Efficient SPA Countermeasures using Redundant Number Representation with Application to ML-KEM

Rishub Nagpal[1], Vedad Hadžić[2], Robert Primas[2], and Stefan Mangard[1]

[1] Graz University of Technology, Graz, Austria, firstname.lastname@tugraz.at
[2] Intel Labs, Portland, OR, United States, firstname.lastname@intel.com

Keywords: Power Analysis · Countermeasures · SASCA · ML-KEM

Abstract. Simple power analysis (SPA) attacks and their extensions, profiled and soft-analytical side-channel attacks (SASCA), represent a significant threat to the security of cryptographic devices and remain among the most powerful classes of passive side-channel attacks. In this work, we analyze how numeric representations of secrets can affect the amount of exploitable information leakage available to the adversary. We present an analysis of how mutual information changes as a result of the integer ring size relative to the machine word-size. Furthermore, we study the Redundant Number Representation (RNR) countermeasure and show that its application to ML-KEM can resist the most powerful SASCA attacks and provides a low-cost alternative to shuffling. We evaluate the performance of RNR-ML-KEM with both simulated and practical SASCA experiments on the ARM Cortex-M4 based on a worst-case attack methodology. We show that RNR-ML-KEM sufficiently renders these attacks ineffective. Finally, we evaluate the performance of the RNR-ML-KEM NTT and INTT and show that SPA security can be achieved with a 62.8% overhead for the NTT and 0% overhead for the INTT relative to the ARM Cortex-M4 reference implementation used.

1 Introduction

In 2022, the National Institute of Standards and Technology (NIST) selected the CRYSTALS-Kyber [Ava+21] and CRYSTALS-Dilithium [Duc+18] to serve as the basis for new PQC standards and these algorithms were standardized as FIPS203 [ST24b] under the name ML-KEM and FIPS204 [ST24a] (ML-DSA) in August 2024.

Although these algorithms are thought to be resistant against quantum cryptanalysis, side-channel analysis (SCA) attacks still pose a threat to the security of their implementations. These attacks exploit information leakage about secret data from physical side-effects such as the power consumption, timing or electromagnetic radiation of a device computing a cryptographic algorithm [KJJ99; Koc96; QS01]. Power SCA attacks can be categorized as either simple power analysis (SPA), where an adversary inspects a small number of measurements

of leaky functions ("traces"), or differential power analysis (DPA), where an adversary has access to a large number of traces. The discovery of SCA has motivated the design of countermeasures intended to minimize the amount of information leakage from these side-channels by e.g., increasing the number of traces needed to extract secret information. Countermeasures against DPA such as masking, where sensitive variables are split into $t + 1$ randomized shares, are well-studied and proven effective [ISW03; DDF19; Cha+99]. In contrast, developing countermeasures against SPA attacks is still a challenging topic. Profiled or template attacks [CRR02] and their extensions [SLP05; CK13; CK14; Cas+23; SA08; Ren+11; Bro+19; VGS14; Mas+23] are a particularly strong class of SPA attacks which use powerful statistical models of device leakage to perform key-recovery. These attacks can circumvent masking [BS21] and can recover secrets with a single attack trace cf. [CRR02]. Countermeasures against SPA typically employ masking and randomize or shuffle the order of operations between executions to make it more difficult to perform profiling, but are costly in terms of performance [Vey+12]. SPA has been shown to be effective against several cryptographic algorithms such as the AES, RSA, ECC and post-quantum candidates including ML-KEM and Dilithium [Bol+19; KPP20; Ngo+21; Ham+21; Her+23a; Str+23; HME24; Qia+24; KPP20; Bro+24].

Lattice-based cryptography is particularly susceptible to a more optimal form of template attacks known as Soft-Analytical Side-Channel Attacks (SASCA) [VGS14]. These attacks exploit knowledge of device operations, combining leakages from multiple points in a trace into a factor graph encoding. An inference method like loopy belief propagation (BP) then computes posterior probabilities of secret-key variables based on these leakages. The Number Theoretic Transform (NTT) operation, which is essential to lattice-based cryptography efficiently, is susceptible to SASCA [PPM17; PP19; Her+23a] and shuffling countermeasures are particularly costly [Rav+20]. Hence, it is important to find SPA countermeasures for the NTT which are cheap to implement and effective against powerful attacks such as SASCA.

Related Works. Zijlstra et al. proposed applying Redundant Number Representation (RNR) to general RLWE public-key cryptographic schemes [ZBT19]. This technique randomizes secret-key coefficients with multiples of the group order to hide their values. Later, Heinz et al. showed that this technique is effective against DPA and fault attacks against ML-KEM [HP23] when using a large redundancy factor. Recently, Tosun et al. observed side-channel distinguishers in the distribution of Hamming weights due to the signed modular reduction arithmetic used in lattice-based cryptoschemes and utilized these distinguishers to enhance DPA attacks on both ML-KEM and ML-DSA [TMS24]. Although not studied by Tosun et al., these distinguishers have huge implications on the effectiveness of SPA attacks. Previous SPA attacks on lattice cryptography in the literature may have unknowingly benefited from these distinguishers; therefore, it is essential to thoroughly study the impact these distinguishers have on SPA attacks.

Our Contributions. In this paper, we study the effectiveness of RNR against SPA attacks to better understand how this technique can reduce the amount of information leakage available to an adversary. In particular, we compare information leakage between different machine representations of data and show that there exists optimal numerical representations for storing secrets. We apply our findings to ML-KEM and show that the RNR-protected ML-KEM NTT^{-1} is resistant to state-of-the-art SASCA attacks in both simulation and practice. Our RNR-ML-KEM implementation incurs a 62.8% performance overhead to compute the NTT and 0% overhead to compute the INTT on the ARM Cortex-M4.

Concretely:

- We analyze the power side-channel leakage of arbitrary integer rings which are smaller than the machine-word size of a computing device and quantify the information leakage from this discrepancy.
- We analyze the Redundant Number Representation (RNR) countermeasure, which reduces the total information leakage by replacing a given integer ring with a larger, harmonic ring which contains multiples of the original.
- In particular, we create a RNR implementation of ML-KEM and demonstrate its effectiveness against SASCAs in simulation and on a ARM Cortex-M4 embedded target.
- We then show that one of the best known SPA attack on ML-KEM, the k-trace attack of Hamburg et al. [Ham+21], is rendered ineffective on RNR-ML-KEM - without the need for additional countermeasures such as shuffling.
- We open-source our implementations, datasets and additional scripts for the broader SCA research community.[3]

2 Preliminaries

We provide a brief review of the ML-KEM algorithm and the Number Theoretic Transform. Following, we go into detail on modular arithmetic and the algorithms commonly implemented. Finally, we discuss Soft-Analytical Side-channel Attacks implemented with belief propagation and metrics to evaluate the attacks.

2.1 Lattice Cryptography

We briefly introduce the Module Learning with Errors (M-LWE) hard problem proposed by Langlois and Stehlé [LS15], which merges the LWE [Reg05] and Ring-LWE [LPR10] hard problems.

Let $\mathbb{Z}_q$ be an integer ring modulo q and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$ be the polynomial ring of polynomials modulo x^n+1 for some degree n. Further, let β_η denote the centered binomial distribution with parameter η, and $\mathcal{U}$ denote the uniform distribution over $\mathbb{Z}_q$. Then, the Module-LWE problem is formally defined as

[3] `https://github.com/rishubn/rnr-kyber-spa`

Algorithm 1 ML-KEM.PKE KeyGen [ST24b]

Output: pk, sk

1: $(\rho, \sigma) \leftarrow \texttt{SHA3-512}(\{0,1,\ldots,255\}^{32})$
2: $\hat{A} \leftarrow \texttt{SamplePolyMatrix}(\rho)$
3: $s \leftarrow \texttt{SamplePolyCBD}_{\eta_1}(\texttt{SHAKE-256}_{\eta_1}(\sigma, 0))$
4: $e \leftarrow \texttt{SamplePolyCBD}_{\eta_1}(\texttt{SHAKE-256}_{\eta_1}(\sigma, 1))$
5: $\hat{s} \leftarrow \mathsf{NTT}(s)$
6: $\hat{e} \leftarrow \mathsf{NTT}(e)$
7: $\hat{t} \leftarrow \hat{A} \circ \hat{s} + \hat{e}$
8: **return** $pk = (\hat{t}, \rho), sk = \hat{s}$

Algorithm 2 ML-KEM.PKE Encryption [ST24b]

Input: $pk = (\hat{t}, \rho)$, message $m \in \mathcal{R}_q$, seed $r \in \{0,1,\ldots,255\}^{32}$
Output: ciphertext $c = (c_1, c_2)$

1: $\hat{A} \leftarrow \texttt{SamplePolyMatrix}(\rho)$
2: $\hat{y} \leftarrow \mathsf{NTT}(\texttt{SamplePolyCBD}_{\eta_1}(\texttt{SHAKE-256}_{\eta_1}(r, 0)))$
3: $e_1 \leftarrow \texttt{SamplePolyCBD}_{\eta_1}(\texttt{SHAKE-256}_{\eta_1}(r, 1))$
4: $e_2 \leftarrow \texttt{SamplePolyCBD}_{\eta_2}(\texttt{SHAKE-256}_{\eta_2}(r, 2))$
5: $u \leftarrow \mathsf{NTT}^{-1}(\hat{A}^{\mathsf{T}} \circ \hat{y}) + e_1$
6: $v \leftarrow \mathsf{NTT}^{-1}(\hat{t}^{\mathsf{T}} \circ \hat{y}) + e_2 + m$
7: **return** $c = (\texttt{Compress}(u), \texttt{Compress}(v))$

$$(\boldsymbol{A}, \boldsymbol{b}) = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{s} + \boldsymbol{e}, \tag{1}$$

with $\boldsymbol{s}$ being the secret-key, $\boldsymbol{A}$ the public matrix sampled from $\mathcal{U}$ and $\boldsymbol{e}$ an error vector sampled from β_η. The decisional M-LWE problem asserts that the tuple $(\boldsymbol{A}, \boldsymbol{b})$ is indistiguishable from a random sampling. The Kyber algorithm [Bos+18] is a public-key encryption (PKE) scheme which is now standardized as ML-KEM. Kyber is converted into a KEM via the Fujisaki-Okamoto (FO) transform [FO13; HHK17] and is conjectured to be IND-CCA2 secure. The polynomial ring in ML-KEM is parameterized with $q = 3329$ and $n = 256$ for all parameter sets. We give descriptions for the ML-KEM.PKE KeyGen (Algorithm 1), the ML-KEM.PKE Encryption (Algorithm 2) and the ML-KEM.PKE Decryption (Algorithm 3) algorithms and refer the reader to the ML-KEM specification [ST24b] for more details.

2.2 Number Theoretic Transform

The Number Theoretic Transform (NTT) is an algorithm analogous to the Discrete Fourier Transform (DFT) which allows to compute the product of two polynomials efficiently. We specify our description of the NTT on the polynomial ring of ML-KEM where the prime modulus $q = 3329$ and $n = 256$ and use the explanation of [ST24b]. Since there are 128 256-th roots-of-unity but no

Algorithm 3 ML-KEM.PKE Decryption [ST24b]

Input: ciphertext $c = (c_1, c_2)$, $sk = \hat{s}$
Output: message $m \in \mathcal{R}_q$
1: $(u, v) = (\texttt{Decompress}(c_1), \texttt{Decompress}(c_2))$
2: **return** $m = v - \mathsf{NTT}^{-1}(\hat{s}^{\mathsf{T}} \circ \mathsf{NTT}(u))$

512-th root-of-unity, the NTT is considered partial. Therefore the polynomial $x^{256} + 1$ is factored with small polynomials of degree 2,

$$(x^{256} + 1) = \prod_{i=0}^{127} \left(x^2 - \zeta^{2i+1}\right).$$

We denote with ζ^n the n-th root-of-unity. The NTT of the polynomial a is:

$$\mathsf{NTT}(a) = \hat{a} = \hat{a}_0 + \hat{a}_1 x + \ldots \hat{a}_{255} x^{255},$$

where

$$\hat{a}_i = \sum_{j=0}^{127} a_{2j} \zeta^{(2i+1)j} \qquad \text{and} \qquad \hat{a}_{2i+1} = \sum_{j=0}^{127} a_{2j+1} \zeta^{(2i+1)j}.$$

Multiplication of polynomials in the NTT domain is done by $\mathsf{NTT}^{-1}(\mathsf{NTT}(f) \circ \mathsf{NTT}(g))$. The basecase multiplication $\hat{h} = \hat{f} \circ \hat{g}$ is,

$$\hat{h}_{2i} + \hat{h}_{2i+1} x = (\hat{f}_{2i} + \hat{f}_{2i+1} x)(\hat{g}_{2i} + \hat{g}_{2i+1} x) \bmod \left(x^2 - \zeta^{2i+1}\right)$$

Implementing the NTT is similar to the DFT and common implementations utilize the Cooley-Tuckey (CT) butterfly [CT65] and the Gentleman-Sande (GS) butterfly [GS66] circuits. Hybrid designs utilize the CT butterfly on the forward pass and the GS butterfly on the inverse pass to avoid necessary bit-reversal steps. The incomplete ML-KEM NTT can be computed in seven butterfly layers.

2.3 Machine Representation of Integers

Computers natively work with integers in a finitely-sized domain $\mathbb{Z}_{2^l}$, in contrast to the infinitely large integer ring $\mathbb{Z}$ in mathematics. Here, l is usually referred to as the word size of a machine integer, and represents the number of bits needed to store the integer. Hence, an integer x is stored in a machine word as $\langle x \rangle_l = x - 2^l \left\lfloor \frac{x}{2^l} \right\rfloor$. Moreover, a stored integer can be interpreted either as *signed* or *unsigned*. The unsigned interpretation of $\langle x \rangle_l$ is $\langle x \rangle_l^+ = \langle x \rangle_l$, meaning that only integers $x \in \left[0, 2^l\right)$ have an unique representation and subsequent interpretation. The signed interpretation of $\langle x \rangle_l$ is given in two's complement, with $\langle x \rangle_l^{\pm} = \langle x \rangle_l - 2^l \left\lfloor \frac{\langle x \rangle_l}{2^{l-1}} \right\rfloor$, where only integers in the $x \in \left[-2^{l-1}, 2^{l-1}\right)$ have an unique representation and interpretation.

2.4 Modular Arithmetic

In this work, we are interested in addition and multiplication in modular integer rings $\mathbb{Z}_q$. More precisely, we are especially interested in rings $\mathbb{Z}_q$ where a canonical representation of each element fits into a machine word, e.g., $\forall x \in \mathbb{Z}_q : \langle x \rangle_l^+ = x$. For more efficient computation, we often also impose the additional limitation $q \leq 2^{l-1}$ to ensure that additions do not overflow.

The results of an addition or multiplication are not necessarily the canonical representation of an element in $\mathbb{Z}_q$. In general, for $x, y \in \left[0, 2^{l-1}\right)$, we have the (loose) result ranges $z = x + y \in \left[0, 2^l\right)$ and $w = x \cdot y \in \left[0, 2^{2l}\right)$. While this preserves congruence, i.e., $\langle z \rangle_l^+ = \langle x + y \rangle_l^+ \equiv x + y \mod q$ and $\langle w \rangle_{2l}^+ = \langle x \cdot y \rangle_{2l}^+ \equiv xy \mod q$, one must reduce the results back to the appropriate range $\left[0, 2^{l-1}\right)$ in order to perform further operations. Commonly, due to the methods used for reduction, one goes a step further and reduces the results into their canonical represesentation, e.g., the range $[0, q)$, or a representation with at most one redundant encoding e.g., the range $[0, 2q)$. Effectively, such a reduction would compute z, respectively w as

$$z = (x + y) - q \left\lfloor \frac{x + y}{q} \right\rfloor \qquad \text{and} \qquad w = (xy) - q \left\lfloor \frac{xy}{q} \right\rfloor .$$

While we have illustrated these operations in modular arithmetic with the unsigned canonical representations, it is also possible to represent the elements of $\mathbb{Z}_q$ as signed numbers in the range $\left[-\left\lfloor \frac{q}{2} \right\rfloor, \left\lceil \frac{q}{2} \right\rceil\right)$, cf. Hua et al. [Hua+22; Hua+24]. Here, having $x, y \in \left[-2^{l-2}, 2^{l-2}\right)$ is enough to prevent overflows or underflows in $z = x + y$ and $w = x \cdot y$. However, reductions are still necessary in order to establish these congruence invariants across multiple modular arithmetic operations.

In cryptographic implementations, the modular reduction is implemented using contant-time algorithms such as the ones of Barrett's [Bar86], Montgomery's [Mon85] and, most recently, Plantard's[Hua+22]. For lattice cryptography, the signed variants of these algorithms are perferred for computing the NTT for performance reasons. A detailed description of the Barrett and Montgomery reduction algorithms is given in Appendix A.

2.5 Soft-Analytical Side-Channel Attacks

Soft-Analytical Side-Channel Attacks (SASCAs) [VGS14] perform low-trace key-recovery attacks on cryptographic implementations. SASCAs improve upon template attacks by leveraging greater side-channel leakage using algorithmic knowledge and probabilistic modeling. In SASCAs, the algorithm is represented as a *factor graph*, encoding variables as circles and algorithmic operations as squares (called factors), connected by edges to illustrate their relationships. Side-channel leakage is incorporated through leakage factors which assign probability distributions to variables. An inference method, such as belief propagation [Pea82], computes marginal probabilities for secret-key variables which can then be used

in a rank estimation. We formally describe the *loopy* belief propagation variant based on [VGS14; MM09; Nag+25] and refer readers there for additional details.

Let $\boldsymbol{x}$ be a set of n random variables, $\boldsymbol{x} \equiv \{x_i\}_{i=1,\dots,n}$. The joint probability of the set can be expressed as the factorization,

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_{j=1}^{m} f_j(\boldsymbol{x}_I(j)).$$

Here, $\boldsymbol{x}_{I(j)}$ is the subset of variables in $\boldsymbol{x}$ which correspond to the factor f_j and Z is a normalization constant. The corresponding factor graph $\mathcal{G} = (\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{e})$ is a bipartite graph which consists of the variable nodes $\boldsymbol{x}$, factor nodes $\boldsymbol{f}$ and edges $\boldsymbol{e}$ which connect subsets $\boldsymbol{x}_{I(j)}$ to f_j.

Marginal probability distributions of variable nodes are computed iteratively using belief propagation [Pea82]. Updates occur in two steps per iteration t: variables send messages to factors (Equation 2), then factors send messages back to variables (Equation 3), repeating until convergence.

$$\mu_{(t+1)}^{x_i \to f_j} = \prod_{j' \in I(i) \setminus j} \mu_{(t)}^{f_{j'} \to x_i}. \tag{2}$$

$$\mu_{(t)}^{f_j \to x_i} = \sum_{k \in I(j) \setminus i} f_j(\boldsymbol{x}_k) \prod_{i' \in I(j) \setminus i} \mu_{(t)}^{x_{i'} \to f_j}. \tag{3}$$

Finally, the marginal for a variable $\Pr[X_i = x_i]$ is computed as the product of all incoming messages:

$$\Pr[X_i = x_i] = \prod_{\{j | i \in I(j)\}} \mu^{f_j \to x_i}. \tag{4}$$

2.6 Information Theoretic Metrics.

The Mutual Information (MI) metric relates the amount of information learned about a random variable X with its true leakage distribution $\mathcal{L}$.

$$\mathrm{I}[X; L] = \mathrm{H}[X] + \sum_{x \in \mathcal{X}} \Pr[x] \sum_{\ell \in \mathcal{L}} \Pr[\ell \mid x] \cdot \log_2 \Pr[x \mid \ell] \tag{5}$$

If the true leakage distribution is unknown, then a model of the leakage distribution may be used for the quantity $\Pr[\ell \mid x]$. If the leakage model differs from the true leakage, then the MI cannot be captured, instead the Perceived Information (PI) can be computed:

$$\mathrm{PI}[X; L] = \mathrm{H}[X] + \sum_{x \in \mathcal{X}} \Pr[x] \sum_{l \in \mathcal{L}_m} \Pr[l \mid x] \cdot \log_2 \Pr[x \mid l] \tag{6}$$

The PI is a lower bound of the MI [Bro+19] that converges to the MI if the model can match the true leakage distribution [Mas+23].

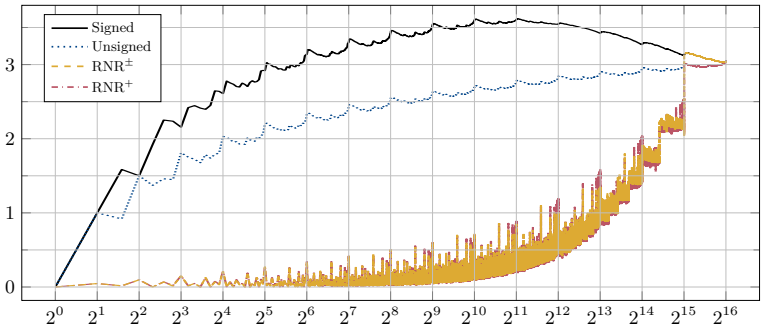

Fig. 1: Mutual information $\mathrm{I}[X;\mathrm{W}(X)]$ for an integer ring element $X \in \mathbb{Z}_q$ and its Hamming weight $\mathrm{W}(X)$ in 16-bit machine words, for different ring sizes $q \in [1, 2^{16})$.

3 Redundant Number Representation (RNR)

In this section, we first analyze the connection between the integer ring $\mathbb{Z}_q$, its representation, and the information gain an adversary receives through Hamming weights. Afterwards, we discuss ways of reducing the information leaked through Hamming weight, and analyze the countermeasure called Redundant Number Representation (RNR), which significantly reduces the leakage, effectively eliminating it for ring sizes q far lower than the maximal representable integers in an l-bit machine word.

3.1 Information Gain from Hamming Weights

In the Hamming weight leakage model we assume that the sum of all machine-word bits are leaked to an adversary through the power side channel. In the past, this model, while crude, has proven to closely match the realities of physical observations. While noisy, observations of both power consumption and EM field perturbations strongly correlate with the number of '1' bits in the data processed by a CPU.

However, the information contained within the Hamming weight of a machine word is not constant and heavily depends on the actual data. For example, if we know that a cryptographic algorithm works with bits, and each bit is stored in a separate machine word, the information revealed through the Hamming weight, and thus power side channels, is perfect, i.e., for $X \in \{0, 1\}$, we have $\mathrm{W}(X) = X$ and thus $\mathrm{I}[X;\mathrm{W}(X)] = 1$. On the contrary, for cryptographic algorithms that work with large integer rings, e.g., filling the whole machine word with $X \in \mathbb{Z}_{2^l}$, the Hamming weight $\mathrm{W}(X)$ leaks more information about X in an absolute sense, with $\mathrm{I}[X;\mathrm{W}(X)] \approx 3$ for $l = 16$, but this is due to the high starting entropy of X, with $\mathrm{H}[X] = l = 16$. Here, the relative information gain $\mathrm{I}[X;\mathrm{W}(X)] \,/\, \mathrm{H}[X]$ is much lower.

Additionally, the representation of the integer ring $\mathbb{Z}_q$ matters, that is, whether we encode it in an unsigned manner as $X \in [0, q)$ or in a signed manner with $X \in \left[-\left\lfloor \frac{q}{2} \right\rfloor, \left\lceil \frac{q}{2} \right\rceil\right)$. Here, due to the nature of two's complement encoding for negative numbers, we expect that, for $q \ll 2^l$, the Hamming weight $\mathrm{W}(X)$ directly reveals whether $X < 0$ or $X \geq 0$, i.e., about one additional bit of information about X. Figure 1 illustrates the amount of information that $\mathrm{W}(X)$ leaks about X for $q \in \left[1, 2^l\right)$ and a machine-word size of $l = 16$.

In almost all cases, the signed integer ring representation leaks more information through the Hamming weight. Prior work by Tosun et al. [TMS24] was the first to observe this discrepancy and use it to build better distinguishers for differential power analysis attacks on lattice-based post-quantum standardization candidates which perform computations in prime fields, i.e., where q is a prime number.

3.2 Reducing the Mutual Information

From the identity $\mathrm{I}[X; \mathrm{W}(X)] = \mathrm{H}[\mathrm{W}(X)] - \mathrm{H}[\mathrm{W}(X) \mid X]$, it is apparent that in order to reduce $\mathrm{I}[X; \mathrm{W}(X)]$, one must either:

1. decrease the entropy $\mathrm{H}[\mathrm{W}(X)]$, i.e., distribute the weights more evenly, or
2. increase the conditional entropy $\mathrm{H}[\mathrm{W}(X) \mid X]$, i.e., make it less certain what the weight $\mathrm{W}(X)$ is for any $X \in \mathbb{Z}_q$.

Here, for the encoding range $[q_\mathrm{l}, q_\mathrm{h})$ representing $\mathbb{Z}_q$, the entropies $\mathrm{H}[\mathrm{W}(X)]$ and $\mathrm{H}[\mathrm{W}(X) \mid X]$ are defined as

$$\mathrm{H}[\mathrm{W}(X)] = -\sum_{w=0}^{l} \Pr[\mathrm{W}(X) = w] \log_2(\Pr[\mathrm{W}(X) = w]) \quad \text{and}$$

$$\mathrm{H}[\mathrm{W}(X) \mid X] = \sum_{x \in [0,q)} \Pr[X \equiv x \bmod q] \, \mathrm{H}[\mathrm{W}(X) \mid X \equiv x \bmod q], \quad ;$$

$$\mathrm{H}[\mathrm{W}(X) \mid X \equiv x \bmod q] = -\sum_{w=0}^{l} \Pr[\mathrm{W}(X) = w \mid X \equiv x \bmod q] \cdot \log_2(\Pr[\mathrm{W}(X) = w \mid X \equiv x \bmod q]).$$

Additionally, the probabilities, assuming X is uniformly distributed in $[q_\mathrm{l}, q_\mathrm{h})$, are

$$\Pr[\mathrm{W}(X) = w] = \frac{|\{x \mid q_\mathrm{l} \leq x < q_\mathrm{h}, \mathrm{W}(x) = w\}|}{q_\mathrm{h} - q_\mathrm{l}},$$

$$\Pr[X \equiv x \bmod q] = \frac{|\{x' \mid q_\mathrm{l} \leq x' < q_\mathrm{h}, x' \equiv x \bmod q\}|}{q_\mathrm{h} - q_\mathrm{l}}, \quad \text{and}$$

$$\Pr[\mathrm{W}(X) = w \mid X \equiv x \bmod q] = \frac{|\{x' \mid q_\mathrm{l} \leq x' < q_\mathrm{h}, \mathrm{W}(x') = w, x' \equiv x \bmod q\}|}{|\{x' \mid q_\mathrm{l} \leq x' < q_\mathrm{h}, x' \equiv x \bmod q\}|}.$$

From this perspective, we also see why both the unsigned encoding with $[q_\mathrm{l}, q_\mathrm{h}) = [0, q)$ and the signed encoding $[q_\mathrm{l}, q_\mathrm{h}) = \left[-\left\lfloor \frac{q}{2} \right\rfloor, \left\lceil \frac{q}{2} \right\rceil\right)$ lead to high

mutual information $\mathrm{I}[X; \mathrm{W}(X)] = \mathrm{H}[\mathrm{W}(X)]$, since $\mathrm{H}[\mathrm{W}(X) \mid X] = 0$ because $\mathrm{W}(X)$ is completely determined by X, with $\Pr[\mathrm{W}(X) = w \mid X \equiv x \bmod q] \in \{0, 1\}$. Moreover, it is apparent that the signed encoding reveals more information in most cases, as the probabilities $\Pr[\mathrm{W}(X) = w]$ are better distributed, i.e., look more uniform, leading to a larger $\mathrm{H}[\mathrm{W}(X)]$.

In order to increase $\mathrm{H}[\mathrm{W}(X) \mid X]$, one must necessarily get rid of the constraint that $q_\mathrm{h} - q_\mathrm{l} = q$, i.e., one must allow for redundant encodings of $\mathbb{Z}_q$, where each remainder class $x \in [0, q)$ has multiple $x' \in [q_\mathrm{l}, q_\mathrm{h})$ such that $x' \equiv x \bmod q$. More importantly, one must redefine the operations of addition and multiplication, with the associated modular reductions, such that the results remain in the full range $[q_\mathrm{l}, q_\mathrm{h})$.

3.3 The Redundant Number Representation (RNR) Countermeasure

As outlined in the previous section, one can reduce the information about elements of $X \in \mathbb{Z}_q$ that is leaked through Hamming weights $\mathrm{W}(X)$, by increasing the range domain $[q_\mathrm{l}, q_\mathrm{h})$ and thus also increasing the entropy $\mathrm{H}[\mathrm{W}(X) \mid X \equiv x \bmod q]$.

In the following, we describe an implementation of Redundant Number Representation (RNR), a countermeasure that achieves such an increased conditional entropy by computing in the integer ring $\mathbb{Z}_{\eta q}$ instead of $\mathbb{Z}_q$, where η is a positive integer. Here, the workflow of RNR is as follows:

1. Start with algorithm that works in integer ring $\mathbb{Z}_q$, with inputs $\mathbf{X} = (X_i)_{i=0}^n$ and outputs $\mathbf{Y} = (Y_i)_{i=0}^m$, where $X_i, Y_i \in \mathbb{Z}_q$.
2. Determine η as the largest positive integer such that the algorithm operations do not overflow or underflow and break the congruences if they were computed in $\mathbb{Z}_{\eta q}$.
3. Encode algorithm inputs X_i as $X_i' \in \mathbb{Z}_{\eta q}$, where $X_i' = X_i + Kq$ and K is sampled uniformly at random from $[0, \eta)$ for unsigned representations and from $\left[\left\lfloor -\frac{\eta}{2} \right\rfloor, \left\lceil \frac{\eta}{2} \right\rceil\right)$ for signed representations.
4. Compute the algorithm while all operations are over $\mathbb{Z}_{\eta q}$.
5. Decode the outputs Y_i' as $Y_i = Y_i' - q \left\lfloor \frac{Y_i'}{q} \right\rfloor$, and return them as the results of the algorithm.

Figure 1 shows the advantage of employing RNR, where we set $\eta = \left\lfloor \frac{2^l - 1}{q} \right\rfloor$ because there are no operations computed. Impressively, RNR manages to virtually eliminate any information gained about X by knowing $\mathrm{W}(X)$ for $q \leq 2^{l/2}$, with $\mathrm{I}[X; \mathrm{W}(X)] < 0.5$. After that point, for larger ring sizes q, the mutual information increases, matching the non-RNR case at $q = 2^{l-1}$.

In the rest of this work, we discuss an application of RNR to the ML-KEM algorithm and show that RNR completely mitigates SASCAs.

4 Case Study: Applying Redundant Number Representation to the ML-KEM NTT

In this section, we begin with an analysis of the information leakage present in the ML-KEM NTT^{-1}. Then, we compare between simulated SASCAs of the NTT^{-1} assuming signed and unsigned Hamming weight leakage using an attack similar to the worst-case k-trace attack of Hamburg et al. [Ham+21]. Finally, we describe how to implement RNR-protected ML-KEM and show that it sufficiently thwarts the attack.

Mutual Information Analysis of the ML-KEM NTT. As described in Subsection 2.1, ML-KEM is parameterized on a polynomial ring parameterized by $n = 256$ and prime module $q = 3329$. The NTT function is an incomplete NTT which takes as input a vector of the 256 polynomial coefficients in $\boldsymbol{a} = (a_i)_{i=0}^{255}, a_i \in \mathbb{Z}_q$ and consists of seven butterfly layers. Furthermore, efficient implementations of the NTT store polynomial coefficients as signed 16-bit machine words e.g., $\langle a_i \rangle_{16}^{\pm}$ for performance reasons (cf. Algorithm 5, Algorithm 7, [Hua+22]). Moreover, the NTT and its inverse directly operate on the secret-key and thus requires protections (cf. Algorithm 1, Algorithm 3).

The coefficients of an input polynomial can be modeled with a vector of 256 random variables, $\boldsymbol{X} = (X_i)_{i=0}^{255}$ where $X_i \in \mathbb{Z}_q$. The entropy of each X_i is, $\mathrm{H}[X_i] \approx 11.701$ bits, and the total entropy an adversary needs to eliminate is, $\mathrm{H}[\boldsymbol{X}] = \sum_{\boldsymbol{X}} \mathrm{H}[X_i] \approx 2995.4$ bits. The mutual information between a coefficient X_i and its Hamming weight $\mathrm{I}[X_i; \mathrm{W}(X_i)] \approx 3.561$ bits in a signed 16-bit machine word setting, $\langle a_i \rangle_{16}^{\pm}$. Thus, an adversary learns $\sum_{\boldsymbol{X}} \mathrm{I}[X_i; \mathrm{W}(X_i)] \approx 911.6$ bits of information from the Hamming weights of the coefficients alone, and the remaining entropy conditioned on this knowledge that must be eliminated is $\mathrm{H}[\boldsymbol{X}] - \sum_{\boldsymbol{X}} \mathrm{I}[X_i; \mathrm{W}(X_i)] \approx 2083.8$ bits. Following the analysis of Section 3, storing coefficient as unsigned words, $\langle a_i \rangle_{16}$ would reduce the amount of information learned from the Hamming weight to 2.756 bits and thus the total (conditional) entropy to eliminate is raised to ≈ 2289.9 bits. Thus, switching from signed to unsigned storage of polynomial coefficients would reduce the information leakage from Hamming weights by ≈ 206.092 bits in total.

The NTT butterfly operations further leak information jointly about the polynomial coefficient operands. Let $A, B \in \mathbb{Z}_q$ be random variables which represent butterfly inputs and $A', B' \in \mathbb{Z}_q$ represent butterfly outputs. The joint entropy of the inputs is, $\mathrm{H}[A, B] = \log_2 q^2 \approx 23.402$ bits. By enumerating the joint Hamming weight distribution of every possible input (A, B) and output (A', B') pairs, the information learned about A, B from knowing the Hamming weights $\mathrm{W}(A), \mathrm{W}(B), \mathrm{W}(A'), \mathrm{W}(B')$ is,

$$\mathrm{I}[A, B; \mathrm{W}(A), \mathrm{W}(B), \mathrm{W}(A'), \mathrm{W}(B')] \approx 13.879 \text{ bits}$$

for signed and ≈ 10.877 bits for unsigned representations of the integers. It holds that A and B are independent considering that the input polynomial coefficients are randomly sampled, then the adversary learns an additional ≈ 1.5 bits about

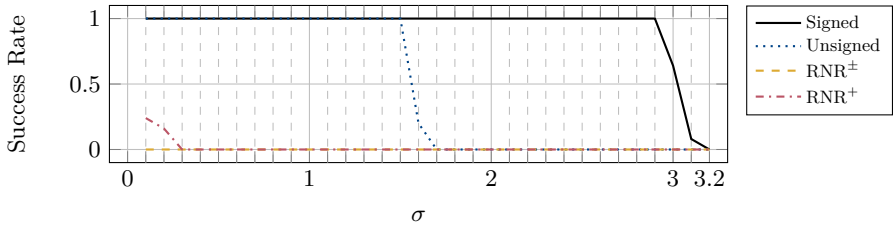

Fig. 2: SASCA on the signed, unsigned and the corresponding RNR variants of the ML-KEM NTT^{-1} using simulated leakage traces.

each coefficient from signing after the first layer of butterflies from the Hamming weights.

4.1 Comparing SASCAs on Different Machine representations.

SASCAs on ML-KEM typically exploit the (inverse-)NTT steps and such attacks have been well-studied [PPM17; PP19; Ham+21; AER23]. The k-trace attack of Hamburg et al. [Ham+21] describes a methodology to perform private-key recovery efficiently using a decryption oracle. In the attack, ciphertexts are crafted such that a chosen number of polynomial coefficients will be equal to zero in the NTT domain. Hence, the input to the NTT^{-1} transformation will be a sparse vector that is the product of the secret-key and the ciphertext NTT-polynomials: $\mathsf{NTT}^{-1}(\hat{s} \circ \mathsf{NTT}(u))$. Then, combined with side-channel information, an adversary can perform an efficient attack e.g., SASCA, to recover the NTT^{-1} sparse input coefficients and solve for the secret-key. In our experimental analysis, we adapt this attack under the simplified assumption that successfully recovering only a block of 32 non-zero contiguous coefficients is sufficient. This scenario, originally applicable only to Kyber1024 in Hamburg et al.'s work, was chosen for its effectiveness in high-noise cases, which better allows to interpolate the effectiveness of the RNR countermeasure. Furthermore, it assumes a more powerful adversary requiring less profiling and attack effort. In our setting, an adversary needs to eliminate ≈ 374.4 bits of entropy in total.

Simulating SASCAs. As a starting point, we simulated SASCAs on the ML-KEM NTT^{-1} using leakages generated from a Hamming weight leakage model. In this model, a leakage trace, ℓ, is generated from the Hamming weight of an intermediate, x, plus the addition of normally distributed noise with mean zero and standard deviation parameterized by σ, i.e., $\ell = \mathrm{HW}(x) + \mathcal{N}(0, \sigma)$. The factor graph model of the NTT^{-1} consists of variable nodes representing the input coefficients followed by the 7-layers GS-butterfly layers. We merge the arithmetic butterfly operations into a single butterfly factor based on the work Pessl et al., who showed that this improves the outcome of the belief propagation by eliminating small loops [PP19]. In total, the factor graph contains 1792 factors.

Table 1: Parameter sets for implementing RNR-ML-KEM variants

Parameter	RNR^{+}	$\text{RNR}^{\pm}$
η	5	9
Barrett Radix R	2^{31}	2^{29}
Montgomery q^{-1}	39987	-22215

In our experiments, we compute the NTT^{-1} on a random sparse input vector with 32 contiguous non-zero values, $\boldsymbol{x} = \left(x_0, \ldots x_{31}, \{0\}^{224}\right)$ with $x_i \in \mathbb{Z}_q$. We generate simulated Hamming weight leakages of the input vector and of the vector after each butterfly layer in the NTT^{-1} for a given machine representation. The leakages are assigned to the leakage factors in the factor graph and then we run belief propagation for 25 iterations and use no additional heuristics, as recommended by Nagpal et al. [Nag+25]. After BP, we compute the ranks of the marginal distributions for each of the non-zero input coefficients and the attack is considered successful if their sum is zero. In Figure 2, we compare the impact of different machine representations on the SASCA. We report the average success rate of 25 SASCA experiments for noise levels $0.1 \leq \sigma \leq 3.2$ in increments of 0.1. The SASCA performs demonstrably better for signed storage: It achieves a 100% success rate until $\sigma \geq 3.0$. In contrast, the SASCA attacking unsigned storage does not fair so well: The performance of the attack degrades significantly when $\sigma \geq 1.6$. The implications of these results indicates that the faster performance of the NTT from using the signed machine storage comes at the cost of security. Overall, the total information available to the adversary is decreased by 25.7 bits by the change from signed to unsigned machine storage.

4.2 RNR-ML-KEM

From Section 3, it is clear that the ML-KEM integer field $\mathbb{Z}_q$ is an ideal candidate for protection with RNR since the prime modulus is much smaller than the maximum representable integer in both signed and unsigned 16-bit machine word sizes. To do so, we define a larger integer ring whose module is the product of the ML-KEM prime and an odd integer, η, such that $\eta q < 2^l$. We propose signed and unsigned variants, denoted as $\text{RNR}^{\pm}$-ML-KEM and RNR^{+}-ML-KEM, with parameters $\eta^{\pm}$ and η^{+} respectively. Here, according to the description of RNR in Subsection 3.3 we must choose $\eta^{\pm}$ and η^{+} in such a way that no overflows occur during the computation of the forward and inverse NTT, as well as other operations in ML-KEM. The butterfly operation in the forward NTT takes coefficients a and b, as well as a root of unity ζ, and produces the results $a' \equiv a + b\zeta \mod p$ and $b' \equiv a - b\zeta \mod p$.

Signed case. For the signed case, assume that $a, b \in [-c, c]$ and $\zeta \in [-z, z]$, and that $b\zeta$ is reduced with the signed Montgomery reduction to get the term t. According to the definition of Montgomery reduction, the result t is in the range

$[-u, u]$, with $u = \frac{cz}{2^{16}} + \frac{\eta^{\pm} q}{2}$. The results a', b' of the butterfly operation are thus in the range $[-c-u, c+u]$. In the first layer of the forward NTT operation, we have $c = \frac{\eta^{\pm} q}{2}$ and $z = \frac{q}{2}$. We can then determine the upper bound for $\eta^{\pm}$ by enforcing that the computation of a' and b' does not overflow, i.e., $c + u < 2^{15}$. Therefore,

$$\begin{aligned} \frac{\eta^{\pm} q}{2} + \left(\frac{\frac{\eta^{\pm} q}{2} \frac{q}{2}}{2^{16}} + \frac{\eta^{\pm} q}{2} \right) &< 2^{15} \\ \eta^{\pm} \cdot \left(q + \frac{q^2}{2^{18}} \right) &< 2^{15} \\ \eta^{\pm} &< \frac{2^{33}}{2^{18} q + q^2} < 10 \end{aligned} \tag{7}$$

Hence, one can pick any $\eta^{\pm} \in [1, 9]$. After performing a similar computation for the inverse NTT and getting the same positive integer range for $\eta^{\pm}$, we pick $\eta^{\pm} := 9$ for our implementation. Importantly, this forces us to perform a Barret reduction on a' and b' in the forward NTT butterfly operation to maintain the invariant $a, b \in [-c, c]$ for the next layer of the forward NTT. For the inverse NTT, one can perform fixpoint-based derivation to show that output b' produced via Montgomery reduction does not need to be additionally reduced via a Barrett reduction. For smaller choices of $\eta^{\pm}$, one may have to perform a Barret reduction only after a number forward or inverse NTT layers, e.g., in the conventional implementation with $\eta^{\pm} = 1$, the forward NTT only performs a Barrett reduction after the last NTT layer.

Unsigned case. For the unsigned case, the considerations are similar. Assume that $a, b \in [0, c]$ and $\zeta \in [0, z]$, and that t is the result of $b\zeta$ reduced with the unsigned Montgomery reduction. Consequently, t is in the range $[0, u]$, with $u = \frac{cz}{2^{16}} + \eta^{+} q$. We again determine the upper bound on η^{+} by enforcing that the computation of a' and b' does not overflow. We compute a' as usual, and therefore have the constraint $c+u < 2^{16}$, while b' must be cmoputed as $b' = a + \left(\left\lceil \frac{u}{q} \right\rceil q - t \right)$ to prevent underflows, and hence getting the constraint $c + \left(\left\lceil \frac{u}{q} \right\rceil q - t \right) < c + u + q < 2^{16}$. For the first layer, assuming $c = \eta^{+} q$ and $z = q$, we have

$$\begin{aligned} \eta^{+} q + \left(\frac{\eta^{+} q^2}{2^{16}} + \eta^{+} q \right) + q &< 2^{16} \\ \eta^{+} \cdot \left(2q + \frac{q^2}{2^{16}} \right) &< 2^{16} - q \\ \eta^{+} &< \frac{2^{32} - 2^{16} q}{2^{17} q + q^2} < 10 \end{aligned} \tag{8}$$

A similar analysis of the butterfly operation in the inverse NTT confirms that $\eta^{+} \in [1, 9]$, and thus we can choose $\eta^{+} := 9$ and perform necessary Barrett reductions after additions and subtractions. Applying RNR to ML-KEM is straightforward, as it does not require additional randomness or operations: It requires

changes only to the rejection sampling used in the functions `GeneratePolyMatrix`, `SamplePolyCBD` in Algorithm 1 and Algorithm 2 such that it outputs coefficients in $\mathbb{Z}_{\eta q}$ instead of $\mathbb{Z}_q$, and to modify the modular reduction parameters. As stated above, for larger η, it may be necessary to reduce intermediate additions and subtractions. We give pseudocode for implementing signed and unsigned RNR-variants of the $\mathsf{NTT}/\mathsf{NTT}^{-1}$ in Appendix B. With the above analysis, we use the parameter sets given in Table 1 for our RNR to ML-KEM implementations discussed in the rest of this work. Analyzing the information learned from Hamming weights similarly to the beginning of Section 4, RNR coefficients have greater entropy: ≈ 14.871 bits for $X \in \mathbb{Z}_{\eta^{\pm} q}$ and 14.023 bits for $X \in \mathbb{Z}_{\eta^{+} q}$, respectively. On the other hand, RNR coefficients exhibits far less information leakage: 0.888 bits for $X \in \mathbb{Z}_{\eta^{\pm} q}$ (1.219 bits in $\mathbb{Z}_{\eta^{+} q}$ resp.). The difference in leakage between RNR variants is due to the smaller gap between storing integers integers in $\mathbb{Z}_{\eta^{\pm} q}$ with the signed storage,

$$2^{16} - \eta^{+} q > 2^{16-1} - \eta^{\pm} q.$$

Revisiting the NTT butterfly analysis, the joint entropy for two random variables representing integer coefficients in $\mathbb{Z}_{\eta^{\pm} q}$ is $\mathrm{H}[A, B] \approx 29.741$ bits ($\approx$ 28.046 bits for $\mathbb{Z}_{\eta^{+} q}$ resp.) and the information learned from knowing the Hamming weights of $A, B, A', B' \in \mathbb{Z}_{\eta q}$ is,

$$\mathrm{I}[A, B; \mathrm{W}(A), \mathrm{W}(B), \mathrm{W}(A') \mathrm{W}(B')] \approx 12.549 \text{ bits}$$

for coefficients in $\mathbb{Z}_{\eta^{\pm} q}$ (≈ 11.744 bits in $\mathbb{Z}_{\eta^{+} q}$ resp.). The information learned from the first butterfly layer due to signed representations is reduced to 0.804 bits, while at the same time, the overall entropy is higher. We next evaluated both RNR-ML-KEM variants with the simulated SASCA experiments of Subsection 4.1, also depected in Figure 2. For all noise levels, the RNR$^{\pm}$ variant renders the attack completely ineffective. When Hamming weights are known with certainty ($\sigma \leq 0.1$), attacks on RNR^{+}-ML-KEM succeed about 25% of the time. Compared to the experiments on the unprotected ML-KEM, the experiments show that RNR is effective at mitigating the attacks.

5 Experimental Analysis of RNR

In this section, we evaluate the effectiveness of the RNR countermeasure against profiled SASCA attacks using both simulated and ARM Cortex-M4 traces of the reference ML-KEM NTT implementation. In both simulated and practical scenarios, we analyze the *signed*, *unsigned*, *RNR*$^{\pm}$ and *RNR*$^{+}$ NTT implementations discussed in the previous sections. As in Subsection 4.1, we perform the same chosen-ciphertext attack of Hamburg et al. [Ham+21] with the same assumptions.

5.1 Attacking RNR-ML-KEM on the ARM Cortex-M4

To better understand how RNR performs on real devices, we analyzed the four ML-KEM NTT implementations on the ARM Cortex-M4. Our experimental

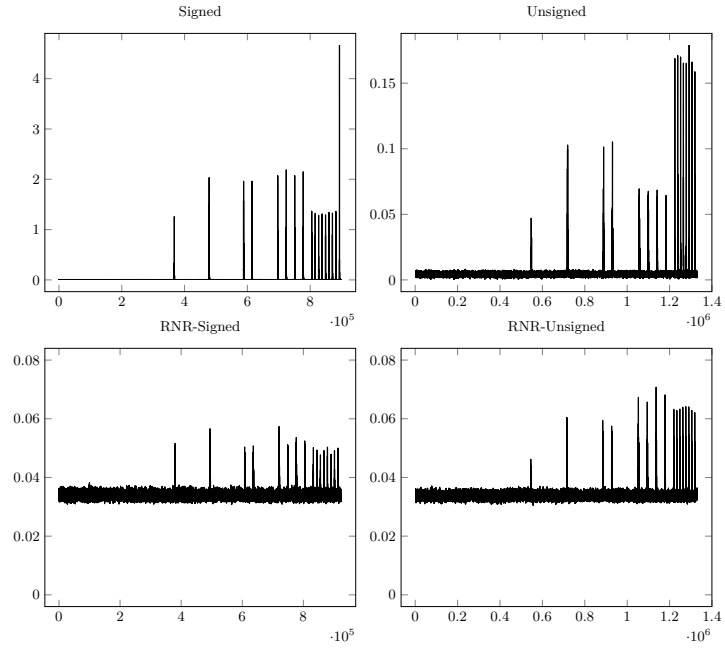

Fig. 3: The SNR of the first intermediate in the third layer of the NTT^{-1} computed from 100 000 profiling traces.

setup uses the Chipwhisperer CW308 UFO board [Inc24] with the STM32F303 target measured by the Picoscope 6404C oscilloscope. The measurement probe is amplified via a 20 dB external LNA. The target is clocked at 10 MHz and synchronized to the oscilloscope sampling frequency. We measure the full NTT^{-1} operation.

Profiling. The seven-layer NTT^{-1} used in ML-KEM, including the input vector, has 1792 intermediates which can be potentially profiled. We assume that only the first 32 intermediates in the input to the NTT^{-1} are non-zero. This leads to several interesting effects which benefit the profiling. First, the zeroed intermediates are known with absolute certainty and templates do not need to be constructed for those positions. Second, when a zeroed intermediate is an operand to a GS-butterfly, the result is either the unchanged or inverted input (prior to the ζ-multiplication). Thus the same value can appear in subsequent layers of the NTT^{-1} causing an "amplification" effect which can be exploited for a particular intermediate and results in a stronger template. Using both of these facts, we reduced the number of needed templates to 864 and the quality of templates in deeper layers of the NTT^{-1} is much better than shallow layers.

For each attack case, we built our templates as follows. We collected 100 000 profiling traces of the NTT^{-1} where the first 32 intermediate were random and the rest zeroed. Then, we computed the signal-to-noise ratio (SNR) to find points-

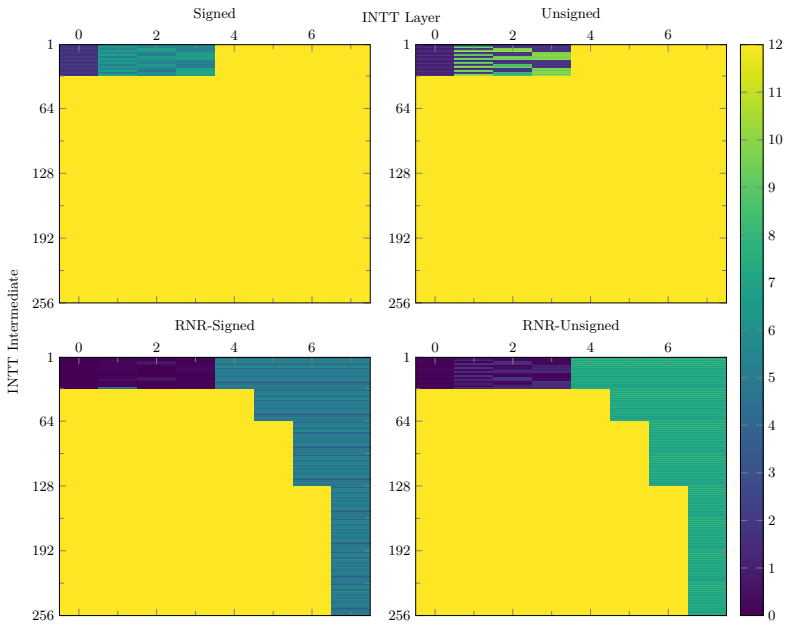

Fig. 4: The mean PI of the leakage models for all four cases as a result of profiling. We computed for nine levels of intermediates of the NTT^{-1}: The input, the seven layers of GS butterflies and the final finite-field multiplication. The quantities are given within ± 0.1 bits where possible with 95% confidence.

of-interest to build the models. The SNR for the first intermediate in the fourth layer of the NTT^{-1} is shown in Figure 3. This position in the NTT^{-1} shows the amplification effect: There are multiple peaks throughout the trace which can be used for template building. The signed attack case shows high SNR peaks, with several points above 1. To the right, the unsigned case already exhibits a strong reduction in SNR, reducing the maximum SNR by an order-of-magnitude. On the bottom row, both RNR implementations reduce the SNR by two orders-of-magnitude compared to the signed case with the $\text{RNR}^{\pm}$ implementation having the lowest SNR overall.

Next, after computing the SNR for every relevant intermediate, we selected points-of-interest (POIs) above an arbitrarily selected threshold, sorting them and selecting up to 1000 points per intermediate. We then constructed leakage models using an LDA classifier (using the implementation of SCALib [CB23]) individually. The number of subspace dimensions was tuned per model using a parameter sweep on a subset of intermediates. To assess the quality of the models, we computed the average perceived information computed using a fresh set of random traces, which we report in Figure 4 within ± 0.1 bits with 95% confidence. In these plots, we show that RNR greatly reduces the quality of the leakage models. For the signed and unsigned case, the models are able to

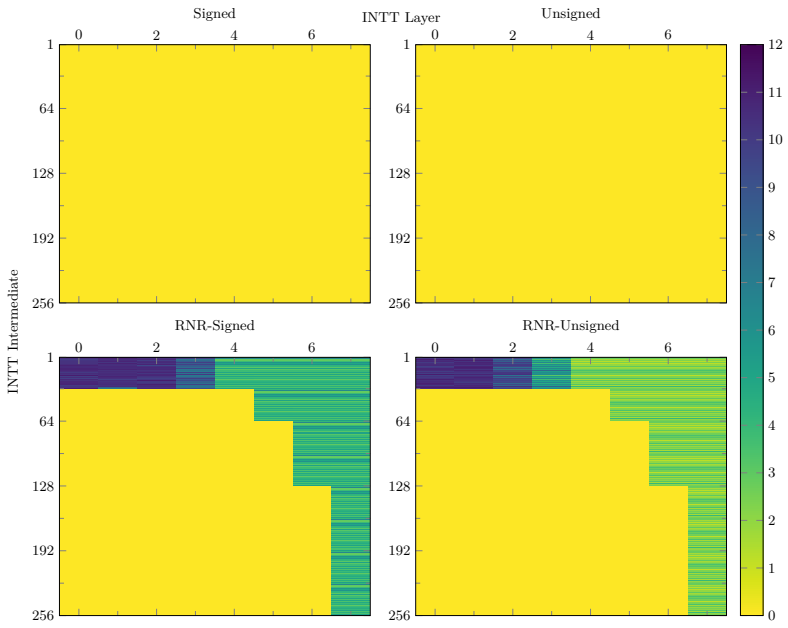

Fig. 5: Guessing entropy of each intermediate in the ML-KEM NTT^{-1} after a single-trace SASCA. 0 GE means the value is known without any uncertainty. Note, the colormap is inverted to make it easier to compare to Figure 4.

predict the correct value with near certainty by the fourth layer of the NTT^{-1} due to the amplification effect. In particular for the unsigned case, the required additional Barrett reduction step with known quantities in the first, second and third layer further improves the quality of the models. The RNR implementations are able to reduce the information the models can provide substantially. The shallow layers of the NTT^{-1} have PI close to zero, and even with the amplification effect, the maximum PI given by a model are 5.8 bits and 7.8 bits for $\text{RNR}^{\pm}$ and RNR^{+} respectively. Thus, we conclude that RNR greatly increases the difficulty of carrying out a template attack on the ML-KEM NTT^{-1} in this ideal adversarial scenario.

Attack. We performed SASCAs on each implementation with our templates. In Figure 5, we report the average guessing entropy of every intermediate in the eight layers of the NTT^{-1} over 25 independent experiments for each point. The attacks show that the inputs to both the signed and unsigned NTT^{-1} can be recovered after a single-trace, consistent with our results from the simulated SASCAs. Furthermore, the attack is rendered ineffective against both the $\text{RNR}^{\pm}$ and RNR^{+} implementations and the SASCA is unable to significantly improve the GE of the output of the templates. In the deeper layers, the belief propagation is able to reduce the GE of some intermediates due to the many known

Table 2: Performance evaluation of the four implementations of the ML-KEM-NTT^{-1} measured on the ARM Cortex-M4 compiled with `arm-none-eabi-gcc` version 10.3.1 for `-O0` and `-O3` with flags `-mcpu=cortex-m4 -mthumb -mfloat-abi=soft`. The results are reported in thousands ($\cdot 10^3$) of clock cycles.

Implementation	KCycles ($\cdot 10^3$)	
	-00	-03
Signed-NTT	127.02	26.48
Unsigned-NTT	158.00	36.75
$\text{RNR}^{\pm}$-NTT	196.01	50.70
RNR^{+}-NTT	260.52	84.74
Signed-NTT^{-1}	202.04	42.61
Unsigned-NTT^{-1}	270.39	64.91
$\text{RNR}^{\pm}$-NTT^{-1}	203.19	42.61
RNR^{+}-NTT^{-1}	305.59	91.15

zeroed coordinates. However, it is unable to significantly reduce the GE in shallower layers due to the large uncertainty, which is a known problem with belief propagation on high entropy distributions [Nag+25].

Discussion. In our attacks, we utilized an adversarial setting highly favorable to the attacker in which they are able to craft ciphertexts to create many zeroed coefficients in the input polynomial to the NTT^{-1}, based on the k-trace attack of Hamburg et al. [Ham+21]. This attack is effective against masked implementations of the NTT^{-1} since shares can be attacked in a divide-and-conquer approach with SPA and the attack can be considered a worst-case scenario for evaluating the security of NTT^{-1} implementations against SPA.

The result of our simulated and experimental SASCAs show that the signed and unsigned implementation of the ML-KEM NTT^{-1} are highly vulnerable to SPA attacks. Based on the analysis of Section 3 and the PI estimation in Figure 4, it is clear that the integer ring used in ML-KEM is small enough relative to the word-size to yield a large amount of information leakage that can be exploited. After applying RNR, the information leakage is reduced substantially – enough to render this worst-case attack ineffective.

Previous literature on SPA countermeasures on the ML-KEM NTT employ combinations of masking and shuffling [ZBT19; Rav+20]. Such countermeasures make it more difficult for an adversary to create templates and shuffling is a known to be effective against SASCA [VGS14]. Ravi et al. proposed several combinations of masking and shuffling to prevent SASCAs on ML-KEM and Dilithium [Rav+20] between butterflies ("course-grained") or within butterflies ("fine-grained"). An evaluation by Hermelink et al. [Her+23b] showed that this countermeasure was effective against SASCAs on the ML-KEM-NTT and re-

quires a more powerful adversary to execute an attack. However, the countermeasure of Ravi et al. comes with a significant performance overhead: They report between 7%-78% performance overhead across all ML-KEM operations on the ARM Cortex-M4 target. In contrast, RNR-ML-KEM achieves a similar margin of security against SPA with lower implementation effort: Only constants in the reduction algorithms and rejection sampling need to be adjusted. Furthermore, RNR can be applied seamlessly on top of masking, if necessary. As masking is generally required to protect against DPA attacks, RNR offers a low-cost yet effective alternative to shuffling. Heinz et al. suggested that RNR on ML-KEM may demask certain coefficients if a ζ is a multiple of the redundancy factor η [HP23]. Our experimental results show that this is not an issue in the context of SPA on the ML-KEM-NTT, even for the small η we chose. For further protections, one can mask the input ciphertexts with randomly sampled integers $r \in [0, \eta)$ to prevent any such demasking. In Table 2 we report the clock cycles needed to compute the ML-KEM-NTT and ML-KEM-NTT^{-1} on the ARM Cortex-M4 target for all four implementations we evaluate. With full optimizations (-O3), applying RNR to the signed NTT incurs a performance overhead of 62.8% (resp. 42.7% for -O0) and, for the unsigned NTT, RNR incurs a 79.0% overhead (resp. 49.0% for -O0). Both RNR-variants require two additional barrett reductions to ensure that intermediate sums do not exceed the 16-bit (signed) integer range. In contrast, the baseline signed and the $\text{RNR}^{\pm}$ NTT^{-1} implementations achieve the same performance since no additional changes are required to the NTT^{-1} other than changes to constants. On the otherhand, the RNR^{+} variant incurs a 27.7% penalty compared to the unsigned counterpart. This is due to a 64-bit multiplication in the Barrett reduction which is a costly operation on the ARM Cortex-M4. Since $\text{RNR}^{\pm}$ suggests less information leakage from the analysis and PI measurements and does not compromise on performance, it is our primary recommendation for ML-KEM.

6 Conclusion

In this paper, we discuss the application of Redundant Number Representation (RNR) to counter SPA and SASCA. We analyzed how information can leak as a result of integer representations and reveal secrets more easily. We then applied these strategies to protect the ML-KEM NTT^{-1}, which deals directly with secrets, and showed that RNR-ML-KEM renders worst-case SASCA attacks ineffective in simulation and in practice on the ARM Cortex-M4. Finally, we show that our RNR implementations achieve comparable performance to state-of-the-art masking/shuffling countermeasures. In particular, we highlight that the signed variant of the RNR-ML-KEM NTT^{-1} can compute with the same performance as an unprotected version. Applying RNR to optimized ML-KEM implementations as well as other schemes remain as directions for future works.

Acknowledgements. This research was funded in whole or in part by the Austrian Science Fund (FWF) (FWF SFB project SPyCoDe 10.55776/F85).

A Modular Reduction Algorithms

Algorithm 4 Unsigned Barrett Reduction

Input: $a \in [0, \beta), R = 2^k$
Output: $x \in [0, q), x \equiv a \bmod q$
1: $v \leftarrow \left\lfloor \frac{R}{q} \right\rceil / R$ ▷ *Pre-computed constant*
2: $t \leftarrow \left\lfloor \frac{a \cdot v}{R} \right\rfloor$
3: **return** $x = a - (t \cdot q)$

Algorithm 5 Signed Barrett Reduction [Sei18]

Input: $a \in [-\frac{\beta}{2}, \frac{\beta}{2}), R = 2^k$
Output: $x \in [-\frac{q}{2}, \frac{q}{2}), x \equiv a \bmod q$
1: $v \leftarrow \left\lfloor \frac{2^{\lfloor \log_2(q) \rfloor - 1} \beta}{q} \right\rceil$ ▷ *Pre-computed constant*
2: $t \leftarrow \left\lfloor \frac{av}{2^{\lfloor \log_2(q) \rfloor - 1} \beta} \right\rfloor$
3: $t \leftarrow tq \bmod \beta$
4: **return** $x = a - t$

Barrett Reduction [Bar86]. The Barrett reduction algorithm efficiently computes the modular reduction without the use of division by approximating the multiplicative inverse of q with a constant that is proportional to a power of two. Let $\beta = 2^l$ be the word-size large enough to fit the prime modulus q; if $q < 2^{16}$ then $\beta = 2^{16}$ and if $q < 2^{32}$ then $\beta = 2^{32}$. The original unsigned Barrett reduction computes $a \bmod q = a - q \cdot \lfloor a \cdot v \rceil$ with the quantity

$$v = \left\lfloor \frac{R}{q} \right\rceil / R,$$

where $R = 2^k$ and k is an arbitrary integer such that v approximates q^{-1}. The signed Barrett reduction [Sei18] computes the symmetric modulus $a \bmod^{\pm} q$ using a tighter approximation of q^{-1} when $0 < q < \frac{\beta}{2}$,

$$v = \left\lfloor \frac{2^{\lfloor \log_2 q \rfloor - 1} \beta}{q} \right\rceil.$$

Montgomery Reduction [Mon85]. The Montgomery modular reduction algorithm is similar to the Barrett reduction in that it computes remainders via approximation, but for a larger domain $x \in [0, q\beta)$. Consider that x is the product of two integers $x = ab$ such that $0 \leq a < q$ and $0 \leq b < q$ and the goal is to compute $x \bmod q$. The algorithm requires the input $x = ab$ to be in the so-called

Algorithm 6 Unsigned Montgomery Reduction

Input: $a \in [0, q\beta), q^{-1} \bmod \beta$ ▷ $q^{-1} \bmod \beta$ *is pre-computed.*
Output: $x \in [0, 2q), x \equiv a\beta^{-1} \bmod q$
1: $t \leftarrow a(-q^{-1}) \bmod \beta$
2: **return** $x \leftarrow (a + tq)/\beta$ ▷ *Division is implemented as right shift.*

Algorithm 7 Signed Montgomery Reduction [Sei18]

Input: $a \in [-\frac{q\beta}{2}, \frac{q\beta}{2})$
Output: $x \in [-q, q), x \equiv a\beta^{-1} \bmod q$
1: $v \leftarrow a(-q^{-1}) \bmod^{\pm} \beta$ ▷ q^{-1} *pre-computed*
2: $t \leftarrow \left\lfloor \frac{vq}{\beta} \right\rfloor$
3: **return** $x = \left\lfloor \frac{a}{\beta} \right\rfloor - t$

Montgomery domain by multiplying the quantity $\beta \bmod q$ making the input to the algorithm $ab\beta \bmod q$. Since the CT and GS butterfly circuits are used to compute the NTT (and its inverse), the second multiplicand b is a power of the root-of-unity ζ and thus $b\beta \bmod q$ can be precomputed. Precomputing these constants in the Montgomery domain yields the output of the reduction in the normal domain, saving additional operations for the translation [Alk+16]. The signed variant(Algorithm 7) [Sei18] returns the signed result $x \in [-q, q)$ with slight modifications.

B NTT Algorithms

Algorithm 8 RNR$^{\pm}$-NTT

Input: $f \in \mathbb{Z}^{256}_{\eta^{\pm}q}$
Output: $\hat{f} \in \mathbb{Z}^{256}_{\eta^{\pm}q}$
1: $\hat{f} \leftarrow f$
2: $k \leftarrow 1; j \leftarrow 0$
3: **for** len $\leftarrow 128;$ len $\geq 2;$ len $\leftarrow$ len$/2$ **do**
4: **for** start $\leftarrow 0;$ start $< 256;$ start $\leftarrow j +$ len **do**
5: **for** $j \leftarrow$ start; $j <$ start + len; $j++$ **do**
6: $t \leftarrow \mathsf{montgomery\_reduce}_{\eta^{\pm}}(\zeta^k \cdot \hat{f}_{j+\text{len}})$
7: ▷ *The additional barrett reductions in the addition and subtraction below are only necessary for larger* $\eta^{\pm}$ ◁
8: $\hat{f}_{j+\text{len}} \leftarrow \mathsf{barrett\_reduce}_{\eta^{\pm}}(\hat{f}_j - t)$
9: $\hat{f}_j \leftarrow \mathsf{barrett\_reduce}_{\eta^{\pm}}(\hat{f}_j + t)$
10: $k \leftarrow k + 1$

Algorithm 9 RNR^{+}-NTT

Input: $f \in \mathbb{Z}_{\eta^+ q}^{256}$
Output: $\hat{f} \in \mathbb{Z}_{\eta^+ q}^{256}$

$\hat{f} \leftarrow f$
$k \leftarrow 1; j \leftarrow 0$
for $\mathrm{len} \leftarrow 128; \mathrm{len} \geq 2; \mathrm{len} \leftarrow \mathrm{len}/2$ **do**
  for $\mathrm{start} \leftarrow 0; \mathrm{start} < 256; \mathrm{start} \leftarrow j + \mathrm{len}$ **do**
    for $j \leftarrow \mathrm{start}; j < \mathrm{start} + \mathrm{len}; j++$ **do**
      $t \leftarrow \mathsf{montgomery\_reduce}_{\eta^+}(\zeta^k \cdot \hat{f}_{j+\mathrm{len}})$
      ▷ *The additional barrett reductions in the addition and subtraction below are only necessary for larger* η^+ ◁
      $\hat{f}_{j+\mathrm{len}} \leftarrow \mathsf{barrett\_reduce}_{\eta^+}((\eta^+ + 1) \cdot q + \hat{f}_j - t)$
      $\hat{f}_j \leftarrow \mathsf{barrett\_reduce}_{\eta^+}(\hat{f}_j + t)$
    $k \leftarrow k + 1$

Algorithm 10 $\mathrm{RNR}^{\pm}$-NTT^{-1}

Input: $\hat{f} \in \mathbb{Z}_{\eta^\pm q}^{256}$
Output: $f \in \mathbb{Z}_{\eta^\pm q}^{256}$

$f \leftarrow \hat{f}$
$k \leftarrow 127; j \leftarrow 0$
for $\mathrm{len} \leftarrow 2; \mathrm{len} \leq 128; \mathrm{len} \leftarrow 2 \cdot \mathrm{len}$ **do**
  for $\mathrm{start} \leftarrow 0; \mathrm{start} < 256; \mathrm{start} \leftarrow j + \mathrm{len}$ **do**
    for $j \leftarrow \mathrm{start}; j < \mathrm{start} + \mathrm{len}; j++$ **do**
      $t \leftarrow f_j$
      $f_j \leftarrow \mathsf{barrett\_reduce}_{\eta^\pm}(t + f_{j+\mathrm{len}})$
      $f_{j+\mathrm{len}} \leftarrow f_{j+\mathrm{len}} - t$
      $f_{j+\mathrm{len}} \leftarrow \mathsf{montgomery\_reduce}_{\eta^\pm}(\zeta^k \cdot f_{j+\mathrm{len}})$
    $k \leftarrow k - 1$

Algorithm 11 RNR^{+}-NTT^{-1}

Input: $\hat{f} \in \mathbb{Z}_{\eta^+ q}^{256}$
Output: $f \in \mathbb{Z}_{\eta^+ q}^{256}$

$f \leftarrow \hat{f}$
$k \leftarrow 127; j \leftarrow 0$
for $\mathrm{len} \leftarrow 2; \mathrm{len} \leq 128; \mathrm{len} \leftarrow 2 \cdot \mathrm{len}$ **do**
  for $\mathrm{start} \leftarrow 0; \mathrm{start} < 256; \mathrm{start} \leftarrow j + \mathrm{len}$ **do**
    for $j \leftarrow \mathrm{start}; j < \mathrm{start} + \mathrm{len}; j++$ **do**
      $t \leftarrow f_j$
      $f_j \leftarrow \mathsf{barrett\_reduce}_{\eta^+}(t + f_{j+\mathrm{len}})$
      ▷ *The additional barrett reduction is only necessary for larger* η^+ ◁
      $f_{j+\mathrm{len}} \leftarrow \mathsf{barrett\_reduce}_{\eta^+}((\eta^+ + 1) \cdot q + f_{j+\mathrm{len}} - t)$
      $f_{j+\mathrm{len}} \leftarrow \mathsf{montgomery\_reduce}_{\eta^+}(\zeta^k \cdot f_{j+\mathrm{len}})$
    $k \leftarrow k - 1$

References

[AER23] Guilhèm Assael, Philippe Elbaz-Vincent, and Guillaume Reymond. "Improving Single-Trace Attacks on the Number-Theoretic Transform for Cortex-M4". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2023, San Jose, CA, USA, May 1-4, 2023*. IEEE, 2023, pp. 111–121. DOI: 10.1109/HOST55118.2023.10133270.

[Alk+16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope". In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. USENIX Association, 2016, pp. 327–343. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

[Ava+21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Kyber algorithm specifications and supporting documentation". In: (2021). URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[Bar86] Paul Barrett. "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor". In: *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*. Springer, 1986, pp. 311–323. DOI: 10.1007/3-540-47721-7_24.

[Bol+19] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. "Order-LWE and the Hardness of Ring-LWE with Entropic Secrets". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*. Springer, 2019, pp. 91–120. DOI: 10.1007/978-3-030-34621-8_4.

[Bos+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 2018, pp. 353–367. DOI: 10.1109/EUROSP.2018.00032.

[Bro+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. "Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*. Springer, 2019, pp. 713–737. DOI: 10.1007/978-3-030-26948-7_25.

[Bro+24] Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. "Exploiting Small-Norm Polynomial Multiplication with Physical Attacks Application to CRYSTALS-Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2024), pp. 359–383. DOI: 10.46586/TCHES.V2024.I2.359-383.

[BS21] Olivier Bronchain and François-Xavier Standaert. "Breaking Masked Implementations with Many Shares on 32-bit Software Platforms or When the Security Order Does Not Matter". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2021), pp. 202–234. DOI: 10.46586/TCHES.V2021.I3.202-234.

[Cas+23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. "Efficient Regression-Based Linear Discriminant Analysis for Side-Channel Security Evaluations Towards Analytical Attacks against 32-bit Implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2023), pp. 270–293. DOI: 10.46586/TCHES.V2023.I3.270-293.

[CB23] Gaëtan Cassiers and Olivier Bronchain. "SCALib: A Side-Channel Analysis Library". In: *Journal of Open Source Software* 8 (2023), p. 5196. DOI: 10.21105/joss.05196.

[Cha+99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Springer, 1999, pp. 398–412. DOI: 10.1007/3-540-48405-1_26.

[CK13] Omar Choudary and Markus G. Kuhn. "Efficient Template Attacks". In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. Springer, 2013, pp. 253–270. DOI: 10.1007/978-3-319-08302-5_17.

[CK14] Marios O. Choudary and Markus G. Kuhn. "Efficient Stochastic Methods: Profiled Attacks Beyond 8 Bits". In: *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*. Springer, 2014, pp. 85–103. DOI: 10.1007/978-3-319-16763-3_6.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Springer, 2002, pp. 13–28. DOI: 10.1007/3-540-36400-5_3.

[CT65] James Cooley and John Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of Computation* 19 (1965), pp. 297–301. URL: http://dx.doi.org/10.1090/S0025-5718-1965-0178586-1.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *J. Cryptol.* 32 (2019), pp. 151–177. DOI: 10.1007/S00145-018-9284-1.

[Duc+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2018), pp. 238–268. DOI: 10.13154/TCHES.V2018.I1.238-268.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *J. Cryptol.* 26 (2013), pp. 80–101. DOI: 10.1007/S00145-011-9114-1.

[GS66] W. M. Gentleman and G. Sande. "Fast Fourier Transforms: for fun and profit". In: *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*. Association for Computing Machinery, 1966, pp. 563–578. DOI: 10.1145/1464291.1464352.

[Ham+21] 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". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2021), pp. 88–113. DOI: 10.46586/TCHES.V2021.I4.88-113.

[Her+23a] 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". In: *IACR Cryptol. ePrint Arch.* (2023), p. 98. URL: https://eprint.iacr.org/2023/098.

[Her+23b] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. "Adapting Belief Propagation to Counter Shuffling of NTTs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2023), pp. 60–88. DOI: 10.46586/TCHES.V2023.I1.60-88.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*. Springer, 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12.

[HME24] Xunyue Hu, Quentin L. Meunier, and Emmanuelle Encrenaz. "Blind-Folded: Simple Power Analysis Attacks using Data with a Single Trace and no Training". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025 (2024), pp. 475–496. DOI: 10.46586/tches.v2025.i1.475-496.

[HP23] Daniel Heinz and Thomas Pöppelmann. "Combined Fault and DPA Protection for Lattice-Based Cryptography". In: *IEEE Trans. Computers* 72 (2023), pp. 1055–1066. DOI: 10.1109/TC.2022.3197073.

[Hua+22] Junhao Huang, Jipeng Zhang, Haosong Zhao, Zhe Liu, Ray C. C. Cheung, Çetin Kaya Koç, and Donglong Chen. "Improved Plantard Arithmetic for Lattice-based Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2022), pp. 614–636. DOI: 10.46586/TCHES.V2022.I4.614-636.

[Hua+24] Junhao Huang, Haosong Zhao, Jipeng Zhang, Wangchen Dai, Lu Zhou, Ray C. C. Cheung, Çetin Kaya Koç, and Donglong Chen. "Yet Another Improvement of Plantard Arithmetic for Faster Kyber on Low-End 32-bit IoT Devices". In: *IEEE Trans. Inf. Forensics Secur.* 19 (2024), pp. 3800–3813. DOI: 10.1109/TIFS.2024.3371369.

[Inc24] NewAE Technology Inc. *Cw308 UFO Target Board*. 2024. URL: https://www.newae.com/products/nae-cw308.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Springer, 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. Springer, 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. "Single-Trace Attacks on Keccak". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2020), pp. 243–268. DOI: 10.13154/TCHES.V2020.I3.243-268.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*. Springer, 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Des. Codes Cryptogr.* 75 (2015), pp. 565–599. DOI: 10.1007/S10623-014-9938-4.

[Mas+23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. "Information Bounds and Convergence Rates for Side-Channel Security Evaluators". In: *IACR Trans. Cryptogr. Hardw.*

Embed. Syst. (2023), pp. 522–569. DOI: 10.46586/TCHES.V2023.I3.522-569.

[MM09] Marc Mezard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, Inc., 2009. ISBN: 019857083X.

[Mon85] Peter L. Montgomery. "Modular multiplication without trial division". In: *Mathematics of Computation* 44 (1985), pp. 519–521.

[Nag+25] Rishub Nagpal, Gaëtan Cassiers, Robert Primas, Christian Knoll, Franz Pernkopf, and Stefan Mangard. "On Loopy Belief Propagation for SASCAs". In: *IACR Communications in Cryptology* 1 (2025). DOI: 10.62056/ayl8ksdja.

[Ngo+21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. "A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2021), pp. 676–707. DOI: 10.46586/TCHES.V2021.I4.676-707.

[Pea82] Judea Pearl. "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach". In: *Proceedings of the Second AAAI Conference on Artificial Intelligence*. AAAI Press, 1982, pp. 133–136. URL: http://dl.acm.org/citation.cfm?id=2876686.2876719.

[PP19] Peter Pessl and Robert Primas. "More Practical Single-Trace Attacks on the Number Theoretic Transform". In: *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*. Springer, 2019, pp. 130–149. DOI: 10.1007/978-3-030-30530-7_7.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. "Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption". In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Springer, 2017, pp. 513–533. DOI: 10.1007/978-3-319-66787-4_25.

[Qia+24] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Yuhan Zhao, and Shuyi Chen. "Single Trace is All It Takes: Efficient Side-channel Attack on Dilithium". In: *IACR Cryptol. ePrint Arch.* (2024), p. 512. URL: https://eprint.iacr.org/2024/512.

[QS01] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards". In: *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*. Springer, 2001, pp. 200–210. DOI: 10.1007/3-540-45418-7_17.

[Rav+20] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. "On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT - A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4". In: *Security, Privacy, and Applied Cryptography Engineering - 10th International*

Conference, SPACE 2020, Kolkata, India, December 17-21, 2020, Proceedings. Springer, 2020, pp. 123–146. DOI: 10.1007/978-3-030-66626-2_7.

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[Ren+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. "A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices". In: *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. Springer, 2011, pp. 109–128. DOI: 10.1007/978-3-642-20465-4_8.

[SA08] François-Xavier Standaert and Cédric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Springer, 2008, pp. 411–425. DOI: 10.1007/978-3-540-85053-3_26.

[Sei18] Gregor Seiler. "Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography". In: *IACR Cryptol. ePrint Arch.* (2018), p. 39. URL: http://eprint.iacr.org/2018/039.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. "A Stochastic Model for Differential Side Channel Cryptanalysis". In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Springer, 2005, pp. 30–46. DOI: 10.1007/11545262_3.

[ST24a] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. Tech. rep. U.S. Department of Commerce, 2024.

[ST24b] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Tech. rep. U.S. Department of Commerce, 2024.

[Str+23] Emanuele Strieder, Manuel Ilg, Johann Heyszl, Florian Unterstein, and Silvan Streit. "ASCA vs. SASCA - A Closer Look at the AES Key Schedule". In: *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023, Munich, Germany, April 3-4, 2023, Proceedings*. Springer, 2023, pp. 65–85. DOI: 10.1007/978-3-031-29497-6_4.

[TMS24] Tolun Tosun, Amir Moradi, and Erkay Savas. "Exploiting the Central Reduction in Lattice-Based Cryptography". In: *IEEE Access* 12 (2024), pp. 166814–166833. DOI: 10.1109/ACCESS.2024.3494593.

[Vey+12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. "Shuffling against Side-Channel At-

tacks: A Comprehensive Study with Cautionary Note". In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*. Springer, 2012, pp. 740–757. DOI: 10.1007/978-3-642-34961-4_44.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. "Soft Analytical Side-Channel Attacks". In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. Springer, 2014, pp. 282–296. DOI: 10.1007/978-3-662-45611-8_15.

[ZBT19] Timo Zijlstra, Karim Bigou, and Arnaud Tisserand. "FPGA Implementation and Comparison of Protections Against SCAs for RLWE". In: *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*. Springer, 2019, pp. 535–555. DOI: 10.1007/978-3-030-35423-7_27.