High-Order Masking of Lattice Signatures in Quasilinear Time

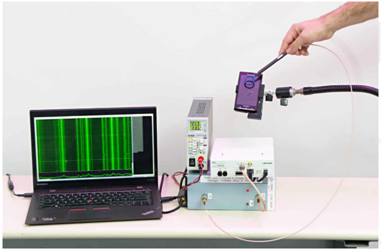
Rafaël del Pino [1]   Thomas Prest [1]
Mélissa Rossi [2]   Markku-Juhani O. Saarinen [1]

[1] PQShield   [2] ANSSI

44th IEEE Symposium on Security and Privacy
22 May 2023, San Francisco, CA

Side-Channel Attacks and Post-Quantum

- **Side-Channel Attacks (SCA)** use external measurements such as latency (TA), power consumption (SPA/DPA), or electromagnetic emissions ([S/D]EMA) to extract secrets.
- *SCA resistance is important for PC, IoT, and mobile device "platform security" (secure boot, firmware updates, attestation), authentication tokens, smart cards, HSMs / secure elements..*
- Common compliance & market requirement for hardware (Common Criteria / AVA_VAN, FIPS 140-3 / ISO 17825).



- **Post-Quantum Cryptography (PQC)** implementations – e.g. lattice-based signature schemes **Dilithium** and **Falcon** inherit all of the security and compliance requirements of Elliptic Curve or RSA based solutions in applications.

Masked Raccoon: Side-Channel Secure Signatures

① **Masked Raccoon** is a member of the new **Raccoon** family of lattice-based PQC signature schemes.

② **Side-Channel Security** is proved in the Strong Non-Inteference (SNI) framework.

③ **Cryptanalytic Security** is proved in relation to well-studied MLWE and SelfTargetMSIS problems.

④ **Performance** is evaluated with both PC and a constrained FPGA hardware target.

Masking: Non-Invasive SCA Security

- **Masking:** Secret data $[\![\mathbf{s}]\!]$ is processed in d = **order** + 1 randomized shares $\mathbf{s}_i$.

$$\text{Boolean Masking: } [\![\mathbf{s}]\!] = \mathbf{s}_1 \oplus \mathbf{s}_2 \oplus \cdots \oplus \mathbf{s}_d$$
$$\text{Arithmetic Masking: } [\![\mathbf{s}]\!] = \mathbf{s}_1 + \mathbf{s}_2 + \cdots + \mathbf{s}_d \pmod{q}.$$

- Like secret sharing: Knowledge of $d - 1$ shares $\mathbf{s}_i$ does not reveal anything about $[\![\mathbf{s}]\!]$.
- If you only have partial or "noisy" measurements (traces), it has been shown that the number of such observations required to learn $[\![\mathbf{s}]\!]$ grows *exponentially* with d.

- **Masking proofs** give formal, algorithm-level assurance against side-channel leakage.
- The proofs can be made in several models; the Ishai-Sahai-Wagner (ISW) t-probing security requires that any t internal intermediate values don't reveal secrets.
- The noisy leakage model is an alternative; links have been proven between t-probing security, noisy leakage model, and information-theoretic attack complexity bounds.

Complexity of Masking: Linear, Quadratic, Quasilinear

- Linear operations only need **linear** $O(d)$ effort to mask:

  Addition / subtraction / XOR of masked variables ($[\![\mathbf{s}]\!] + [\![\mathbf{r}]\!]$), *multiplication (or Boolean AND, OR) with a scalar constant or a public variable* ($\mathbf{c} \cdot [\![\mathbf{s}]\!]$), *or share-independent linear operations such as NTT (Number Theoretic Transform.)*

- Non-linear operations generally require **quadratic** $O(d^2)$ effort:

  Multiplication (Boolean AND, OR) between secret variables ($[\![\mathbf{s}]\!] \cdot [\![\mathbf{r}]\!]$), *conversion between Arithmetic and Boolean masking (A2B and B2A), or symmetric cryptography like SHA3.*

- But some non-linear operations can be done with **quasilinear** $O(d \log d)$ effort:

  Practical quasilinear techniques are known only for a limited number of computational tasks.

Why is Dilithium Hard To Mask?

- **Dilithium** requires a masked SHAKE; mixes bit manipulations with (mod q) arithmetic, requiring A2B and B2A; has masked comparisons / rejection sampler.

  (For these non-linear operations only quadratic $O(d^2)$ gadgets are known.)

- **Raccoon** avoids quadratic operations.

  The cost of additional shares is nearly constant. *(Cycles/share even decreases initially due to a small constant overhead.)*

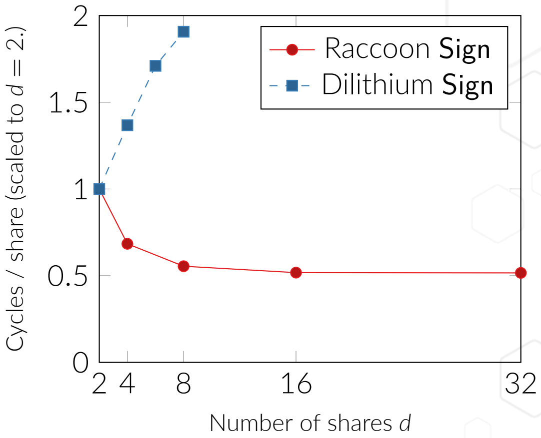


Figure 1: Cost of masking: Signing cycle count divided by d, normalized to a common start at 1 for $d = 2$. Dilithium data from [24, Table 3].

Blueprint: “Fiat–Shamir with aborts”

- Blueprint from Lyubashevsky [15,16], refined by Bai and Galbraith [17], and used in Dilithium and this work.
- Public key $\mathsf{vk} = (\mathbf{A}, \mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ is a *Module Learning With Errors*, or **MLWE** sample. Additionally, the security proof uses **SelfTargetMSIS** (as in Dilithium).
- There actually aren’t “secret-secret” multiplications in the blueprint! *Could we build it entirely with quasilinear gadgets?*

Algorithm 1 PrototypeSign(sk, vk, msg)

```
Input: A signing key sk = s, a verification key
    vk = (A, t), a message msg.
Output: A signature sig of msg under sk.
1: Sample r uniformly in a small set S
2: u := A · r
3: w := Truncate(u)                        ▷ Commitment
4: c := H(w, msg)                          ▷ Challenge
5: z := r + c · s                          ▷ Response
6: y := A · z − c · t
7: if CheckCondition(z, y) = False then
8:     goto Line 1                         ▷ Rejection sampling
9: return sig := (c, z)
```

Raccoon Masking Proof: Composition of Gadgets

- Cryptanalytic sensitivity analysis: Which variables need to be protected?
- Raccoon signature and key generation functions are composed of **Masking Gadgets** that are individually t-non-intefering ($t -$ NI) or t-strong non-interfering ($t -$ SNI).
- The scheme is designed to be “masking friendly,” so the proofs are quite standard.

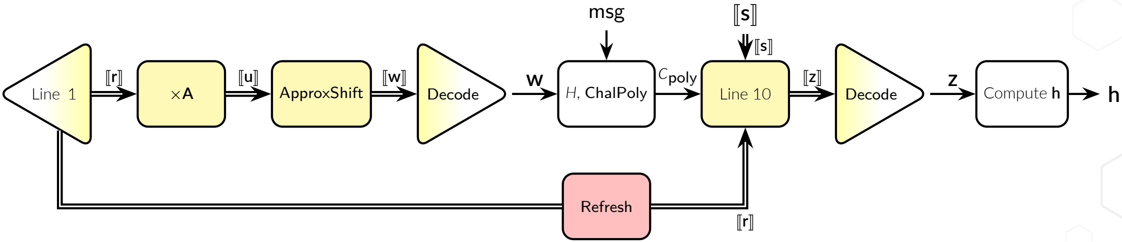


Figure 2: Raccoon signature function. Colors: gadget proven $t -$ NI, gadget proven $t -$ SNI, gadget unmasked. Single arrows ($\longrightarrow$) and double arrows ($\Longrightarrow$) represent plain and masked values.

Example: Gadgets, Proofs, and Implementation

Example: $O(d \log d)$ masking refresh (re-randomization) gadget, proven t – SNI [35,36,37].

Algorithm 2 Refresh()

```
Input: A d-shared $[\![x]\!]$ of $x \in \mathbb{Z}_q$
Output: A fresh d-shared $[\![x]\!]$ of $x$
$[\![z]\!] \leftarrow$ ZeroEncoding()
return $[\![x]\!] = [\![x]\!] + [\![z]\!]$
```

Algorithm 3 ZeroEncoding()

```
Input: A power-of-two integer $d$, a ring $\mathbb{Z}_q$
Output: A d-shared $[\![z]\!] \in \mathbb{Z}_q^d$ of $0 \in \mathbb{Z}_q$
if $d = 1$ then
    return $[\![z_1]\!] = (0)$                         ▷ Order zero.
$[\![z_1]\!]_{d/2} \leftarrow$ ZeroEncoding($d/2$)
$[\![z_2]\!]_{d/2} \leftarrow$ ZeroEncoding($d/2$)
$[\![r]\!]_{d/2} \leftarrow \mathbb{Z}_q^{d/2}$         ▷ Uniform random vector.
$[\![z_1]\!]_{d/2} = [\![z_1]\!]_{d/2} + [\![r]\!]_{d/2}$
$[\![z_2]\!]_{d/2} = [\![z_2]\!]_{d/2} - [\![r]\!]_{d/2}$
return $[\![z]\!]_d = \left([\![z_1]\!]_{d/2} \,\|\, [\![z_2]\!]_{d/2}\right)$
```

- Proofs examine correlations between intermediate variables, input/output.
- (Hardware implementation has circuits to generate masking randomness efficiently and perform all the ring arithmetic ops.)

MLWE/MSIS Security Proof and Parameter Selection

- Hybrid Lemma 3 bounds a forger $\frac{\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sign}}}{Q_s}$ to distinguishing public key from uniform $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PK}}$.
- Thm. 1 provides a reduction to MSIS. Further consideration of SelfTargetMSIS is used in parameter selection (BKZ attack Core-SVP.)
- Thm. 2 reduces PK distinguishability to MLWE.

Parameter Selection

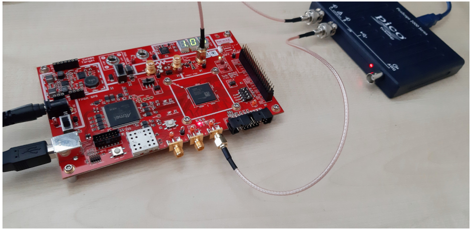
- As usual for lattice schemes, parameter selection for each security target λ_{target} is a complex multi-objective optimization problem.
- However, the core problems and high-level structure are well-studied, so we can rely on a large body of existing research.

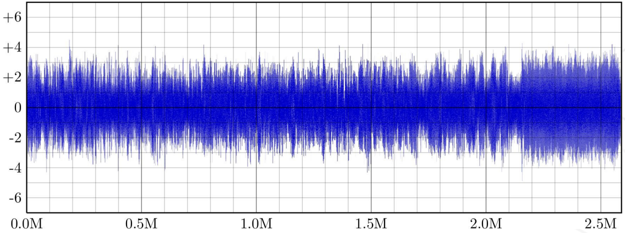
Name	Raccoon-λ_{target}		
λ_{target}	128	192	256
Q_s	2^{48}	2^{48}	2^{49}
d	32	-	-
$\log q$	$49^{\dagger}$	-	-
$\log p_{\mathbf{t}}$	10	6	7
$\log p_{\mathbf{w}}$	43	40	42
n	512	-	-
k	8	11	14
ℓ	3	5	6
ω	19	31	44
B_2^2	2^{14}	2^{14}	2^{15}
B_∞	8	-	-
$\lvert\mathbf{vk}\rvert$	19 968	30 272	37 632
$\lvert\mathbf{sig}\rvert$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

†Across all parameter sets, we set $q = (2^{25} - 2^{18} + 1) \cdot (2^{24} - 2^{18} + 1)$.

Implementations and Experiments

- Portable C Implementation was developed to assess the relative speed to other algorithms. Unmasked Dilithium runs at about $1/2$ time of than Raccoon with $d = 2$ masking. Unfortunately not many comparison points (no open masked SW Dilithium.)
- Artix7 FPGA target implements Raccoon up $d = 32$ and also has $d = 2$ proprietary Dilithium HW. Raccoon is already faster at first order, tens of times faster with higher d.
- No secret key leakage was detected in a 200,000-trace ISO 17825 / “TVLA” style leakage assessment of Raccoon-128 ($d = 2$) signature function on the FPGA target.





Masked Raccoon: Conclusions

Contributions:

1. In this work, we have shown that lattice-based signature schemes can be masked with quasilinear complexity – giving the "defenders" a significant asymptotic advantage.
2. Proposed new algorithmic techniques, as well as new proof techniques.
3. Software and hardware experiments show that the performance and concrete leakage profile of Raccoon are consistent with our theoretical analyses (+new masking records!)

Note: *We have further developed the Raccoon framework since this work was submitted and have found new techniques and applications. Also, the parameter selection has changed.*

We are currently working (with an expanded team) to release a new version of Raccoon.