to the OLS solution under those assumptions.

4.3 Attacker and Leakage Models

Previous attacks rely on either bit or HW leakage models. The BP-based solvers [13, 38, 39] may be used in either case and may deal with combined leakage, i.e., leakage on bits and HW. In power SCA, the latter will often be the case, but in case of pure bit leakage, deriving ILWE samples—while keeping track of the error distributions—does not lose information.

Side-channel attacks that derive CILWE problems, e.g., [81], usually suffer from severe information loss. The attacker model from which CILWE was devised assumes that the adversary is able to detect that a coefficient of $\mathbf{y}$ is zero—this gives a single sample. However, $\mathbf{y}, \mathbf{w}$ are sampled uniformly over a range of 2^{γ_1+1} or 2^{γ_2+1}, which gives the adversary a *chance* of less than 2^{-18} to derive a relation (equality **if** the adversary can detect the zero coefficient with accuracy 1), while an attacker typically obtains information on *all* coefficients in power SCA. Thus, applying [13, 38, 39] allows deriving much more information *and* outperforms regression-based CILWE solvers. On the other hand, fault injection attacks that derive CILWE problems, e.g., [80], by faulting $\mathbf{y}$ to zero do not suffer from the same problem and can tolerate high concealment rates arising from probabilistic faults.

Given that the attacker and leakage models of the BP-based works allow deriving the same or more information, and that BP outperforms regression-based methods, we believe it is sensible to recommend relying on BP, i.e., [13, 38, 39] for leakage on $\mathbf{y}$ and $\mathbf{w}$.

5 Evaluation

We now complement the conceptual analysis with an empirical evaluation of regression-based and BP-based solvers on generic ILWE and CILWE instances.

Experimental Setup. We conducted our experiments on 3 different servers featuring the following configurations: Intel Xeon 6342 with 512GB RAM, ARM Neoverse N1 with 512GB RAM, and AMD EPYC 9754 with 1TB RAM. Unless otherwise stated, we conduct 10 independent trials for each configuration presented and report the mean number of samples required for key recovery. In addition, we report results

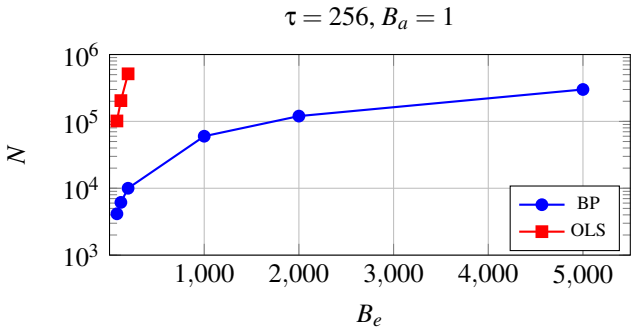


Figure 2: Sample complexity of 100% ILWE secret recovery. N is the number of ILWE required samples, and B_e is the error bound such that $\chi_e = \mathcal{U}(B_e)$. We use dense sampling vectors $\mathbf{a}$ ($\tau = 256$) with $a_i \sim \mathcal{U}(B_a)$ and sparse secrets.

only when all 10 out of 10 trials were successful, i.e., 100% success rate was achieved. To select the number of samples N, we used two methods: a binary search in which we multiplicatively increased the number of points until recovery succeeded and then bisected the interval to find the minimum required number of samples, and linearly or logarithmically spaced points, for which results from lower-noise instances were used to set the lower bound of N for higher-noise instances. Per solver, we stop increasing N once all 10 trials recover all 256 coefficients.

Number of samples. Note that for a fixed number of relations, the variance in the distance of the guess and the secret key is generally low, we evaluate many settings that are within close range of each other, and lattice reduction may recover the secret key even if the distance is non-zero [36]. Thus, we believe the 10 trials per setting are sufficient. While we do not characterize every individual setting with high certainty about the exact number of samples required, we believe the overall picture is sufficiently clear.

ILWE. For our evaluation, we instantiate ILWE instances with $n = 256$ due to its usage in ML-DSA and other deployed LWE schemes. Now, we evaluate the difficulty of solving ILWE using the sparse secret distribution $\mathcal{S}$ defined in [11] and used in [85]. Concretely, the sparse distribution $\mathcal{S}$ has $\mathbf{s} \in \{-2,-1,0,1,2\}^{256}$ with $\tau = 78$ non-zero coefficients: 39 randomly selected coefficients have magnitude 1, another 39 have magnitude 2, each with an independently uniform sign, while the remaining 178 coefficients are zero. For the error and sampling vector distributions, errors were drawn from a centred discrete uniform distribution, while we used dense sampling vectors $\mathbf{a}$ ($\tau = 256$) with $a_i \sim \mathcal{U}(1)$ in Figure 2 and $a_i \sim \mathcal{U}(5)$ in Figure 6.

Next, we investigate the effect of different parameters on the difficulty of solving ILWE with BP or OLS, namely: error distribution, secret distribution and the sparsity (τ) of the

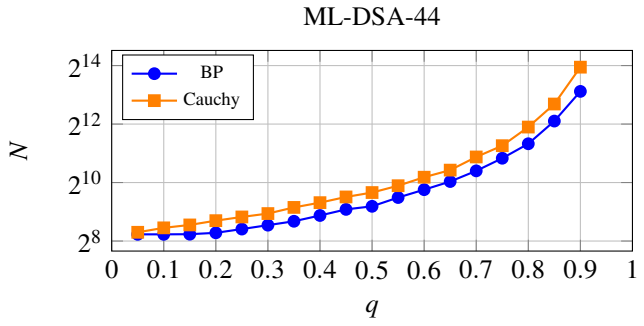


Figure 3: Sample complexity of 100% CILWE secret recovery for ML-DSA-44. N is the number of CILWE required samples, and q is the concealment rate as defined in [20].

sampling vector $\mathbf{a}$. First, we investigate the effect of error distribution by comparing both solvers with uniform and discrete Gaussian errors in Figure 7. We instantiate two ILWE instances with nearly equivalent variances, one having its errors drawn from a uniform distribution ($\chi_e = \mathcal{U}(17)$) and the other from a discrete Gaussian distribution ($\chi_e = D_{\mathbb{Z},10}$). For discrete Gaussian errors, which more closely match the Gaussian error model underlying OLS, BP has a modest advantage over OLS. For uniform errors, BP exploits the known error distribution and substantially outperforms OLS.

Second, we investigate the effect of secret distribution by running both solvers against secrets drawn from the sparse secret distribution ($\chi_s = \mathcal{S}$) and from the uniform distribution ($\chi_s = \mathcal{U}(2)^n$) in Figure 8. BP benefits from secret sparsity because the sparse prior provides more initial information about each secret coefficient. Specifically, the marginal sparse prior has entropy $H(s_i) \approx 1.50$ bits, whereas a uniform coefficient $s_i \sim \mathcal{U}(2)$ has entropy $H(s_i) \approx 2.32$ bits. The lower per-coefficient uncertainty produces more concentrated BP messages even though the factor-graph connectivity remained unchanged. Indeed, this is consistent with the dependence of the information gain on the secret prior in Equation (16). In contrast, OLS does not incorporate this prior and, therefore, has nearly identical recovery curves for both secret types.

Finally, the effect of the sparsity (τ) of the sampling vector $\mathbf{a}$ is investigated in Figure 9 by instantiating ILWE instances with $\tau \in \{39, 60, 256\}$. As predicted by Equation (16), increasing (τ) broadens the distribution of ($\langle \mathbf{a}, \mathbf{s} \rangle$) and thereby increases the expected information gain of each sample for a fixed error distribution. Moreover, because the non-zero coefficients are drawn from a fixed distribution, increasing τ increases the coefficient variance σ_a^2. Since the sample complexity of OLS scales inversely with σ_a^2, denser sampling vectors require fewer samples [11]; thus, $\tau = 256$ represents the most informative setting considered. Nonetheless, BP requires fewer samples for every evaluated value of τ than OLS requires in the most informative OLS setting. On the other hand, BP does exhibit slight variation across τ. Although increasing τ generally increases the information gain per sample, $\tau = 60$ produces a lower-degree factor graph that propagates more concentrated messages than those produced with $\tau = 256$ and, therefore, slightly outperforms $\tau = 256$ with the selected parameters.

CILWE. In Figure 10, the evaluation of the BP-based solver for CILWE is reported for ML-DSA-44, while Figure 10a and Figure 10b report the evaluations for ML-DSA-65 and ML-DSA-87, respectively. BP requires fewer samples throughout the evaluated range, with the advantage increasing as q increases (1.78x and 1.92x at $q = 0.90$ for ML-DSA-44 and ML-DSA-87, respectively). For ML-DSA-65, Cauchy regression failed to recover within our sample limit of 1×10^5 at $q \in \{0.85, 0.90\}$, while BP required an average of 12,636 samples at $q = 0.90$[5]. Furthermore, we evaluate our BP solver on CILWE instances having very high concealment rates $q \in [0.91, 0.99]$ in Figure 11. With a sample limit of 1×10^6, CILWE secret recovery was achieved for concealment rates up to $q = 0.99$ for all ML-DSA parameter sets except ML-DSA-65. It can be seen that ML-DSA-65's secrets with $H(s_i) \approx 3.17$ bits—compared with $H(s_i) \approx 2.32$ bits for other levels—result in a significantly higher aggregate entropy, while simultaneously decreasing the information gain of individual CILWE samples. In addition, Figure 11 shows ML-DSA-87 recovery with 14.9% fewer samples than ML-DSA-44 for $q = 0.99$ due to its higher τ. Indeed, both observations can be predicted by the previously investigated generic ILWE instances.

6 Timing Attack

We now complement the evaluation in the previous section by applying our BP solver to a recent timing side-channel leakage [43] in PQClean's implementation of ML-DSA [46]. Notably, the attack is applicable to the hedged and deterministic variants of ML-DSA. Our empirical evaluation shows that our solver is able to decrease the estimated number of signatures required by a factor of 62 on average compared to OLS in noise-free conditions. Furthermore, we evaluate the attack under simulated bit errors up to $p_{\text{err}} = 0.45$ to demonstrate key recovery under very high bit error rates. Although the issue has been addressed by the mldsa-native team [43], how the resulting leakage can be exploited for key recovery has not been analyzed.

6.1 Attack Description

The vulnerability originates from the secret-dependent branches that leak the Most Significant Bit (MSB) of the $\mathbf{w}_0$

[5]The baseline for Cauchy regression differs from that of [20]. This is due to a minor bug in the sampler of the original evaluation which was acknowledged by the authors in personal communication.

```
1 int32_t decompose(int32_t *w0,
2                    int32_t w) {
3   int32_t w1;
4   w1  = (w + 127) >> 7;
5   w1  = (w1 * 11275
6       + (1 << 23)) >> 24;
7
8   // If w1 == 44, set w1 = 0;
9    w1 ^= ((43 - w1) >> 31)
10   & w1;
11
12  // Compute w0 before the
13  // wraparound correction.
14  *w0 = w - w1 * 2 * GAMMA2;
15
16  % // If w0 > (Q - 1) / 2,
17  % // subtract Q.
18   *w0 -= (((Q - 1) / 2 - *w0)
19   >> 31) & Q;
20
21  return w1;
22 }
```

(a) Branch-free C code.

```
1 xor  eax, eax
2 cmp  ecx, 44
3 jl .LBB0_1
4 ...
5 cmp  ecx, 4190209
6 jge .LBB0_3
7
8 .LBB0_4:
9 mov  dword ptr [rdi], ecx
10 ret
11
12 .LBB0_1:
13 mov  eax, ecx
14 ...
15 cmp  ecx, 4190209
16 jl .LBB0_4
17
18 .LBB0_3:
19 lea  ecx, [rdx + rsi]
20 add  ecx, -8380417
21 mov  dword ptr [rdi], ecx
22 ret
23
```

(b) Vulnerable assembly.

Figure 4: Code of `PQCLEAN_MLDSA44_CLEAN_poly_decompose`. The sensitive conditional branches are highlighted in color. To align the code with our notation, the original variables `a`, `a1`, and `a0` have been renamed to `w`, `w1`, and `w0`, respectively.

intermediate in the `Decompose` function. These branches are not present in the source code but may be introduced during compilation under certain compiler configurations.

The `Decompose` function operates coefficient-wise on $\mathbf{w}$. For each coefficient of $\mathbf{w}$, it computes the corresponding coefficients of its high part $\mathbf{w}_1$ and low part $\mathbf{w}_0$. We present the C code implementation, taken from the PQClean library, in Figure 4a. Recall from Section 2.1 that `Decompose` computes $w = 2\gamma_2 w_1 + w_0$ for a given w where w_0 has absolute value smaller than γ_2. Lines 7–8 of Figure 4a first compute the raw high part $\widetilde{w}_1 = \left\lceil \frac{w}{2\gamma_2} \right\rfloor$ and store it in `w1`. Then, Line 18 computes the preliminary low part using the resulting value of `w1`. If $\widetilde{w}_1 < 44$, no wraparound has occurred and the computed value is already the centered low part w_0. If $\widetilde{w}_1 = 44$, however, `w1` has been set to zero, and Line 18 temporarily sets `w0` to w. Lines 23–24 handle this wraparound case by subtracting q, resulting in the centered low part $w_0 = w - q < 0$.

Although the `C` implementation contains no explicit conditional branches, the compiler may introduce them under certain compilation [75]. For example, compiling the code with `Clang-21.1.8` using the flags `-O3 -fno-vectorize -mllvm -x86-cmov-converter-force-mem-operand` produces the conditional branches shown in Figure 4b. In essence, the branch at Line 3 is taken if $\widetilde{w}_1 < 44$ and is not taken if $\widetilde{w}_1 = 44$. For coefficients with reconstructed $w_1 = 0$, the raw high part $\widetilde{w}_1$ is either 0 or 44. The branch distinguishes these cases: if it is taken, then $w_0 \in [0, \gamma_2]$; otherwise, $w_0 \in [-\gamma_2, -1]$. Thus, for these coefficients, the branch outcome reveals $\nu_{[j,i]} = \mathbf{1}[(\mathbf{w}_0)_{[j,i]} < 0]$.

6.2 Key Recovery

We now relate how the recovered $\nu_{[j,i]}$ (indicating the wraparound case) can be used to determine the MSB of coefficients of $\mathbf{w}_0$ and, therefore, form inequalities on $\mathbf{s}_2$ leading to its recovery.

Recall that part $\mathbf{w}_1$ can be reconstructed from a released signature. As established earlier, $\nu_{[j,i]}$ determines that sign bit of $(\mathbf{w}_0)_{[j,i]}$ for coefficients for which $(\mathbf{w}_1)_{[j,i]} = 0$. For coefficients with $(\mathbf{w}_1)_{[j,i]} \neq 0$, the branch outcome does not determine the sign of $(\mathbf{w}_0)_{[j,i]}$, and we discard these coefficients during the attack. Furthermore, we recompute $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$ using $\tilde{\mathbf{r}} = \mathbf{A}\mathbf{z} - c\mathbf{t} - 2\gamma_2\mathbf{w}_1$ as described in [4] and discard coefficients that do not satisfy $-\beta \leq \tilde{\mathbf{r}}_{[j,i]} < \beta$ to remove non-informative inequalities. Hence, together with the publicly reconstructed high-bit coefficient $(\mathbf{w}_1)_{[j,i]}$, the trace yields the sign bit $\nu_{[j,i]} = \mathbf{1}[(\mathbf{w}_0)_{[j,i]} < 0]$ for coefficient index $i \in \{0, \ldots, n-1\}$ of polynomial $j \in \{0, \ldots, k-1\}$. This gives the following inequalities constraining the secret key $\mathbf{s}_2$:

$$(c\mathbf{s}_2)_{[j,i]} \begin{cases} \geq -\tilde{\mathbf{r}}_{[j,i]}, & \nu_{[j,i]} = 0, \\ < -\tilde{\mathbf{r}}_{[j,i]}, & \nu_{[j,i]} = 1. \end{cases} \tag{25}$$

Alternatively, each inequality can be converted into an ILWE sample as described in [21] when linear regression is used. As discussed in the preceding sections, the inequalities and the resulting ILWE samples contain the same information when the respective error distributions are retained. Assuming that the coefficients of $\mathbf{w}$ are uniformly distributed over $\mathbb{Z}_q$, the probability that $(\mathbf{w}_1)_{[j,i]} = 0$ is $\frac{2\gamma_2+1}{q}$, and the corresponding probability that $\tilde{r}$ is informative is $\frac{2\beta}{2\gamma_2+1}$. Therefore, for each subkey, we obtain ≈ 0.004765 inequality constraints per signature—or one constraint every ≈ 210 signatures—for ML-DSA-44. Then, to evaluate key recovery for varying error probabilities p_{err}, the inferred bit $\widehat{\nu}_{[j,i]}$ of each retained coefficient is independently flipped from its true value $\nu_{[j,i]}$ with probability p_{err} to simulate measurement noise.

We follow the experimental setup described in Section 5, but limit each subkey instance to 2×10^6 retained constraints because of the memory constraints of our setup. Figure 5 reports the number of signatures required for the recovery of all 1024 coefficients of $\mathbf{s}_2$ at varying error probabilities p_{err} for BP and the scaled OLS baseline described in [22].

6.3 Instruction Counting with SGX-Step

We demonstrate the exploitability of the aforementioned non-constant time behavior in `Decompose` with an instruction counting attack in Intel SGX [61, 84]. For our experiments, we run the attack on an `Intel Xeon Gold 6526Y` server with Ubuntu 26.04 LTS. When compiling the SGX enclave with `Clang-21.1.8` using the flags `-O3 -fno-vectorize -mllvm -x86-cmov-converter-force-mem-operand`, the compiler produces assembly with non-constant-time behavior as shown in Figure 4.

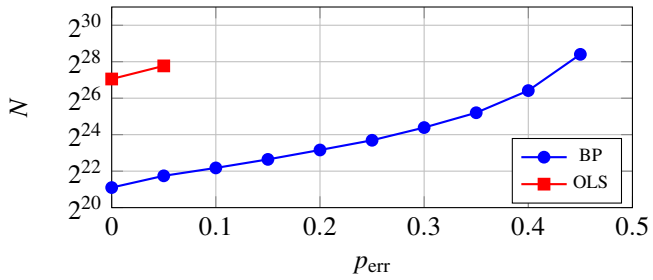


Figure 5: Attack complexity of the proposed ML-DSA-44 timing attack for varying error probabilities p_{err}. N is the number of signatures required by each solver for 100% recovery of $\mathbf{s}_2$. Recall that, as described in Section 6.2, we retain only coefficients with $(\mathbf{w}_1)_{[j,i]} = 0$. A direct bit leakage attack on the sign bits of $\mathbf{w}_0$ would therefore require $\approx N/44$ traces with the corresponding solver.

To launch an instruction counting attack on an SGX enclave we make use of the SGX-Step [82] framework to single-step the enclave and to design a page-fault sequence to find the start of the target function `Decompose`. The page-fault controlled-channel alone, however, is not sufficient to uniquely identify the start of `Decompose`. Therefore, as an additional metric, we consider the instruction count between the page-fault identification sequence and the next relevant page-fault. We find that `Decompose` is uniquely identifiable by the page-fault sequence plus its variable length of 12 to 15 instructions. The instruction count serves as both identification metric and the side-channel, allowing us to recover the outcomes of the two data-dependent branches for $\mathbf{w}_1$ and $\mathbf{w}_0$.

We observe two distinct event combinations. As can be seen in Figure 4b, when the branch in line 3 is not taken and in line 6 is taken, this corresponds to 14 counted instructions. Alternatively, if the branches in line 3 and in line 16 are taken, this corresponds to 13 instructions.

As a proof-of-concept, we collect 1000 traces of the `PQClean` sign function and evaluate the noise, execution time and correctness of the predicted signs for $\mathbf{w}_0$. For validation and comparison, we run 1000 additional traces in the same SGX enclave without single-stepping. Of the 1000 traces, only 936 yielded a complete trace usable for information extraction.

We use only the complete traces to compare the run time of computing a signature with and without the attack. Since every signature has a varying number of rejection sampling rounds, we report the timing per round instead of the average per signature. The average execution time of 936 traces without the single-stepping attack is 0.251 ms per rejection sampling round. With single-stepping, we observe an execution time of 7603 ms per rejection sampling round. This results in an average slowdown of approximately $\times 30000$ through the attack. Although the majority of the slowdown is attributed to the enclave exit and resume, our current implementation is a proof-of-concept that can be further optimized by more selectively single-stepping only the relevant `Decompose` calls or more efficient logging, which we leave for future work. To serially collect the $\approx 2^{21}$ traces in a noise-free setting, the attack would thus require approximately 2 years on average.

For checking the accuracy and correctness of our attack, we compare the branching information extracted from the side-channel information, as well as the predicted sign for $\mathbf{w}_0$ against the ground truth. Therefore, we first check that an observation of a non taken branch in line 3 of Figure 4 (b) corresponds with a zero coefficient in the public $\mathbf{w}_1$ and second that the recorded branching information correctly predicts the sign of $\mathbf{w}_0$ for every zero coefficient in $\mathbf{w}_1$. Due to the rejection sampling in ML-DSA, only the last rejection sampling round can be considered. For the first check, we find that from the 936 considered traces, approximately 95% correspond to a zero coefficient in $\mathbf{w}_1$, thus the number of false positives is low. Recall from Section 6.2 that an attacker will only consider those leakage points for which they know that the coefficient in $\mathbf{w}_1$ is 0. In the last rejection sampling round of all considered traces, we count a total of 21897 zero coefficients. We have a loss of 1013 classifications due to multi-stepping or other errors. Missing classifications are detected and sorted out. Additionally, 543 signs of $\mathbf{w}_0$ were incorrectly predicted. Thus, we are able to measure the true value of the MSB of $\mathbf{w}_0$ with an accuracy of $\approx 97\%$.

7 Conclusion

In this work, we showed that a soft-analytic approach to ILWE and CILWE has substantial advantages over prior recovery methods. This is in-line with improvements achieved by applying a soft-analytic point of view to attacks on post-quantum schemes, such as, e.g., in the recent improvements in PC-oracle attacks [37] or when targeting ML-KEM's noise term [39]. Conceptually, our advantage largely comes keeping track of the error distributions when deriving relations[6] and making use of priors on the key coefficients.

Our approach to filtering follows a similar idea: instead of heuristics, we apply principled information theory. However, as we do not observe any numerical issues or oscillations at practically relevant parameters in our attacks, we leave a thorough evaluation in settings where BP oscillates to future work. Another important area for future work is a clearer and conceptual understanding of the relation between ILP-based and soft-analytic solvers such as the one presented in the recent work of [8] and adjusting ours or the implementation of [38] to single-bit leakage, in particular to compare it to [8].

[6] Note the conceptual similarities to the concurrent work of [67] that also achieves improvements by keeping track of error probabilities.

8 Acknowledgments

This work was funded as part of the "Bavarian Chip-Design-Center" by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (funding number: RMF-SG20-3410-2-17-14).

References

[1] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the third round of the nist post-quantum cryptography standardization process. https://csrc.nist.gov/pubs/ir/8413/upd1/final.

[2] Amazon Web Services. Aws kms adds support for post-quantum ml-dsa digital signatures, June 2025.

[3] Chromium Blog Authors. Protecting chrome traffic with hybrid kyber kem, 2023. https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html.

[4] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting Dilithium against leakage revisited sensitivity analysis and improved implementations. *IACR TCHES*, 2023(4):58–79, 2023.

[5] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium – algorithm specifications and supporting documentation (version 3.1), 2021. https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[6] Chloé Baïsse, Antoine Moran, Guillaume Goy, Julien Maillard, Nicolas Aragon, Philippe Gaborit, Maxime Lecomte, and Antoine Loiseau. Secret and shared keys recovery on hamming quasi-cyclic with SASCA. *DCC*, 93(6):2137–2157, 2025.

[7] Brieuc Balon, Gaëtan Cassiers, Thibaud Schoenauen, and François-Xavier Standaert. How strong is the FOcalypse, really? instantiating plaintext-checking oracles against masked software implementations of ML-KEM. Cryptology ePrint Archive, Paper 2026/777, 2026. https://eprint.iacr.org/2026/777.

[8] Kaveh Bashiri, Jan Geuenich, and Johannes Mittmann. Exploiting noisy single-bit leakage in ML-DSA. Cryptology ePrint Archive, Paper 2026/580, 2026. https://eprint.iacr.org/2026/580.

[9] Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in Dilithium: A template-based approach. *IACR TCHES*, 2023(4):188–210, 2023.

[10] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[11] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 494–524. Springer, Cham, December 2018.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

[13] Olivier Bronchain, Melissa Azouaoui, Mohamed El-Ghamrawy, Joost Renes, and Tobias Schneider. Exploiting small-norm polynomial multiplication with physical attacks application to CRYSTALS-Dilithium. *IACR TCHES*, 2024(2):359–383, 2024.

[14] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. Cryptology ePrint Archive, Report 2021/817, 2021.

[15] Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *Journal of Open Source Software*, 8(86):5196, 2023.

[16] Cloudflare. Post-quantum encryption worldwide — cloudflare radar, 2026. https://radar.cloudflare.com/post-quantum. Accessed August 11, 2026.

[17] Jean-Sébastien Coron, François Gérard, Tancrède Lepoint, Matthias Trannoy, and Rina Zeitoun. Improved high-order masked generation of masking vector and rejection sampling in Dilithium. *IACR TCHES*, 2024(4):335–354, 2024.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[19] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Cham, August 2020.

[20] Simon Damm, Asja Fischer, Alexander May, Soundes Marzougui, Leander Schwarz, Henning Seidler, Jean-Pierre Seifert, Jonas Thietke, and Vincent Quentin Ulitzsch. Solving concealed ILWE and its application for breaking masked dilithium. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part II*, volume 16246 of *LNCS*, pages 608–640. Springer, Singapore, December 2025.

[21] Simon Damm, Nicolai Kraus, Alexander May, Julian Nowakowski, and Jonas Thietke. One bit to rule them all - imperfect randomness harms lattice signatures. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part I*, volume 15674 of *LNCS*, pages 284–316. Springer, Cham, May 2025.

[22] Simon Damm, Nicolai Kraus, Alexander May, Julian Nowakowski, and Jonas Thietke. One (Noisy) Bit to Rule Them All: Key Recovery from Randomness Leakage in ML-DSA. *Journal of Cryptology 39*, 2026.

[23] Jeroen Delvaux. Roulette: A diverse family of feasible fault attacks on masked Kyber. *IACR TCHES*, 2022(4):637–660, 2022.

[24] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Berlin, Heidelberg, August 2013.

[25] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018.

[26] Apple Security Engineering and Architecture. imessage with pq3: The new state of the art in quantum-secure messaging at scale, 2024. https://security.apple.com/blog/imessage-pq3/.

[27] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1857–1874. ACM Press, October / November 2017.

[28] Matt Etemad. Future-proofing data integrity: Quantum-safe digital signatures in cloud kms, July 2026. https://cloud.google.com/blog/products/identity-security/future-proofing-data-integrity-quantum-safe-digital-signatures-in-cloud-kms.

[29] Firefox. Firefox release notes 146.0, 2025. https://www.firefox.com/en-US/firefox/146.0/releasenotes/.

[30] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[31] Julie Godard, Nicolas Aragon, Philippe Gaborit, Antoine Loiseau, and Julien Maillard. Single trace side-channel attack on the MPC-in-the-head framework. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part II*, pages 267–293. Springer, Cham, April 2025.

[32] Guillaume Goy, Julien Maillard, Philippe Gaborit, and Antoine Loiseau. Single trace HQC shared key recovery with SASCA. *IACR TCHES*, 2024(2):64–87, 2024.

[33] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with enumeration: Which one beats the other and when? In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 291–312. Springer, Berlin, Heidelberg, November / December 2015.

[34] Qian Guo, Denis Nabokov, Alexander Nilsson, and Thomas Johansson. SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 203–236. Springer, Singapore, December 2023.

[35] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked CCA2 secure Kyber. *IACR TCHES*, 2021(4):88–113, 2021.

[36] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. *IACR TCHES*, 2023(4):287–317, 2023.

[37] Julius Hermelink, Erik Mårtensson, and Maggie Tran. Noise-tolerant plaintext-checking oracle attacks – A soft-analytic approach applied to ML-KEM. Cryptology ePrint Archive, Report 2025/1496, 2025.

[38] Julius Hermelink, Kai-Chun Ning, and Richard Petri. Finding and protecting the weakest link - on side-channel attacks on y in masked ML-DSA. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part V*, volume 16004 of *LNCS*, pages 3–37. Springer, Cham, August 2025.

[39] Julius Hermelink, Silvan Streit, Erik Mårtensson, and Richard Petri. A generic framework for side-channel attacks against LWE-based cryptosystems. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 3–32. Springer, Cham, May 2025.

[40] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[41] Peter J Huber and Elvezio M Ronchetti. *Robust statistics, second edition*. Wiley series in probability and statistics. John Wiley & Sons, Inc, Hoboken, N.J, 2nd ed. edition, 2009.

[42] Brian Jarvis. How to tune tls for hybrid post-quantum cryptography with Kyber, 2022. https://aws.amazon.com/blogs/security/how-to-tune-tls-for-hybrid-post-quantum-cryptography-with-kyber.

[43] Matthias J. Kannwischer. Constant-time hardening for `caddq`, `poly_chknorm`, and `decompose`. GitHub pull request #371, 2025. https://github.com/pq-code-package/mldsa-native/pull/371. Accessed August 11, 2026.

[44] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on Keccak. *IACR TCHES*, 2020(3):243–268, 2020.

[45] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. Improving software quality in cryptography standardization projects. Cryptology ePrint Archive, Report 2022/337, 2022.

[46] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. Improving software quality in cryptography standardization projects. In *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*, pages 19–30, Los Alamitos, CA, USA, 2022. IEEE Computer Society.

[47] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Cham, April / May 2018.

[48] Ehren Kret. Quantum resistance and the signal protocol, 2023. https://signal.org/blog/pqxdh/.

[49] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[50] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[51] Reimar H. Leike and Torsten A. Enßlin. Optimal belief approximation. *Entropy*, 19(8):402, 2017.

[52] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, and Rui Zhang. On security of Fiat-Shamir signatures over lattice in the presence of randomness leakage. Cryptology ePrint Archive, Report 2019/715, 2019.

[53] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Trans. Inf. Forensics Secur.*, 16:1868–1879, 2021.

[54] Zheng Liu, An Wang, Congming Wei, Yaoling Ding, Jingqi Zhang, Annyu Liu, and Liehuang Zhu. Release the power of rejected signatures: An efficient side-channel attack on the ML-DSA cryptosystem. Cryptology ePrint Archive, Report 2025/582, 2025.

[55] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

[56] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[57] Julien Maillard, Thomas Hiscock, Maxime Lecomte, and Christophe Clavier. Simulating SASCA on Keccak: Security Implications for Post-Quantum Cryptographic Schemes. In *21st International Conference on Security and Cryptography*, pages 518 – 527, Dijon, France, July 2024. SCITEPRESS - Science and Technology Publications.

[58] Dmitry M Malioutov, Jason K Johnson, and Alan S Willsky. Walk-sums and belief propagation in gaussian graphical models. *The Journal of Machine Learning Research*, 7:2031–2064, 2006.

[59] Alexander May and Julian Nowakowski. Too many hints - when LLL breaks LWE. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 106–137. Springer, Singapore, December 2023.

[60] Microsoft. What is ml-dsa support in ad cs?, May 2026.

[61] Daniel Moghimi, Jo Van Bulck, Nadia Heninger, Frank Piessens, and Berk Sunar. Copycat: Controlled instruction-level attacks on enclaves. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security*

Symposium, USENIX Security 2020, August 12-14, 2020, pages 469–486. USENIX Association, 2020.

[62] Harry Richard Moore. Robust regression using maximum- likelihood weighting and assuming cauchy-distributed random error. PhD thesis, Naval Postgraduate School, 1977.

[63] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 467–475. Morgan Kaufmann, 1999.

[64] Rishub Nagpal, Gaëtan Cassiers, Robert Primas, Christian Knoll, Franz Pernkopf, and Stefan Mangard. On loopy belief propagation for SASCAs: An analysis and empirical study of the inference problem. *CiC*, 1(4):15, 2024.

[65] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical report, Department of Commerce, Washington, D.C., 2024. Federal Information Processing Standards Publication (FIPS) NIST FIPS 204. https://doi.org/10.6028/NIST.FIPS.204.

[66] Paco Azevedo Oliveira, Andersson Calle Viera, Benoît Cogliati, and Louis Goubin. Uncompressing dilithium's public key. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part I*, volume 16000 of *LNCS*, pages 417–443. Springer, Cham, August 2025.

[67] Niklas Paskarbeit, Carsten Schubert, Marian Margraf, and Jean-Pierre Seifert. Statistical inference from noisy randomness leakage for ML-DSA attacks. Cryptology ePrint Archive, Paper 2026/1712, 2026. https://eprint.iacr.org/2026/1712.

[68] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In David L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence , Pittsburgh, PA, USA, August 18-20, 1982*, pages 133–136. AAAI Press, 1982.

[69] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *LNCS*, pages 130–149. Springer, Cham, October 2019.

[70] Yury Polyanskiy and Yihong Wu. *Information Theory: From Coding to Learning*. Cambridge University Press, 2025.

[71] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 513–533. Springer, Cham, September 2017.

[72] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. Practical public template attack attacks on crystals-dilithium with randomness leakages. *IEEE Trans. Inf. Forensics Secur.*, 18:1–14, 2023.

[73] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Side-channel assisted existential forgery attack on Dilithium - A NIST PQC candidate. Cryptology ePrint Archive, Report 2018/821, 2018.

[74] Prasanna Ravi, Thales Paiva, Dirmanto Jap, Jan-Pieter D'Anvers, and Shivam Bhasin. Defeating low-cost countermeasures against side-channel attacks in lattice-based encryption A case study on Crystals-Kyber. *IACR TCHES*, 2024(2):795–818, 2024.

[75] Moritz Schneider, Daniele Lain, Ivan Puddu, Nicolas Dutly, and Srdjan Capkun. Breaking bad: How compilers break constant-time implementations. In *ASIACCS 25*, pages 1690–1706. ACM Press, August 2025.

[76] Carsten Schubert, Niklas Julius Müller, Jean-Pierre Seifert, and Marian Margraf. Descent into broken trust: Uncovering ML-DSA subkeys with scarce leakage and local optimization. Cryptology ePrint Archive, Paper 2026/472, 2026. https://eprint.iacr.org/2026/472.

[77] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[78] Hauke Malte Steffen, Georg Land, Lucie Johanna Kogelheide, and Tim Güneysu. Breaking and protecting the crystal: Side-channel analysis of Dilithium in hardware. In Thomas Johansson and Daniel Smith-Tone, editors, *PQCrypto 2023*, pages 688–711. Springer, Cham, August 2023.

[79] Emanuele Strieder, Manuel Ilg, Johann Heyszl, Florian Unterstein, and Silvan Streit. ASCA vs. SASCA - A closer look at the AES key schedule. In Elif Bilge Kavun and Michael Pehl, editors, *COSADE 2023*, volume 13979 of *LNCS*, pages 65–85. Springer, Cham, April 2023.

[80] Vincent Quentin Ulitzsch, Soundes Marzougui, Alexis Bagia, Mehdi Tibouchi, and Jean-Pierre Seifert. Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ILP. *IACR TCHES*, 2023(4):367–392, 2023.

[81] Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on Dilithium - A small bit-fiddling leak breaks it all. In Benjamin Smith and Huapeng Wu, editors, *SAC 2022*, volume 13742 of *LNCS*, pages 3–32. Springer, Cham, August 2024.

[82] Jo Van Bulck, Frank Piessens, and Raoul Strackx. SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, pages 1–6, Shanghai China, October 2017. ACM.

[83] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 282–296. Springer, Berlin, Heidelberg, December 2014.

[84] Luca Wilke, Florian Sieck, and Thomas Eisenbarth. Tdxdown: Single-stepping and instruction counting attacks against intel TDX. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 79–93. ACM, 2024.

[85] Peiheng Zhang, Yuejun Liu, Wei Cheng, Muye Li, Honglin Shao, and Yongbin Zhou. From regression to prior-aware inference: Solving the ilwe family in randomness leakage attacks against ml-dsa, 2026. https://arxiv.org/abs/2607.05921.

[86] Yuhan Zhao, Wei Cheng, Zehua Qiao, Yuejun Liu, and Yongbin Zhou. Rejection matters: Efficient non-profiling side-channel attack on ML -DSA via exploiting public templates. Cryptology ePrint Archive, Paper 2026/056, 2026. https://eprint.iacr.org/2026/056.

A Extended Evaluation

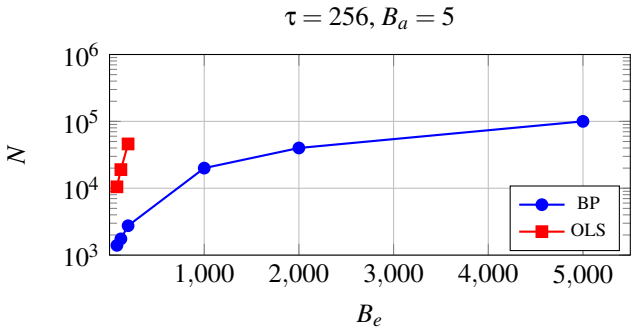


Figure 6: Sample complexity of 100% ILWE secret recovery. N is the number of ILWE samples required by each solver, while B_e is the error bound such that $\chi_e = \mathcal{U}(B_e)$. We use dense sampling vectors $\mathbf{a}$ ($\tau = 256$) with $a_i \sim \mathcal{U}(B_a)$ and sparse secrets ($\chi_s = \mathcal{S}$).

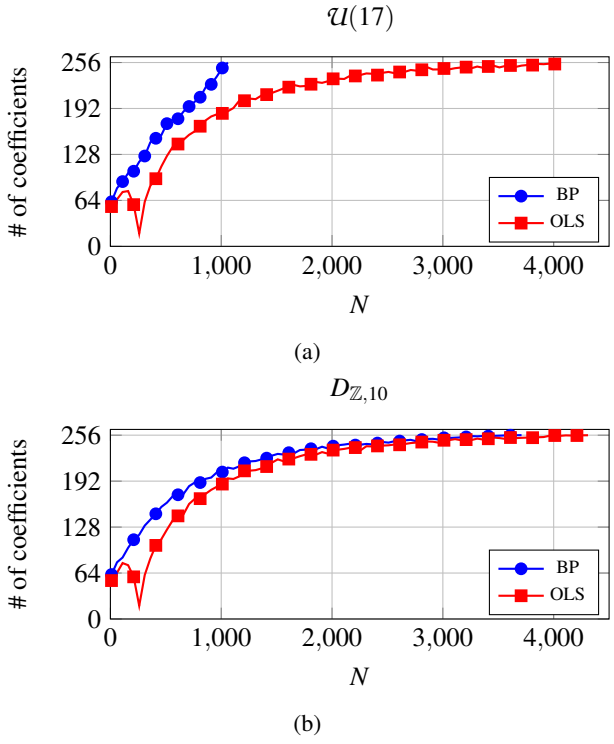
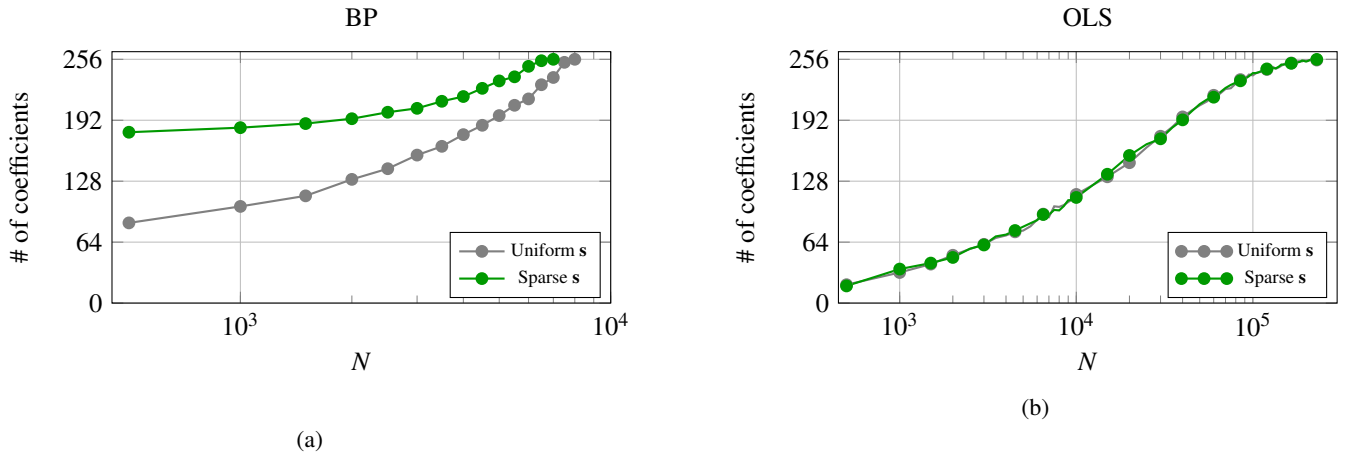
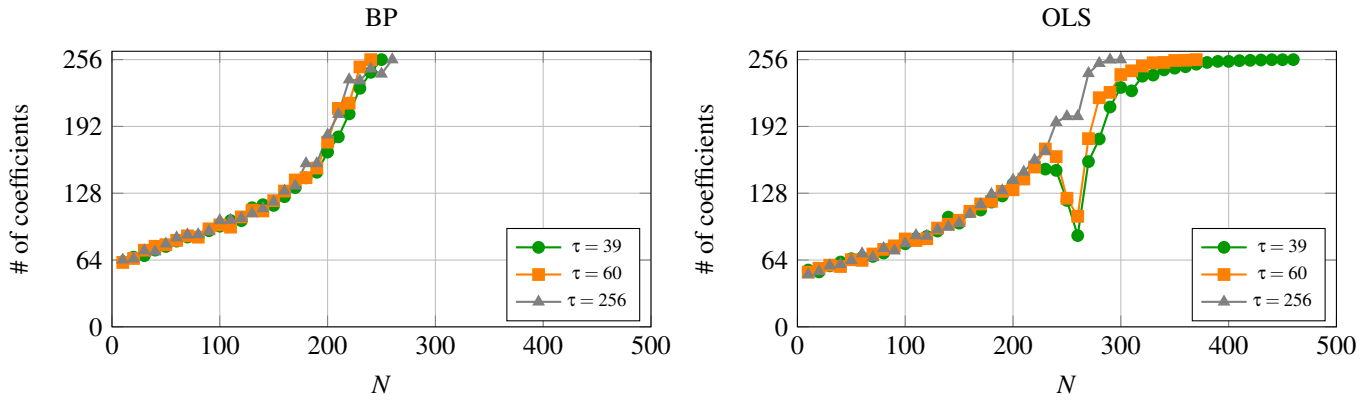


Figure 7: Effect of the error distribution on ILWE secret recovery. Each plot reports the mean number of correctly recovered secret coefficients as a function of the number of ILWE samples N. We compare uniform errors $\chi_e = \mathcal{U}(17)$, for which $\sigma_e^2 = 102$, with discrete Gaussian errors $\chi_e = D_{\mathbb{Z},10}$, for which $\sigma_e^2 \approx 100$. We use dense sampling vectors $\mathbf{a}$ ($\tau = 256$) with $a_i \sim \mathcal{U}(1)$ and uniform secrets $s_i \sim \mathcal{U}(2)$.



(a) (b)

Figure 8: Effect of the secret distribution on ILWE secret recovery. Each plot reports the mean number of correctly recovered secret coefficients as a function of the number of ILWE samples N. We use dense sampling vectors $\mathbf{a}$ ($\tau = 256$) with $a_i \sim \mathcal{U}(1)$, and errors $e \sim \mathcal{U}(120)$.



(a) (b)

Figure 9: Effect of sampling vector $\mathbf{a}$ sparsity on ILWE recovery. Each vector $\mathbf{a}$ has $\tau \in \{39, 60, 256\}$ randomly selected coefficients drawn from $\mathcal{U}(20)$, with all remaining coefficients set to zero. Errors are drawn from $D_{\mathbb{Z},10}$. We use uniform secrets $s_i \sim \mathcal{U}(2)$. Notably, when the number of samples is close to the number of dimensions of the secret, OLS turns into a fully determined system of equations, so the solution passes through all (erroneous) points. Hence, the accuracy drops.

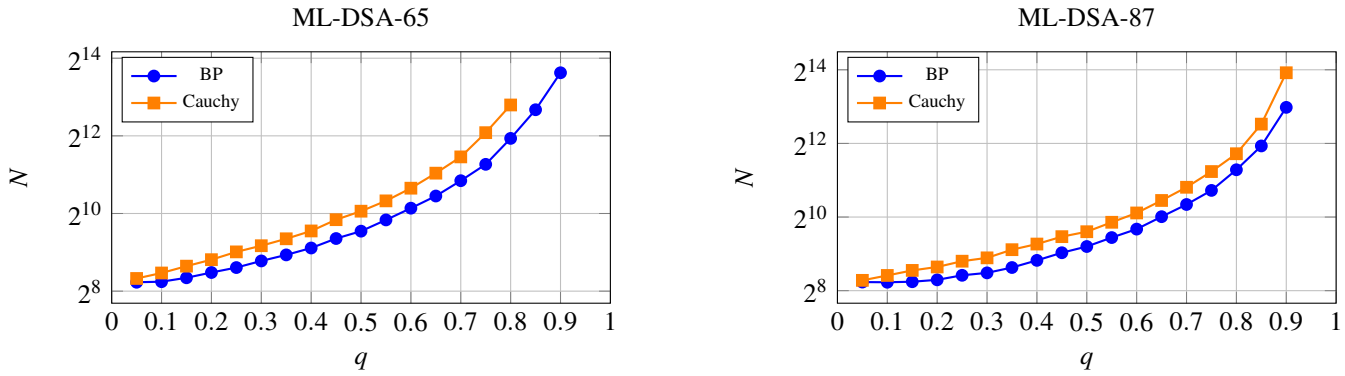


Figure 10: Sample complexity of CILWE secret recovery for ML-DSA-65 and ML-DSA-87. N is the number of CILWE samples, while q is the concealment rate as defined in [20]. See Footnote 5.

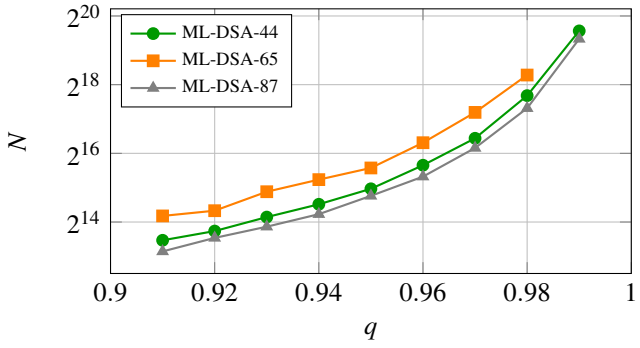


Figure 11: Sample complexity of CILWE secret recovery for instances with $q > 0.90$. N is the number of CILWE samples, while q is the concealment rate as defined in [20]. See Footnote 5.