Plover

Muhammed F. Esgin
Monash University

Thomas Espitau
PQShield

Guilhem Niot
PQShield
University Rennes 1

Thomas Prest[1]
PQShield

Amin Sakzad
Monash University

Ron Steinfeld
Monash University

[1]Thanks to Thomas for letting me reuse his slides on Raccoon.

Motivation

Signature schemes strike a balance between:

- Sizes (verification key and signatures)
- Speed (signing, verification)
- Portability
- Conservative assumptions
- **Resistance against side-channel attacks**

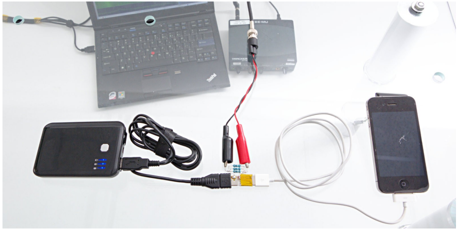
And so on...

Criteria					
Dilithium	★★½	★★★	★★★	★★	😭
Falcon	★★★	★★★	★★	★★	😭
SPHINCS+	★½	★★	★★	★★★	😭
Raccoon	★★	★★★	★★★	★★	★★★
Plover	★★	★★★	★★★	★★	★★★

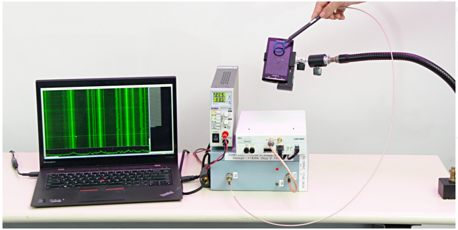
Side-Channel Attacks

Side-channel attacks in cryptography

Power consumption [KJJ99]



Electromagnetic emissions [Eck85]



Timing measurement [Koc96]

Acoustic emissions [AA04]

Example with Falcon

In Falcon, a signature **sig** is distributed as a Gaussian.
The signing key **sk** should remain private.
The power consumption leaks information about the dot product $\langle \mathbf{sig}, \mathbf{sk} \rangle$, or **sk** itself.

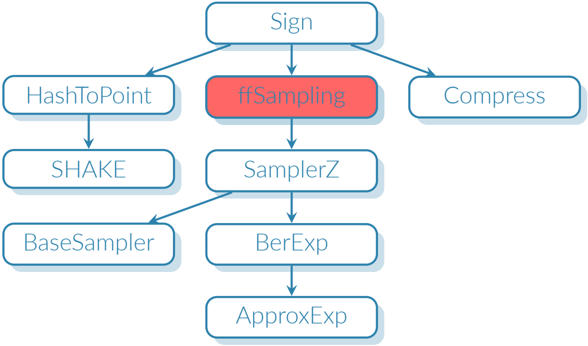
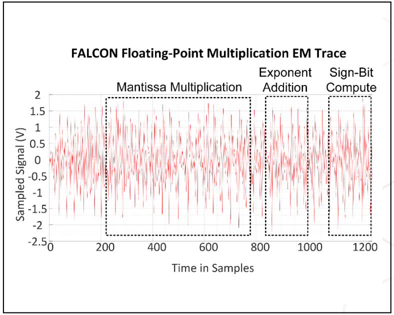


Figure 1: Flowchart of the signature



Learning **sk** directly

[1] *FALCON Down: Breaking FALCON Post-Quantum Signature Scheme through Side-Channel Attacks* [KA21]

Example with Falcon

In Falcon, a signature **sig** is distributed as a Gaussian.
The signing key **sk** should remain private.
The power consumption leaks information about the dot product $\langle \mathbf{sig}, \mathbf{sk} \rangle$, or **sk** itself.

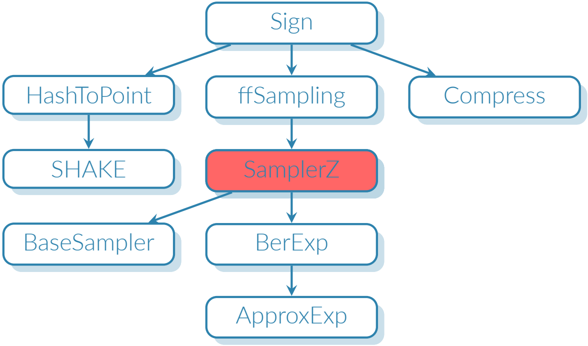
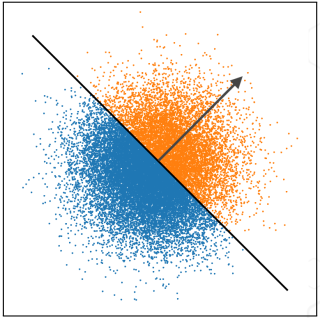


Figure 1: Flowchart of the signature



Filtering $\langle \mathbf{sig}, \mathbf{sk} \rangle > 0$

[2] *Improved Power Analysis Attacks on Falcon* [ZLYW23]

t-probing model

- Adversary can probe t circuit values at runtime
- Unrealistic but a good starting point

Masking

- Each sensitive value x is split in d shares:

$$[\![x]\!] = (x_0, x_1, \ldots, x_{d-1}) \tag{1}$$

such that

$$x_0 + x_1 + \cdots + x_{d-1} = x \bmod q \qquad \text{(additive)}$$

$$\text{or } x_0 \oplus x_1 \oplus \cdots \oplus x_{d-1} = x \qquad \text{(boolean)}$$

- In t-probing model, ideally 0 leakage if $d > t$
- In "real life", security is exponential in d
- What about computations?

Masking in practice?

PQ SHIELD

How difficult are operations to mask?

- **Addition ($[\![c]\!] = [\![a+b]\!]$)?**
  - Compute $[\![c]\!] = (a_0 + b_0, \ldots, a_{d-1} + b_{d-1})$, simple and fast: $\Theta(d)$ operations
- **Multiplication ($[\![c]\!] = [\![a \cdot b]\!]$)?**
  - Complex and slower: $\Theta(d^2)$ operations
- **More complex operations?**
  - Use so-called *mask conversions* to convert between additive and boolean masking, very slow: $\gg \Theta(d^2)$ operations

PQSHIELD

How difficult are operations to mask?

- **Dilithium**
  - Generation of short secrets:
    - Sample a uniform value in boolean masking.
    - Convert boolean mask to arithmetic mask ☹
  - Comparison for rejection sampling: conversion to boolean masking ☹
- **Falcon**
  - Gaussian sampling: no efficient way known 😱

Masking Dilithium

Dilithium - signature

PQSHIELD

Dilithium follows the Fiat-Shamir **with aborts** paradigm.

$\mathsf{Sign}(\mathsf{sk} = \mathbf{s}, \mathsf{vk} = (\mathbf{A}, \mathbf{t}), \mathsf{msg}) \to \mathsf{sig}$

1. Generate a short ephemeral secret $\mathbf{r}$
2. Compute the commitment $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$
3. Compute challenge $c = \mathsf{H}(\mathbf{w}, \mathsf{msg}, \mathsf{vk})$
4. Compute the response $\mathbf{z} = \mathbf{s} \cdot c + \mathbf{r}$
5. Check that $\mathbf{z}$ is in a given interval. If not, restart.
6. Signature is $\mathsf{sig} = (c, \mathbf{z})$

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig} = (c, \mathbf{z}))$

1. Verify that $\mathbf{z}$ is small.
2. Recover $\mathbf{w} = \mathbf{A} \cdot \mathbf{z} - c \cdot \mathbf{t}$
3. Verify that $c = \mathsf{H}(\mathbf{w}, \mathsf{msg}, \mathsf{vk})$

Dilithium - signature

Dilithium follows the Fiat-Shamir **with aborts** paradigm.

$\mathsf{Sign}(\mathsf{sk} = \mathbf{s}, \mathsf{vk} = (\mathbf{A}, \mathbf{t}), \mathsf{msg}) \rightarrow \mathsf{sig}$

1. Generate a short ephemeral secret $\mathbf{r}$ ▷ Hard
2. Compute the commitment $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$ ▷ Easy
3. Compute challenge $c = \mathsf{H}(\mathbf{w}, \mathsf{msg}, \mathsf{vk})$ ▷ No mask
4. Compute the response $\mathbf{z} = \mathbf{s} \cdot c + \mathbf{r}$ ▷ Easy
5. Check that $\mathbf{z}$ is in a given interval. If not, restart. ▷ Hard
6. Signature is $\mathbf{sig} = (c, \mathbf{z})$

Masking bottlenecks:

- 😱 Short secret generation (1) requires B2A.
- 😱 Rejection sampling (5) requires A2B and B2A.

Total masking overhead: $\Theta(d^2 \log q)$

Raccoon - masked signature

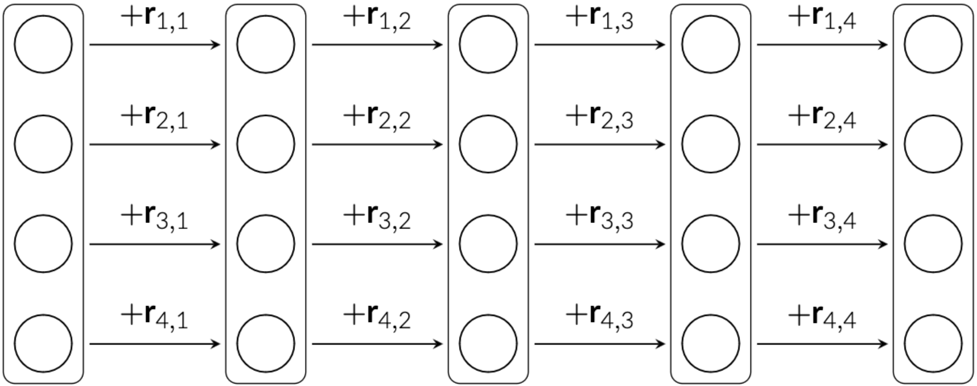
PQ SHIELD

$\mathsf{Sign}(\mathsf{sk} = [\![\mathbf{s}]\!], \mathsf{vk} = (\mathbf{A}, \mathbf{t}), \mathsf{msg}) \to \mathsf{sig}$

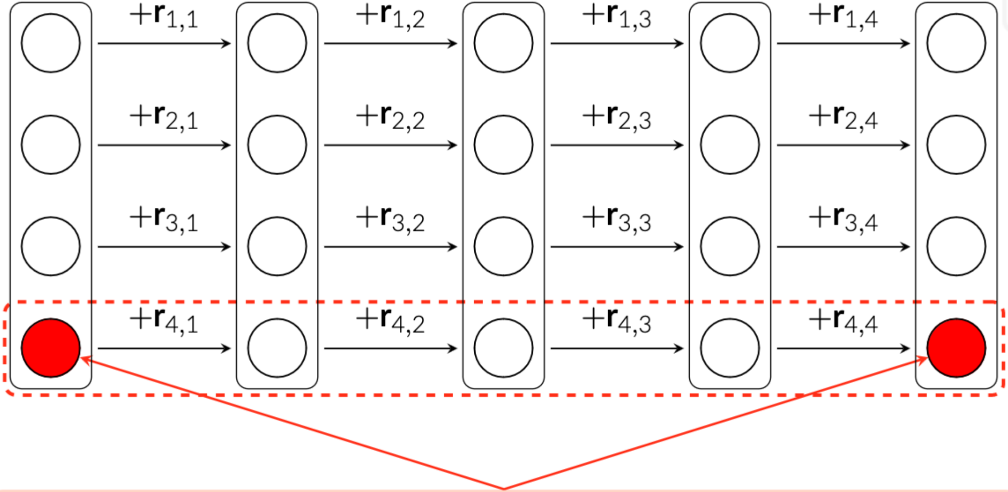
1. Generate a masked short ephemeral secret $[\![\mathbf{r}]\!]$ using "AddRepNoise" ▷ Easy
2. Compute the commitment $[\![\mathbf{w}]\!] = \mathbf{A} \cdot [\![\mathbf{r}]\!]$ ▷ Easy
3. Unmask $[\![\mathbf{w}]\!]$ to obtain $\mathbf{w}$ ▷ Easy
4. Compute the challenge $c = \mathsf{H}(\mathbf{w}, \mathsf{msg}, \mathsf{vk})$ ▷ No mask
5. Compute the response $[\![\mathbf{z}]\!] = [\![\mathbf{s}]\!] \cdot c + [\![\mathbf{r}]\!]$ ▷ Easy
6. Unmask $[\![\mathbf{z}]\!]$ to obtain $\mathbf{z}$ ▷ Easy
7. (No more rejection sampling!)
8. Signature is $\mathsf{sig} = (c, \mathbf{z})$

Total masking overhead: $O(d \log d)$

What happens inside AddRepNoise?

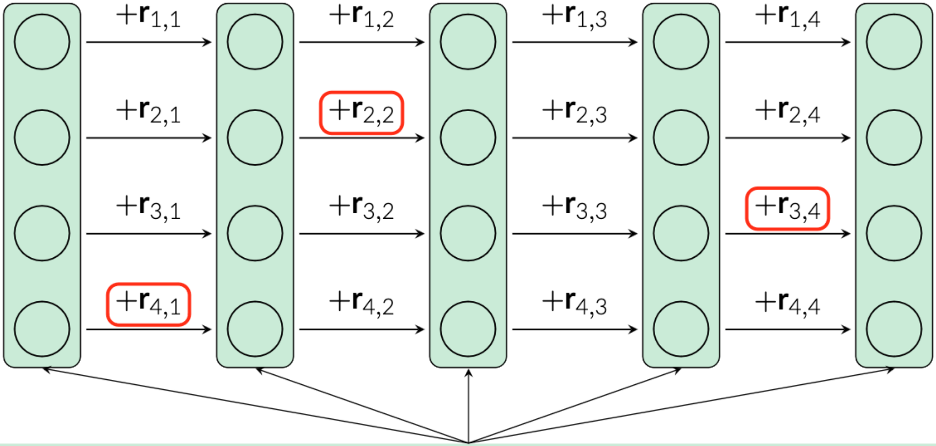


What happens inside AddRepNoise?



Problem: a probing adversary can learn the sum of T random in 2 probes.

What happens inside AddRepNoise?



Solution: add refresh gadgets to separate the algorithm in independent layers
Now a probing adversary learns at most (the sum of) t short noises.

Security of Raccoon?

- Vanilla Raccoon,
  - Randomness $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$ is public
  - No rejection sampling: signatures leak part of the secret

Security of Raccoon?

- Vanilla Raccoon,
  - Randomness $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$ is public
  - No rejection sampling: signatures leak part of the secret

Definition 1 (Hint-MLWE)

It is hard to distinguish $(\mathbf{A}, \mathbf{u}, (c_i \cdot \mathbf{s} + \mathbf{r}_i)_i)$ with (c_i) small,

- when $\mathbf{u}$ is random
- or, when $\mathbf{u} = \mathbf{A} \cdot \mathbf{s}$ an MLWE sample

Assuming at most Q hints, Hint-MLWE is as hard as MLWE when taking $\mathbf{r}_i$ of standard deviation $\approx \sqrt{Q}\|c\|$.

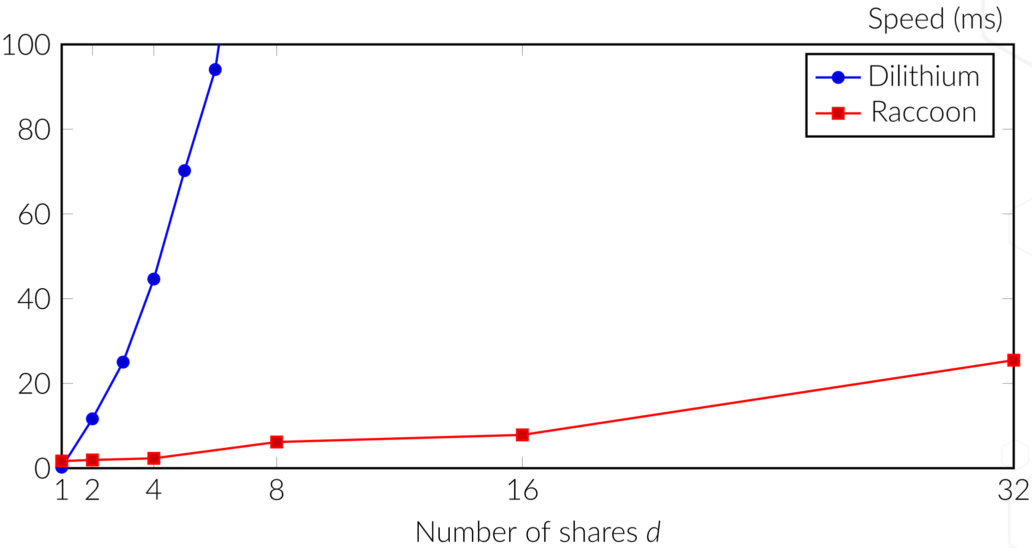
☺ Vanilla Raccoon is secure when taking large enough perturbations $\mathbf{r}_i$.

Security of Raccoon?

- Vanilla Raccoon,
  - Randomness $\mathbf{w} = \mathbf{A} \cdot \mathbf{r}$ is public
  - No rejection sampling: signatures leak part of the secret
- Masked Raccoon:
  - Leak part of the perturbation $\mathbf{r} = \mathsf{AddRepNoise}()$.

In t-probing model, write $\mathbf{r} = \mathbf{r}_{\mathsf{safe}} + \mathbf{r}_{\mathsf{leaked}}$.
If **rep** iterations in AddRepNoise, $\mathbf{r}_{\mathsf{safe}}$ has standard deviation $\sqrt{d \cdot \mathbf{rep} - t} \cdot \sigma_{\mathbf{r}}$.
Security of **Masked Raccoon reduces to Vanilla Raccoon** with small loss.

Performances on a Desktop



With some tricks [SR23], RAM consumption is < 128 kB

Masked Hash-and-Sign signatures

Keygen(1^λ)

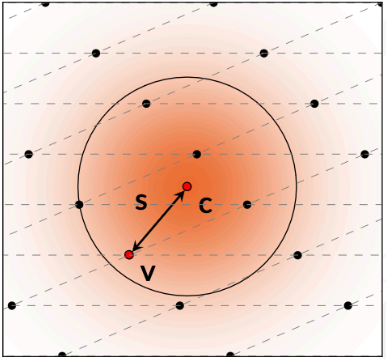
1. Gen. matrices $\mathbf{A}$, $\mathbf{B}$ s.t.:
   - $\mathbf{A}$ is pseudo-random.
   - $\mathbf{B} \cdot \mathbf{A} = 0$.
   - $\mathbf{B}$ has small coefficients.
2. $\mathsf{vk} := \mathbf{A}$, $\mathsf{sk} := \mathbf{B}$

Sign$(\mathsf{sk} = \mathbf{B}, \mathsf{msg})$

1. Compute $\mathbf{c}$ such that $\mathbf{c} \cdot \mathbf{A} = H(\mathsf{msg})$
2. $\mathbf{v} \leftarrow$ vector in $\mathcal{L}(\mathbf{B})$, close to $\mathbf{c}$.
3. $\mathsf{sig} := \mathbf{s} = (\mathbf{c} - \mathbf{v})$

Verify$(\mathsf{vk} = \mathbf{A}, \mathsf{msg}, \mathsf{sig} = \mathbf{s})$

Check that ($\mathbf{s}$ is short) and ($\mathbf{s} \cdot \mathbf{A} = H(\mathsf{msg})$)



😱 But masking Gaussian sampling efficiently remains an open problem.

What about Mitaka?

- In 2022, *Mitaka: a simpler, parallelizable, maskable variant of Falcon* [EFG+22]
  - But, *A Key-Recovery Attack against Mitaka in the t-Probing Model* [Pre23]

Mitaka cannot be masked efficiently with current techniques.

A masking friendly hash-and-sign scheme?

Eagle was recently introduced by Yu et al. in [YJW23].

Eagle.Keygen()

1. Generate matrices $\mathbf{A}, \mathbf{T}$ s.t.:
   - $\mathbf{A}$ is pseudo-random
   - $\mathbf{T} \cdot \mathbf{A} = \beta \cdot \mathbf{I}$
   - $\mathbf{T}$ has small coefficients
2. $\mathsf{vk} := \mathbf{A}, \mathsf{sk} := \mathbf{T}$

Eagle.Verify(msg, sig $= \mathbf{z}$)

1. $u := H(\mathsf{msg})$
2. Check that ($\mathbf{z}$ is small) and ($\mathbf{A} \cdot \mathbf{z} = u$)

Eagle.Sign(sk, msg)

1. $\mathbf{u} := H(\mathsf{msg})$
2. $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2\mathbf{I} - r^2\mathbf{T}\mathbf{T}^*}}$
3. $\mathbf{c} := \mathbf{u} - \mathbf{A} \cdot \mathbf{p}$
4. Decompose $\mathbf{c}$ as $\mathbf{c} = \beta \cdot \mathbf{c}_1 + \mathbf{c}_2$
5. $\mathbf{y} \leftarrow D_{\lfloor q/\beta \rceil \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$
6. $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$
7. **return** sig $:= \mathbf{z}$

A masking friendly hash-and-sign scheme?

Eagle was recently introduced by Yu et al. in [YJW23].

Eagle.Sign(sk, msg)

1. $\mathbf{u} := H(\mathsf{msg})$
2. $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2\mathbf{I} - r^2\mathbf{T}\mathbf{T}^*}}$
3. $\mathbf{w} := \mathbf{A} \cdot \mathbf{p}$
4. $\mathbf{c} := \mathbf{u} - \mathbf{w}$
5. Decompose $\mathbf{c}$ as $\mathbf{c} = \beta \cdot \mathbf{c}_1 + \mathbf{c}_2$
6. $\mathbf{y} \leftarrow D_{\lfloor q/\beta \rceil \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$
7. $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$
8. **return** $\mathsf{sig} := \mathbf{z}$

☺ Almost linear scheme, maybe we can do something with it!

A masking friendly hash-and-sign scheme?

Eagle was recently introduced by Yu et al. in [YJW23].

Eagle.Sign(sk, msg)

1. $\mathbf{u} := H(\mathsf{msg})$ ▷ No mask
2. $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2\mathbf{I} - r^2\mathbf{T}\mathbf{T}^*}}$ ▷ Hard
3. $\mathbf{w} := \mathbf{A} \cdot \mathbf{p}$ ▷ Easy
4. $\mathbf{c} := \mathbf{u} - \mathbf{w}$ ▷ No mask
5. Decompose $\mathbf{c}$ as $\mathbf{c} = \beta \cdot \mathbf{c}_1 + \mathbf{c}_2$
6. $\mathbf{y} \leftarrow D_{\lfloor q/\beta \rceil \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$ ▷ Hard
7. $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$ ▷ Easy
8. **return** sig $:= \mathbf{z}$

☺ Almost linear scheme, maybe we can do something with it!

A masking friendly hash-and-sign scheme?

Eagle was recently introduced by Yu et al. in [YJW23].

Eagle.Sign(sk, msg)

1. $\mathbf{u} := H(\mathsf{msg})$ ▷ No mask
2. $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2\mathbf{I} - r^2\mathbf{T}\mathbf{T}^*}}$ ▷ Hard
3. $\mathbf{w} := \mathbf{A} \cdot \mathbf{p}$ ▷ Easy
4. $\mathbf{c} := \mathbf{u} - \mathbf{w}$ ▷ No mask
5. Decompose $\mathbf{c}$ as $\mathbf{c} = \beta \cdot \mathbf{c}_1 + \mathbf{c}_2$
6. $\mathbf{y} \leftarrow D_{\lfloor q/\beta \rceil \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$ ▷ Hard
7. $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$ ▷ Easy
8. **return** $\mathsf{sig} := \mathbf{z}$

Plover.Sign(sk, msg)

1. $\mathbf{u} := H(\mathsf{msg})$
2. $[\![\mathbf{p}]\!] \leftarrow \mathsf{AddRepNoise}()$
3. $\mathbf{w} := \mathsf{Unmask}(\mathbf{A} \cdot [\![\mathbf{p}]\!])$
4. $\mathbf{c} := \mathbf{u} - \mathbf{w}$
5. Decompose $\mathbf{c}$ as $\mathbf{c} = \beta \cdot \mathbf{c}_1 + \mathbf{c}_2$
6. $\mathbf{z} := \mathsf{Unmask}([\![\mathbf{p}]\!] + [\![\mathbf{T}]\!] \cdot \mathbf{c}_1)$
7. **return** $\mathsf{sig} := \mathbf{z}$

☺ Almost linear scheme, maybe we can do something with it!

😁 Introducing Plover, the first hash-and-sign masking-friendly signature scheme.

Security of Plover

- Vanilla Plover
  - Returns responses of the form $\mathbf{z} = \mathbf{p} + \mathbf{T} \cdot \mathbf{c}_1$: hints on the secret.
    ☺ Like Raccoon, rely on Hint-MLWE. Secure for large enough perturbation $\mathbf{p}$.

Security of Plover

- Vanilla Plover
  - Returns responses of the form $\mathbf{z} = \mathbf{p} + \mathbf{T} \cdot \mathbf{c}_1$: hints on the secret.
    ☺ Like Raccoon, rely on Hint-MLWE. Secure for large enough perturbation $\mathbf{p}$.
- Masked Plover
  - As in Raccoon, **AddRepNoise** leaks only a small part of the perturbation $\mathbf{p}$.
    ☺ Unforgeability of Masked Plover in the t-probing model reduces to unforgeability of Vanilla Plover.

A generic framework

Plover introduces a very generic framework for masking friendly schemes:

- Replace non-linear operations with noise flooding. Leakage on the secret mitigated by taking large perturbations **p**.
- Analyse leakage with Hint-MLWE problem.
- Use **AddRepNoise** to sample short vectors. New composable notion $t-SNIu$ to prove security in the t-probing model.

Proofs in the t-probing model

t-probing model

- Adversary can probe t circuit values at runtime

Masking

- Each sensitive value x is split in d shares:

$$[\![x]\!] = (x_0, x_1, \ldots, x_{d-1})$$

such that

$$x_0 + x_1 + \cdots + x_{d-1} = x \bmod q \qquad \text{(additive)}$$

- In t-probing model, ideally 0 leakage if $d > t$

PQ SHIELD

t-probing model

- Adversary can probe t circuit values at runtime

Definition 1 (t-probing security)

A circuit C is t-probing secure, if there exists a simulator $\mathcal{S}$ such that for any input $\mathbf{x}$, and set $\mathcal{P}$ of up to t probes:

$$\mathcal{S}(\mathcal{P}, C_{\textsf{public}}([\![\mathbf{x}]\!])) = \underbrace{C_{\mathcal{P}}([\![\mathbf{x}]\!])}_{\text{Probes on C executed with } \mathbf{x}}$$

i.e., probes are simulatable without knowledge of the circuit input $\mathbf{x}$, only from public output $C_{\textsf{public}}(\mathbf{x})$.

(strong) non-interference framework

- The (strong) non-interference (or (S)NI) framework eases proofs in the t-probing model.
  Composition of simple gadgets: masked additions, multiplications, etc.

(strong) non-interference framework

PQSHIELD

→ The (strong) non-interference (or (S)NI) framework eases proofs in the t-probing model.
Composition of simple gadgets: masked additions, multiplications, etc.

Definition 2 (t-NI)

A circuit C is t-NI, if there exists simulators $\mathcal{S}_1, \mathcal{S}_2$ such that for any input $[\![\mathbf{x}]\!]$, and any set $\mathcal{P}$ of at most t probes:

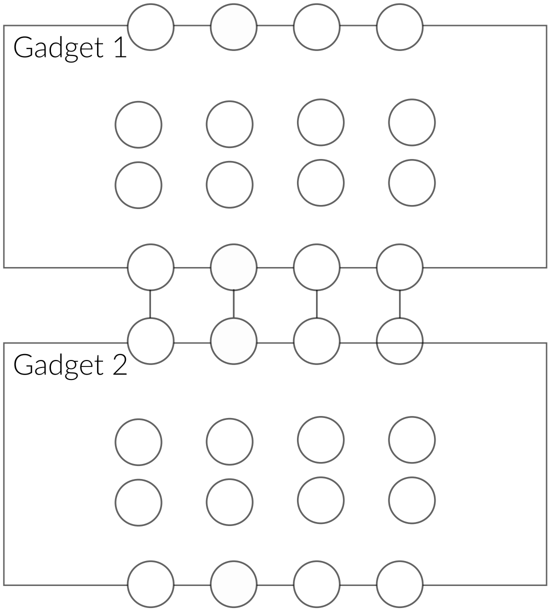
$$i_1, \ldots, i_t := S_1(\mathcal{P})$$
$$S_2(\mathcal{P}, [\![\mathbf{x}]\!]_{i_1}, \ldots, [\![\mathbf{x}]\!]_{i_t}) = C_{\mathcal{P}}([\![\mathbf{x}]\!])$$

i.e. probes are simulatable from at most t shares of the input.

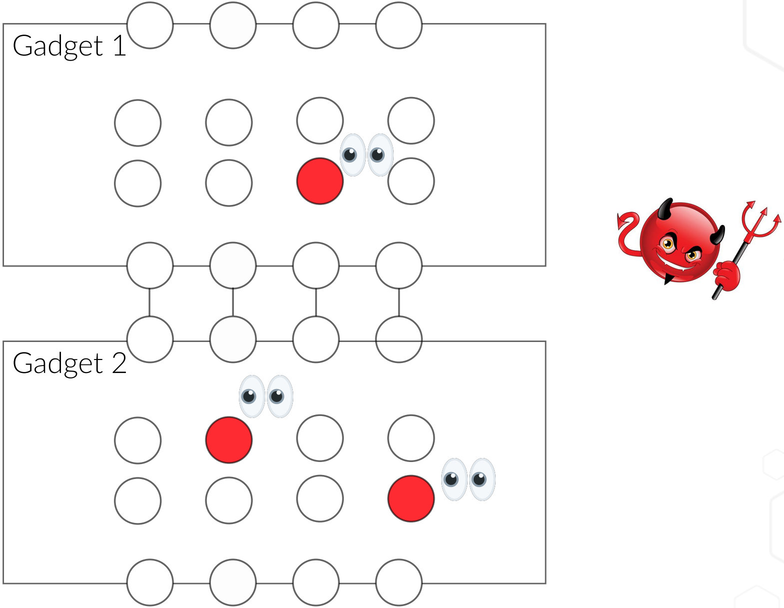
Definition 3 (t-SNI)

Same, but output probes are simulated from internal probes only. Formally, there exists an extra simulator S_3 for probes on output: $S_3(\mathcal{P}_{\mathbf{out}}, S_2(\mathcal{P}_{\mathbf{in}}, \ldots)) = C_{\mathcal{P}_{\mathbf{out}}}([\![\mathbf{x}]\!])$.

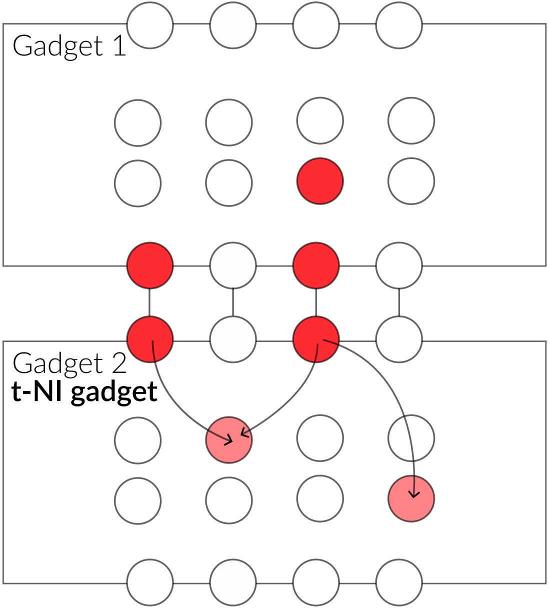
(strong) non-interference framework



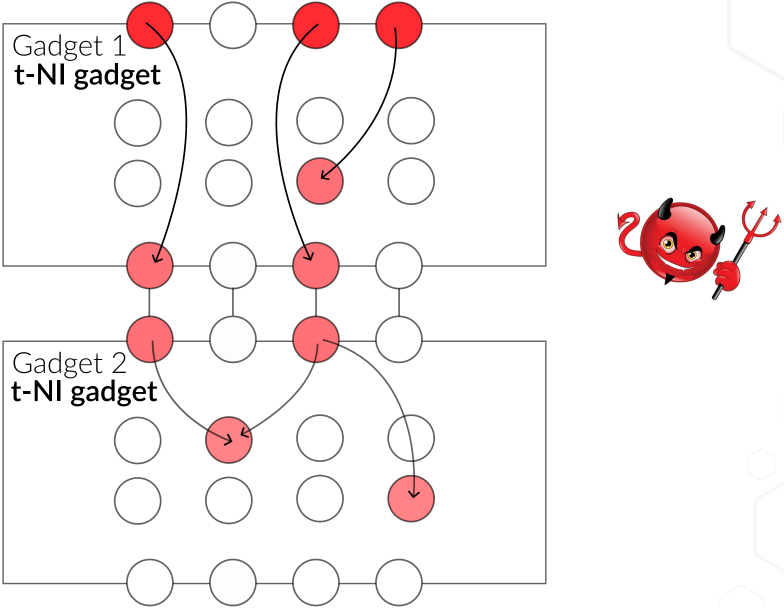
(strong) non-interference framework



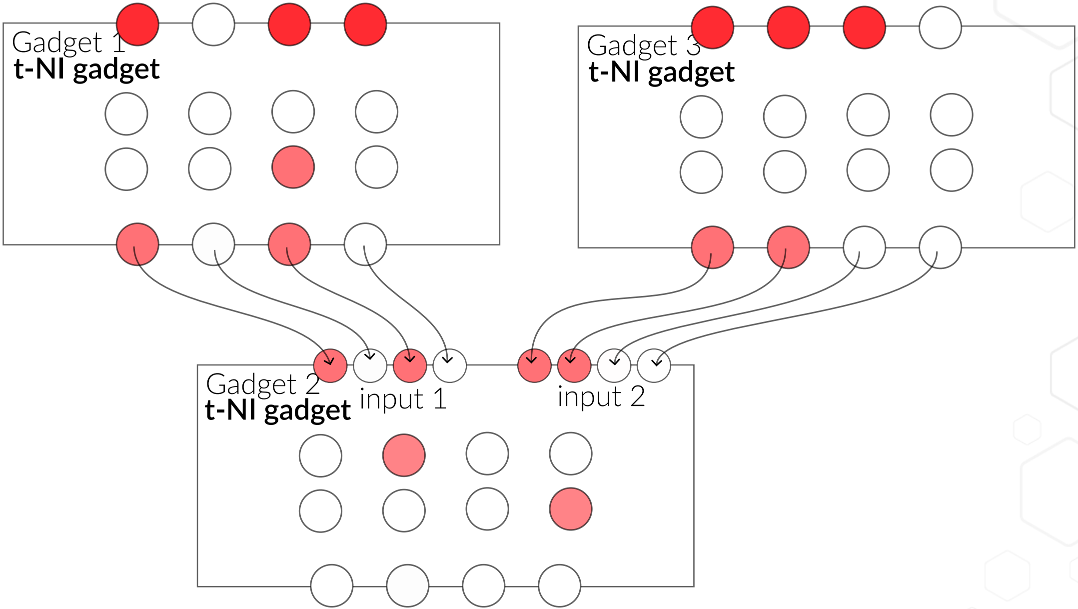
(strong) non-interference framework



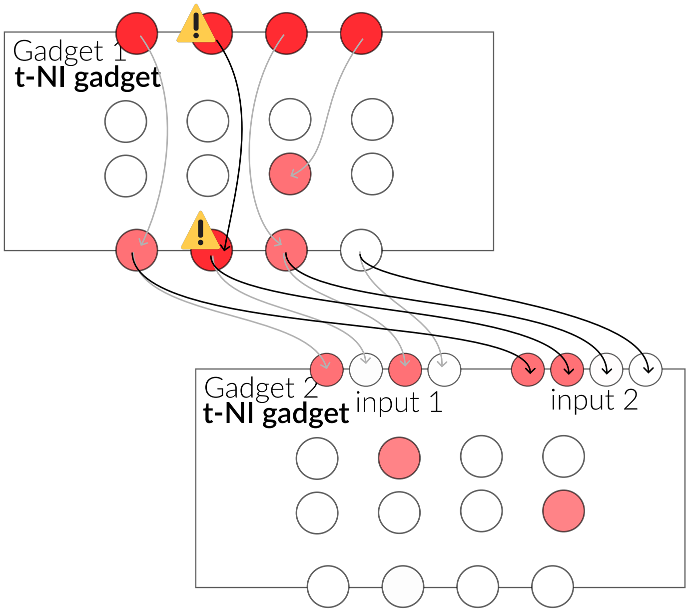
(strong) non-interference framework



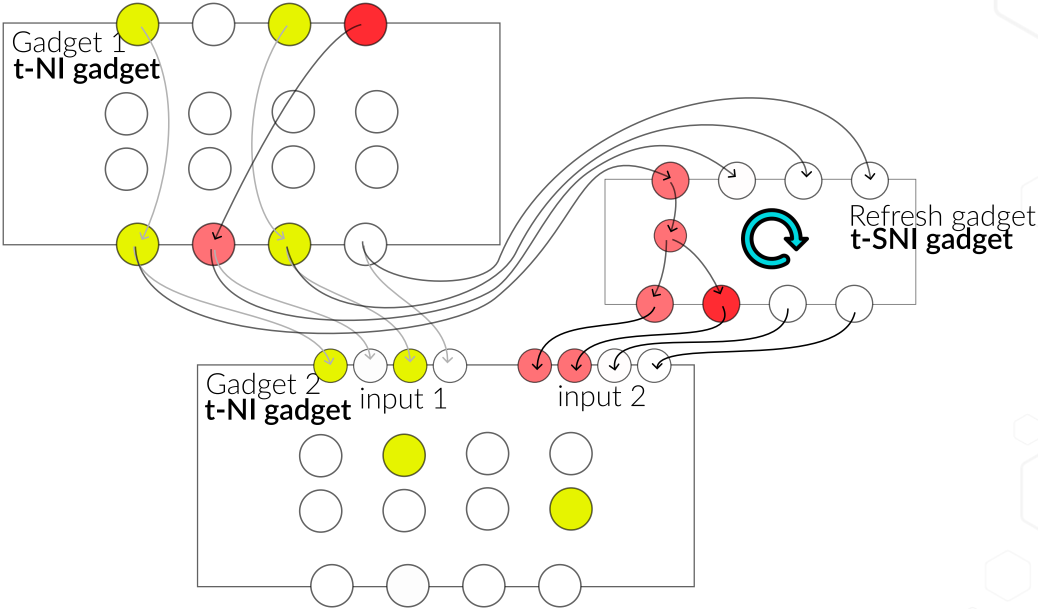
(strong) non-interference framework



(strong) non-interference framework



(strong) non-interference framework



Handling AddRepNoise

- The randomness added in AddRepNoise are secret inputs to the signature circuit, but some of them leak.
  (S)NI model does not capture partial leakage of input.

Handling AddRepNoise

- The randomness added in **AddRepNoise** are secret inputs to the signature circuit, but some of them leak.
  (S)NI model does not capture partial leakage of input.
- New notion: t-SNIu, strong non-interference with unmasked inputs.

Definition 4 (t-SNIu)

A circuit C is t-SNIu, if there exists simulators $\mathcal{S}_1, \mathcal{S}_2$ such that for any input $[\![\mathbf{x}]\!]$, unmasked values $(\mathbf{v}_i)_i$, and any set $\mathcal{P}$ of at most t probes:

$$i_1, \ldots, i_t, i'_1, \ldots, i'_t := S_1(\mathcal{P})$$
$$S_2(\mathcal{P}, [\![\mathbf{x}]\!]_{i_1}, \ldots, [\![\mathbf{x}]\!]_{i_t}, \mathbf{v}_{i'_1}, \ldots, \mathbf{v}_{i'_t}) = C_{\mathcal{P}}([\![\mathbf{x}]\!], (\mathbf{v}_i)_i)$$

i.e. probes are simulatable from at most t shares of the input $\mathbf{x}$, and t values $(\mathbf{v}_i)$.

Handling AddRepNoise

- The randomness added in **AddRepNoise** are secret inputs to the signature circuit, but some of them leak.
  (S)NI model does not capture partial leakage of input.
- New notion: t-SNIu, strong non-interference with unmasked inputs.
- We can show that **AddRepNoise** is t-SNIu secure for $t < d$.

Handling AddRepNoise

- The randomness added in **AddRepNoise** are secret inputs to the signature circuit, but some of them leak.
  (S)NI model does not capture partial leakage of input.
- New notion: t-SNIu, strong non-interference with unmasked inputs.
- We can show that **AddRepNoise** is t-SNIu secure for $t < d$.

- t-SNIu is composable: probes on Raccoon/Plover signing procedure can be simulated with at most t inputs shares, and t unmasked values.
  - t shares of masked input: independent from actual input
  - t unmasked values: remains $d \cdot \mathbf{rep} - t$ safe values to ensure security

Conclusion

PQ SHIELD

Raccoon and Plover are specific-purpose scheme aimed at high side-channel resistance:

- ☺ Standard assumptions: MLWE, MSIS
- ☺ Simpler
- ☺ Verification key size is similar
- ☹ Signatures are larger ($\approx$ 10kB)
- 🤩 **When masked, orders of magnitude faster than other schemes are**

General framework to create masking friendly schemes:

- → Noise-flooding to replace non-linear operations
- → Prove unmasked security with Hint-MLWE
- → Sample short vectors with **AddRepNoise** and use t-SNIu notion to prove security in the t-probing model

Questions?

Dmitri Asonov and Rakesh Agrawal.
Keyboard acoustic emanations.
pages 3–11, 2004.

Wim Van Eck.
Electromagnetic radiation from video display units: An eavesdropping risk?
Computers & Security, 4:269–286, 1985.

Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu.
Mitaka: A simpler, parallelizable, maskable variant of falcon.
pages 222–253, 2022.

Emre Karabulut and Aydin 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*, pages 691–696. IEEE, 2021.

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun.
Differential power analysis.
pages 388–397, 1999.

Paul C. Kocher.
Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems.
pages 104–113, 1996.

Thomas Prest.
A key-recovery attack against mitaka in the t-probing model.
pages 205–220, 2023.

Markku-Juhani O. Saarinen and Mélissa Rossi.
Mask compression: High-order masking on memory-constrained devices.
Cryptology ePrint Archive, Paper 2023/1117, 2023.
https://eprint.iacr.org/2023/1117.

Yang Yu, Huiwen Jia, and Xiaoyun Wang.
Compact lattice gadget and its applications to hash-and-sign signatures.
pages 390–420, 2023.

Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang.
Improved power analysis attacks on falcon.
Cryptology ePrint Archive, Paper 2023/224, 2023.
https://eprint.iacr.org/2023/224.

Masked Dilithium (Graph)