

Plover

A masking-friendly lattice-based Hash-and-Sign signature

Muhammed F. Esgin
Monash University

Thomas Espitau
PQShield

Guilhem Niot
PQShield & Uni. Rennes 1

Thomas Prest¹
PQShield

Amin Sakzad
Monash University






Ron Steinfeld
Monash University











¹ Thanks to Thomas Prest for letting me reuse several of his slides.



Signature schemes strike a balance between:

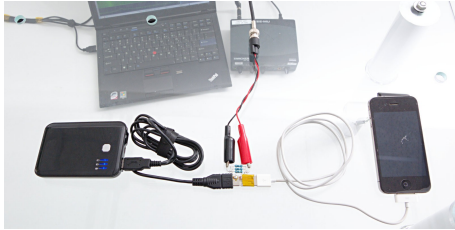
-  Sizes (verification key and signatures)
-  Speed (signing, verification)
-  Portability
-  Conservative assumptions
-  **Resistance against side-channel attacks**

And so on...

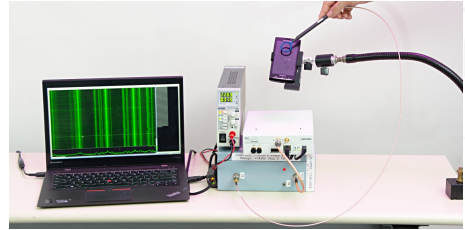
Criteria					
Dilithium	★★★½	★★★★	★★★★	★★★	
Falcon	★★★★	★★★★	★★★	★★★	
SPHINCS+	★★½	★★★	★★★	★★★★	
Raccoon	★★★	★★★★	★★★★	★★★	★★★★
Plover	★★★	★★★★	★★★★	★★★	★★★★

Side-Channel Attacks

Power consumption [KJJ99]



Electromagnetic emissions [Eck85]



Timing measurement [Koc96]



Acoustic emissions [AA04]



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.

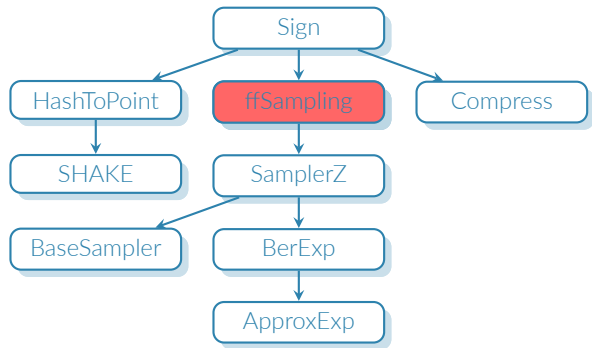
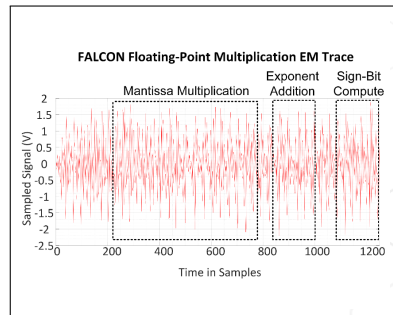


Figure 1: Flowchart of the signature



Learning **sk** directly

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.

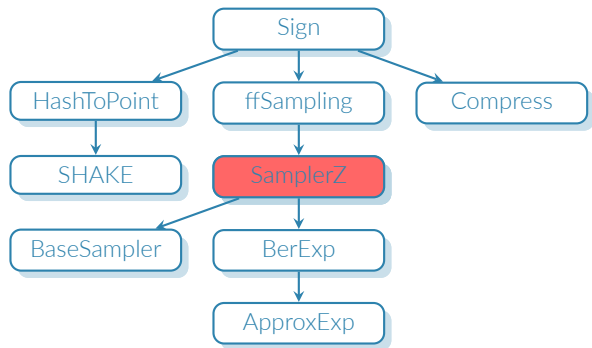
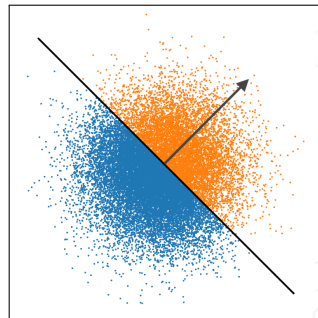




Figure 1: Flowchart of the signature



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

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 $t + 1$ shares:

$$[x] = (x_1, x_2, \dots, x_{t+1}) \quad (1)$$

such that

$$x_1 + x_2 + \dots + x_{t+1} = x \pmod{q} \quad (\text{additive})$$

$$\text{or } x_1 \oplus x_2 \oplus \dots \oplus x_{t+1} = x \quad (\text{boolean})$$

-  In “real life”, attacks cost is exponential in t
-  What about computations?



How difficult are operations to mask?

😊 Addition ($\llbracket c \rrbracket = \llbracket a + b \rrbracket$)?

- Compute $\llbracket c \rrbracket = (a_1 + b_1, \dots, a_{t+1} + b_{t+1})$, simple and fast: $\Theta(t)$ operations

😞 Multiplication ($\llbracket c \rrbracket = \llbracket a \cdot b \rrbracket$)?

- Complex and slower: $\Theta(t^2)$ operations

🤖 More complex operations?

- Use so-called *mask conversions* to convert between additive and boolean masking, very slow: $\gg \Theta(t^2)$ operations

Dilithium

- Requires costly *mask conversions*. Does not scale well with t .
- Or, masking-friendly variant *Raccoon*²

Falcon



²Submitted at the NIST 2023 Call for Additional Digital Signature Schemes.

Masked Hash-and-Sign signatures

- In 2017, *Falcon* was submitted to NIST.
 - Gaussian sampling and floating-point are challenging to mask.
- In 2022, *Mitaka* [EFG⁺22], attempted to solve this.
 - But, A *Key-Recovery Attack against Mitaka in the t-Probing Model* [Pre23]

Masking hash-and-sign signature scheme efficiently remains an open problem.

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

Eagle.Sign(sk, msg)

- 1 $\mathbf{u} := H(\text{msg})$
- 2 $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2 I - r^2 \mathbf{T} \mathbf{T}^*}}$
- 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 $\mathbf{y} \leftarrow D_{\lfloor q/\beta \rfloor \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$
- 7 $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$
- 8 return sig := \mathbf{z}

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

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

Eagle.Sign(sk, msg)

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

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

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

Eagle.Sign(sk, msg)

- 1 $\mathbf{u} := H(\text{msg})$ ▷ No mask
- 2 $\mathbf{p} \leftarrow D_{\mathcal{R}^\ell, \sqrt{s^2 I - r^2 \mathbf{T} \mathbf{T}^*}}$ ▷ **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 $\mathbf{y} \leftarrow D_{[q/\beta] \cdot \mathcal{R}^\ell + \mathbf{c}_1, r}$ ▷ **Hard**
- 7 $\mathbf{z} := \mathbf{p} + \mathbf{T} \cdot \mathbf{y}$ ▷ **Easy**
- 8 return sig := \mathbf{z}

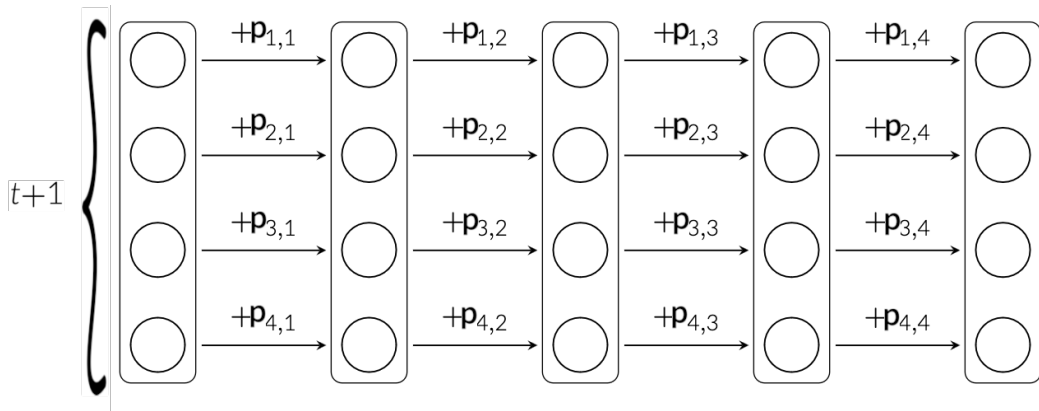
Plover.Sign(sk, msg)

- 1 $\mathbf{u} := H(\text{msg})$ ▷ No mask
- 2 $[[\mathbf{p}]] \leftarrow \text{AddRepNoise}(\sigma_{\mathbf{p}})$ ▷ **Easy**
- 3 $\mathbf{w} := \text{Unmask}(\mathbf{A} \cdot [[\mathbf{p}]])$ ▷ **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 // $\mathbf{y} := \mathbf{c}_1$
- 7 $\mathbf{z} := \text{Unmask}([[\mathbf{p}]] + [[\mathbf{T}]] \cdot \mathbf{c}_1)$ ▷ **Easy**
- 8 return sig := \mathbf{z}

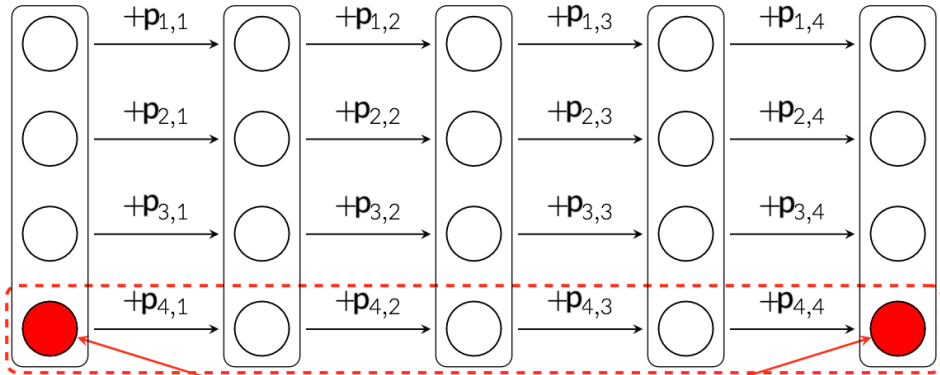
😊 Almost linear scheme, maybe we can do something with it!

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

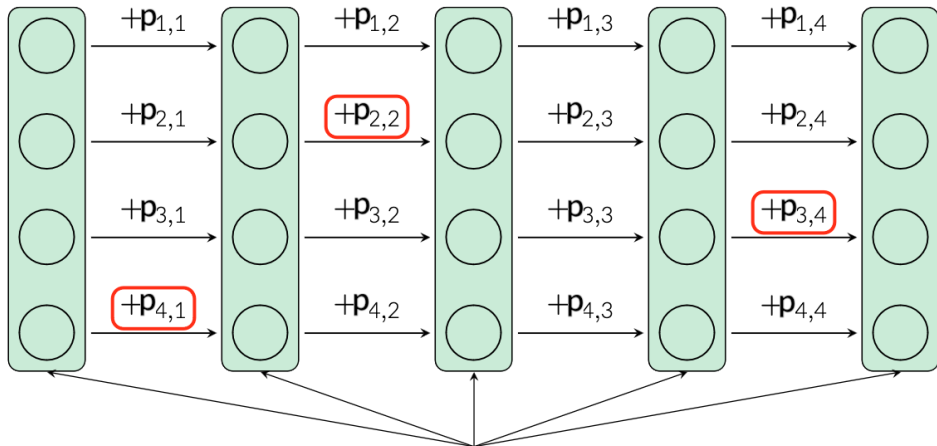
What happens inside AddRepNoise?



What happens inside AddRepNoise?



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



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

- Vanilla Plover,
 - Output of **AddRepNoise** looks like a Gaussian.
 - No Gaussian sampling: signatures leak part of the secret

Definition 1 (Hint-MLWE)

It is hard to distinguish $(\mathbf{A}, \mathbf{u}, (c_i \cdot \mathbf{s} + \mathbf{p}_i)_i)$ 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{p}_i of standard deviation $\approx \sqrt{Q} \|c\|$.

😊 Vanilla Plover is secure when taking large enough perturbations \mathbf{p}_i .

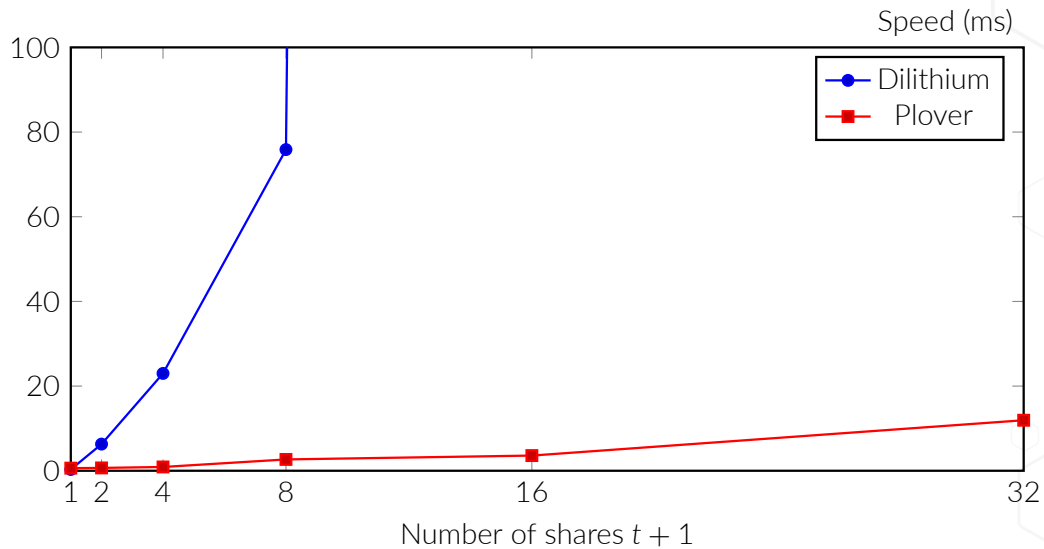
→ Masked Plover:

→ Leak part of the perturbation $\mathbf{p} = \text{AddRepNoise}(\sigma_p)$.

In t -probing model, write $\mathbf{p} = \mathbf{p}_{\text{safe}} + \mathbf{p}_{\text{leaked}}$.

If rep iterations in AddRepNoise , \mathbf{p}_{safe} has standard deviation $\sqrt{(t+1) \cdot \text{rep} - t} \cdot \sigma_p$.

Security of **Masked Plover reduces to Vanilla Plover** with small loss.



Proof in the t-probing model

- Generally, proofs are easy with the **SNI** framework.
 - Prove security of masked gadgets.
 - Compose them securely.
- But doesn't work for **AddRepNoise**: **AddRepNoise** leaks some secret randomness, and not only shares.
 - New composable notion t -SNI_l, which covers partial leakage of secret values.

Conclusion

Plover highlights a very generic framework for masking friendly schemes:

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

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

- 😊 Standard assumptions: MLWE, MSIS
- 😊 Simpler
- 😊 Verification key size is similar
- 😞 Signatures are larger ($\approx 10\text{kB}$)
- 😊 **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 -SNL_u notion to prove security in the t -probing model

Questions?



²Image from Emma Scheltema, <https://drawingscape.wordpress.com>

- 
-  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.
-  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>.