Raccoon

Rafael del Pino
PQShield

Thomas Espitau
PQShield

Shuichi Katsumata
PQShield
AIST

Mary Maller
PQShield
Ethereum Foundation

Fabrice Mouhartem
CryptPad

Thomas Prest
PQShield

Mélissa Rossi
ANSSI

Markku-Juhani Saarinen
PQShield
Tampere University

Motivation

Signature schemes strike a balance between:

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

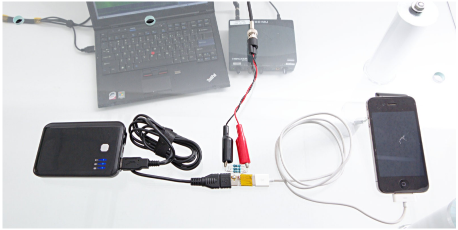
And so on...

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

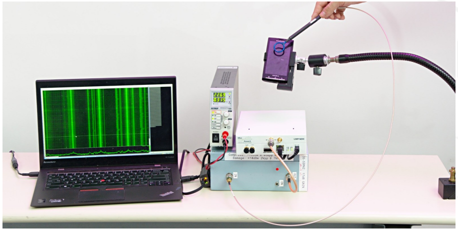
Side-Channel Attacks

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.

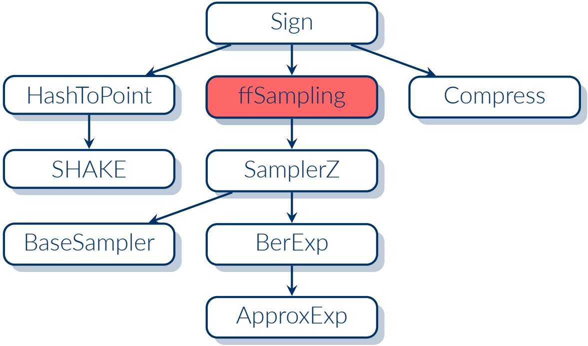
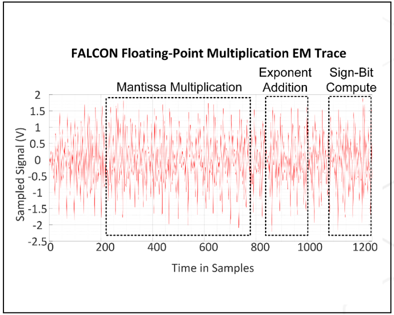


Figure 1: Flowchart of the signature



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.

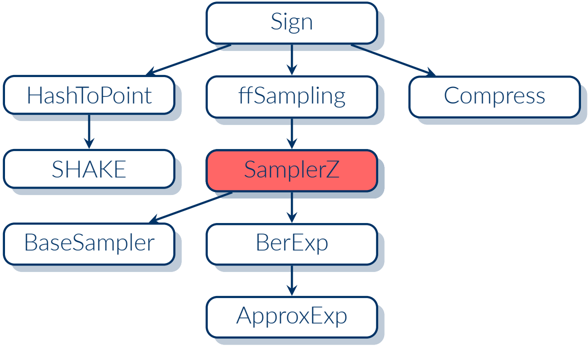
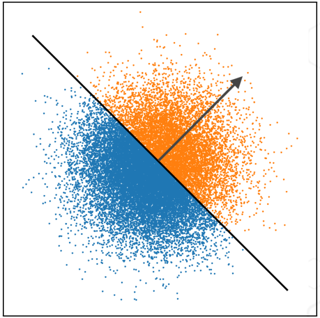


Figure 1: Flowchart of the signature



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

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

Masking and the t-probing model

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

$$x_0 + x_1 + \cdots + x_{d-1} = x \tag{2}$$

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

Interlude: river-crossing puzzles

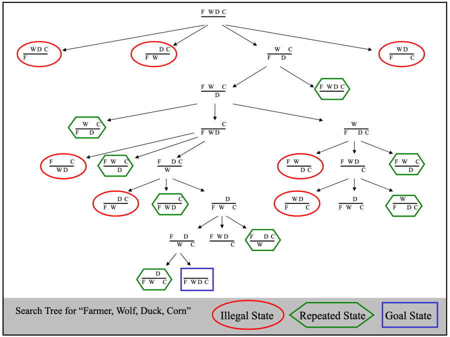
Remember this puzzle?

> " A farmer with a wolf, a goat, and a cabbage must cross a river by boat. The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten? "

Interlude: river-crossing puzzles

Remember this puzzle?

> “ A farmer with a wolf, a goat, and a cabbage must cross a river by boat. The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten? ”



It gets quickly complicated...

From river-crossing to masking

Now replace:

1. The set { farmer, wolf, goat, cabbage } by the shares $(x_0, \ldots, x_{d-1})$
2. The operation "*everyone crosses the river*" by an arbitrary function $f(\llbracket x \rrbracket) \to \llbracket y \rrbracket$
3. The constraints "*never leave A alone with B*" by "*a probing adversary shall not learn anything*"

... and you obtain an inexhaustible source of headaches for cryptographers.

How difficult are operations to mask?

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

Dilithium and Raccoon (simplified)

Dilithium - key generation

$\mathsf{Keygen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$

1. Generate a large matrix $\mathbf{A} = [\,\mathbf{I}\,|\,\bar{\mathbf{A}}\,] \in \mathcal{R}_q^{k\times(k+\ell)}$ ▷ No mask
2. Generate a short secret $\mathbf{s}$ ▷ Slow
3. Compute $\mathbf{t} = \mathbf{A}\cdot\mathbf{s}$ ▷ Fast
4. Verification key $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$ ▷ No mask
5. Signing key $\mathsf{sk} = \mathbf{s}$ ▷ No mask

When masking this algorithm, the bottleneck is sampling $\mathbf{s}$ (2):

→ Concretely, start with boolean masking, then apply B2A conversions

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

Raccoon - masked key generation

Keygen$(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$

1. Generate a large matrix $\mathbf{A} = [\,\mathbf{I} \mid \bar{\mathbf{A}}\,] \in \mathcal{R}_q^{k\times(k+\ell)}$ ▷ No mask
2. $[\![\mathbf{s}]\!] = (\mathbf{0}, \ldots, \mathbf{0})$ ▷ Fast
3. For $i \in [T]$: ▷ We call this “AddRepNoise”
   1. Sample short random shares in parallel: $[\![\mathbf{r}]\!] = (\mathbf{r}_0, \ldots, \mathbf{r}_{d-1})$ ▷ Fast
   2. $[\![\mathbf{s}]\!] := [\![\mathbf{s}]\!] + [\![\mathbf{r}]\!]$ ▷ Fast
   3. Refresh $[\![\mathbf{s}]\!]$ ▷ Fast
4. Compute $\mathbf{t} = \mathbf{A} \cdot [\![\mathbf{s}]\!]$ ▷ Fast
5. Unmask $[\![\mathbf{t}]\!]$ to obtain $\mathbf{t}$ ▷ Fast
6. Verification key is $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$ ▷ No mask
7. Signing key is $\mathsf{sk} = [\![\mathbf{s}]\!]$ ▷ No mask

Raccoon - masked key generation

Keygen$(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$

1. Generate a large matrix $\mathbf{A} = [\,\mathbf{I} \,|\, \bar{\mathbf{A}}\,] \in \mathcal{R}_q^{k \times (k+\ell)}$ ▷ No mask
2. $[\![\mathbf{s}]\!] = (\mathbf{0}, \ldots, \mathbf{0})$ ▷ Fast
3. For $i \in [T]$: ▷ We call this "AddRepNoise"
   1. Sample short random shares in parallel: $[\![\mathbf{r}]\!] = (\mathbf{r}_0, \ldots, \mathbf{r}_{d-1})$ ▷ Fast
   2. $[\![\mathbf{s}]\!] := [\![\mathbf{s}]\!] + [\![\mathbf{r}]\!]$ ▷ Fast
   3. Refresh $[\![\mathbf{s}]\!]$ ▷ Fast
4. Compute $\mathbf{t} = \mathbf{A} \cdot [\![\mathbf{s}]\!]$ ▷ Fast
5. Unmask $[\![\mathbf{t}]\!]$ to obtain $\mathbf{t}$ ▷ Fast
6. Verification key is $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$ ▷ No mask
7. Signing key is $\mathsf{sk} = [\![\mathbf{s}]\!]$ ▷ No mask

We show that $\mathbf{s}$ retains a large amount of randomness **even in the presence of a probing adversary**.

Dilithium - signature

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

Masking bottlenecks:

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

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

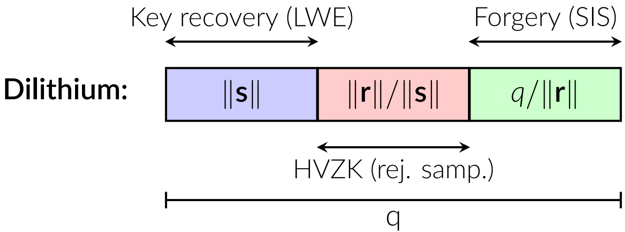
Raccoon - masked signature

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

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

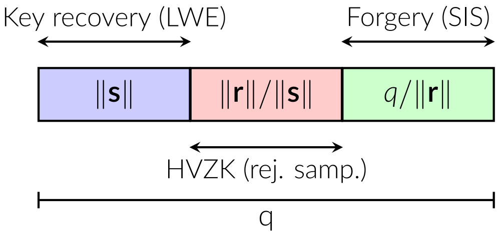
Total masking overhead: $O(d \log d)$

Impact on the modulus

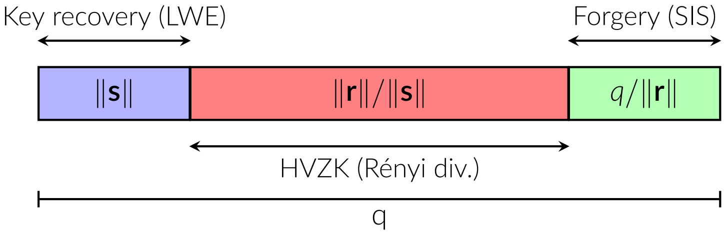


Impact on the modulus

Dilithium:

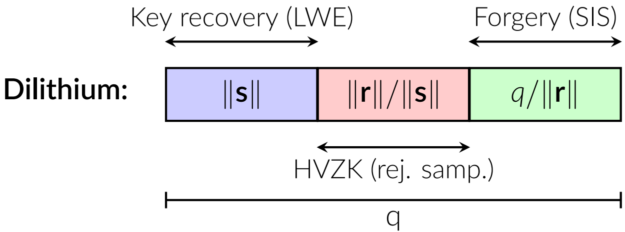


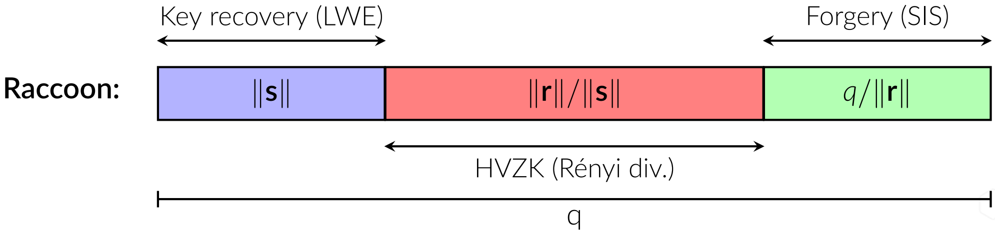
Raccoon:



1. Removing rejection sampling increases $\|\mathbf{r}\|/\|\mathbf{s}\|$ from $\Theta(\dim \mathbf{s})$ to $\Theta\left(\|c\|\sqrt{\text{Queries}}\right)$

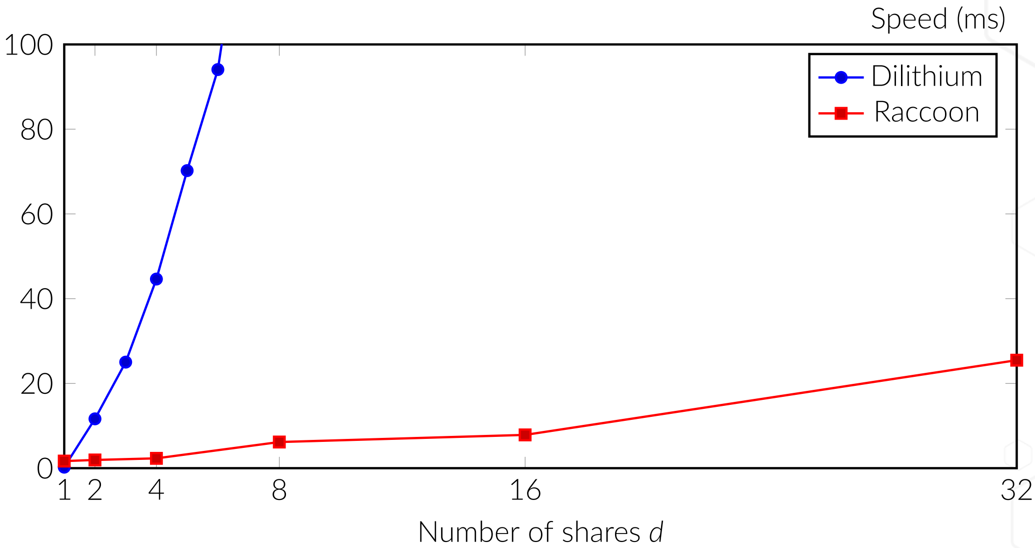
Impact on the modulus





1. Removing rejection sampling increases $\|\mathbf{r}\|/\|\mathbf{s}\|$ from $\Theta(\dim \mathbf{s})$ to $\Theta\left(\|c\|\sqrt{\text{Queries}}\right)$
2. The increased q in turn requires increasing $\|\mathbf{s}\|$, $q/\|\mathbf{r}\|$ and/or the dimensions.

Performances on a Desktop



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

Conclusion

Raccoon is a specific-purpose scheme aimed at high side-channel resistance:

- ☺ Same assumptions as Dilithium
- ☺ Simpler
- ☺ Verification key size is similar
- ☹ Signature is 4x larger
- 🤩 **When masked, orders of magnitude faster than other schemes are**

Questions?

Dmitri Asonov and Rakesh Agrawal.
Keyboard acoustic emanations.
In *2004 IEEE Symposium on Security and Privacy*, pages 3–11. IEEE Computer Society Press, May 2004.

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

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.
In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

Paul C. Kocher.

Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems.
In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.

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.

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.