High-Order Masking of Lattice Signatures in Quasilinear Time

Rafaël del Pino
PQShield SAS, France
rafael.del.pino@pqshield.com

Thomas Prest
PQShield SAS, France
thomas.prest@pqshield.com

Mélissa Rossi
ANSSI, France
melissa.rossi@ssi.gouv.fr

Markku-Juhani O. Saarinen
PQShield LTD, UK
mjos@pqshield.com

***Abstract*—In recent years, lattice-based signature schemes have emerged as the most prominent post-quantum solutions, as illustrated by NIST's selection of Falcon and Dilithium for standardization. Both schemes enjoy good performance characteristics. However, their efficiency dwindles in the presence of side-channel protections, particularly masking – perhaps the strongest generic side-channel countermeasure. Masking at order d-1 requires randomizing all sensitive intermediate variables into d shares. With existing schemes, signature generation complexity grows quadratically with the number of shares, making high-order masking prohibitively slow.**

In this paper, we turn the problem upside-down: We design a lattice-based signature scheme specifically for side-channel resistance and optimize the masked efficiency as a function of the number of shares. Our design avoids costly operations such as conversions between arithmetic and boolean encodings (A2B/B2A), masked rejection sampling, and does not require a masked SHAKE implementation or other symmetric primitives. The resulting scheme is called Raccoon and belongs to the family of Fiat-Shamir with aborts lattice-based signatures. Raccoon is the first lattice-based signature whose key generation and signing running time has only an O(d log(d)) overhead, with d being the number of shares.

Our Reference C implementation confirms that Raccoon's performance is comparable to other state-of-the-art signature schemes, except that increasing the number of shares has a near-linear effect on its latency. We also present an FPGA implementation and perform a physical leakage assessment to verify its basic security properties.

***Index Terms*—Post-Quantum Cryptography, Side-Channel Security, Masking Countermeasures, Raccoon signature.**

1. Introduction

Lattice-based signatures have emerged as prominent post-quantum solutions. They are computationally efficient and backed up by strong mathematical assumptions that are conjectured post-quantum secure. In July 2022, NIST announced [1] that it will standardize two lattice-based signatures: Dilithium [2] and Falcon [3].

The picture is less favorable when it comes to the physical protection of implementations, partially because these schemes have been highly optimized for performance. Such optimizations often induce side-channel weaknesses, as highlighted in several attacks, for example Karabulut and Aysu [4] and Guerreau et al. [5] for Falcon, or Ravi et al. [6] and Marzougui et al. [7] for Dilithium.

Masking is a generic countermeasure that offers provable protection against physical side-channel attacks and is broadly applied in embedded systems. Masking lattice-based signatures with an arbitrarily high number of shares has been achieved for schemes such as GLP [8], qTESLA [9] and BLISS [10], but the performance penalty factor is high (for 2 shares: around $\times 15$ for GLP [11] and $\times 4$ for qTESLA-I [12]; for 4 shares: around $\times 73$ for GLP and $\times 37$ for qTESLA-I). Many works have also tried to mask variants of the NIST standards efficiently, but the performance penalty is still high, see a variant of Dilithium by Migliore et al. [13], or Mitaka by Espitau et al. [14]. Let us present the masking challenges in more detail.

1.1. The challenges of masking lattice signatures

As an illustration, Algorithm 1 presents a prototypical Fiat–Shamir with aborts lattice signature scheme based on Module Learning with Errors (or MLWE, see Definition 1). In this work, we focus on Fiat–Shamir with aborts lattice signature schemes as the requirements can be met with masking-friendly operations in a more natural way compared to the Falcon-like complex Gaussian sampling over lattices that belong in the hash-and-sign paradigm. We assume that the verification key vk is a small-secret MLWE sample $(\mathbf{A}, \mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$.

The blueprint of Algorithm 1 was introduced by Lyubashevsky [15], [16], then refined by Bai and Galbraith [17], and in Dilithium [2]. We now explain why masking such a scheme is challenging.

Masking consists of randomizing any secret-dependent intermediate variable. Each of these secret-dependent intermediate variables, say $\mathbf{x}$, is split into $t+1$ variables $(\mathbf{x}_i)_{0 \leq i \leq t}$ called "shares". The integer t is referred to as the masking order. We define $d = t + 1$ to differentiate the masking order and the number of shares.

The two most deployed types of masking are *arithmetic masking* and *Boolean masking*. Concretely, a sensitive vari-

Algorithm 1 $\mathsf{Sign}(\mathsf{sk}, \mathsf{vk}, \mathsf{msg})$

Require: A signing key $\mathsf{sk} = \mathbf{s}$, a verification key $(\mathbf{A}, \mathbf{t})$, a message msg
Ensure: A signature sig of msg under sk
1: Sample $\mathbf{r}$ uniformly in a small set S
2: $\mathbf{u} := \mathbf{A}\mathbf{r}$
3: $\mathbf{w} := \mathsf{Truncate}(\mathbf{u})$ ▷ Commitment
4: $c := H(\mathbf{w}, \mathsf{msg})$ ▷ Challenge
5: $\mathbf{z} := \mathbf{r} + c \cdot \mathbf{s}$ ▷ Response
6: $\mathbf{y} := \mathbf{A} \cdot \mathbf{z} - c \cdot \mathbf{t}$
7: **if** $\mathsf{CheckCondition}(\mathbf{z}, \mathbf{y}) = \mathsf{False}$ **then**
8: **goto** Line 1 ▷ Rejection sampling
9: **return** $\mathsf{sig} := (c, \mathbf{z})$

able $\mathbf{x}$ is shared in $(\mathbf{x}_i)_{0 \le i \le t}$ such that

$$\mathbf{x} = \mathbf{x}_0 + \cdots + \mathbf{x}_t \bmod q \text{ for arithmetic masking,} \quad (1)$$

$$\mathbf{x} = \mathbf{x}_0 \oplus \cdots \oplus \mathbf{x}_t \text{ for Boolean masking.} \quad (2)$$

We will use Algorithm 1 to examine the many challenges of masking Fiat–Shamir lattice signatures. We first observe that the most natural type of masking for MLWE is arithmetic masking. Indeed, it is compatible with the basic algebraic operations used in these algorithms. On the plus side, the knowledge of $\mathbf{w}$ does not provide any advantage for the attacker under no assumptions (or in some cases under mild assumptions [8, Def. 2 and Th. 1]). So, Line 4 does not need to be masked. The main issue in masking Algorithm 1 boils down to masking non-linear operations:

- Sampling $\mathbf{r}$ from a small set S (Line 1) is challenging in an arithmetic masked form. The most efficient known approach is to sample $\mathbf{r}$ in *Boolean* masked form, then convert the result in arithmetic masked form. This requires so-called mask conversions, see Coron et al. [18], [19], [20]. Despite efficiency improvements since their introduction in 2014, known secure mask conversion algorithms run in time at least $O(d^2)$. See Espitau et al. [8, Alg. 15], Migliore et al. [13, Alg. 13] and Gérard and Rossi [9, §3.2] for concrete instantiations of randomness generation with mask conversions.
  Some schemes replace Line 1 by sampling $\mathbf{r}$ according to a discrete Gaussian. This operation is even harder to mask, and the most masking-friendly approach requires cumulative distribution table look-up and thus even more comparisons are needed. See Barthe et al. [10, §5.3.2] and Gérard and Rossi [9, Alg. 9] for more details.
- $\mathsf{Truncate}$ (Line 3) discards the least significant bits of $\mathbf{u}$. This is straightforwardly expressed as a Boolean circuit, but challenging as an arithmetic circuit. Thus, mask conversions are also employed, which again slows down performance. Such a masked operation with conversions has been introduced by Gérard and Rossi [9, §3.3] Migliore et al. [13, Alg. 14-15].
- $\mathsf{CheckCondition}$ (Line 7) verifies that $(\mathbf{z}, \mathbf{y})$ satisfies some constraints and outputs a boolean value. This step, called *rejection sampling*, is critical for the correctness and zero-knowledge property of the scheme. It is critical to mask this part, since disclosing the output of $\mathsf{CheckCondition}(\mathbf{z}, \mathbf{y})$ typically leaks the "direction" of the private key, which can lead to its recovery.
  In practice, most schemes verify that (a) $\mathbf{z}$ is short, and (b) $\mathbf{y}$ belongs to a set that guarantees $\{\mathsf{Truncate}(\mathbf{y}) = \mathbf{w}\}$. Once again, the most efficient known techniques require expensive mask conversions – it has been performed this way in existing masked designs. See Barthe et al. [8, Alg. 16], [10, §4], Migliore et al. [13, §5.3.3], and Gérard and Rossi [9, Alg. 8] for concrete instantiations of masked rejection sampling.

As explained above, while existing lattice-based signature schemes can be masked, they are not optimized for good masking performance. To improve efficiency, some works have slightly modified the schemes to remove some steps and ease the conversions. Migliore et al. [13] forego Dilithium's NTT-friendly modulus q for a power-of-two modulus that is more amenable to conversions. Espitau et al. [14] changes the Gaussian sampler in Falcon in a simpler way to avoid complex operations that could not be masked. However, the performance penalty factor is still non-linear in the masking order, making it quite unsatisfying and prohibitive for practical use in embedded devices.

1.2. Our contributions

In this work, we study the masking of lattice-based signatures from a different angle. Instead of tweaking existing schemes to improve their masked efficiency, we build from the ground up a lattice-based signature that is tailor-made to be masked efficiently. Our scheme is called Raccoon[1]. In the process, we managed to remove mask conversions from our design, as well as any masked gadget with an overhead higher than quasilinear. As a result, consider the question:

> *"Is it possible to achieve a quasilinear masking overhead with lattice-based signatures?"*

We answer this question positively.

Main contribution. Our main contribution is to propose a framework for building Fiat-Shamir lattice signatures that can be masked at order d with quasilinear overhead $O(d \log d)$. We achieve that by eliminating all the efficiency bottlenecks identified in Algorithm 1 and thus removing the need for mask conversions:

1) We switch from small-secret MLWE to uniform-secret MLWE, a well-known reduction (c.f. Appendix A.1) proves that both problems are equivalent. This trivializes the generation of $\mathbf{s}$ and $\mathbf{r}$ in a masked fashion since they are now uniformly random over the underlying space. This also eliminates the need for checking that $\mathbf{z}$ is short in $\mathsf{CheckCondition}$.
2) We introduce a gadget (called $\mathsf{ApproxShift}$) for $\mathsf{Truncate}(\mathbf{u})$ in a masked fashion in time $O(d \log d)$,

1. We continue the cryptographic "tradition" of naming masking-friendly schemes after masked characters. Indeed, the face of a raccoon looks like it wears a mask.

with one big caveat: the output is only an *approximation* of the actual output of Truncate. Nevertheless, for key generation this is not an issue, and we can replace the expensive masked error generation of $\mathbf{e}$ by ApproxShift.

3) Using ApproxShift to compute the commitment $\mathbf{w}$ (Line 3) is more problematic since its approximate output may break the correctness of the signature scheme. In addition, we still need to compute in masked form whether $\mathbf{y}$ belongs to a certain set which guarantees $\mathsf{Truncate}(\mathbf{y}) = \mathbf{w}$. This is important for correctness, but also for the zero-knowledge property.
It turns out that both issues can be solved simultaneously. As in Dilithium, $\mathsf{Truncate}(\mathbf{y})$ and $\mathbf{w}$ may differ, and a hint $\mathbf{h} = \mathbf{w} - \mathsf{Truncate}(\mathbf{y})$ is sent in the signature to allow the verifier to recompute $\mathbf{w}$. We also accept that a probing adversary may completely learn $\mathbf{y}$; since $\mathbf{w} = c \cdot \mathbf{e}$ plus some noise, this might leak information about the private key. A careful Rényi-divergence based study shows that if we limit the number of signatures per key (in practice, to about 2^{48}), then our scheme remains secure in the presence of this leakage. In doing so, we completely remove the need to perform expensive check conditions on masked values.

By combining all these individual improvements, we obtain a scheme whose masked operations are only refresh gadgets, additions, and multiplication with an unmasked operand and ApproxShift. As a result, the scheme can be masked in quasilinear time.

Security proofs. We complement the design with two security proofs. The first one reduces the black-box security of the signature scheme to the MLWE problem. The second proof ensures the masking security in the probing model by Ishai, Sahai and Wagner [21].

Implementations. Masking is an efficient countermeasure: It is expected that the required number of side-channel observations to mount an attack grows exponentially in relation to the number of shares d, see Chari et al. [22] or Duc et al. [23]. We demonstrate that the performance of Raccoon is practical even at higher masking levels, leading to greater security margins against side-channel attacks.

A 2-share version of the Raccoon C implementation exhibits roughly half of the signing performance of unmasked Dilithium (Table 4). Raccoon masking overhead grows sub-linearly up to $d = 32$ shares (Table 4, Fig. 1.) By comparison, a Dilithium3 implementation was reported to have an increasing $1.37 \times d$, $1.71 \times d$, and $1.91 \times d$ per-share complexity for $d \in \{4, 6, 8\}$ shares when compared to $d = 2$ [24, Table 3].

We also perform a TVLA-style leakage assessment of an RTL implementation of Raccoon-128 with $d = 2$. The implementation is tested on a resource-constrained Artix 7 FPGA platform. Overall, the leakage characteristics, performance, and other implementation metrics demonstrate the feasibility of higher-order lattice signatures in applications such as authentication tokens, SoC platform security, smart cards, and cryptocurrency wallets.

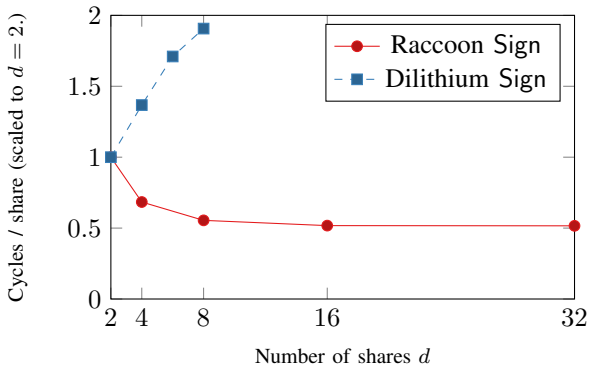

Figure 1. Cost of masking: Signing cycle count divided by d, normalized to a common start at 1 for $d = 2$ (unmasked implementations are not comparable.) For Raccoon, the cost of each additional share plateaus and grows very slowly due to quasilinear $O(d \log d)$ complexity. For Dilithium, each new share needs more resources as its masking complexity is polynomial; close to quadratic $O(d^2)$. Raccoon data is for CRACCOON (Table 4). Dilithium data is for randomized Dilithium3 [24, Table 3]. No masking data is available for Falcon.

1.3. Comparison with Falcon and Dilithium

It is illustrative to compare Raccoon to Dilithium [2], and Falcon [3], the two lattice-based signature schemes selected by NIST for standardization in 2022 [1]. All three schemes are believed to be resistant to attacks with quantum computers due to being based on well-studied lattice assumptions that are conjectured to be quantum-resistant. We provide a security proof in the ROM; in the quantum ROM (or QROM), we expect the proof strategy of Kiltz et al. [25] for Dilithium to be applicable to our scheme.

1.3.1. Dilithium. In terms of cryptanalytic security, Raccoon uses assumptions very close to the ones of Dilithium. Instead of MLWE, Raccoon relies on an assumption we call Module Learning with Rounding and Errors (MLWRE), a hybrid between MLWE and MLWR. Since there is no known gap in concrete security between MLWR and MLWE, it is reasonable to assume that MLWRE is just as hard.

Dilithium was not explicitly designed to facilitate side-channel countermeasures beyond timing attacks. As a result, masking Dilithium requires relatively complex gadgets as reported in works by Migliore et al. [13] and Azouaoui et al. [24]: a masked Keccak (SHAKE) permutation, masked conversions between Arithmetic and Boolean representations (A2B and B2A), masked comparisons, and a masked rejection sampler. Overall, these gadgets make the masking overhead of Dilithium polynomial (almost quadratic) in d (See Fig. 1.) The design of Raccoon avoids these bottlenecks.

1.3.2. Falcon. Cryptanalytic security of Falcon relies on the NTRU assumption. Although this assumption supports fewer security reductions than MLWE, it is widely believed to be as secure as MLWE for well-chosen parameters.

We are not aware of any comparable masked Falcon implementations, even at first order ($d = 2$.) Masking Falcon represents a very significant technical challenge due to Falcon's reliance on floating point (FP) arithmetic; IEEE 754 [26] double-precision arithmetic is needed to manipulate secret variables. Additive or multiplicative masking can not be directly applied to floating point numbers, as the resulting distributions would be biased. To mask Falcon, it seems necessary to develop a masked floating point emulation library with individual (Boolean) masking gadgets for normalization, rounding, and other tasks required for FP multiplication and addition/subtraction. This is a complex engineering task and is likely to cause a high performance (or area) overhead.

1.4. Structure of the paper

In Section 2, we introduce the necessary preliminaries for lattice-based signatures, proof techniques and masking. Next, in Section 3, we introduce our new masking-friendly Raccoon scheme directly in its masked form. Section 4 contains both the black-box security proof of the scheme and the masking proof in the probing model. We present the parameter selection strategy in Section 5. We finish with extensive experimental tests in Section 6.

2. Preliminaries

2.1. Notations

We note scalars, vectors, and matrices in italic (i.e. x), lowercase bold (i.e. $\mathbf{x}$), and uppercase bold (i.e. $\mathbf{X}$), respectively.

For n, q integer parameters, such that $X^n + 1$ is irreducible in $\mathbb{Z}_q$, we define $\mathcal{R}_q = \frac{\mathbb{Z}_q[X]}{X^n+1}$. We use the notation $a \gg k$ for the arithmetic right shift of a by k bits. Finally, we note $\log$ the logarithm in base 2.

2.2. Sets, functions and distributions

Given an integer $n \in \mathbb{N}$, we note $[n]$ the set $\{0, 1, \ldots, n-1\}$. We use the notation $y := f(x)$ (resp. $y \leftarrow f(x)$) to indicate that the function f is deterministic (resp. randomized).

Given a finite set S, we note $x \leftarrow S$ to indicate that x is sampled uniformly at random in S. Finally, $x \leftarrow_{\mathsf{state}} S$ indicates that the object state is used as a source of pseudorandomness in order to select x (pseudo-)uniformly in S. Concretely, state will be a SHAKE256 hash object initialized with an input string.

2.3. Hardness assumptions

Definition 1 (Module Learning With Errors, or MLWE)**.** *Let ℓ, k, q be integers and $\mathcal{D}$ be a probability distribution over $\mathcal{R}_q$. The Module Learning With Errors, noted $\mathsf{MLWE}_{\mathcal{R}_q,\ell,k,\mathcal{D}}$, comes in two variants. For $\mathbf{s} \leftarrow \mathcal{D}^\ell$:*

- ***Decisional:*** *distinguish between $(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{b})$, where $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$, $\mathbf{e} \leftarrow \mathcal{D}^k$ and $\mathbf{b} \leftarrow \mathcal{R}_q^k$.*
- ***Computational:*** *recover $\mathbf{s}$ from $(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e})$, where $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ and $\mathbf{e} \leftarrow \mathcal{D}^k$.*

In this work, we rely on a less used variant of MLWE *called uniform secret* MLWE *(*UMLWE*). Its definition is identical to* MLWE*'s, except that $\mathbf{s}$ can be any vector in $\mathcal{R}_q^{k'}$ instead of $\mathbf{s} \leftarrow \mathcal{D}^k$, with $k' = k + \ell$.*

There is a well known reduction from $\mathsf{UMLWE}_{\mathcal{R}_q,\ell,k+\ell,\chi}$ to $\mathsf{MLWE}_{\mathcal{R}_q,\ell,k,\chi}$ given in Applebaum et al. [27], we are however concerned with the other direction since we want to rely on the well studied security of MLWE. We give the reduction from MLWE to UMLWE in Section A.1 as it is very close to the one of [27].

The security reduction will be under a hybrid problem between MLWE and MLWR, which is at least as hard as the harder of the two problems. However, for concrete parameters we will consider the best known attacks for such a problem which will be to consider it as an MLWE problem with a larger distribution since the best known attacks against MLWR consist in solving a related MLWE instance.

Definition 2. Hybrid problem MLWRE. *Let $\ell, k, q, t, q' = q \gg t$ be integers and $\mathcal{D}$ be a probability distribution over $\mathcal{R}_{q'}$. The Module Learning With Rounding and Errors, noted $\mathsf{MLWRE}_{\mathcal{R}_q,\ell,k,t,\mathcal{D}}$, comes in two variants. For $\mathbf{s} \leftarrow \mathcal{R}_q^\ell$:*

- ***Decisional:*** *distinguish between $(\mathbf{A}, ((\mathbf{A}\cdot\mathbf{s}) \gg t)+\mathbf{e})$ and $(\mathbf{A}, \mathbf{b})$, where $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ and $\mathbf{e} \leftarrow \mathcal{D}^k$ and $\mathbf{b} \leftarrow \mathcal{R}_{q'}^k$.*
- ***Computational:*** *recover $\mathbf{s}$ from $(\mathbf{A}, ((\mathbf{A}\cdot\mathbf{s}) \gg t)+\mathbf{e})$, where $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$ and $\mathbf{e} \leftarrow \mathcal{D}^k$.*

Definition 3. MSIS. *Let ℓ, k, q be integers and $\beta > 0$ a real number. The Module Short Integer Solution $\mathsf{MSIS}_{q,\ell,k,\beta}$ is as follows: Given $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times\ell}$, find $\mathbf{s} \in \mathcal{R}^\ell$ such that*

$$\|\mathbf{s}\|_2 \leq \beta \wedge \mathbf{A}\cdot\mathbf{s} = 0 \mod q$$

2.4. Rényi divergence

The Rényi divergence [28] is a tool from information theory which has recently found many applications in lattice-based cryptography, see Bai et al. [29] and Prest [30]. We use the "exponential form" of the Rényi divergence, as is common in lattice-based cryptography.

Definition 4 (Rényi divergence)**.** *Let $\mathcal{P}, \mathcal{Q}$ be two discrete distributions such that $\operatorname{Supp}\mathcal{P} \subseteq \operatorname{Supp}\mathcal{Q}$, and $\alpha \in (1; +\infty)$. The Rényi divergence of order α is:*

$$R_\alpha(\mathcal{P}; \mathcal{Q}) = \left(\sum_{x\in X} \frac{\mathcal{P}(x)^\alpha}{\mathcal{Q}(x)^{\alpha-1}}\right)^{\frac{1}{\alpha-1}}$$

Following Csiszár's f-divergence framework [31], $R_\alpha^{\alpha-1} - 1$ is an f-divergence for $f : x \mapsto x^\alpha - 1$. Lemma 1 presents some properties of the Rényi divergence; proofs can be found in van Erven and Harremoës [32] or Bai et al.

[29]. Note that (R1) is a generic property of f-divergences, and it implies (R2).

Lemma 1. *For two distributions $\mathcal{P}, \mathcal{Q}$ and two finite families of distributions $(\mathcal{P}_i)_{i\in[n]}, (\mathcal{Q}_i)_{i\in[n]}$, the Rényi divergence verifies these properties:*

(R1) ***Data processing inequality.*** *For a (randomized) function f,*

$$R_\alpha(f(\mathcal{P}); f(\mathcal{Q})) \leq R_\alpha(\mathcal{P}; \mathcal{Q}).$$

(R2) ***Probability preservation.*** *For any event $E \subseteq \mathrm{Supp}(\mathcal{Q})$:*

$$\begin{aligned}\mathcal{P}(E) &\leq (\mathcal{Q}(E) \cdot R_\alpha(\mathcal{P}; \mathcal{Q}))^{\frac{\alpha-1}{\alpha}} \\ &\leq \mathcal{Q}(E)^{\frac{\alpha-1}{\alpha}} \cdot R_\alpha(\mathcal{P}; \mathcal{Q}),\end{aligned}$$

(R3) ***Multiplicativity.*** $R_\alpha(\prod_i \mathcal{P}_i; \prod_i \mathcal{Q}_i) = \prod_i R_\alpha(\mathcal{P}_i; \mathcal{Q}_i)$.

2.5. Masking

In this paper, the masking will be of only one type: *arithmetic masking*. A d-shared variable $(\mathbf{x}_i)_{0\leq i\leq d-1}$ will be denoted $[\![\mathbf{x}]\!]$ for readability. In some parts of the paper, the number of shares may be $d' \neq d$, in that case, we will speficy it in the index as $[\![\mathbf{x}]\!]_{d'}$.

In other words, a sensitive variable $\mathbf{x}$ belonging in a finite abelian group is shared in $[\![\mathbf{x}]\!] = (\mathbf{x}_i)_{i\in[d]}$ following Eq. (1). Note that for a masking order t, there are $d = t + 1$ shares.

Definition 5 (t-probing Security or ISW security)**.** *An algorithm is t-probing secure iff the joint distribution of any set of at most t internal intermediate values is independent of the secrets.*

Masking is one of the most deployed side-channel countermeasures. Indeed, the theoretical model reductions of Duc et al. [33] relates the t-probing security to side-channel security up to a certain level of noise. Moreover, it has been confirmed in practice: the number of measurements required to mount a successful side-channel attack usually increases exponentially in the masking order, see Chari et al. [22] and Duc et al. [23].

Although the t-probing security seems straightforward on linear operations on $\mathbb{Z}_q$, proving it on non-linear operations is much more complicated. The mixture of shares to compute the final result makes it often mandatory to introduce random variables and the bigger the algorithm is, the more dependencies to be considered. To resolve this problem, extra security properties that are slightly stronger than t-probing have been introduced in the literature. They are applied to small sub-algorithms called *gadgets*. More precisely, a gadget is a probabilistic algorithm that takes shared and un-shared inputs values and returns shared and un-shared values. These new security properties open the door for composing the security of gadgets, see Barthe et al. [34].

Definition 6 (Non interference [34])**.** *A gadget is:*

- *t-non-interfering (t-NI) iff any set of at most t observations can be perfectly simulated from at most t shares of each input.*
- *t-strong non-interfering (t-SNI) iff any set of at most t observations whose t_{int} observations on the internal data and t_{out} observations on the outputs can be perfectly simulated from at most t_{int} shares of each input.*

It is easy to check that t-SNI implies t-NI, which implies t-probing security. Also note that any linear gadget for $\mathbb{Z}_q$ is immediately t-NI. An additional notion was introduced by Barthe et al. [8] to reason on the security of lattices-based schemes in which some intermediate variables may be revealed to the adversary.

Definition 7. *A gadget with public outputs X is t-non-interfering with public outputs (t-NIo) iff every set of at most t intermediate variables can be perfectly simulated with the public outputs X and at most t shares of each input.*

Definition 8. *We introduce a* Decode *gadget that takes $[\![\mathbf{x}]\!] = (\mathbf{x}_i)_{i\in[t+1]}$ as input, refreshes it with Algorithm 4, then computes the sum $\mathbf{x}_0 + \cdots + \mathbf{x}_t \bmod q$. It is identical to the* FullAdd *algorithm in [8, Alg. 16].*

Conversely, we define the Encode *gadget that takes an unmasked value $\mathbf{x}$ as input and outputs the shares by generating the first t shares uniformly at random and fixing $\mathbf{x}_t$ as $\mathbf{x} - \mathbf{x}_0 - \cdots - \mathbf{x}_{t-1} \bmod q$.*

3. The Raccoon Signature Scheme

3.1. Additional notations

Let $q \in \mathbb{N}$ and $p_\mathbf{t}, p_\mathbf{w} < q$ be powers-of-two: $p_\mathbf{t} = 2^{\kappa_\mathbf{t}}$ and $p_\mathbf{w} = 2^{\kappa_\mathbf{w}}$ for $\kappa_\mathbf{t}, \kappa_\mathbf{w} \in \mathbb{N}$. We note $q_\mathbf{t} = \lfloor q/p_\mathbf{t} \rfloor$ and $q_\mathbf{w} = \lfloor q/p_\mathbf{w} \rfloor$. Observe that $q \approx q_\mathbf{t} \cdot p_\mathbf{t} \approx q_\mathbf{w} \cdot p_\mathbf{w}$.

For $x \in \mathbb{Z}_q$, we also note $\lfloor x \rfloor_{q\to q_\mathbf{t}} = \lfloor x/p_\mathbf{t} \rfloor$, and similarly $\lfloor x \rfloor_{q\to q_\mathbf{w}} = \lfloor x/p_\mathbf{w} \rfloor$.

3.2. Subroutines ApproxShift, OrderSwitch and Refresh

We present three subroutines that are integral to the efficiency and provable security of Raccoon.

ApproxShift. A first subroutine is ApproxShift (Algorithm 2), which performs the arithmetic right shift $\gg$ on masked integers in an approximate manner. The subroutine is extremely simple to implement and prove secure. As it manipulates the inputs share by share without adding any randomness, ApproxShift is $t-$NI.

Algorithm 2 $\mathsf{ApproxShift}_{q\to q'}()$

Require: $[\![x]\!] = (x_i)_{i\in[d]} \in \mathbb{Z}_q^d$, an integer $q' = q \gg k$ for $k \in \mathbb{N}$, a precomputed value $\delta = \delta(q, d, k) \in \mathbb_q$
Ensure: $[\![y]\!] \in \mathbb{Z}_{q'}^d$
1: **for** $i \in [d]$ **do**
2: $\quad y_i := (x_i + \delta) \gg k$
3: **return** $[\![y]\!] = (y_i)_{i\in[d]}$

Algorithm 2 requires one to precompute a value $\delta = \delta(q, d, k)$. Lemma 2 shows how to set to set δ in a way that bounds the output error of Algorithm 2.

Lemma 2. *Let* $[\![x]\!] \in \mathbb{Z}_q^d, q' = q \gg k$, $[\![y]\!] = \mathsf{ApproxShift}_{q\to q'}([\![x]\!])$, $x := \mathsf{Decode}([\![x]\!])$, $y := \mathsf{Decode}([\![y]\!])$ *and* $y^* = x \gg k$. *If* $d \mid 2^{k-1}$ *and* $\delta = \frac{(d-1)2^{k-1}}{d}$, *it holds that:*

$$|y - y^*| \leq \left\lceil \frac{d+1}{2} \right\rceil . \tag{3}$$

The proof of Lemma 2 is elementary and is provided in Section A.2. In practice, the output error of Algorithm 2 is much better than predicted by Lemma 2. The average error observed in practice grows in $O(\sqrt{d})$, which is easily explained by the fact that each individual share independently adds an error $O(1)$ centered on zero.

OrderSwitch. The OrderSwitch subroutine (Algorithm 3), converts a d_1-encoding of a value $x \in \mathbb{Z}_q$ into a d_2-encoding of the same value, for $d_2 \geq d_1$. The goal of this operation is to reduce the amount of information that an adversary learns by probing $n^{\text{prob}} < d_1$ values. In particular, it will become clear in Section 3.3 that combining Algorithms 2 and 3 provides better security guarantees than applying Algorithm 2 alone. We will show in Lemma 4 that OrderSwitch is $t-$SNI. In practice, we use $d_2 = 2d_1$.

Algorithm 3 $\mathsf{OrderSwitch}_{d_1\to d_2}([\![x]\!]_{d_1})$

Require: Two integers $d_1, d_2 \in \mathbb{N}$ with $d_2 \geq d_1$, a d_1-sharing of a value $x \in \mathbb{Z}_q$, denoted $[\![x]\!]_{d_1}$
Ensure: a d_2-sharing $[\![x]\!]_{d_2}$ of the same value $x \in \mathbb{Z}_q$
1: $[\![z]\!]_{d_2-d_1} := (0)_{i\in[d_2-d_1]}$
2: $[\![x]\!]_{d_2} = ([\![x]\!]_{d_1} \| [\![z]\!]_{d_2-d_1})$
3: $[\![x]\!]_{d_2} \leftarrow \mathsf{Refresh}([\![x]\!]_{d_2})$
4: **return** $[\![x]\!]_{d_2}$

Refresh. Finally, we require an efficient Refresh procedure as a subroutine of OrderSwitch (Algorithm 3) and Sign (Algorithm 7). This Refresh procedure is functionally equivalent to the identity function: the unmasked input value is equal to the unmasked output value. But, the shares are renewed in a way that the gadget enjoys the t-SNI property, which is necessary in the masking composition proofs.

We use an evolution of a refresh gadget described by Battistelo et al. [35], which appears independently in works by Mathieu-Mahias [36] and Goudarzi et al. [37]. It was proven SNI in [36]. Its time and randomness complexity are $O(d \log d)$.

Algorithm 4 $\mathsf{Refresh}()$

Require: A d-shared $[\![x]\!]$ of $x \in \mathbb{Z}_q$
Ensure: A fresh d-shared $[\![x]\!]$ of x
1: $[\![z]\!] \leftarrow \mathsf{ZeroEncoding}()$
2: **return** $[\![x]\!] = [\![x]\!] + [\![z]\!]$

Algorithm 5 $\mathsf{ZeroEncoding}()$

Require: A power-of-two integer d, a ring $\mathbb{Z}_q$
Ensure: A d-shared $[\![z]\!] \in \mathbb{Z}_q^d$ of $0 \in \mathbb{Z}_q$
1: **if** $d = 1$ **then**
2: **return** $[\![z_1]\!] = (0)$ ▷ Masking order zero.
3: $[\![z_1]\!]_{d/2} \leftarrow \mathsf{ZeroEncoding}(d/2)$
4: $[\![z_2]\!]_{d/2} \leftarrow \mathsf{ZeroEncoding}(d/2)$
5: $[\![r]\!]_{d/2} \leftarrow \mathbb{Z}_q^{d/2}$ ▷ Uniform random vector.
6: $[\![z_1]\!]_{d/2} = [\![z_1]\!]_{d/2} + [\![r]\!]_{d/2}$
7: $[\![z_2]\!]_{d/2} = [\![z_2]\!]_{d/2} - [\![r]\!]_{d/2}$
8: **return** $[\![z]\!]_d = \left([\![z_1]\!]_{d/2} \| [\![z_2]\!]_{d/2}\right)$ ▷ Concatenate.

3.3. Key generation

Recall that we use the notation $[\![x]\!]$ to refer to a sharing of x. The number of shares is always d except if indicated otherwise.

Algorithm 6 describes the key generation of Raccoon, in which sensitive variables are masked with (at least) d shares. Intuitively, one can think of Algorithm 6 as generating d independent UMLWR samples, briefly expanding them into $2d$ shares, then collapsing them into a verification key.

First, consider a simplified variant of Algorithm 6: (a) A public matrix $\mathbf{A}$ is uniformly generated, (b) d secret key shares $\mathbf{s}_i$ are generated in parallel, (c) d rounded products $\mathbf{t}_i := (\mathbf{A}\mathbf{s}_i + \boldsymbol{\delta}) \gg k$ are computed, and finally (d) the private key is defined as $[\![\mathbf{s}]\!]$ and the public key is computed as $(\mathbf{A}, \mathbf{t} = \sum_{i\in[d]} \mathbf{t}_i)$. Note that each $(\mathbf{A}, \mathbf{t}_i)$ is an UMLWR sample and $(\mathbf{A}, \mathbf{t})$ is also such a sample with a larger noise. Intuitively, the optimal strategy for a t-probing adversary $\mathcal{A}$ wishing to infer a coefficient of $\mathbf{A} \cdot \mathbf{s}$ is to probe the corresponding coefficients of $\mathbf{A} \cdot \mathbf{s}_i$ and compute the corresponding coefficient of $\mathbf{t} - \sum_{i\in[t]} \mathbf{A} \cdot \mathbf{s}_i$. As long as $n^{\text{prob}} < d$, at least one share $\mathbf{t}_i$ is pseudorandom under the UMLWR assumption.

For proving such a key generation, one would need to assume that a t-probing adversary can decrease by a factor $\sqrt{d}$ the error rate in the LWE/LWR problem and thus adapting the choice of parameters for preventing any attack. Adding a temporary increase of the number of shares in Line 4 prevents this degradation; in its presence, if an adversary probes $n^{\text{prob}} < d$ values, at least $2d - n^{\text{prob}}$ shares $\mathbf{t}_i$ remain pseudorandom under the LWR assumption, and the error rate is only decreased by a factor $\sqrt{\frac{2d}{2d-\delta}} < \sqrt{2}$. This gain of a factor $O(\sqrt{d})$ allows one to select more efficient parameters.

3.4. Signing procedure

The signing procedure follows the usual commit-challenge-response flow, and we are able to perform all

2. Following the standard practice in MLWE/MLWR schemes [2], [38], $\mathbf{A}$ is actually generated in a pseudorandom manner from a public seed. This has no influence on security and, for simplicity, is omitted in our description.

Algorithm 6 Keygen()

Require: $\emptyset$
Ensure: A signature keypair $([\![\mathsf{sk}]\!], \mathsf{vk})$, where $[\![\mathsf{sk}]\!]$ is an order-d sharing of sk

1: $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$ ▷ See Footnote 2
2: $[\![\mathbf{s}]\!] \leftarrow (\mathcal{R}_q^{\ell})^d$
3: $[\![\mathbf{u}]\!] := \mathbf{A} \cdot [\![\mathbf{s}]\!]$
4: $[\![\mathbf{v}]\!]_{2d} \leftarrow \mathsf{OrderSwitch}_{d \to 2d}([\![\mathbf{u}]\!])$
5: $[\![\mathbf{t}]\!]_{2d} := \mathsf{ApproxShift}_{q \to q_{\mathbf{t}}}([\![\mathbf{v}]\!]_{2d})$
6: $\mathbf{t} := \mathsf{Decode}([\![\mathbf{t}]\!]_{2d})$ ▷ $\mathbf{t} \in \mathcal{R}_{q_{\mathbf{t}}}^k$
7: **return** $([\![\mathsf{sk}]\!] := [\![\mathbf{s}]\!], \mathsf{vk} := (\mathbf{A}, \mathbf{t}))$

sensitive operations in a masked fashion.

Sensitive components. Let us first discuss which components need to be masked, and which don't.

- Operations involving $\mathbf{r}$ need to be masked. This involves computing the commitment $\mathbf{w}$ and the response $\mathbf{z}$, although $\mathbf{w}$ and $\mathbf{z}$ are revealed as part of the signing procedure.
- On the other hand, the challenge and its computation do not need to be masked. Note that for Fiat-Shamir *with aborts* schemes, this may require some additional assumptions due to the fact that some signatures are not supposed to be output, see Barthe et al. [8, Def. 2 and Th. 1]. We do not require such an assumption; as explained in Remark 1, $\mathbf{w}$ can be recomputed from public information.

Algorithm 7 Sign($[\![\mathsf{sk}]\!], \mathsf{vk}, \mathsf{msg}$)

Require: A masked signing key $[\![\mathsf{sk}]\!]$, a message msg
Ensure: A signature sig of msg under sk

1: $[\![\mathbf{r}]\!] \leftarrow (\mathcal{R}_q^{\ell})^d$ ▷ Masked ephemeral secret
2: $[\![\mathbf{u}]\!] := \mathbf{A} \cdot [\![\mathbf{r}]\!]$
3: $[\![\mathbf{u}]\!] \leftarrow \mathsf{Refresh}([\![\mathbf{u}]\!])$ ▷ Alg. 4
4: $[\![\mathbf{w}]\!] := \mathsf{ApproxShift}_{q \to q_{\mathbf{w}}}([\![\mathbf{u}]\!])$ ▷ Alg. 2
5: $\mathbf{w} := \mathsf{Decode}([\![\mathbf{w}]\!])$ ▷ Commitment, Def. 8
6: $c_{\mathsf{hash}} := H(\mathbf{w}, \mathsf{msg})$ ▷ Challenge hash
7: $c_{\mathsf{poly}} := \mathsf{ChalPoly}(c_{\mathsf{hash}})$ ▷ Challenge. Alg. 8
8: $[\![\mathbf{s}]\!] \leftarrow \mathsf{Refresh}([\![\mathbf{s}]\!])$ ▷ Alg. 4
9: $[\![\mathbf{r}]\!] \leftarrow \mathsf{Refresh}([\![\mathbf{r}]\!])$ ▷ Alg. 4
10: $[\![\mathbf{z}]\!] := c_{\mathsf{poly}} \cdot [\![\mathbf{s}]\!] + [\![\mathbf{r}]\!]$
11: $\mathbf{z} := \mathsf{Decode}([\![\mathbf{z}]\!])$ ▷ Response, Def. 8
12: $\mathbf{y} := \mathbf{A} \cdot \mathbf{z} - p_{\mathbf{t}} \cdot c_{\mathsf{poly}} \cdot \mathbf{t}$
13: $\mathbf{y}^{\mathsf{top}} := \lfloor \mathbf{y} \rceil_{q \to q_{\mathbf{w}}}$
14: $\mathbf{h} := \mathbf{w} - \mathbf{y}^{\mathsf{top}}$ ▷ Hint
15: **if** $(\|\mathbf{h}\|_2 > B_2)$ **or** $(\|\mathbf{h}\|_\infty > B_\infty)$ **then**
16: **goto** Line 1 ▷ Check the hint's norms
17: **return** $\mathsf{sig} := (c_{\mathsf{hash}}, \mathbf{z}, \mathbf{h})$

Masking the signing algorithm. The signing procedure is presented in Algorithm 7. We explain the intuition of (a) how we can mask all relevant operations efficiently, and (b) why our scheme is secure. The full proofs will be detailed in Section 4.

- Since the ephemeral randomness $\mathbf{r}$ is uniformly random, it is straightforward to generate in masked form (Line 1). Similarly, computing $[\![\mathbf{u}]\!]$ is a linear operation (Line 2) and can be done with computational overhead $O(d)$.
- The commitment $\mathbf{w}$ needs to be computed in masked form. In Dilithium [2] and related schemes, this is done by rounding (or equivalently, shifting) the coefficients of $\mathbf{u}$. With known techniques, masking this operation incurs an overhead at least $O(d^2)$, see Coron et al. [39, §5].
  Instead, we approximately shift the coefficients of $\mathbf{u}$ using ApproxShift (Line 4). This has the major advantage of incurring an overhead $O(d)$. The drawback is that this adds a small error term to $\mathbf{w}$, which breaks the correctness of the scheme. We discuss below how this is solved.

The hint $\mathbf{h}$. We repair correctness by computing a hint $\mathbf{h}$ (Line 14) which encodes the difference between the commitment $\mathbf{w}$ computed by the signer, and the commitment $\mathbf{y}^{\mathsf{top}}$ the verifier initially computes. This hint $\mathbf{h}$ is output as part of the signature, and it allows the verifier to recompute $\mathbf{w}$.

However, if $\mathbf{h}$ is left unconstrained, then an adversary $\mathcal{A}$ could forge a signature by $\mathcal{A}$ first generating $\mathbf{w}$ and $\mathbf{z}$ randomly, then compute c_{poly}, $\mathbf{y}^{\mathsf{top}}$ and $\mathbf{h}$ honestly.

As in Dilithium, we prevent this type of attack by imposing bounds on the L_2 and L_∞ norms of $\mathbf{h}$, enabling a clean security reduction to MSIS and a variant of UMLWE.

Number of queries. Compared to similar schemes like Dilithium [2] and qTESLA [40], Raccoon has no masked rejection sampling. This makes the implementation much simpler, but it also means that signatures can leak information about the private key. Indeed, if we note $\mathbf{w} = \mathbf{u} + \mathbf{e}_{\mathbf{w}}$, $\mathbf{y}^{\mathsf{top}} = \mathbf{y} + \mathbf{e}_{\mathbf{y}}$ and $p_{\mathbf{t}}\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_{\mathbf{t}}$, then one can check that:

$$\mathbf{h} = \mathbf{e}_{\mathbf{w}} - \mathbf{e}_{\mathbf{y}} - c_{\mathsf{poly}} \cdot \mathbf{e}_{\mathbf{t}} \tag{4}$$

There is therefore a correlation between $\mathbf{h}$ and the signing key $\mathbf{s}$. In Section 4.1, we show via a Rényi divergence argument that as long as Q_s is upper bounded by a function of Raccoon's parameters, the leakage provoked by $\mathbf{h}$ does not weaken the security of Raccoon. In Section 5.3, we provide an explicit formula (with some heuristics) for setting Q_s.

Challenge computation. The challenge c_{poly} is selected uniformly in the subset $\mathcal{C} \subsetneq R$ of polynomials such that $\|c_{\mathsf{poly}}\|_\infty = 1$ and $\|c_{\mathsf{poly}}\|_1 = \omega$. As in Dilithium [2], we separate the challenge computation in two parts:

- The challenge hash c_{hash} is computing via a hash function $H : \{0,1\}^* \to \{0,1\}^{\lambda_{\text{TARGET}}}$;
- The challenge polynomial c_{poly} is derived from c_{hash} through an unmasked function ChalPoly (Algorithm 8). Note that Algorithm 8 is not constant-time, but this is not an issue since it is always performed on public data.

This two-step description is convenient from an implementation point of view, since the signing procedure

Algorithm 8 $\mathsf{ChalPoly}(c_{\mathsf{hash}})$

Require: A hash digest $c_{\mathsf{hash}} \in \{0,1\}^\lambda$
Ensure: A polynomial $c_{\mathsf{poly}} \in \mathcal{C}$
1: $\mathsf{shake} := \mathsf{SHAKE256.new}()$
2: $\mathsf{shake} := \mathsf{SHAKE256.update}(c_{\mathsf{hash}})$
3: $\mathbf{c} = (c_i)_{i\in[n]} := 0^n$
4: **while** $\|\mathbf{c}\|_1 \leq \omega$ **do**
5: $\quad (i,b) \leftarrow_{\mathsf{shake}} [n] \times \{0,1\}$
6: $\quad$ **if** $(c_i = 0)$ **then**
7: $\quad\quad c_i = (-1)^b$
8: **return** $c_{\mathsf{poly}} = \sum_{i\in[n]} c_i x^n$

computes c_{hash} then c_{poly}, whereas the verification procedure computes c_{poly} then c_{hash}.

3.5. Verification procedure

Compared to the signing procedure, the verification procedure (Algorithm 9) is straightforward since all operations are performed unmasked. As usual in Fiat-Shamir schemes, the verifier recomputes the challenge and checks it against the one in the signature, helped in our case by the hint $\mathbf{h}$. In addition, the verifier checks the L_2 and L_∞ norms of the hint.

Algorithm 9 $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Require: A verification key vk, a message msg, a signature $\mathsf{sig} = (c_{\mathsf{hash}}, \mathbf{z}, \mathbf{h})$
Ensure: **accept** or **reject**
1: $c_{\mathsf{poly}} := \mathsf{ChalPoly}(c_{\mathsf{hash}})$
2: $\mathbf{y} = \mathbf{A}\cdot\mathbf{z} - p_{\mathbf{t}} \cdot c_{\mathsf{poly}} \cdot \mathbf{t}$
3: $\mathbf{w}' = \lfloor \mathbf{y} \rceil_{q\to q_{\mathbf{w}}} + \mathbf{h}$
4: $c_{\mathsf{hash}}{}' := H(\mathbf{w}', \mathsf{msg})$
5: **if** $(\|\mathbf{h}\|_2 > B_2)$ **or** $(\|\mathbf{h}\|_\infty > B_\infty)$ **or** $(c_{\mathsf{hash}} \neq c_{\mathsf{hash}}{}')$ **then**
6: $\quad$ **reject**
7: **accept**

4. Security Proof

4.1. Security reduction

In this section we prove the unforgeability of our signature scheme. Given that we prove in Section 4.2 that the signing protocol is t-NI we can assume that the adversary is only given the output of the signing oracle (as the t-NI property guarantees that any set of at most t internal probes can be simulated with at most t shares of each input). For our proof, we introduce $\mathsf{ApproxShift}'_{q\to q'}$, which replicates in unmasked form the functionality of $\mathsf{ApproxShift}_{q\to q'}$:

$$\mathsf{ApproxShift}'_{q\to q'} = \mathsf{Decode} \circ \mathsf{ApproxShift}_{q\to q} \circ \mathsf{Encode} \quad (5)$$

The security proof of Raccoon is given in Lemma 3.

Lemma 3. *Let $\mathcal{A}$ be a forger with advantage $\mathsf{Adv}^{\mathsf{Sign}}_{\mathcal{A}}$, making Q_s signing queries. Let $\mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}$ be the advantage of an adversary in distinguishing the public key from uniform. Then:*

$$\frac{\mathsf{Adv}^{\mathsf{Sign}}_{\mathcal{A}}}{Q_s} \leq 2\left(\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}} + \mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}\right) \frac{R^{Q_s}_\alpha(\mathcal{D}_4;\mathcal{D}_5)}{Q_s} + \frac{1}{q^{\ell n}_{\mathbf{w}}}, \quad (6)$$

where $\mathcal{D}_i$ is the distribution of the quadruple $(c_{\mathsf{poly}}, \mathbf{z}, \mathbf{h}, \mathbf{w})$ output by Hybrid_i of Fig. 2, and $\alpha = -\log_2\left(\mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}}\right)$.

Proof. We use a series of hybrids as defined in Fig. 2, with Hybrid_0 corresponding to the unforgeability security game.

Hybrid_0**:** This hybrid is the signing protocol

$$\mathsf{Adv}^{\mathsf{Hybrid}_0}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Sign}}_{\mathcal{A}}$$

Hybrid_1**:** This hybrid is just a rewriting of the signing protocol where we assume the t-NI property. We also consider the security of the scheme in the random oracle model (ROM) and hence replace the challenge computation with a random oracle H. Observe that the output of this hybrid has the same distribution as the signing protocol.

$$\mathsf{Adv}^{\mathsf{Hybrid}_1}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Hybrid}_0}_{\mathcal{A}}$$

Hybrid_2**:** In this hybrid we replace the hash function with a random element and program the hash function to said element. The games are indistinguishable except if the signing algorithm has to program a value that was already queried by the adversary.Using the min-entropy of $\mathbf{w}$ we get:

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_1}_{\mathcal{A}}\right| \leq Q_s \cdot \frac{1}{q^{\ell n}_{\mathbf{w}}}$$

Hybrid_3**:** One can check that $\mathbf{y} + \mathbf{z}' = \mathbf{A}\mathbf{r}$. Therefore this hybrid is simply a rewriting of the previous one.

$$\mathsf{Adv}^{\mathsf{Hybrid}_3}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}}$$

Hybrid_4**:** In this hybrid we sample $\mathbf{z}$ uniformly. Note that the distribution of $\mathbf{z}$ is unchanged since $\mathbf{z} = c_{\mathsf{poly}}\cdot\mathbf{s} + \mathbf{r}$ and $\mathbf{r}$ is uniform.

$$\mathsf{Adv}^{\mathsf{Hybrid}_4}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Hybrid}_3}_{\mathcal{A}}$$

Hybrid_5**:** In this game we set $\mathbf{z}' := 0$. This part of the proof uses a Rényi divergence argument. For $i \in \{4,5\}$, let $\epsilon_i = \mathsf{Adv}^{\mathsf{Hybrid}_i}_{\mathcal{A}}$. It follows from Lemma 1 (more precisely, Item (R1) then Item (R3)) that:

$$\epsilon_4 \leq \epsilon_5^{\frac{\alpha-1}{\alpha}} \cdot R^{Q_s}_\alpha(\mathcal{D}_4;\mathcal{D}_5), \quad (7)$$

We can set $\alpha = -\log_2(\epsilon_5)$, so that Eq. (7) implies:

$$\mathsf{Adv}^{\mathsf{Hybrid}_4}_{\mathcal{A}} \leq 2\cdot\mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}} \cdot R^{Q_s}_\alpha(\mathcal{D}_4;\mathcal{D}_5). \quad (8)$$

Hybrid_6**:** In this hybrid we replace $\mathbf{t}$ with a uniform vector. Note that since the secret key is not used in the previous hybrid this does not change the signing algorithm.

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}}\right| \leq \mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}$$

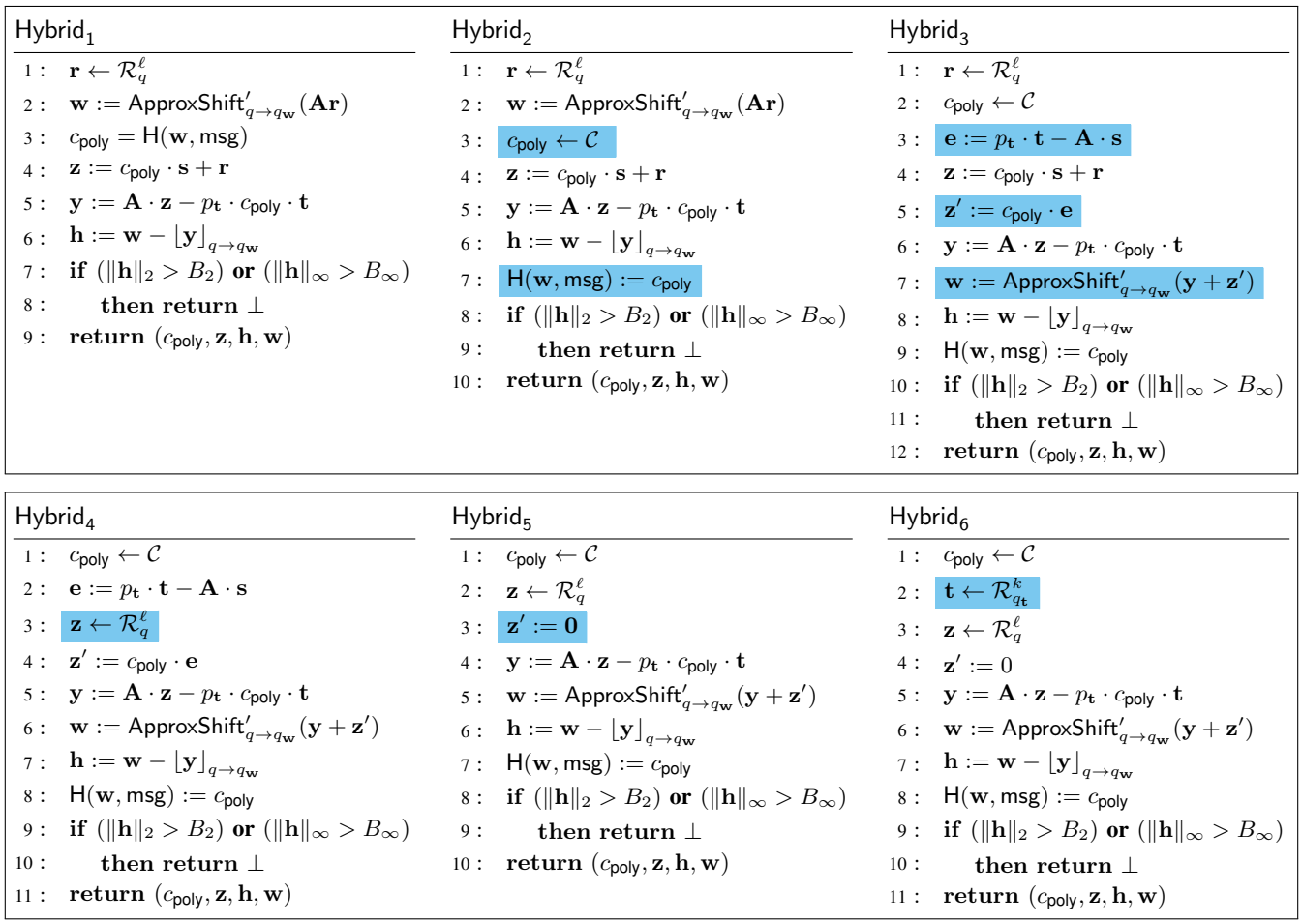

Figure 2. The security hybrids in the proof of Lemma 3. Changes between each hybrid are highlighted in blue.

□

An adversary breaking Hybrid_6 can be used to solve MSIS. This is stated in Theorem 1, and proven in Section A.3.

Theorem 1. *Let $\mathcal{A}$ be a forger for Hybrid_6 which is as defined above. Using $\mathcal{A}$ we construct an adversary for the $\mathsf{MSIS}_{q,k-\ell,B_{\mathsf{MSIS}}}$ where $B_{\mathsf{MSIS}} = 2p_\mathbf{w}B_2 + p_\mathbf{w}\sqrt{kn} + 2p_\mathbf{t}\sqrt{\omega}$.*

Theorem 2. *An adversary adversary who distinguishes the public key* vk *from uniform can break the* MLWRE *problem with the same advantage. I.e.*

$$\mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{MLWRE}_{\mathcal{R}_{q'},\ell,k,k,\chi}}_{\mathcal{A}}$$

Where χ is the Irwin-Hall distriubtion of degree d shifted by k bits.

Proof. The proof is given in Section A.4. □

4.2. Masking Proof

As the verification only uses public information, to achieve t-probing security, we need to prove the masking security of both the key generation scheme in Algorithm 6 and the signature scheme in Algorithm 7. This scheme has been designed to be "masking friendly", so the proofs are quite standard. However, in Section 4.2.1, we deal with a technical issue in the key generation because an intermediate value is correlated to the secret in an unintuitive way.

Let us first decompose both the Keygen and Sign algorithms into a sequence of elementary gadgets. We present the different gadgets and their security properties in Table 1.

TABLE 1. SECURITY PROPERTIES OF THE KNOWN AND NEW GADGETS

Name	Property	Reference
Refresh	t-SNI	Section 3.2 [35], [36], [37]
Decode	t-NIo	Elementary, see [8]
Line 1	t-NI	$\mathbb{Z}_q-$linear
$\times\mathbf{A}$	t-NI	$\mathbb{Z}_q-$linear
Line 10	t-NI	$\mathbb{Z}_q-$linear
$\mathsf{ApproxShift}_{q\to q'}$	t-NI	$\mathbb{Z}_q-$linear
OrderSwitch	t-SNI	Lemma 4
H and ChalPoly	None	Section 3.4
Computing the hint $\mathbf{h}$	None	Section 3.4

The gadget represented in Line 1 and the $\mathsf{ApproxShift}_{q\to\cdot}$ gadget can both be easily proved secure in the t-probing model because of their linearity with the addition modulo q.

Concerning OrderSwitch, note that even though the number of shares may increase inside this gadget, the targeted security is $t-\mathsf{NI}$ and not $t'-\mathsf{NI}$ (where $t'=2d-1$). Indeed, this temporary mask increase is performed for achieving the general $t-\mathsf{NIo}$ security of the Keygen as will be presented in Section 4.2.1.

Lemma 4. OrderSwitch *(Algorithm 3) is* $t-\mathsf{SNI}$.

Proof. This OrderSwitch gadget corresponds to a $\mathbb{Z}_q-$linear operation at Lines 1 to 3 and a Refresh (that is $t'-\mathsf{SNI}$ and thus $t-\mathsf{SNI}$ because $t'\geq t$). Hence, the $t-\mathsf{SNI}$ security is directly inherited from the Refresh gadget. □

Remark 1. *For a signature* $(c_{\mathsf{hash}},\mathbf{z},\mathbf{h})$ *generated with the secret key associated with the public key* $(\mathbf{A},\mathbf{t})$*, the internal value* $\mathbf{w}$ *can be recomputed only with public information,*

$$\mathbf{w}=\mathbf{h}+\lfloor\mathbf{Az}-p_{\mathbf{t}}\cdot c_{\mathsf{poly}}\cdot\mathbf{t}\rfloor_{q\to q_{\mathbf{w}}}.$$

Thus, $\mathbf{w}$ *can be considered as available for any attacker and, for proof matters, it will be part of the public outputs of the* Sign *algorithm.*

4.2.1. Masking security of the signature algorithm. Let us now introduce the composition proofs of the signature scheme.

Theorem 3. Sign *(Algorithm 7) algorithm is* $t-\mathsf{NIo}$ *with public outputs* $(c_{\mathsf{hash}},\mathbf{z},\mathbf{h},\mathbf{w})$.

Proof. The overall gadget decomposition of the algorithm is presented in Fig. 3. Let us assume that an attacker has access to $n^{\text{prob}}\leq t$ observations on the whole signature scheme. Then, we want to prove that all these n^{prob} observations can be perfectly simulated with at most n^{prob} shares of $[\![\mathbf{s}]\!]$ and the public variables. With such a result, the signature scheme is then secure in the t-probing model since no set of at most t observations would give information on the secret values. To fix notations, let us consider the following distribution of the attacker's n^{prob} observations: n_1^{prob} on the last instance of Decode, n_2^{prob} on Line 10, n_3^{prob} on the first instance of Decode, n_4^{prob} on $\mathsf{ApproxShift}_{q\to q_{\mathbf{w}}}$, n_5^{prob} on the multiplication with $\mathbf{A}$, n_6^{prob} on the randomness generation, and n_7^{prob} on the refresh algorithm. Some other observations can be made on the computation of the hash of the message and the computation of the hint h, their number will not matter during the proof. Finally, we have $\sum_{i=1}^{7}n_i^{\text{prob}}\leq n^{\text{prob}}$.

Now we build the proof from right to left as follows.

As $\mathbf{z}$ is a public output and the last instance of Decode in $t-\mathsf{NIo}$, all the observations performed during the execution of Decode and after can be perfectly simulated with at most n_1^{prob} shares of $[\![\mathbf{z}]\!]$ and the public outputs $(\mathbf{w},\mathbf{z},c_{\mathsf{hash}},\mathbf{h})$.

The gadget in line Line 10 is $t-\mathsf{NI}$. Thus, all the observations performed after Line 10, Decode and after can be perfectly simulated with at most $n_1^{\text{prob}}+n_2^{\text{prob}}$ shares of $[\![\mathbf{r}]\!]$ and $[\![\mathbf{s}]\!]$ and the public outputs.

We continue with the refresh gadget. As it is $t-\mathsf{SNI}$, the observations performed in Refresh, MultAdd, Decode and after can be perfectly simulated with at most $n_1^{\text{prob}}+n_2^{\text{prob}}$ shares of $[\![\mathbf{s}]\!]$, n_7^{prob} shares of $[\![\mathbf{r}]\!]$ and the public outputs.

The first Decode is $t-\mathsf{NIo}$ with public output $\mathbf{w}$, thus, all the observations performed during the execution of the first Decode, of the Refresh, of Line 10, of the Decode and after can be perfectly simulated with at most $n_1^{\text{prob}}+n_2^{\text{prob}}$ shares of $[\![\mathbf{s}]\!]$, n_7^{prob} shares of $[\![\mathbf{r}]\!]$, n_3^{prob} shares of $[\![\mathbf{w}]\!]$ and the public outputs.

Then, since both $\mathsf{ApproxShift}_{q\to q_{\mathbf{w}}}$ and the multiplication with $\mathbf{A}$ are $t-\mathsf{NI}$, we propagate the observations : all the observations performed during the signature strictly after the randomness generation can be perfectly simulated with at most $n_1^{\text{prob}}+n_2^{\text{prob}}$ shares of $[\![\mathbf{s}]\!]$, $n_7^{\text{prob}}+(n_3^{\text{prob}}+n_5^{\text{prob}}+n_5^{\text{prob}})$ shares of $[\![\mathbf{r}]\!]$ and the public outputs.

Finally, the left randomness generation gadget is also t-NI secure and does require any input; thus, the observations cannot be propagated any more. We need to ensure that the number of reported observations does not exceed n^{prob}. Finally, the simulation relies on $n_1^{\text{prob}}+n_2^{\text{prob}}$ shares of $[\![\mathbf{s}]\!]$ and the public outputs, which is sufficient to ensure the $t-\mathsf{NIo}$ security with public outputs $(\mathbf{w},\mathbf{z},c_{\mathsf{hash}},\mathbf{h})$. □

4.2.2. Masking security of the key generation algorithm. We present here a technical overview of the challenge of proving the masking of the Keygen. The detailed proof can be found in Section A.4.

Even though the gadget structure of the Keygen, presented in Fig. 4, would enable a straightforward masking proof at first sight, the t-probing security of Keygen is actually unintuitively technical. Indeed, unlike usual key generations in Fiat–Shamir with aborts signatures schemes, the error of the MLWE instance defining the pseudorandomness of vk relies on noise induced by the approximate shift of the shares of $[\![\mathbf{v}]\!]$. Thus, if an attacker probes the shares of $[\![\mathbf{v}]\!]$, the MLWE instance of vk gets easier and sk can be recovered more easily.

Hence, as highlighted in Lemma 6 in Section A.4, there exist a correlation between the joint distribution of $(\mathsf{vk},(\mathbf{v}_i))$ and sk. This prevents from simulating all intermediate values of Algorithm 6 with random and public data. We resolve this problem by introducing a new masking proof relying on a MLWE hypothesis.

The final result of Section A.4 is as follows. Let the discrete Irwin–Hall distribution be the distribution of the sum of a number of independent random variables, each having a uniform distribution.

Theorem 4. Keygen *(Algorithm 6) is* $t-\mathsf{NIo}$ *with public output* vk *under the* $\mathsf{MLWE}_{\mathcal{R}_q,\ell,k-\ell,\chi}$ *hypothesis where* χ *is the discrete Irwin–Hall distribution of standard deviation* $p_{\mathbf{t}}\sqrt{t/12}$ *and center* 0.

To close up the masking security, Theorems 3 and 4 allow concluding that Raccoon is EUF-CMA secure in the

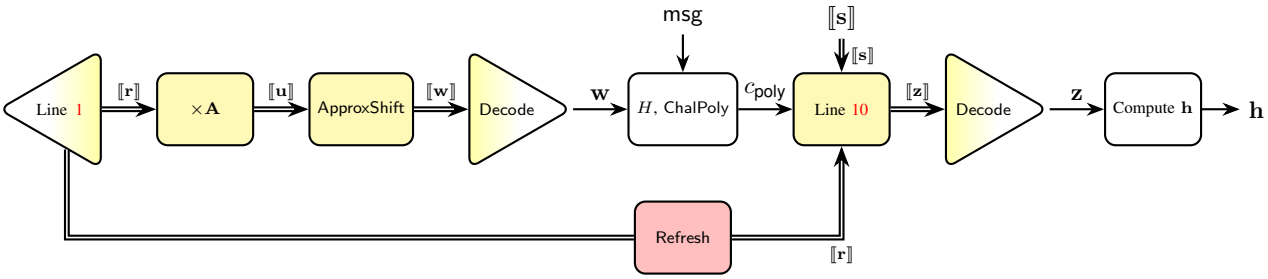

Figure 3. Structure of Sign (Algorithm 7). A gadgets proven $t-\mathsf{NI}$ (resp. $t-\mathsf{SNI}$, resp. unmasked) is noted gadget (resp. gadget, resp. gadget). Single arrows (⟶) and double arrows (⟹) represent plain and masked values, respectively. Triangular gadgets either start from a masked input and output an unmasked value, or the other way around.

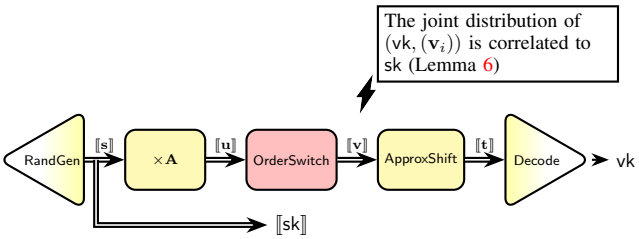

Figure 4. Structure of Keygen (Algorithm 6). Terminology as in Fig. 3.

t-probing model if the secret key ⟦s⟧ is refreshed at each signature query. We refer to Barthe et al. [8, Def. 1] for more details about this masking security model.

5. Parameter Selection

We now discuss parameter selection for Raccoon. The overarching goal is to ensure that Lemma 3 guarantees a bit-security λ_{TARGET}. More precisely, individual terms in the right side of Eq. (6) should guarantee $\mathsf{Adv}^{\mathsf{Sign}}_{\mathcal{A}}/Q_s \leq 2^{-\lambda_{\text{TARGET}}}$ under well-defined, explicit assumptions.

There are four important terms in Lemma 3, Eq. (6):

A. $\mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}$ is the advantage of $\mathcal{A}$ when distinguishing the verification key vk from a uniformly random vector. This is studied in Section 5.1.

B. $\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$ is the advantage of $\mathcal{A}$ when forging a signature for a pseudorandom vk. This is studied in Section 5.2.

C. $R^{Q_s}_{\alpha}(\mathcal{D}_4;\mathcal{D}_5)/Q_s$ quantifies an upper bound on the security loss that is entailed in the $(\mathsf{Hybrid}_4 \to \mathsf{Hybrid}_5)$ game hop. This bound can be expressed as a function of Q_s. Conversely, this means we can set a number of queries Q_s which guarantees a limited security loss. This is studied in Section 5.3.

D. $|\mathcal{C}|$ quantifies the security loss in the game hop $(\mathsf{Hybrid}_1 \to \mathsf{Hybrid}_2)$. This is studied in Section 5.4.

As illustrated by Table 2, each relevant term in Eq. (6) of Lemma 3 depends on the parameter set in a distinct way, which makes a global parameter selection slightly delicate. We start by studying each term separately, in Sections 5.1 to 5.4. We also briefly discuss the bounds B_2 and B_∞ in Section 5.5. Finally, we propose parameter sets in Section 5.7.

TABLE 2. CONSTRAINTS FOR PARAMETER SELECTION. TERMINOLOGY: ↗↗ (RESP. ↗, ↘, ↘↘) INDICATES THAT INCREASING THIS PARAMETER HAS A VERY POSITIVE (RESP. POSITIVE, NEGATIVE, VERY NEGATIVE) IMPACT ON THE CONSIDERED SECURITY METRIC.

Name	Key-recovery	Forgery	Rényi div.	Size of $\mathcal{C}$
Lemma 3	$\mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}$	$\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$	$R^{Q_s}_{\alpha}/Q_s$	$1/\lvert\mathcal{C}\rvert$
In Sect.	5.1	5.2	5.3	5.4
Q_s			↘	
q	↘	↗		
$p_{\mathbf{t}}$	↗↗		↘↘	
$p_{\mathbf{w}}$		↘↘	↗↗	
n	↗↗	↗↗	↘	↗
k	↘	↗↗	↘	
ℓ	↗↗	↘		
ω			↘	↗↗
d	↗			

5.1. Pseudorandomness of vk

We model the sum of d UMLWR samples with implicit error rate α as being hard to distinguish from a rounded sample of identical dimensions and error rate $\alpha\sqrt{d}$. Following the security reduction of Section A.1, an adversary solving $\mathsf{UMLWE}_{R_q,\ell,k,\chi}$ can be used to solve the $\mathsf{MLWE}_{R_q,\ell,k-\ell,\chi}$ problem, where χ is the distribution induced by the summation of d roundings. We use the lattice estimator by Albrecht et al. [41] to estimate the hardness of this problem. As is usual, we ignore the ring structure and model χ as a discrete Gaussian of standard deviation $p_{\mathbf{t}}\sqrt{\frac{d}{12}}$.

Let us note Hardness the function which takes as input an MLWE instance, computes its Core-SVP hardness using the lattice estimator [41], and applies the dimensions-for-free optimization by Ducas [42]. The best known attacks according to the lattice estimator are the primal uSVP attack by by Alkim et al. [43] and the dual/hybrid attack by by Espitau et al. [44]. The key-only bit-security $\lambda_{\text{KO-KR}}$ of Raccoon against a key-recovery attack is:

$$\lambda_{\text{KO-KR}} = \mathsf{Hardness}(\mathsf{MLWE}_{R_q,\ell,k-\ell,\chi}) \tag{9}$$

In practice, the number of MLWE samples $(k-\ell)$ has a negligible influence on the hardness of Eq. (9). The concrete evolution of $\mathsf{Adv}^{\mathsf{PK}}_{\mathcal{A}}$ as a function of the most influential parameters is given in Fig. 5.

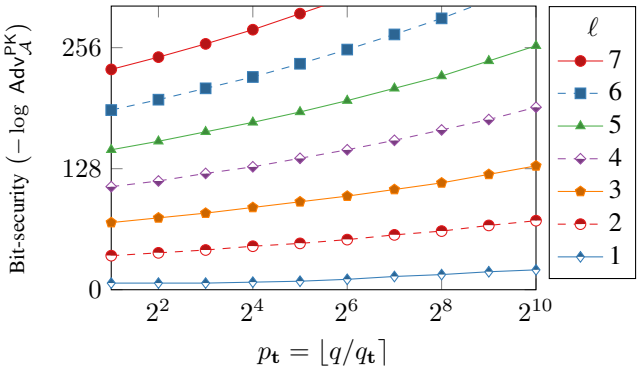

Figure 5. Key-only bit-security of key-recovery as a function of $p_{\mathbf{t}} = \lfloor q/q_{\mathbf{t}} \rceil$ and ℓ, using Eq. (9) with $q = 2^{49} - 2^{18} + 1, n = 512, d = 32$, $k = \infty$.

5.2. Forgery

The second important step is to bound $\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$, which corresponds to forging a signature for a pseudorandom verification key vk. While Theorem 1 provides a reduction to the MSIS problem, we consider directly the hardness of finding $(c_{\mathsf{poly}}, \mathbf{z}, \mathbf{h})$ with small $\mathbf{h}$ such that:

$$H\left(\mathbf{A}\cdot\mathbf{z} - p_{\mathbf{t}}\cdot c_{\mathsf{poly}}\cdot\mathbf{t} + \mathbf{h}, \mathsf{msg}\right) = c_{\mathsf{hash}} \tag{10}$$

This choice of considering the direct forgery problem mirrors what is done in Dilithium [2]. The similarity doesn't end there: the problem underlying Eq. (10) is a variant of SelfTargetMSIS problem [2], and many of the observations made in [2] are applicable here. The best known ways to solve Eq. (10) are to either break the hash function H or, using the same transformation as in the proof of Theorem 1, to find $\mathbf{v} \in \mathcal{R}_q^k$ such that:

$$\mathbf{B}\cdot\mathbf{v} = \mathbf{w}', \quad \text{where} \begin{cases} \mathbf{w}' = \mathbf{B}\left(\mathbf{w} + p_{\mathbf{t}}\cdot c_{\mathsf{poly}}\cdot\mathbf{t}\right), \\ \mathbf{v} = \boldsymbol{\delta} + p_{\mathbf{w}}\cdot\mathbf{h}, \end{cases} \tag{11}$$

and $\mathbf{B} \in \mathcal{R}_q^{(k-\ell)\times k}$ and $\boldsymbol{\delta}$ are defined in the proof of Theorem 1. The best known way to solve Eq. (11) is by solving the *inhomogenous* MSIS problem $\mathsf{IMSIS}_{R_q, k-\ell, k, B_{\mathsf{IMSIS}}}$, where:

$$B_{\mathsf{IMSIS}} = p_{\mathbf{w}}\left(B_2 + \frac{\sqrt{kn}}{2}\right).$$

The best known strategy for solving IMSIS is outlined by Chuengsatiansup et al. [45, §4.2]. Following their analysis, Raccoon is secure against a direct forgery attack using BKZ with blocksize β as long as:

$$B_{\mathsf{IMSIS}} \leq \min_{\ell n \leq m \leq kn}\left(q^{\frac{(k-\ell)n}{m}}\cdot\delta_\beta^m\right), \tag{12}$$

$$\text{where } \delta_\beta = \left(\frac{(\pi\beta)^{1/\beta}\cdot\beta}{2\pi e}\right)^{1/(2(\beta-1))}. \tag{13}$$

Following the Core-SVP methodology and the dimensions-for-free optimization [42], the key-only bit-security of Raccoon against forgery, $\left(-\log \mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}\right)$, is a non-decreasing function of the highest BKZ blocksize β for which Eq. (12) is satisfied.

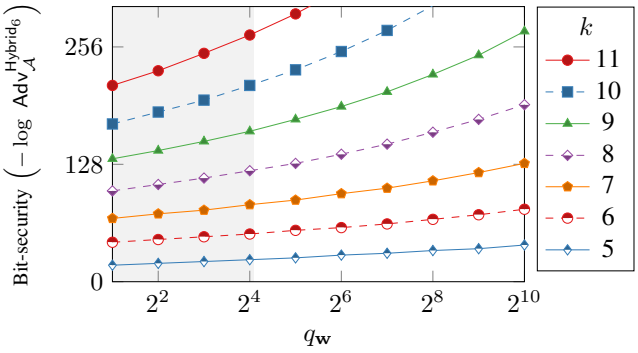

Figure 6. Key-only bit-security of forgery as a function of $q_{\mathbf{w}}$ and k, using Eq. (12) with $q = 2^{49} - 2^{18} + 1, n = 512, d = 32$, $\ell = 3, B_\infty = 8$. We rule out (gray area) parameter sets for which $2B_\infty \geq q_{\mathbf{w}}$ (see Section 5.5).

Fig. 6 provides security estimates for a forgery attack based on our analysis. We note that we only studied whether a forgery attempt is successful with respect to the L_2 bound B_2. A security analysis with respect to the L_∞ bound B_∞ may lead to different conclusions. Of course, since our scheme relies on both metrics, it remains secure as long as the IMSIS problem is secure for *either* the L_2 or L_∞ norms.

5.3. Number of queries Q_s

We now bound $R_\alpha^{Q_s}(\mathcal{D}_4;\mathcal{D}_5)/Q_s$. As a function of Q_s, this term asymptotically grows to infinity since $R_\alpha(\mathcal{D}_4;\mathcal{D}_5) > 1$. Therefore, in order to make Lemma 3 non-vacuous, we impose an upper bound on Q_s. In this section, we show that under mild heuristic assumptions, Q_s can be set to be concretely large with little to no impact on the security of Raccoon.

We recall that $\mathcal{D}_4$ and $\mathcal{D}_5$ are the distribution of the outputs $(c_{\mathsf{hash}}, \mathbf{z}, \mathbf{h}, \mathbf{w})$ of Hybrid_4 and Hybrid_5, respectively. We note that in both hybrids, the distributions of c_{hash} and $\mathbf{z}$ are identical, and $\mathbf{h} = f(c_{\mathsf{hash}}, \mathbf{z}, \mathbf{w})$ for a randomized function f that is exactly the same in both hybrids. By the data processing inequality of the Rényi divergence (Lemma 1, Item (R1)):

$$R_\alpha(\mathcal{D}_4;\mathcal{D}_5) \leq \max_{\{c_{\mathsf{poly}}, \mathbf{z}\}} R_\alpha(\mathcal{P}_4;\mathcal{P}_5), \tag{14}$$

where $\mathcal{P}_i$ is the distribution of $\mathbf{w}$ in Hybrid_i conditioned on $(c_{\mathsf{poly}}, \mathbf{z})$ being fixed to the same values in both hybrids.

As a stepping stone, we rely on Conjecture 1, which bounds the Rényi divergence between the individual integer coefficients of $\mathbf{w}$ in Hybrid_4 and Hybrid_5. This numerical conjecture is supported by experiments available from the public repository. We expect a formal proof to be extremely tedious but feasible to obtain.

Conjecture 1. *Let $q = p_{\mathbf{w}} q_{\mathbf{w}}$, let $u_1, u_2 \in \mathbb{Z}_q$ such that $|u_1 - u_2|^2 \leq \epsilon^2 p_{\mathbf{w}}^2$, and $P_i = \mathsf{ApproxShift}'_{q \to q_{\mathbf{w}}}(u_i)$. With probability at least $1 - {p_{\mathbf{w}}}^{-(d-1)}$, it holds that:*

$$\log R_\alpha(P_1; P_2) \leq \frac{C\alpha\epsilon^2}{d}, \tag{15}$$

for a constant $C \approx 6$.

The next step is to bound a value related to ϵ in Conjecture 1. Note that the inputs of $\mathsf{ApproxShift}'$ in Hybrid_4 and Hybrid_5 differ by an additive term $\mathbf{z}' = c_{\mathsf{poly}} \cdot \mathbf{e_t}$, which is easily characterized under mild heuristics. Recall that the verification key can be written as $p_{\mathbf{t}} \cdot \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e_t}$. Heuristically, each integer coefficient of $\mathbf{e_t}$ can be modeled as the sum of $2d$ errors in $\{-p_{\mathbf{t}}/2, \ldots, p_{\mathbf{t}}/2\}$. From there, we derive a heuristic formula for the expected L_2 norm of $\mathbf{e_t} \cdot c_{\mathsf{poly}}$:

$$\mathbb{E}[\|\mathbf{e_t} \cdot c_{\mathsf{poly}}\|_2^2] \approx \frac{nkd\omega p_{\mathbf{t}}^2}{6} \tag{16}$$

In Eq. (16), the factor nk corresponds to the number of integer coefficients, and the factor $\frac{d\omega p_{\mathbf{t}}^2}{6}$ corresponds to the fact that each integer coefficient is the sum of $2d\omega$ errors in $\{-p_{\mathbf{t}}/2, \ldots, p_{\mathbf{t}}/2\}$.

We can now establish a bound on Q_s. Let $\lambda_{\text{KO}} = \log\left(\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_6} + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PK}}\right)$. Note that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} \leq 2^{-\lambda_{\text{KO}}}$. Since the Rényi divergence R_α is non-decreasing in its parameter α, in order to bound Eq. (6) it suffices to have:

$$2^{-\lambda_{\text{KO}}+1} \cdot \frac{R_{\lambda_{\text{KO}}}^{Q_s}(\mathcal{P}_4; \mathcal{P}_5)}{Q_s} \leq 2^{-\lambda_{\text{TARGET}}} \tag{17}$$

We now estimate $R_{\lambda_{\text{KO}}}(\mathcal{P}_4; \mathcal{P}_5)$. Combining Conjecture 1, the multiplicativity property of the Rényi divergence (R3), and Eq. (16), provides this approximation:

$$\log R_{\lambda_{\text{KO}}}(\mathcal{P}_4; \mathcal{P}_5) \lesssim \frac{C\lambda_{\text{KO}}\epsilon^2}{d}, \text{with } \epsilon^2 = \frac{nkd\omega p_{\mathbf{t}}^2}{6p_{\mathbf{w}}^2}. \tag{18}$$

We can combine Eq. (17) and Eq. (18), which provides the following (heuristic) sufficient condition on Q_s:

$$Q_s \frac{C\lambda_{\text{KO}} nk\omega}{6} \left(\frac{p_{\mathbf{t}}}{p_{\mathbf{w}}}\right)^2 - \log Q_s \leq \lambda_{\text{KO}} - \lambda_{\text{TARGET}} - 1 \tag{19}$$

5.4. Challenge space

It is clear from Eq. (6) that we need $|\mathcal{C}| \geq 2^{\lambda_{\text{TARGET}}}$. This translates to the condition Eq. (20).

$$\binom{n}{\omega} 2^\omega \geq 2^{\lambda_{\text{TARGET}}}. \tag{20}$$

5.5. Bounds B_2 and B_∞ on the norms of the hint

Setting $B_2^2 = \frac{d^2 kn}{4}$ and $B_\infty = \frac{d}{2}$ guarantees that Line 15 of Algorithm 7 returns **true** with probability 1. In practice, setting B_2 and B_∞ lower maintains this probability to $1 - o(1)$ while making forgery harder.

For B_∞ to be non-vacuous, we also require $2B_\infty < q_{\mathbf{w}}$.

5.6. Masking Order

Unlike other security parameters, choosing the masking order t and the number of shares $d = t + 1$ depends on the use case and the physical characteristics of the implementation platform. As reported by Chari et al. [22] or Duc et al. [23], there is an exponential relationship between d and a lower bound on the number of observations required for an attack. In practice, the number of shares depends on the use case and needs to be determined experimentally.

The safety margin for d selection should account for inadvertent mixing of (share) signals in the data paths of the implementation and developments in the efficiency of analysis (e.g., machine learning.) For some low-security applications running in high-noise environments, even $d = 2$ may be sufficient. However, due to its quasilinear masking complexity, Raccoon can have a substantial security margin to prepare for advances in physical attacks (such as Laser Logic State Imaging [46]). We conjecture that $d = 32$ is sufficient for the foreseeable future (if implemented diligently), and propose it as the default value for Raccoon. As the software and low-end FPGA experiments (Section 6) demonstrate, $d = 32$ can be supported in applications such as HSMs, secure elements, and smart cards.

5.7. Parameter sets

Based on this section's analysis, we propose three parameter sets in Table 3. They correspond to at least 128, 192 and 256 bits of security, respectively.

TABLE 3. PARAMETER SETS FOR RACCOON. THE SYMBOL "-" INDICATES THAT THE VALUE IS IDENTICAL ACROSS ALL PARAMETER SETS. THE SIZES |vk| AND |sig| ARE PROVIDED IN BYTES.

Name	Raccoon-128	Raccoon-192	Raccoon-256
λ_{TARGET}	128	192	256
Q_s	2^{48}	2^{48}	2^{49}
d	32	-	-
$\log q$	49†	-	-
$\log p_{\mathbf{t}}$	10	6	7
$\log p_{\mathbf{w}}$	43	40	42
n	512	-	-
k	8	11	14
ℓ	3	5	6
ω	19	31	44
B_2^2	2^{14}	2^{14}	2^{15}
B_∞	8	-	-
|vk|	19 968	30 272	37 632
|sig|	12 000	19 232	23 328

†Across all parameter sets, we set $q = (2^{25} - 2^{18} + 1) \cdot (2^{24} - 2^{18} + 1)$. See Section 6 for a detailed discussion.

6. Implementation Characteristics

Raccoon has been designed not to require masked symmetric cryptography primitives. Unlike Dilithium, a standard, unmasked version of SHA3/SHAKE can be used in Raccoon without negatively affecting the side-channel security of the construction.

6.1. Portable C Implementation: CRACCOON

This reference implementation[3] is written in ANSI C for portability. The purpose of CRACCOON is to evaluate the relative performance characteristics of Raccoon. Note that for actual security one would require d-probing secure random number generators. Simple non-cryptographic masking generators are used in CRACCOON for testing purposes.

Table 4 summarizes the performance characteristics of the CRACCOON in relation to an *unmasked* C reference implementation of Dilithium.

TABLE 4. LATENCY (MILLISECONDS AND MILLIONS OF CYCLES) OF C REFERENCE CODE OF DILITHIUM, FALCON, AND MASKED RACCOON.

		Keygen()		Sign()		Verify()	
Algorithm	d	ms	Mc	ms	Mc	ms	Mc
Raccoon-128	2	0.624	1.308	0.731	1.532	0.559	1.171
Raccoon-128	4	0.970	2.034	1.020	2.139	0.564	1.182
Raccoon-128	8	1.718	3.600	1.673	3.506	0.583	1.222
Raccoon-128	16	3.426	7.181	3.098	6.494	0.584	1.224
Raccoon-128	32	7.127	14.93	6.201	12.99	0.584	1.224
Raccoon-192	32	11.08	23.23	10.30	21.60	1.187	2.487
Raccoon-256	32	14.96	31.37	13.78	28.89	1.752	3.672
Dilithium2	1	0.098	0.205	0.382	0.801	0.109	0.229
Dilithium3	1	0.179	0.375	0.416	0.872	0.174	0.365
Dilithium5	1	0.269	0.564	1.343	2.814	0.283	0.593
Falcon-512	1	15.86	33.25	4.578	9.596	0.043	0.090
Falcon-1024	1	44.03	92.31	10.00	20.97	0.089	0.186

Notes: AMD Ryzen 5 PRO 3500U, gcc 11.3.0, flags `-O3`. Dilithium v3.1 and Falcon v1.2 implementations are "clean" versions from https://github.com/PQClean/PQClean. The benchmarked Falcon code uses floating point emulation, making key generation and signing several times slower, but also more likely to be resistant to timing attacks ("constant-time".)

6.2. Experimental Hardware Validation: t-PANDA

t-PANDA is a further application of an architecture that has been used in a commercial side-channel secure (ASIC) PQC unit. Hence t-PANDA is also able to run first-order masked Dilithium (and Kyber); See Table 5.

On a XC7A100T (Xilinx Artix 7) FPGA target, this size-optimized design (including the simplified control Core, Keccak unit, lattice coprocessor, masking random number generator, and communication peripherals) was 10,638 Slice LUTs (16.78%), 4,140 Slice registers / Flip Flops, (3.26%) and 3 DSPs. The design was rated for 78.3 MHz.

One may compare these results with the Dilithium implementation in [47], which is faster, but also twice the size of our implementation on the same Artix 7 target. However, this architecture lacks the features to achieve side-channel security. The side-channel secure Dilithium implementation reported in [24] requires almost three times more (6.757M / 0.19) cycles for a successful signature with Level 3 / $d = 2$ parameters.

3. CRACCOON artefact: https://github.com/masksign/sp23-craccoon

TABLE 5. FPGA / t-PANDA CYCLE COUNTS AT VARIOUS SIDE-CHANNEL SECURITY LEVELS. THE DEVICE ALSO SUPPORTS TWO-SHARE DILITHIUM.

Algorithm	Shares	Keygen()	Sign()	Verify()
Raccoon-128	$d = 2$	1,366,000	2,402,000	1,438,000
Raccoon-128	$d = 4$	2,945,000	3,714,230	1,433,034
Raccoon-128	$d = 8$	6,100,000	6,345,000	1,389,000
Raccoon-128	$d = 16$	12,413,000	11,605,000	1,389,000
Raccoon-128	$d = 32$	25,073,000	22,160,000	1,393,000
Dilithium2	$d = 1$	572,000	3,102,000	510,000
Dilithium3	$d = 1$	886,000	5,010,000	725,000
Dilithium5	$d = 1$	1,399,000	5,889,000	1,174,000
Dilithium2	$d = 2$	1,633,000	7,866,000	510,000
Dilithium3	$d = 2$	2,538,000	12,326,000	725,000
Dilithium5	$d = 2$	3,389,000	13,489,000	1,174,000

6.3. Leakage Assessment

We performed a leakage assessment on the FPGA implementation, following the general test procedure of ISO 17825:2022 [48]. This "TVLA" type random-vs-fixed test was adapted to detect leakage from $[\![\mathbf{s}]\!]$ in the signature generation function.

Power signal was acquired from connectors on the CW305 board [49, Sect. C.3] with a PicoScope 2208B oscilloscope. The test was run with a 32ns (31.25 MHz) clock cycle. Power samples were gathered at the same rate.

The ISO 17825 testing procedure is generally limited to first-order leakage, and hence a $d = 2$ version of Raccoon was used. At $N = 200,000$ traces, the maximum t-value was 4.89 (Fig. 7). No leakage was detected.

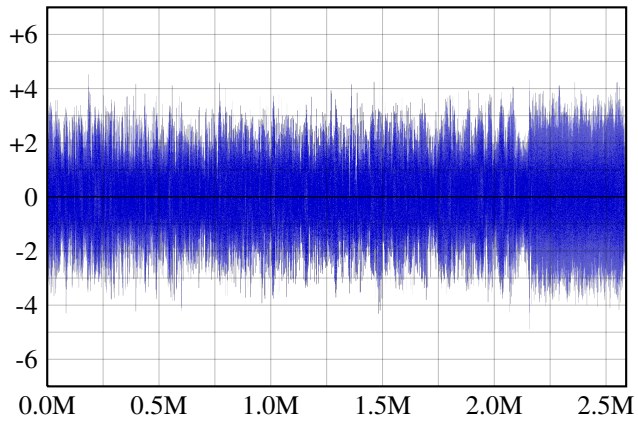

Figure 7. No secret key leakage was detected in a 200,000-trace leakage assessment of Raccoon-128 ($d = 2$) signature function. Each trace had 2.59×10^6 time points (one for each cycle – horizontal axis); a total of 518 billion measurements. The vertical axis has the t-statistic, which was bound by $|t| < 4.89$. This is well under the critical value ($C = 6.94$).

7. Conclusion

In this paper, we propose the first lattice-based signature scheme that can be masked at high order with a quasilinear overhead. We achieved that by proposing new algorithmic techniques, as well as new proof techniques. We prove the security of our design under variants of standard lattice assumptions (MLWE and MSIS) and in the well-known SNI proof framework. Finally, we implemented our signature scheme and measure efficiency and security metrics; both the performance and concrete leakage profile are consistent with our theoretical analyses.

Since our design is rather simple and generic, we hope to see more efficient instantiations and more applications of it in the future.

References

[1] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status report on the third round of the NIST post-quantum cryptography standardization process," NISTIR 8413-upd1, National Institute of Standards and Technology, Interagency or Internal Report, September 2022. [Online]. Available: https://csrc.nist.gov/publications/detail/nistir/8413/final

[2] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai, "CRYSTALS-DILITHIUM," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[3] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "FALCON," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[4] E. Karabulut and A. Aysu, "FALCON down: Breaking FALCON post-quantum signature scheme through side-channel attacks," in *58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, December 5-9, 2021*. IEEE, 2021, pp. 691–696. [Online]. Available: https://doi.org/10.1109/DAC18074.2021.9586131

[5] M. Guerreau, A. Martinelli, T. Ricosset, and M. Rossi, "The hidden parallelepiped is back again: Power analysis attacks on falcon," *IACR TCHES*, vol. 2022, no. 3, pp. 141–164, 2022.

[6] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay, and S. Bhasin, "Side-channel assisted existential forgery attack on Dilithium - A NIST PQC candidate," Cryptology ePrint Archive, Report 2018/821, 2018, https://eprint.iacr.org/2018/821.

[7] S. Marzougui, V. Ulitzsch, M. Tibouchi, and J.-P. Seifert, "Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all," Cryptology ePrint Archive, Report 2022/106, 2022, https://eprint.iacr.org/2022/106.

[8] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi, "Masking the GLP lattice-based signature scheme at any order," in *EUROCRYPT 2018, Part II*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10821. Springer, Heidelberg, Apr. / May 2018, pp. 354–384.

[9] F. Gérard and M. Rossi, "An efficient and provable masked implementation of qtesla," in *Smart Card Research and Advanced Applications - 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11-13, 2019, Revised Selected Papers*, ser. Lecture Notes in Computer Science, S. Belaïd and T. Güneysu, Eds., vol. 11833. Springer, 2019, pp. 74–91. [Online]. Available: https://doi.org/10.1007/978-3-030-42068-0_5

[10] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, M. Rossi, and M. Tibouchi, "GALACTICS: Gaussian sampling for lattice-based constant-time implementation of cryptographic signatures, revisited," in *ACM CCS 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM Press, Nov. 2019, pp. 2147–2164.

[11] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *CHES 2012*, ser. LNCS, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer, Heidelberg, Sep. 2012, pp. 530–547.

[12] N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon, "qTESLA," National Institute of Standards and Technology, Tech. Rep., 2019, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[13] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque, "Masking Dilithium - efficient implementation and side-channel evaluation," in *ACNS 19*, ser. LNCS, R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, Eds., vol. 11464. Springer, Heidelberg, Jun. 2019, pp. 344–362.

[14] T. Espitau, P.-A. Fouque, F. Gérard, M. Rossi, A. Takahashi, M. Tibouchi, A. Wallet, and Y. Yu, "Mitaka: A simpler, parallelizable, maskable variant of falcon," in *EUROCRYPT 2022, Part III*, ser. LNCS, O. Dunkelman and S. Dziembowski, Eds., vol. 13277. Springer, Heidelberg, May / Jun. 2022, pp. 222–253.

[15] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *ASIACRYPT 2009*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer, Heidelberg, Dec. 2009, pp. 598–616.

[16] ——, "Lattice signatures without trapdoors," in *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Springer, Heidelberg, Apr. 2012, pp. 738–755.

[17] S. Bai and S. D. Galbraith, "An improved compression technique for signatures based on learning with errors," in *CT-RSA 2014*, ser. LNCS, J. Benaloh, Ed., vol. 8366. Springer, Heidelberg, Feb. 2014, pp. 28–47.

[18] J.-S. Coron, J. Großschädl, and P. K. Vadnala, "Secure conversion between Boolean and arithmetic masking of any order," in *CHES 2014*, ser. LNCS, L. Batina and M. Robshaw, Eds., vol. 8731. Springer, Heidelberg, Sep. 2014, pp. 188–205.

[19] M. Hutter and M. Tunstall, "Constant-time higher-order Boolean-to-arithmetic masking," *Journal of Cryptographic Engineering*, vol. 9, no. 2, pp. 173–184, Jun. 2019.

[20] J.-S. Coron, J. Großschädl, M. Tibouchi, and P. K. Vadnala, "Conversion from arithmetic to Boolean masking with logarithmic complexity," in *FSE 2015*, ser. LNCS, G. Leander, Ed., vol. 9054. Springer, Heidelberg, Mar. 2015, pp. 130–149.

[21] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *CRYPTO 2003*, ser. LNCS, D. Boneh, Ed., vol. 2729. Springer, Heidelberg, Aug. 2003, pp. 463–481.

[22] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO'99*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Springer, Heidelberg, Aug. 1999, pp. 398–412.

[23] A. Duc, S. Faust, and F.-X. Standaert, "Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version," *Journal of Cryptology*, vol. 32, no. 4, pp. 1263–1297, Oct. 2019.

[24] M. Azouaoui, O. Bronchain, G. Cassiers, C. Hoffmann, Y. Kuzovkova, J. Renes, M. Schönauer, T. Schneider, F.-X. Standaert, and C. van Vredendaal, "Leveling Dilithium against leakage: Revisited sensitivity analysis and improved implementations," Cryptology ePrint Archive, Paper 2022/1406, 2022, fourth PQC Standardization Conference, NIST (Virtual) 29 Nov – 1 Dec 2022. [Online]. Available: https://eprint.iacr.org/2022/1406

[25] E. Kiltz, V. Lyubashevsky, and C. Schaffner, "A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model," in *EUROCRYPT 2018, Part III*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Springer, Heidelberg, Apr. / May 2018, pp. 552–586.

[26] "IEEE standard for floating-point arithmetic," IEEE Std 754-2019 (Revision of IEEE Std 754-2008), p. 84, June 2019.

[27] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *CRYPTO 2009*, ser. LNCS, S. Halevi, Ed., vol. 5677. Springer, Heidelberg, Aug. 2009, pp. 595–618.

[28] A. Rényi, "On measures of entropy and information," in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif.: University of California Press, 1961, pp. 547–561. [Online]. Available: http://projecteuclid.org/euclid.bsmsp/1200512181

[29] S. Bai, A. Langlois, T. Lepoint, D. Stehlé, and R. Steinfeld, "Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance," in *ASIACRYPT 2015, Part I*, ser. LNCS, T. Iwata and J. H. Cheon, Eds., vol. 9452. Springer, Heidelberg, Nov. / Dec. 2015, pp. 3–24.

[30] T. Prest, "Sharper bounds in lattice-based cryptography using the Rényi divergence," in *ASIACRYPT 2017, Part I*, ser. LNCS, T. Takagi and T. Peyrin, Eds., vol. 10624. Springer, Heidelberg, Dec. 2017, pp. 347–374.

[31] I. Csiszár, "Eine informationstheoretische ungleichung und ihre anwendung auf den beweis der ergodizitat von markoffschen ketten," *Magyar. Tud. Akad. Mat. Kutató Int. Közl*, vol. 8, pp. 85–108, 1963.

[32] T. van Erven and P. Harremoës, "Rényi divergence and kullback-leibler divergence," *IEEE Trans. Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014. [Online]. Available: http://dx.doi.org/10.1109/TIT.2014.2320500

[33] A. Duc, S. Dziembowski, and S. Faust, "Unifying leakage models: From probing attacks to noisy leakage," in *EUROCRYPT 2014*, ser. LNCS, P. Q. Nguyen and E. Oswald, Eds., vol. 8441. Springer, Heidelberg, May 2014, pp. 423–440.

[34] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini, "Strong non-interference and type-directed higher-order masking," in *ACM CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM Press, Oct. 2016, pp. 116–129.

[35] A. Battistello, J.-S. Coron, E. Prouff, and R. Zeitoun, "Horizontal side-channel attacks and countermeasures on the ISW masking scheme," in *CHES 2016*, ser. LNCS, B. Gierlichs and A. Y. Poschmann, Eds., vol. 9813. Springer, Heidelberg, Aug. 2016, pp. 23–39.

[36] A. Mathieu-Mahias, "Securisation of implementations of cryptographic algorithms in the context of embedded systems. (sécurisation des implémentations d'algorithmes cryptographiques pour les systèmes embarqués)," Ph.D. dissertation, University of Paris-Saclay, France, 2021. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03537322

[37] D. Goudarzi, T. Prest, M. Rivain, and D. Vergnaud, "Probing security through input-output separation and revisited quasilinear masking," Cryptology ePrint Archive, Report 2022/045, 2022, https://eprint.iacr.org/2022/045.

[38] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé, "CRYSTALS-KYBER," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[39] J.-S. Coron, F. Gérard, S. Montoya, and R. Zeitoun, "High-order table-based conversion algorithms and masking lattice-based encryption," *IACR TCHES*, vol. 2022, no. 2, pp. 1–40, 2022.

[40] N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon, "qTESLA," National Institute of Standards and Technology, Tech. Rep., 2017, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[41] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015. [Online]. Available: https://doi.org/10.1515/jmc-2015-0016

[42] L. Ducas, "Shortest vector from lattice sieving: A few dimensions for free," in *EUROCRYPT 2018, Part I*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10820. Springer, Heidelberg, Apr. / May 2018, pp. 125–145.

[43] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange - A new hope," in *USENIX Security 2016*, T. Holz and S. Savage, Eds. USENIX Association, Aug. 2016, pp. 327–343.

[44] T. Espitau, A. Joux, and N. Kharchenko, "On a dual/hybrid approach to small secret LWE - A dual/enumeration technique for learning with errors and application to security estimates of FHE schemes," in *INDOCRYPT 2020*, ser. LNCS, K. Bhargavan, E. Oswald, and M. Prabhakaran, Eds., vol. 12578. Springer, Heidelberg, Dec. 2020, pp. 440–462.

[45] C. Chuengsatiansup, T. Prest, D. Stehlé, A. Wallet, and K. Xagawa, "ModFalcon: Compact signatures based on module-NTRU lattices," in *ASIACCS 20*, H.-M. Sun, S.-P. Shieh, G. Gu, and G. Ateniese, Eds. ACM Press, Oct. 2020, pp. 853–866.

[46] T. Krachenfels, F. Ganji, A. Moradi, S. Tajik, and J.-P. Seifert, "Real-world snapshots vs. theory: Questioning the t-probing security model," in *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1955–1971.

[47] G. Land, P. Sasdrich, and T. Güneysu, "A hard crystal - implementing Dilithium on reconfigurable hardware," in *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, ser. Lecture Notes in Computer Science, V. Grosso and T. Pöppelmann, Eds., vol. 13173. Springer, 2021, pp. 210–230.

[48] ISO, "Information technology – security techniques – testing methods for the mitigation of non-invasive attack classes against cryptographic modules," International Organization for Standardization, Draft International Standard ISO/IEC DIS 17825:2022(E), 2023.

[49] ——, "IT security techniques – test tool requirements and test tool calibration methods for use in testing non-invasive attack mitigation techniques in cryptographic modules – part 2: Test calibration methods and apparatus," International Organization for Standardization, Standard ISO/IEC 20085-2:2020(E), 2020. [Online]. Available: https://www.iso.org/standard/70082.html

Appendix A. Omitted Proofs

A.1. Reduction from MLWE to UMLWE

Lemma 5. *For any integers ℓ, k, $\mathsf{UMLWE}_{\mathcal{R}_q,\ell,k+\ell,\chi}$ is at least as hard as $\mathsf{MLWE}_{\mathcal{R}_q,\ell,k,\chi}$.*

Proof. We give a transformation that maps any $(\mathbf{A}, \mathbf{b}) \in \mathcal{R}_q^{k\times\ell}$ to $(\mathbf{A}', \mathbf{b}') \in \mathcal{R}_q^{(k+\ell)\times\ell}$. Where if $(\mathbf{A}, \mathbf{b})$ is an $\mathsf{MLWE}_{\mathcal{R}_q,\ell,k,\chi}$ instance then $(\mathbf{A}', \mathbf{b}')$ will be an $\mathsf{UMLWE}_{\mathcal{R}_q,\ell,k+\ell,\chi}$ instance; and if $(\mathbf{A}, \mathbf{b})$ is uniform then $(\mathbf{A}', \mathbf{b}')$ will be uniform.

For an instance $(\mathbf{A};\mathbf{b})$ we sample $\mathbf{A}'_1 \leftarrow \mathcal{R}_q^{\ell\times\ell}$, $\mathbf{b}'_1 \leftarrow \mathcal{R}_q^{\ell}$, and set

$$\mathbf{A}' = \begin{bmatrix}\mathbf{A}'_1 \\ \mathbf{A}\mathbf{A}'_1\end{bmatrix}; \mathbf{b}' = \begin{bmatrix}\mathbf{b}'_1 \\ \mathbf{b} - \mathbf{A}\mathbf{b}'_1\end{bmatrix}.$$

It is clear that if $(\mathbf{A},\mathbf{b})$ is uniform then $(\mathbf{A}',\mathbf{b}')$ is uniform. If $\mathbf{b} = \mathbf{A}\mathbf{s}+\mathbf{e}$ with $\mathbf{s},\mathbf{e} \leftarrow \chi^\ell \times \chi^k$, then we can fix $\mathbf{s}' = {\mathbf{A}'_1}^{-1}(\mathbf{b}_1+\mathbf{s})$ (we can assume that $\mathbf{A}'_1$ is invertible as the reduction can simply sample another matrix if it is not true) and $\mathbf{e}' = \begin{bmatrix}-\mathbf{s} \\ \mathbf{e}\end{bmatrix}$. It is then easily checked that $\mathbf{A}'\mathbf{s}' + \mathbf{e}' = \mathbf{b}'$, with $\mathbf{s}' \in \mathcal{R}_q^\ell$ and $\mathbf{e}' \leftarrow \chi^{k+\ell}$.

□

A.2. Proof of Lemma 2

Proof. On one hand:

$$\begin{aligned} y &= \sum_{i\in[d]} \left\lfloor \frac{x_i+\delta}{2^k} \right\rfloor \le \left\lfloor \sum_{i\in[d]} \frac{x_i+\delta}{2^k} \right\rfloor \\ &\le \left\lfloor \sum_{i\in[d]} \frac{x_i}{2^k} \right\rfloor + \left\lceil \frac{d\delta}{2^k} \right\rceil \le y^* + 1 + \left\lceil \frac{d\delta}{2^k} \right\rceil \end{aligned} \tag{21}$$

On the other hand:

$$\begin{aligned} y &\ge \sum_{i\in[d]} \left\lceil \frac{x_i+\delta}{2^k} \right\rceil - d \\ &\ge \left\lceil \sum_{i\in[d]} \frac{x_i+\delta}{2^k} \right\rceil - d \ge y^* + \left\lfloor \frac{d\delta}{2^k} \right\rfloor - d \end{aligned} \tag{22}$$

Setting $\delta = \frac{(d-1)2^{k-1}}{d}$ in Eq. (21) and Eq. (22) gives Eq. (3). □

A.3. Proof of Theorem 1

Proof. Let $\mathbf{M}$ be the random matrix given in the MSIS instance. W.l.o.g we assume $\mathbf{M}$ is of the form $[\mathbf{M}' \mid -\mathbf{I} \mid \mathbf{v}]$, with $\mathbf{M}' \in \mathcal{R}_q^{(k-\ell)\times\ell}$. We sample $(\mathbf{A}_1, \mathbf{t}_1) \leftarrow \mathcal{R}_q^{\ell\times\ell} \times \mathcal{R}_q^{l}$, and set:

$$\mathbf{A} := \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{M}'\cdot\mathbf{A}_1\end{bmatrix}, \qquad \mathbf{t} := \begin{bmatrix}\mathbf{t}_1 \\ \mathbf{v} - \mathbf{M}'\cdot\mathbf{t}_1\end{bmatrix}.$$

Note that both $\mathbf{A}$ and $\mathbf{t}$ are uniform, and give them to the adversary $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(c_{\mathsf{poly}}, \mathbf{z}, \mathbf{h})$, we can assume w.l.o.g that $H(\lfloor \mathbf{A}\cdot\mathbf{z} - p_\mathbf{t}\cdot c_{\mathsf{poly}}\cdot\mathbf{t}\rceil_{q\to q_\mathbf{w}} + \mathbf{h}, \mathsf{msg})$ was queried by $\mathcal{A}$. When receiving a forgery we rewind the adversary to obtain a new forgery $(c_{\mathsf{poly}}, \mathbf{z}', \mathbf{h}')$ such that

$$\begin{aligned} &\lfloor \mathbf{A}\cdot\mathbf{z} - p_\mathbf{t}\cdot c_{\mathsf{poly}}\cdot\mathbf{t}\rceil_{q\to q_\mathbf{w}} + \mathbf{h} \\ =\ &\lfloor \mathbf{A}\cdot\mathbf{z}' - p_\mathbf{t}\cdot {c_{\mathsf{poly}}}'\cdot\mathbf{t}\rceil_{q\to q_\mathbf{w}} + \mathbf{h}' \end{aligned}$$

Which can be rewritten as:

$$\begin{aligned} &\mathbf{A}\cdot\mathbf{z} - p_\mathbf{t}\cdot c_{\mathsf{poly}}\cdot\mathbf{t} + \boldsymbol{\delta} + p_\mathbf{w}\cdot\mathbf{h} \\ =\ &\mathbf{A}\cdot\mathbf{z}' - p_\mathbf{t}\cdot {c_{\mathsf{poly}}}'\cdot\mathbf{t} + \boldsymbol{\delta}' + p_\mathbf{w}\cdot\mathbf{h}', \end{aligned}$$

where $\boldsymbol{\delta}$ and $\boldsymbol{\delta}'$ are the errors entailed by the modulus shiftings $\lfloor\cdot\rceil_{q\to q_\mathbf{w}}$. Which is equivalent to

$$\mathbf{A}\cdot\tilde{\mathbf{z}} + \tilde{\mathbf{e}} = p_\mathbf{t}\cdot\tilde{c}_{\mathsf{poly}}\cdot\mathbf{t}, \text{ where } \begin{cases}\tilde{\mathbf{z}} = \mathbf{z} - \mathbf{z}', \\ \tilde{\mathbf{e}} = \boldsymbol{\delta} + p_\mathbf{w}\cdot\mathbf{h} - \boldsymbol{\delta}' - p_\mathbf{w}\cdot\mathbf{h}', \\ \tilde{c}_{\mathsf{poly}} = c_{\mathsf{poly}} - {c_{\mathsf{poly}}}'. \end{cases}$$

If we note $\mathbf{B} := \left[\mathbf{A}_2\cdot\mathbf{A}_1^{-1} \mid -\mathbf{I}\right]$, then $\mathbf{B}\mathbf{A} = \mathbf{0}$ and:

$$\mathbf{B}\cdot\tilde{\mathbf{e}} = p_\mathbf{t}\cdot\tilde{c}_{\mathsf{poly}}\cdot\mathbf{B}\cdot\mathbf{t},$$

Since $\mathbf{M} = \left[\mathbf{B} \mid \mathbf{v}\right]$ and by definition of $\mathbf{t}$, this implies:

$$\mathbf{M}\cdot\begin{bmatrix}\tilde{\mathbf{e}} \\ p_\mathbf{t}\cdot c_{\mathsf{poly}}\end{bmatrix} = \mathbf{0},$$

which concludes the proof. □

A.4. Masking security of the key generation algorithm

In this section, we introduce all the technical details for proving the t-probing security of Raccoon Keygen algorithm.

Note that we will ignore the shift δ used in the $\mathsf{ApproxShift}_{q\to q_\mathbf{t}}$ algorithm as this makes proofs a lot more readable, its purpose is to add a static shift to the key so that its error is centered on 0, as such it does not affect the masking security.

We first introduce a lemma highlighting the technical issue of the correlation between an intermediate value and the secret through a MLWRE equation.

Lemma 6. *For $[\![\mathbf{v}]\!]$ the output of* $\mathsf{OrderSwitch}$ *in Fig. 4, let $\mathbf{t}$ be the output of* $\mathsf{Decode}\circ\mathsf{ApproxShift}_{q\to q_\mathbf{t}}$ *in Fig. 4, we have:*

$$\mathbf{t} = (\mathbf{v} \gg k) - \left(\underbrace{\left(\sum_{i=0}^{2t}(\mathbf{v}_i \bmod 2^k)\right)}_{\in[0,2t(2^k-1)]} \gg k\right)$$

Proof. By definition of $\mathsf{ApproxShift}_{q\to q_\mathbf{t}}$, $\mathbf{t} = \sum_0^{2t}(\mathbf{v}_i \gg k)$. We can also rewrite $\mathbf{v}_i$ as:

$$\mathbf{v}_i = (\mathbf{v}_i \gg k)\cdot 2^k + (\mathbf{v}_i \bmod 2^k)$$

By summing, we obtain:

$$\mathbf{v} = \sum_0^{2t}\mathbf{v}_i = \left(\sum_0^{2t}\mathbf{v}_i \gg k\right)\cdot 2^k + \sum_0^{2t}(\mathbf{v}_i \bmod 2^k)$$

Note that:

$$\left(\sum_0^{2t}(\mathbf{v}_i \bmod 2^k)\right) \bmod 2^k = \left(\sum_0^{2t}\mathbf{v}_i\right) \bmod 2^k = 0$$

From which we get the result. □

Lemma 6 underlines a particular link between the shares of an intermediate value $\mathbf{v}$ and the secret key $\mathbf{s}$ via a MLWRE equation involving the public key. We can rewrite the equation of Lemma 6 as

$$\underbrace{\mathbf{t}}_{\in \mathsf{vk}} = \lfloor \mathbf{A}\mathbf{s} \rceil_{q \to q_{\mathbf{w}}} - \underbrace{\left(\sum_{i=0}^{2t} (\mathbf{v}_i \bmod 2^k) \right) \gg k}_{\text{Extra error}}.$$

We see that the first term corresponds to the rounding and the second term adds some extra error. The knowledge of a subset of shares $\mathbf{v}_i$ implies a reduction of the error and a potential easier MLWRE problem. However, we will prove in the sequel (Theorem 6) that the parameters are set up such that the MLWRE instance stays hard even if the extra error is a sum of t terms. In other words, even if the attacker has the exact value of t shares of $\mathbf{v}$, the MLWRE instance stays hard. Let us first introduce a masking result.

Theorem 5. *Assume that an attacker has access to $n^{\text{prob}} \leq t$ observations on the whole key generation scheme (Algorithm 6). All these observations can be perfectly simulated with the public information (*vk *and the parameters) and at most t shares of $[\![\mathbf{v}]\!]$.*

Proof. The overall gadget decomposition of the algorithm is in Fig. 4.

This proof is not a usual composition proof because the joint distribution of $\mathbf{v}$ and vk is correlated to the secret (see Lemma 6).

We will then divide the proof in two parts: simulation before $\mathsf{OrderSwitch}_{d \to d'}$ and one simulation after. First of all, as in hypothesis, let us assume that an attacker has access to $n^{\text{prob}} \leq t$ observations on the whole key generation scheme. Let us consider the following distribution of the attacker's n^{prob} observations: n_1^{prob} on Decode, n_2^{prob} on $\mathsf{ApproxShift}_{q \to q_{\mathbf{w}}}$, n_3^{prob} on $\mathsf{OrderSwitch}_{d \to d'}$, n_4^{prob} on the multiplication with $\mathbf{A}$, n_5^{prob} on the randomness generation. Some other observations can be made on the computation of the matrix $\mathbf{A}$, but we do not fix a notation for this number. Finally, we have $\sum_{i=1}^{5} n_i^{\text{prob}} \leq n^{\text{prob}}$.

Similarly to the proof of Theorem 3, we can prove that all the observations strictly after $\mathsf{OrderSwitch}_{d \to d'}$ can be perfectly simulated with at most $n_1^{\text{prob}} + n_2^{\text{prob}} \leq n^{\text{prob}}$ shares of $[\![\mathbf{v}]\!]$.

Next, we continue the proof. Due to the t-SNI property of $\mathsf{OrderSwitch}_{d \to d'}$, all the observations performed strictly after $\mathsf{OrderSwitch}_{d \to d'}$ are not propagated but due to Lemma 6 they are not discarded. Due to the t-SNI property of $\mathsf{OrderSwitch}_{d \to d'}$ and the t-NI property of the multiplication with $\mathbf{A}$, all the observations performed strictly after the randomness generation and before the end of $\mathsf{ApproxShift}_{q \to q_{\mathbf{w}}}$ can be perfectly simulated with at most $n_3^{\text{prob}} + n_4^{\text{prob}} \leq n^{\text{prob}}$ shares of $[\![\mathbf{s}]\!]$. Finally, the left randomness generation gadget is also t-NI secure and does require any input; thus, these observations are not propagated.

Let us summarize:

- all the observations performed during the key generation before the end of $\mathsf{ApproxShift}_{q \to q_{\mathbf{w}}}$ can be perfectly simulated from public parameters.
- all the observations performed during the key generation strictly after $\mathsf{OrderSwitch}_{d \to d'}$ can be perfectly simulated with at most $n_1^{\text{prob}} + n_2^{\text{prob}} \leq n^{\text{prob}} \leq t$ shares of $[\![\mathbf{v}]\!]$ and public parameters.

□

Theorem 6. *Under* MLWRE, *an adversary with at most d shares of $[\![\mathbf{v}]\!]$ cannot distinguish* vk *from uniform.*

Proof. As per Lemma 6, the output $(\mathbf{A}, \mathbf{t})$ of Keygen is such that

$$\mathbf{t} = \mathbf{v} \gg k - \left(\sum_{i=0}^{2t} (\mathbf{v}_i \bmod 2^k) \right) \gg k.$$

Where $[\![\mathbf{v}]\!]$ is a fresh masking of $\mathbf{A}\mathbf{s}$ of order $2t$. Hence $\mathbf{v} = \mathbf{A}\mathbf{s}$, and given any subset $I \subset [2t]$ of size at most d, after fixing the value of $(\mathbf{v}_i)_{i \in I}$, the distribution of the remaining $(\mathbf{v}_i)_{i \notin I}$ is uniform. By decomposing $\sum_{i \in I} (\mathbf{v}_i \bmod 2^k)$ as

$$\sum_{i \in I} (\mathbf{v}_i \bmod 2^k) = \mathbf{a} \cdot 2^k + \mathbf{b}$$

we can write

$$\left(\sum_{i=0}^{2t} (\mathbf{v}_i \bmod 2^k) \right) \gg k = \mathbf{a} + \left(\mathbf{b} + \sum_{i \notin I} (\mathbf{v}_i \bmod 2^k) \right) \gg k$$
$$= \mathbf{a} + \mathbf{e}$$

where $\mathbf{e}$ follows a discrete Irwin–Hall distribution, denoted as χ, of standard deviation $p_{\mathbf{t}} \sqrt{t/12}$ and center 0.

Let us rewrite the MLWRE instance:

$$\mathbf{t} + \mathbf{a} = \lfloor \mathbf{A}\mathbf{s} \rceil_{q \to q_{\mathbf{w}}} - \mathbf{e}.$$

By Definition 2 the vector $\mathbf{t}$ is indistinguishable from uniform over $\mathcal{R}_{q_{\mathbf{t}}}$ under the $\mathsf{MLWRE}_{\mathcal{R}_q, \ell, k, \chi}$ which is at least as hard as $\mathsf{MLWE}_{\mathcal{R}_q, \ell, k-\ell, \chi}$. □