

# Road to Beam Chain

## A lattice approach

Kurt Pan

ZKPunk.pro

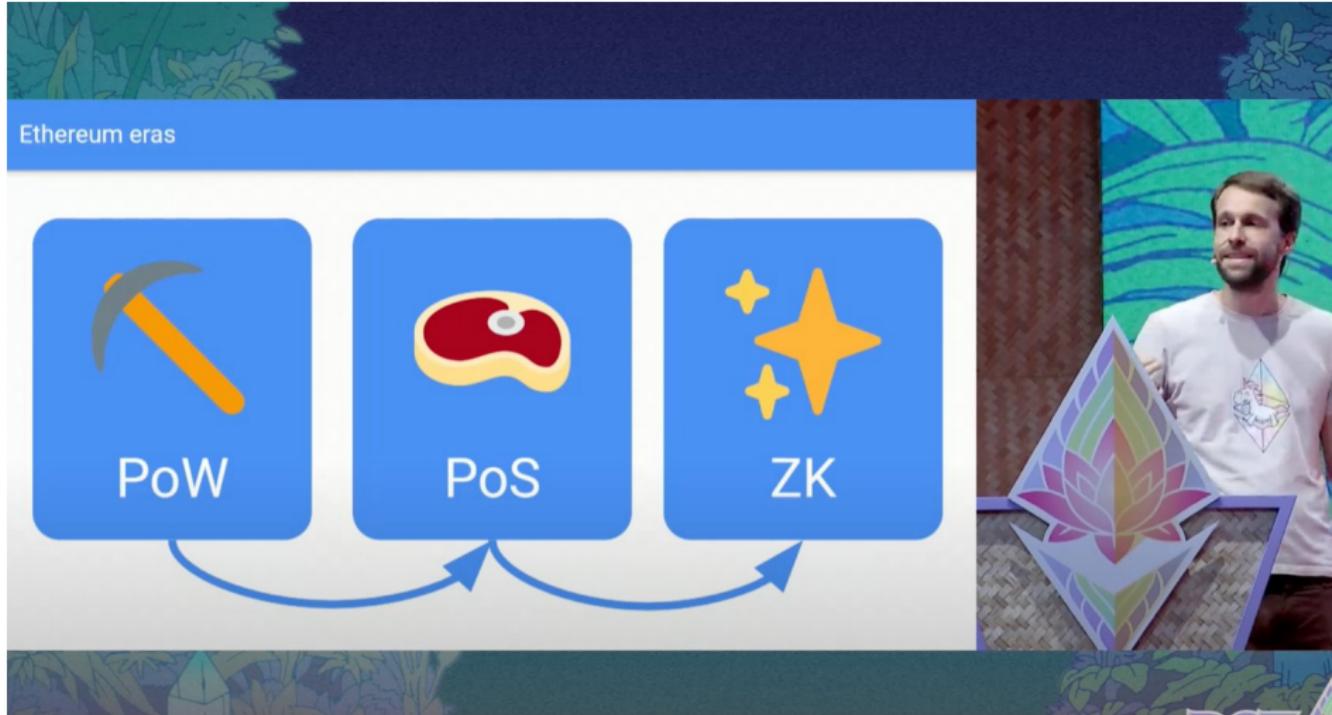
January 12, 2025



# Table of Contents

- 1 Why Post-Quantum
- 2 Why Lattice
- 3 What is lattice
- 4 How to aggregate Falcon using Labrador

# Beam Chain in a nutshell



# Beam Chain Roadmap

|    | Block Production                         | Staking                                  | Cryptography                                |
|----|--|--|---|
| P0 | Censorship Resistance (FOCIL)            | smarter issuance (e.g stake cap)         | chain snarkification (e.g. Poseidon + zkVM) |
| P1 | Isolated Validators (execution auctions) | smaller validators (1 ETH orbit staking) | quantum security (e.g. hash-based sigs)     |
| P2 | <b>Faster Slots (4 sec)</b>              | <b>Faster Finality (3-slot FFG)</b>      | strong randomness (MinRoot VDF)             |

faster block times, faster finality,  
**chain snarkification**, and **quantum resistance**.

## NIST's PQC standards

| Digital Signature Algorithm Family | Parameters                           | Transition   |
|------------------------------------|--------------------------------------|--|
| ECDSA<br>[FIPS186]                 | 112 bits of security strength        | <b>Deprecated</b> after 2030<br><b>Disallowed</b> after 2035 |
|                                    | $\geq 128$ bits of security strength | <b>Disallowed</b> after 2035                                 |
| EdDSA<br>[FIPS186]                 | $\geq 128$ bits of security strength | <b>Disallowed</b> after 2035                                 |
| RSA<br>[FIPS186]                   | 112 bits of security strength        | <b>Deprecated</b> after 2030<br><b>Disallowed</b> after 2035 |
|                                    | $\geq 128$ bits of security strength | <b>Disallowed</b> after 2035                                 |

ECC will be doomed by  
a large scale quantum  
computer, we need to switch to  
quantum-resistant crypto ASAP!!!

# Use ECC



0.1% 2%

IQ score

55

2%

70

34% 34%

68%

14%

95%

115

130

2%

0.1%

145

# Use ECC



# Table of Contents

1 Why Post-Quantum

2 Why Lattice

3 What is lattice

4 How to aggregate Falcon using Labrador

# Why NOT Lattice?



Wechat: XPTY

## attestation snarkification



# FIP 206 : FN-DSA (planned for late 2024)

- **Federal Information Processing Standard (FIPS) 203**, intended as the primary standard for general encryption. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation. The standard is based on the CRYSTALS-Kyber algorithm, which has been renamed ML-KEM, short for Module-Lattice-Based Key-Encapsulation Mechanism.
- **FIPS 204**, intended as the primary standard for protecting digital signatures. The standard uses the CRYSTALS-Dilithium algorithm, which has been renamed ML-DSA, short for Module-Lattice-Based Digital Signature Algorithm.
- **FIPS 205**, also designed for digital signatures. The standard employs the Sphincs+ algorithm, which has been renamed SLH-DSA, short for Stateless Hash-Based Digital Signature Algorithm. The standard is based on a different math approach than ML-DSA, and it is intended as a backup method in case ML-DSA proves vulnerable.

Similarly, when the draft FIPS 206 standard built around FALCON is released, the algorithm will be dubbed FN-DSA, short for FFT (fast-Fourier transform) over NTRU-Lattice-Based Digital Signature Algorithm.

The new standards are designed for two essential tasks for which encryption is typically used: general encryption, used to protect information exchanged across a public network; and digital signatures, used for identity authentication. NIST [announced its selection of four algorithms](#) — CRYSTALS-Kyber, CRYSTALS-Dilithium, Sphincs+ and FALCON — slated for standardization in 2022 and [released draft versions of three of these standards](#) in 2023. The fourth draft standard based on FALCON is planned for late 2024.

1

<sup>1</sup><https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>

- Compared to other signature schemes selected for standardisation by NIST, such as Dilithium [LDK+22] and Sphincs+ [HBD+22],
- Falcon stands out for its compactness, minimising both public key and signature sizes.<sup>2</sup>

| variant     | keygen (ms) | keygen (RAM) | sign/s | verify/s | pub size | sig size |
|-------------|-------------|--------------|--------|----------|----------|----------|
| FALCON-512  | 8.64        | 14336        | 5948.1 | 27933.0  | 897      | 666      |
| FALCON-1024 | 27.45       | 28672        | 2913.0 | 13650.0  | 1793     | 1280     |

Table 2. SLH-DSA parameter sets

|                    | <i>n</i> | <i>h</i> | <i>d</i> | <i>h'</i> | <i>a</i> | <i>k</i> | <i>lg<sub>w</sub></i> | <i>m</i> | security category | pk bytes | sig bytes |
|--------------------|----------|----------|----------|-----------|----------|----------|-----------------------|----------|-------------------|----------|-----------|
| SLH-DSA-SHA2-128s  | 16       | 63       | 7        | 9         | 12       | 14       | 4                     | 30       | 1                 | 32       | 7 856     |
| SLH-DSA-SHAKE-128s | 16       | 66       | 22       | 3         | 6        | 33       | 4                     | 34       | 1                 | 32       | 17 088    |
| SLH-DSA-SHA2-128f  | 16       | 66       | 22       | 3         | 6        | 33       | 4                     | 34       | 1                 | 32       | 17 088    |
| SLH-DSA-SHA2-192s  | 24       | 63       | 7        | 9         | 14       | 17       | 4                     | 39       | 3                 | 48       | 16 224    |
| SLH-DSA-SHAKE-192s | 24       | 63       | 7        | 9         | 14       | 17       | 4                     | 39       | 3                 | 48       | 16 224    |
| SLH-DSA-SHA2-192f  | 24       | 66       | 22       | 3         | 8        | 33       | 4                     | 42       | 3                 | 48       | 35 664    |
| SLH-DSA-SHA2-256s  | 32       | 64       | 8        | 8         | 14       | 22       | 4                     | 47       | 5                 | 64       | 29 792    |
| SLH-DSA-SHAKE-256s | 32       | 64       | 8        | 8         | 14       | 22       | 4                     | 47       | 5                 | 64       | 29 792    |
| SLH-DSA-SHA2-256f  | 32       | 68       | 17       | 4         | 9        | 35       | 4                     | 49       | 5                 | 64       | 49 856    |
| SLH-DSA-SHAKE-256f | 32       | 68       | 17       | 4         | 9        | 35       | 4                     | 49       | 5                 | 64       | 49 856    |

<sup>2</sup><https://falcon-sign.info/>

# Table of Contents

- 1 Why Post-Quantum
- 2 Why Lattice
- 3 What is lattice
- 4 How to aggregate Falcon using Labrador

# What is Lattice

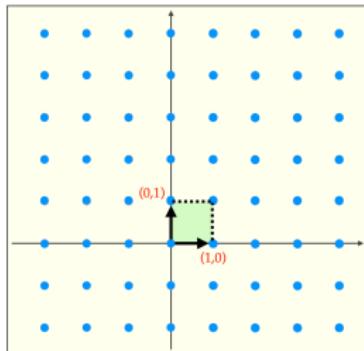
## Definition (Lattice)

A discrete additive subgroup of  $\mathbb{R}^n$

A lattice is the set of all *integer* linear combinations of (linearly independent) *basis* vectors  $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\} \subset \mathbb{R}^n$  :

$$\mathcal{L} = \sum_{i=1}^n \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

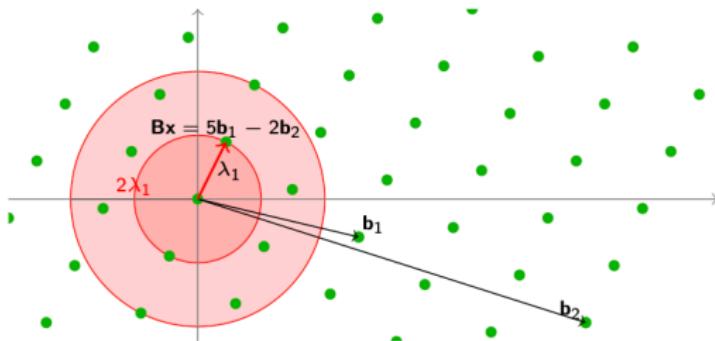
The same lattice has (infinite) many bases.



# Shortest Vector Problem

## Definition (SVP<sub>γ</sub>)

Given a lattice  $\mathcal{L}(\mathbf{B})$ , find a (nonzero) lattice vector  $\mathbf{Bx}$  (with  $\mathbf{x} \in \mathbb{Z}^k$ ) of length (at most)  $\|\mathbf{Bx}\| \leq \gamma \lambda_1$



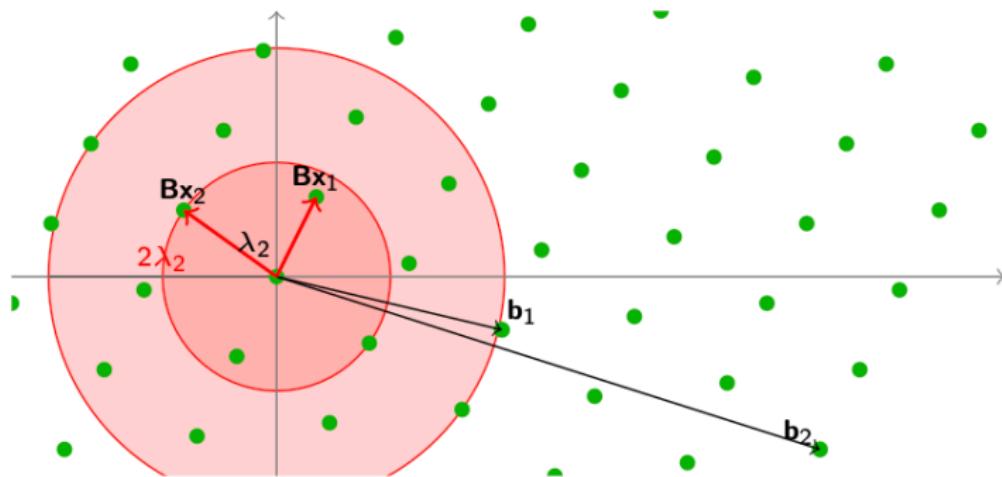
Minimum distance

$$\begin{aligned}\lambda_1 &= \min_{\mathbf{x}, \mathbf{y} \in \mathcal{L}, \mathbf{x} \neq \mathbf{y}} \|\mathbf{x} - \mathbf{y}\| \\ &= \min_{\mathbf{x} \in \mathcal{L}, \mathbf{x} \neq 0} \|\mathbf{x}\|\end{aligned}$$

# Shortest Independent Vectors Problem

## Definition (SIVP $_{\gamma}$ )

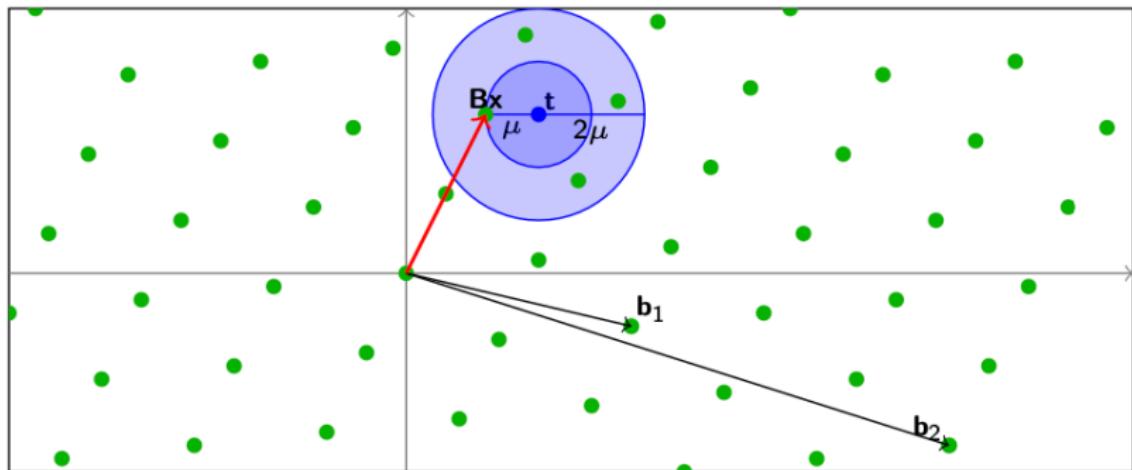
Given a lattice  $\mathcal{L}(\mathbf{B})$ , find  $n$  linearly independent lattice vectors  $\mathbf{Bx}_1, \dots, \mathbf{Bx}_n$  of length (at most)  $\max_i \|\mathbf{Bx}_i\| \leq \gamma \lambda_n$



# Closest Vector Problem

## Definition (CVP<sub>γ</sub>)

Given a lattice  $\mathcal{L}(\mathbf{B})$  and a target point  $\mathbf{t}$ , find a lattice vector  $\mathbf{Bx}$  within distance  $\|\mathbf{Bx} - \mathbf{t}\| \leq \gamma\mu$  from the target



# Special Versions of CVP

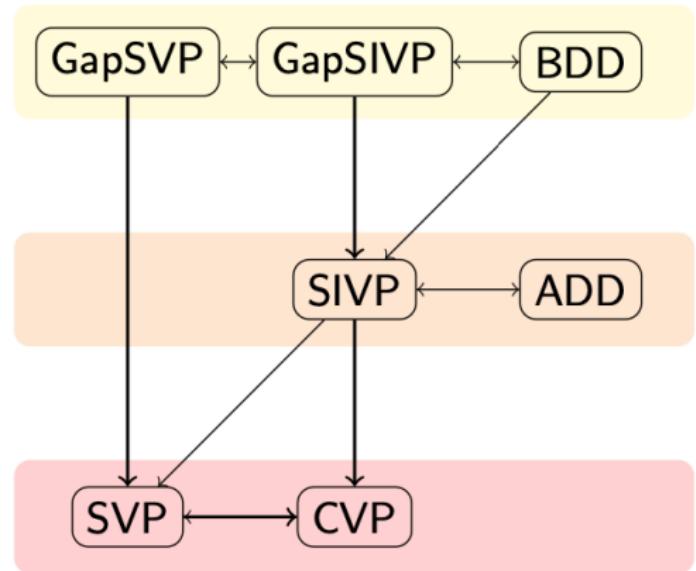
## Definition

Given  $(\mathcal{L}, \mathbf{t}, \mathbf{d})$ , with  $\mu(\mathbf{t}, \mathcal{L}) \leq \mathbf{d}$ , find a lattice point within distance  $d$  from  $\mathbf{t}$ .

- If  $d$  is arbitrary, then one can find the closest lattice vector by binary search on  $d$ .
- *Bounded Distance Decoding (BDD)*: If  $d < \lambda_1(\mathcal{L})/2$ , then there is at most one solution. Solution is the closest lattice vector.
- *Absolute Distance Decoding (ADD)*: If  $d \geq \mu(\mathcal{L})$ , then there is always at least one solution. Solution may not be closest lattice vector.

# Relations among lattice problems

- $\text{SIVP} \approx \text{ADD}$  [MG'01]
- $\text{SVP} \leq \text{CVP}$  [GMSS'99]
- $\text{SIVP} \leq \text{CVP}$  [M'08]
- $\text{BDD} \lesssim \text{SIVP}$
- $\text{CVP} \lesssim \text{SVP}$  [L'87]
- $\text{GapSVP} \approx \text{GapSIVP}$   
[LLS'91, B'93]
- $\text{GapSVP} \lesssim \text{BDD}$  [LM'09]



# Random lattices in Cryptography

- Cryptography typically uses (random) lattices  $\Lambda$  such that  $\Lambda \subseteq \mathbb{Z}^d$  is an integer lattice and  $q\mathbb{Z}^d \subseteq \Lambda$  is periodic modulo a small integer  $q$ .
- Cryptographic functions based on  $q$ -ary lattices involve only arithmetic modulo  $q$ .

## Definition ( $q$ -ary lattice)

$\Lambda$  is a  $q$ -ary lattice if  $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$

Examples (for any  $\mathbf{A} \in \mathbb{Z}_q^{n \times d}$ )

- $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \mid \mathbf{x} \bmod q \in \mathbf{A}^T \mathbb{Z}_q^n\} \subseteq \mathbb{Z}^d$
- $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\} \subseteq \mathbb{Z}^d$  (SIS lattice)

$$\mathcal{L}_q^\perp([\mathbf{A} \mid \mathbf{I}_n]) = \mathcal{L}\left(\left[\begin{array}{cc} -\mathbf{I}_m & \mathbf{0} \\ \mathbf{A} & q\mathbf{I}_n \end{array}\right]\right)$$

# How are they related?

$$\mathbf{A} \in \mathbb{Z}_q^{m \times k}, \mathbf{s} \in \mathbb{Z}_q^k, \mathbf{e} \in \mathcal{E}^m.$$

$$g_{\mathbf{A}}(\mathbf{s}; \mathbf{e}) = \mathbf{As} + \mathbf{e} \bmod q$$

## Theorem (R'05)

The function  $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$  is hard to invert on the average, assuming SIVP is hard to approximate in the worst-case.

## LWE and q-ary lattices

- If  $\mathbf{e} = \mathbf{0}$ , then  $\mathbf{As} + \mathbf{e} = \mathbf{As} \in \Lambda(\mathbf{A}^t)$
- Same as CVP in random  $q$ -ary lattice  $\Lambda(\mathbf{A}^t)$  with random target  $\mathbf{t} = \mathbf{As} + \mathbf{e}$
- Usually  $\mathbf{e}$  is shorter than  $\frac{1}{2}\lambda_1(\Lambda(\mathbf{A}^T))$ , and  $\mathbf{e}$  is uniquely determined
- TAKE AWAY:
- LWE  $\equiv$  Approximate BDD (Bounded Distance Decoding)

# (M)SIS Problem

## Definition

Let  $n, m, \beta \in \mathbb{N}$ . The Module Short Integer Solution problem M-SIS <sub>$n, m, \beta$</sub>  over  $\mathcal{R}_q$  is defined as follows.

Given  $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times m}$ , find an  $\vec{z} \in \mathcal{R}_q^m$  such that  $\mathbf{A}\vec{z} = \vec{0}$  and  $0 < \|\vec{z}\|_2 \leq \beta$ .

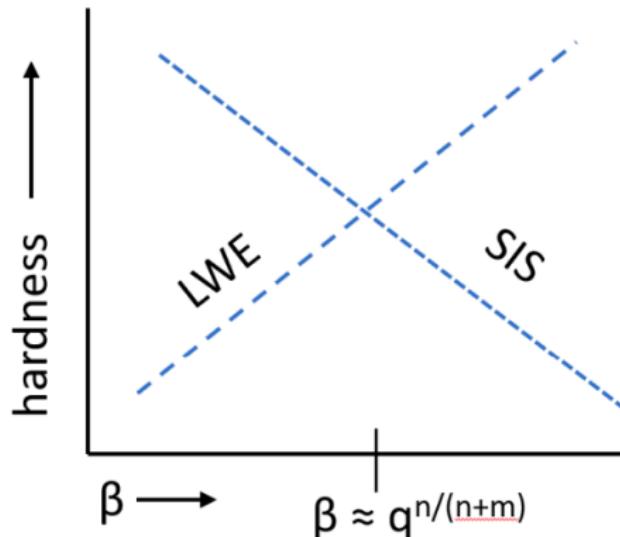
$$\begin{array}{c} A \\ \text{---} \\ n \times m \end{array} = \begin{array}{c} z \\ \text{---} \\ m \times 1 \end{array} = \begin{array}{c} 0 \\ \text{---} \\ n \times 1 \end{array} \pmod{q}$$

# SIS (lattice formulation)

## Definition (SVP in SIS lattice)

Given  $A \in_R \mathbb{Z}_q^{n \times m}$ , find a nonzero  $z \in [-B, B]^m$  in the SIS lattice

$$L_A^\perp = L(C) \text{ where } C = \begin{bmatrix} qI_n & -\bar{A} \\ 0 & I_{m-n} \end{bmatrix}.$$



# SIS application: Collision-resistant hash function (Ajtai)

## Definition (Ajtai Hash)

Select  $A \in_R \mathbb{Z}_q^{n \times m}$ , where  $m > n \log q$ .

Define  $H_A : \{0, 1\}^m \longrightarrow \mathbb{Z}_q^n$  by  $H_A(z) = Az \pmod{q}$ .

## Compression.

Since  $m > n \log q$ , we have  $2^m > q^n$ . □

## Collision resistance.

Suppose that one can efficiently find  $z_1, z_2 \in \{0, 1\}^m$  with  $z_1 \neq z_2$  and  $H_A(z_1) = H_A(z_2)$ . Then  $Az_1 = Az_2 \pmod{q}$ , whence  $Az = 0 \pmod{q}$  where  $z = z_1 - z_2$ . Since  $z \neq 0$  and  $z \in [-1, 1]^m$ ,  $z$  is an SIS solution (with  $B = 1$ ) which has been efficiently found. □

# NTRU: Nth-degree TRUncated polynomial ring

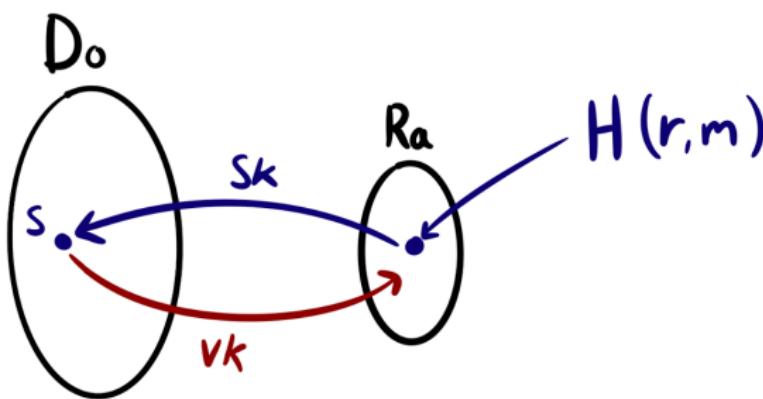
- The NTRU cryptosystem is parameterized by a certain polynomial ring  $R = \mathbb{Z}[X]/(f(X))$ , e.g.,  $f(X) = X^n - 1$  for a prime  $n$  or  $f(X) = X^n + 1$  for an  $n$  that is a power of two, and a sufficiently large odd modulus  $q$  that defines the quotient ring  $R_q = R/qR$ .
- The public key is  $h = 2g \cdot s^{-1} \in R_q$  for two "short" polynomials  $g, s \in R$ , i.e., ones having relatively small integer coefficients, where the secret key  $s$  is also chosen to be invertible modulo both  $q$  and two.
- Can define SVP/CVP in NTRU lattice

# Table of Contents

- 1 Why Post-Quantum
- 2 Why Lattice
- 3 What is lattice
- 4 How to aggregate Falcon using Labrador

# Full-Domain-Hash (FDH) paradigm, hash-and-sign framework

- The verification key  $pk$  is a *trapdoor permutation*  $f$ .
- The signing key  $sk$  is the inverse  $f^{-1}$ .
- To sign a message  $m$ , one first hashes  $m$  to some point  $y = H(m)$  in the range of  $f$ , then outputs the signature  $\sigma = f^{-1}(y)$ .
- Verification consists of checking that  $f(\sigma) = H(m)$ .



Recall RSA-FDH / BLS signature

# GPV framework: lattice-based hash-and-sign

- Generalised the FDH paradigm to work with *preimage sampleable trapdoor functions (PSFs)*.
- $\text{pk}$  defining a PSF  $F_{\text{pk}}$ , and  $\text{sk}$  that allows one to invert  $F_{\text{pk}}$ .
- $y = H(r, m) \in \text{Ra}$ ,
- Then use  $\text{sk}$  to sample a signature  $\sigma = x \in \text{Do}$  following some distribution  $\mathcal{D}(F_{\text{pk}}^{-1}(y))$ .

## Falcon: GPV over NTRU lattice

- $\text{pk} = \mathbf{h} = g \cdot f^{-1} \bmod q \in \mathcal{R}_q$  defines an NTRU-module  
 $\Lambda = \{(\mathbf{u}, \mathbf{v}) \in \mathcal{R}^2 : \mathbf{u} + \mathbf{h}\mathbf{v} = \mathbf{0} \bmod q\}$
- $\text{sk}$  contains a secret (short) basis of  $\Lambda$  that allows sampling module elements following a discrete Gaussian distribution defined over an arbitrary coset  $\Lambda_t = \{(\mathbf{u}, \mathbf{v}) \in \mathcal{R}^2 : \mathbf{u} + \mathbf{h}\mathbf{v} = \mathbf{t} \bmod q\}$ .
- The signing algorithm first hashes  $m$  to  $y = t \in \mathcal{R}_q$ , uses the secret basis to obtain a preimage  $x = (\mathbf{s}_1, \mathbf{s}_2) \in \Lambda_t$ , and outputs  $\sigma = (\mathbf{s}_1, \mathbf{s}_2, r)$  as a signature.
- The verification conditions are simply (1)  $\mathbf{s}_1 + \mathbf{h}\mathbf{s}_2 = \text{H}(r, m) \bmod q$ , and (2)  $\|(\mathbf{s}_1, \mathbf{s}_2)\|_2 \leq \beta \ll q$ , where  $\beta$  is determined by a Gaussian parameter.

Falcon stands out for its compactness, minimising both public key and signature sizes.

# What is LaBRADOR

- The LaBRADOR protocol is an *iterative multi-round public-coin interactive proof* (Bulletproofs-style), which can be made non-interactive in the random oracle model by applying the Fiat-Shamir transform. The security of LaBRADOR relies on the hardness of *the Module Shortest Integer Solution problem*.
- Native language (DPCS) well suited for Falcon verification
- **Lazarus** is a framework for implementing lattice-based zero-knowledge arguments in Rust.
- In active developing, LaBRADOR PoC will be the very 1st milestone.



pls scan and give us a star :)

**Lazarus** Public  
A Framework of Lattice-based Zero-knowledge Arguments in Rust  
[cryptography-library](#) [post-quantum-cryptography](#) [zero-knowledge-proofs](#)  
Rust · Apache License 2.0 · 7 · 39 · 14 · 0 · Updated 5 days ago

**beam-ethla** Public  
Aggregating Falcon Signatures using Lazarus, aligned with ETH Beam roadmap  
MIT License · 0 · 4 · 0 · 0 · Updated last week

**baby-kyber-rs** Public  
Rust · MIT License · 1 · 0 · 0 · 0 · Updated 2 weeks ago

**LV-FHE** Private  
Lattice Verifiable FHE  
0 · 0 · 0 · 0 · Updated last month

**Lazard-VM** Private  
zkVM using Lazarus  
0 · 0 · 0 · 0 · Updated last month

**Latino** Private  
A Lattice-based Indistinguishability Obfuscator  
Apache License 2.0 · 0 · 0 · 0 · 0 · Updated last month

**bitcoin-musig-l** Public  
Introducing lattice-based multi-signature into bitcoin core  
Apache License 2.0 · 0 · 0 · 0 · 0 · Updated on Nov 13, 2024

We are NOT hiring, we BUDL for the future.

Kurt Pan (ZKPunk.pro)

Road to Beam Chain

# Aggregating Falcon Signatures with LaBRADOR

## Aggregating Falcon Signatures with LaBRADOR

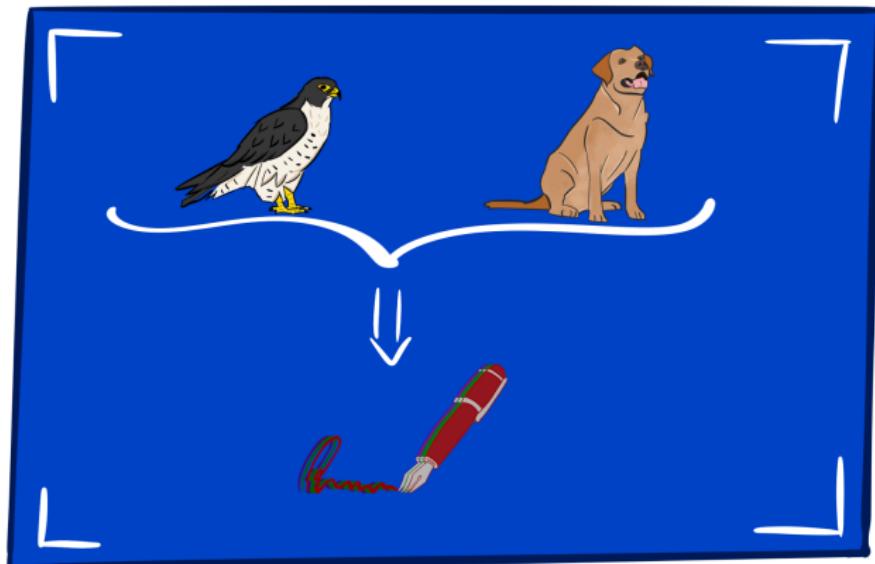
Marius A. Aardal<sup>1</sup>, Diego F. Aranha<sup>1</sup>, Katharina Boudgoust<sup>2\*</sup>, Sebastian Kolby<sup>1</sup>, and Akira Takahashi<sup>3\*\*</sup>

<sup>1</sup> Aarhus University, Denmark

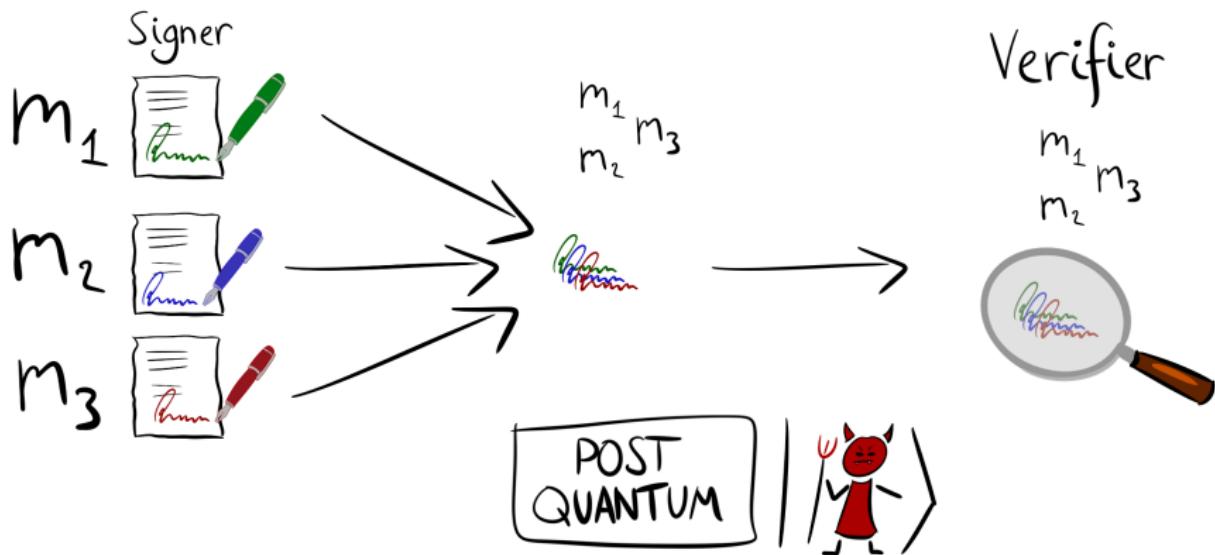
<sup>2</sup> CNRS, Univ Montpellier, LIRMM, France

<sup>3</sup> J.P.Morgan AI Research & AlgoCRYPT CoE, United States

August 9, 2024



# (PQ) Aggregate Signatures



Aggregate BLS Signatures [BGLS'03]

Aggregate Schnorr Signatures [BN'06]

Multi/Threshold ECDSA [Lindell'17, GG'18, GG'20]

# Aggregate Signature: Correctness & Security

- $\text{AggSign} \left( \text{pp}, \{\text{pk}_i, m_i, \sigma_i\}_{i \in [N]} \right) \rightarrow \sigma_{\text{agg}}$
- $\text{AggVer} \left( \text{pp}, \{\text{pk}_i, m_i\}_{i \in [N]}, \sigma_{\text{agg}} \right) \rightarrow b$

## Definition (Correctness)

For all  $\lambda, N \in \mathbb{N}$  it yields

$$\Pr \left[ \text{AggVer} \left( \{\text{pk}_i, m_i\}_{i \in [N]}, \sigma_{\text{agg}} \right) = 1 \right] = 1 - \text{negl}(\lambda)$$

where

$\text{pp} \leftarrow \text{Setup} (1^\lambda), m_i \in M, (\text{sk}_i, \text{pk}_i) \leftarrow \text{Gen}(\text{pp}), \sigma_i \leftarrow \text{Sign} (\text{sk}_i, m_i)$  for all  $i \in [N]$  and  $\sigma_{\text{agg}} \leftarrow \text{AggSign} \left( \{\text{pk}_i, m_i, \sigma_i\}_{i \in [N]} \right)$ .

# Aggregate Signature: Correctness & Security

## Definition (Unforgeability)

An AS scheme satisfies existential unforgeability in the aggregate chosen key model (EU-ACK), if for all PPT adversaries  $\mathcal{A}$ ,

$$\text{Adv}_{\text{AS}}^{\text{EU-ACK}}(\mathcal{A}) := \Pr[\text{EU-ACK}_{\text{AS}}(\mathcal{A}, \lambda) = 1] = \text{negl}(\lambda)$$

### Game 2: $\text{EU-ACK}_{\text{AS}}(\mathcal{A}, \lambda)$

```
1:  $\text{pp} \leftarrow \text{Setup}(1^\lambda)$                                  $\text{OSign}(m)$ 
2:  $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\text{pp})$ 
3:  $\mathcal{Q} := \emptyset$ 
4:  $(\{\text{pk}_i, m_i\}_{i \in [N]}, \sigma_{\text{agg}}) \leftarrow \mathcal{A}^{\text{OSign}}(\text{pp}, \text{pk})$ 
5: if  $\text{AggVer}(\{\text{pk}_i, m_i\}_{i \in [N]}, \sigma_{\text{agg}}) \wedge \exists i^* \in [N] : (\text{pk}_{i^*} = \text{pk} \wedge m_{i^*} \notin \mathcal{Q})$  then
6:   return 1
7: else
8:   return 0
```

## Computing hash locally

Batch-proving knowledge of  $N$  Gaussian samples  $(s_{i,1}, s_{i,2})_{i \in [N]}$  and salts  $(r_i)_{i \in [N]}$  meeting verification conditions w.r.t. a list  $(m_i, h_i)_{i \in [N]}$  of messages and public keys, respectively.

However, generating proof of correct hash computation is not only costly, but also leads to heuristic security guarantees: an aggregator may need a concrete description of  $H$  as a hash function, while Falcon has only been proven secure if  $H$  is modeled as a random oracle. And the size of salt  $r_i$  is much smaller than  $(s_{i,1}, s_{i,2})$

- Aggregator include salts  $r_i$  in the aggregated signature
- generate a proof for  $t_i = s_{i,1} + h_i s_{i,2} \bmod q$  for a public statement  $t_i$ ,
- Verifier compute  $t_i = H(r_i, m_i)$  locally.

# Adapting LaBRADOR for Aggregating Falcon Signatures

Principal relation of LaBRADOR:

$$f(\vec{w}_1, \dots, \vec{w}_r) := \sum_{i,j=1}^r \mathbf{a}_{i,j} \langle \vec{w}_i, \vec{w}_j \rangle + \sum_{i=1}^r \langle \vec{\varphi}_i, \vec{w}_i \rangle - \mathbf{b}$$

- The verification equation and norm bound of a single Falcon signature seem quite compatible with the principal relation of LaBRADOR.
- To aggregate  $N$  signatures, one might then try to extend the statement to contain a verification equation for each signature and a combined norm bound:  $\forall i = 1, \dots, N, \mathbf{s}_{i,1} + \mathbf{h}_i \mathbf{s}_{i,2} - \mathbf{t}_i = \mathbf{0} \bmod q$  and  $\sum_{i=1}^N \sum_{j=1}^2 \|\mathbf{s}_{i,j}\|_2^2 \leq N\beta^2$ .

If LaBRADOR was instantiated over the ring used by Falcon then  $(\mathbf{s}_{i,1}, \mathbf{s}_{i,2})_{i \in [N]}$  could be used directly as witness vector. However, there are several problems with this approach.

# Adapting LaBRADOR for Aggregating Falcon Signatures

- The norm check in LaBRADOR is both approximate and with respect to the entire witness. This introduces a degree of slack that grows with the number of signatures  $N$ . → Modify the first iteration of the LaBRADOR protocol to use the approach of [GHL22] for an exact proof of smallness.
- The modular Johnson-Lindenstrauss projections are used, which require that the norm bound  $b$  of the statement satisfies the inequality  $\sqrt{\lambda}b \leq q/C_1$  for security level  $\lambda$  and some corresponding constant  $C_1$ . For both Falcon parameter sets this is not satisfied. → Reformulate the statement and witness so that the LaBRADOR protocol uses a separate modulus  $q'$ , different from  $q$
- the number of initial witness elements  $r = 2N$  and their rank  $n = 1$  is quite unbalanced (bad for performance). → Present an alternative formulation of the constraints that achieves a better balance between these parameters.

# Reducing proof size by moving to subrings

- To significantly compress the proof sizes by using existing techniques from [LNPS21] to move to subrings  $\mathcal{S}$  of smaller degrees  $d'$ .
- Moving from the ring  $\mathcal{R}$  of degree  $d$  to a subring  $\mathcal{S}$  of smaller degree  $d' = d/c$ , which improves proof sizes.
- Reduces the proof sizes by at least a multiplicative factor 2 .

# Choice of Ring and Challenge Space

- 2-splitting modulus → Lifted to composite modulus for NTT [CHK+'22]
- see more in Greyhound [NS'24]

| Scheme       | Structure | Transparent setup | Proof sizes for |              |              |
|--------------|-----------|-------------------|-----------------|--------------|--------------|
|              |           |                   | $N = 2^{26}$    | $N = 2^{28}$ | $N = 2^{30}$ |
| Brakedown-PC | Hashes    | ✓                 | 49157           | 93767        | 181948       |
| Ligero-PC    | Hashes    | ✓                 | 7256            | 14383        | 28631        |
| FRI-PC       | Hashes    | ✓                 | 740             | —            | —            |
| FMN23-PC     | Lattices  | ✗                 | —               | —            | 8499         |
| SLAP-PC      | Lattices  | ✗                 | —               | —            | 785408       |
| CMNW24-PC    | Lattices  | ✓                 | 1546            | —            | 5294         |
| HSS24-PC     | Lattices  | ✓                 | 48640           | —            | —            |
| Greyhound    | Lattices  | ✓                 | 46              | 53           | 53           |

# Estimates and Comparison

Single Falcon sig size  $\approx 666$  B

| Aggr. Signature Scheme           | # Sign. | N   | Sec. | Param. | $\lambda$ | Life Cycle | $ \sigma_{\text{agg}} $ |
|----------------------------------|---------|-----|------|--------|-----------|------------|-------------------------|
| Chipmunk [24]                    | 1024    | 112 |      |        | 8 months  | 118 kB     |                         |
|                                  | 1024    | 112 |      |        | 5 years   | 133 kB     |                         |
|                                  | 1024    | 112 |      |        | 21 years  | 143 kB     |                         |
| Ours for Falcon-512 with salt    | 1024    | 121 |      |        | $\infty$  | 122 kB     |                         |
| Ours for Falcon-512 without salt | 1024    | 121 |      |        | $\infty$  | 81 kB      |                         |

Concatenation  $\approx 682$  KB , -82% <sup>3</sup>

| Aggr. Signature Scheme           | # Sign. | N   | Sec. | Param. | $\lambda$ | $ \sigma_{\text{agg}} $ |
|----------------------------------|---------|-----|------|--------|-----------|-------------------------|
| Ours for Falcon-512 with salt    | 1024    | 121 |      |        | 122 kB    |                         |
| Ours for Falcon-512 without salt | 1024    | 121 |      |        | 81 kB     |                         |
| Ours for Falcon-512 with salt    | 4096    | 121 |      |        | 252 kB    |                         |
| Ours for Falcon-512 without salt | 4096    | 121 |      |        | 88 kB     |                         |
| Ours for Falcon-512 with salt    | 8192    | 121 |      |        | 417 kB    |                         |
| Ours for Falcon-512 without salt | 8192    | 121 |      |        | 89 kB     |                         |

Annotations: The table shows two rows for each Falcon-512 variant. The first row uses salt and the second row does not. The 'Param.' column is empty. The 'lambda' column is also empty. The 'sigma\_aggr' column is empty. Handwritten annotations include: 'log' with an arrow pointing to the 8192 row; 'linear' with an arrow pointing to the 8192 row; and yellow boxes highlighting the 'with salt' rows and the 'sigma\_aggr' column.

<sup>3</sup><https://github.com/dfaranha/aggregate-falcon>

Thanks!

