Road to Beam Chain

A lattice approach

Kurt Pan

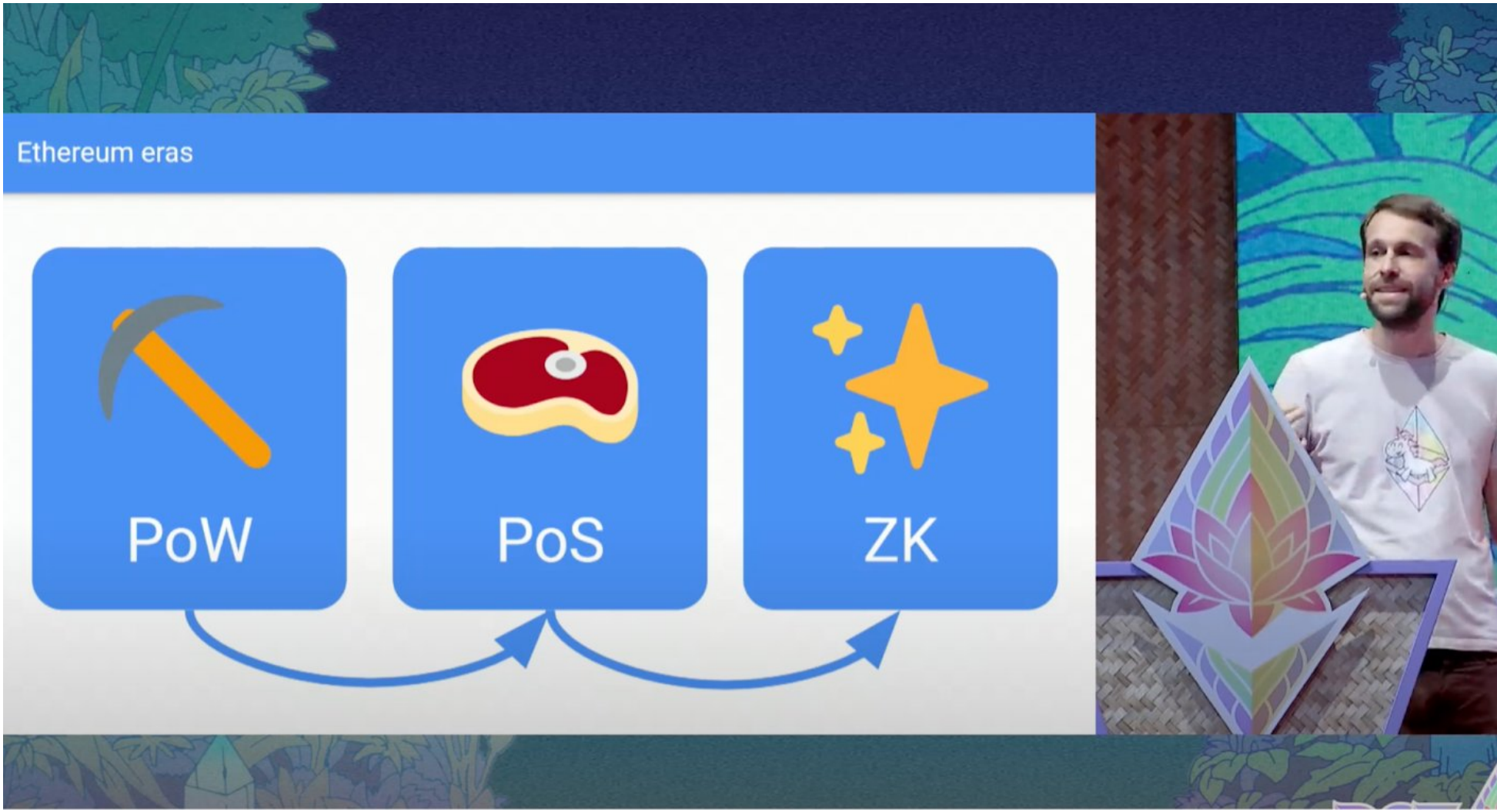
ZKPunk.pro

January 11, 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	**Faster Slots (4 sec)**	**Faster Finality (3-slot FFG)**	strong randomness (MinRoot VDF)

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

Disallowed Pre-Quantum Signature Schemes after 2035

NIST's PQC standards

Digital Signature Algorithm Family	Parameters	Transition
ECDSA [FIPS186]	112 bits of security strength	***Deprecated*** after 2030 ***Disallowed*** after 2035
	≥ 128 bits of security strength	***Disallowed*** after 2035
EdDSA [FIPS186]	≥ 128 bits of security strength	***Disallowed*** after 2035
RSA [FIPS186]	112 bits of security strength	***Deprecated*** after 2030 ***Disallowed*** after 2035
	≥ 128 bits of security strength	***Disallowed*** after 2035

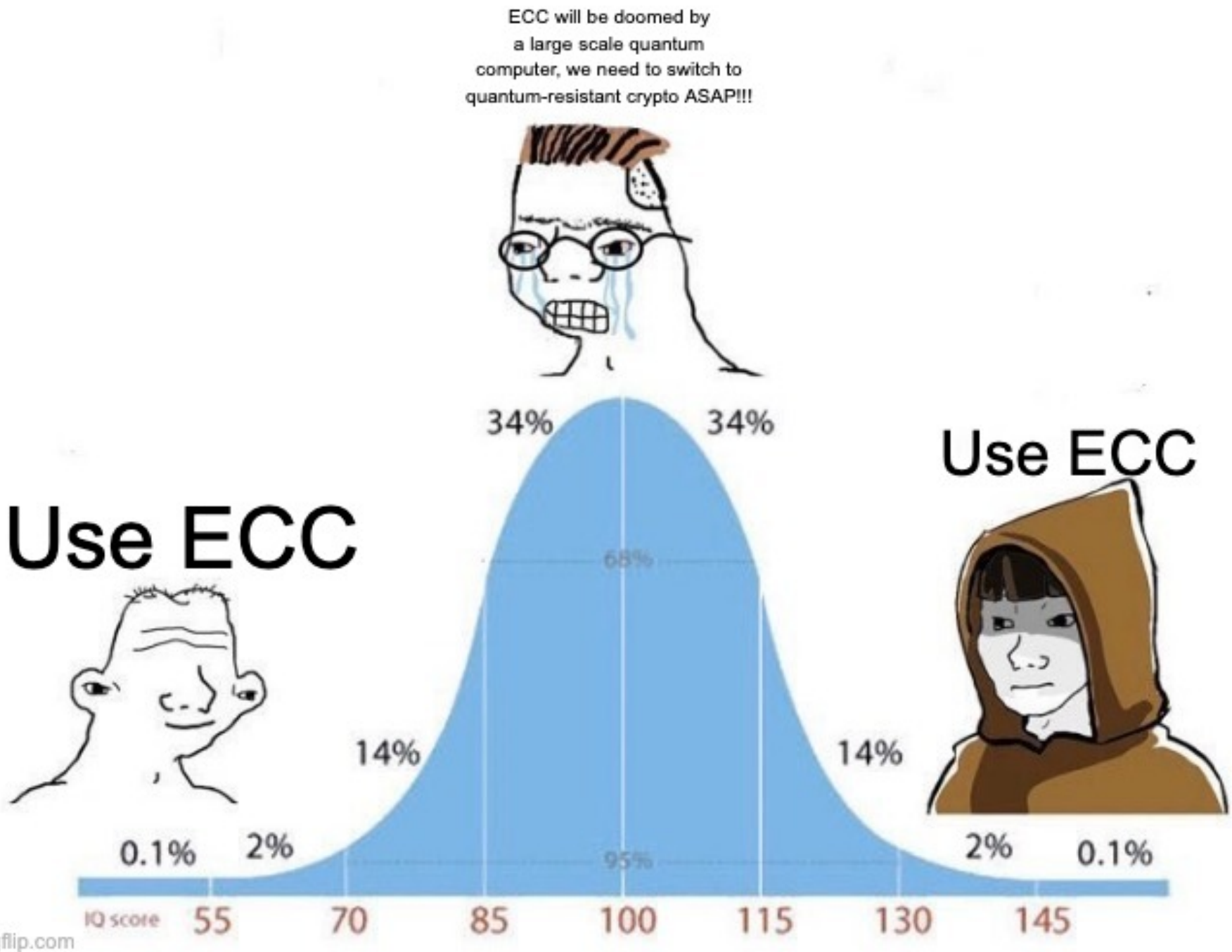
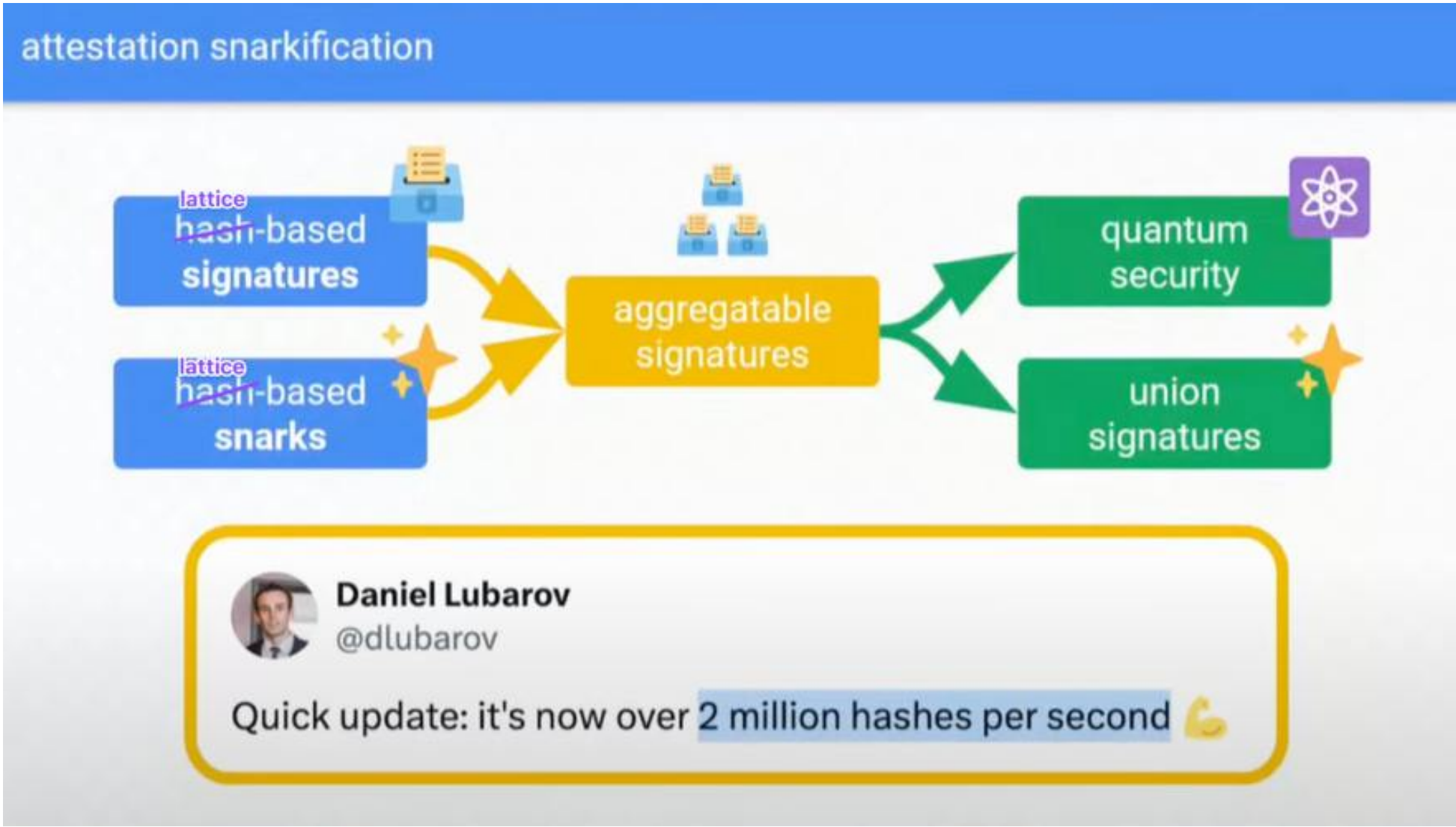

ECC will be doomed by
a large scale quantum
computer, we need to switch to
quantum-resistant crypto ASAP!!!
Use ECC
Use ECC
34%
34%
68%
14%
14%
95%
0.1%
2%
2%
0.1%
IQ score
55
70
85
100
115
130
145
imgflip.com

Table of Contents

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

Why NOT Lattice?



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

[1] 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. [2]

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

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

[2] 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 $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$:

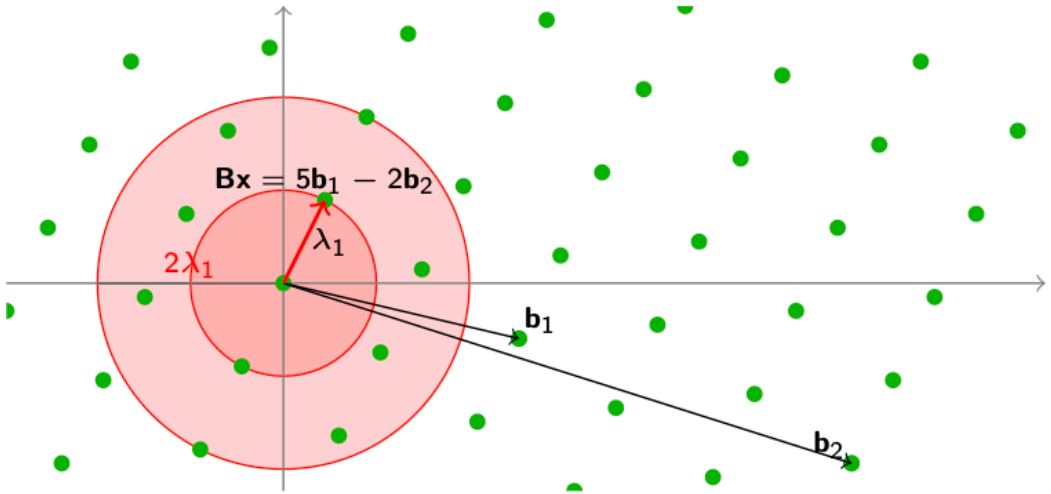
$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases.

Shortest Vector Problem

Definition (SVP_γ)

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) $\|\mathbf{Bx}\| \leq \gamma\lambda_1$



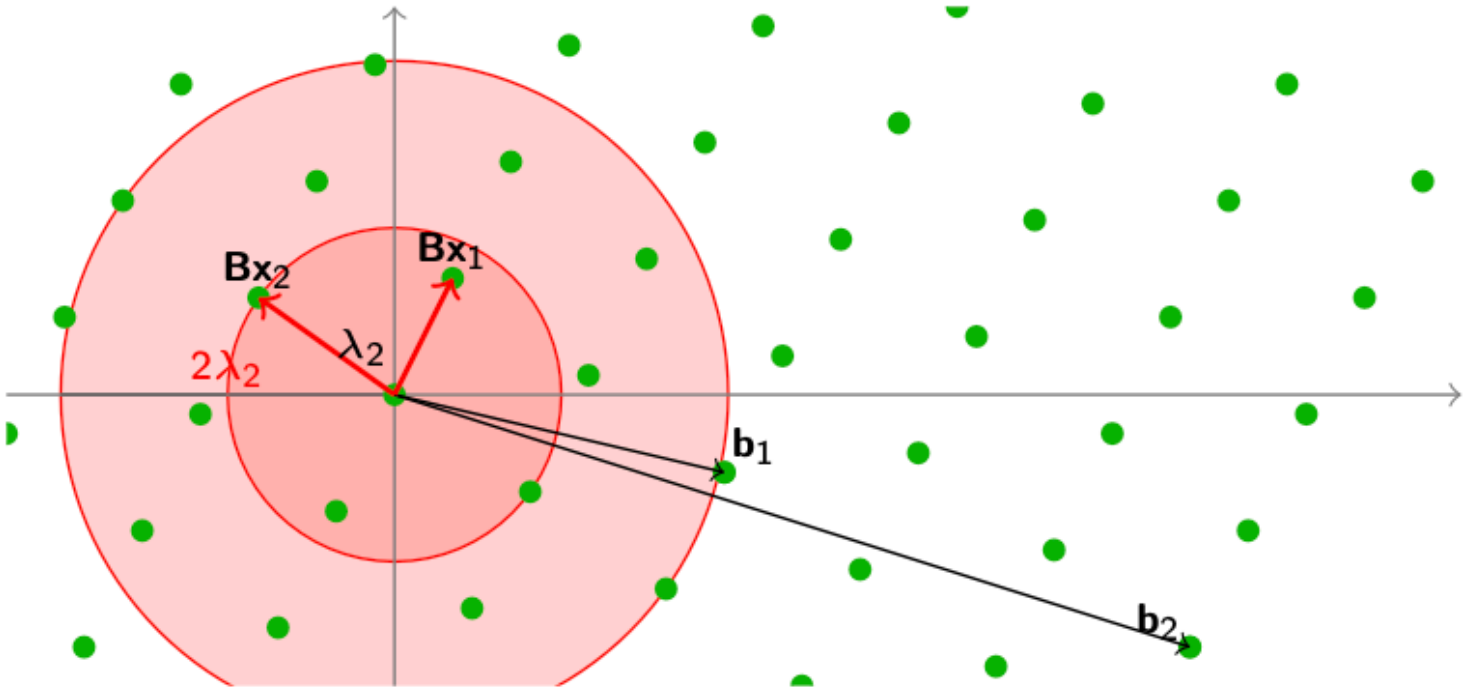
Minimum distance

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

Shortest Independent Vectors Problem

Definition (SIVP_γ)

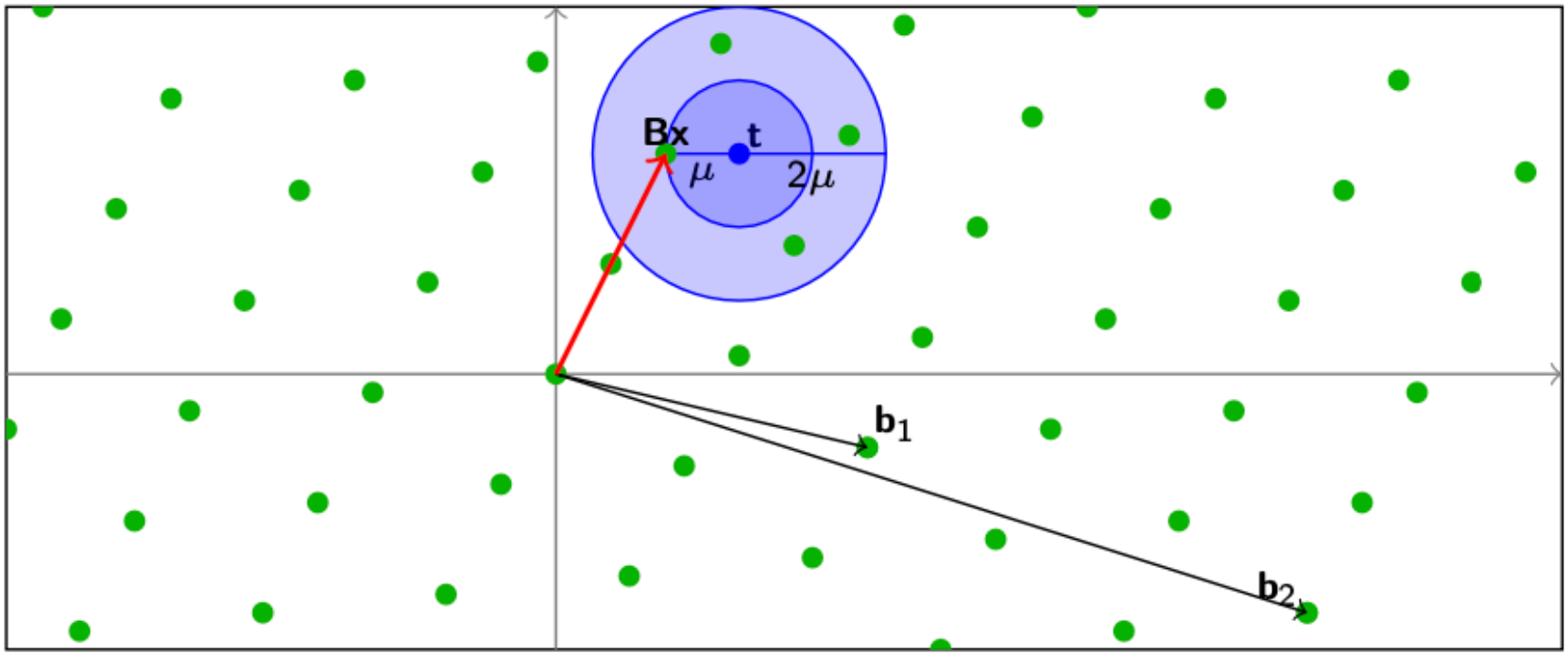
Given a lattice $\mathcal{L}(\mathbf{B})$, find n linearly independent lattice vectors $\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n$ of length (at most) $\max_i \|\mathbf{Bx}_i\| \leq \gamma\lambda_n$



Closest Vector Problem

Definition (CVP_γ)

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

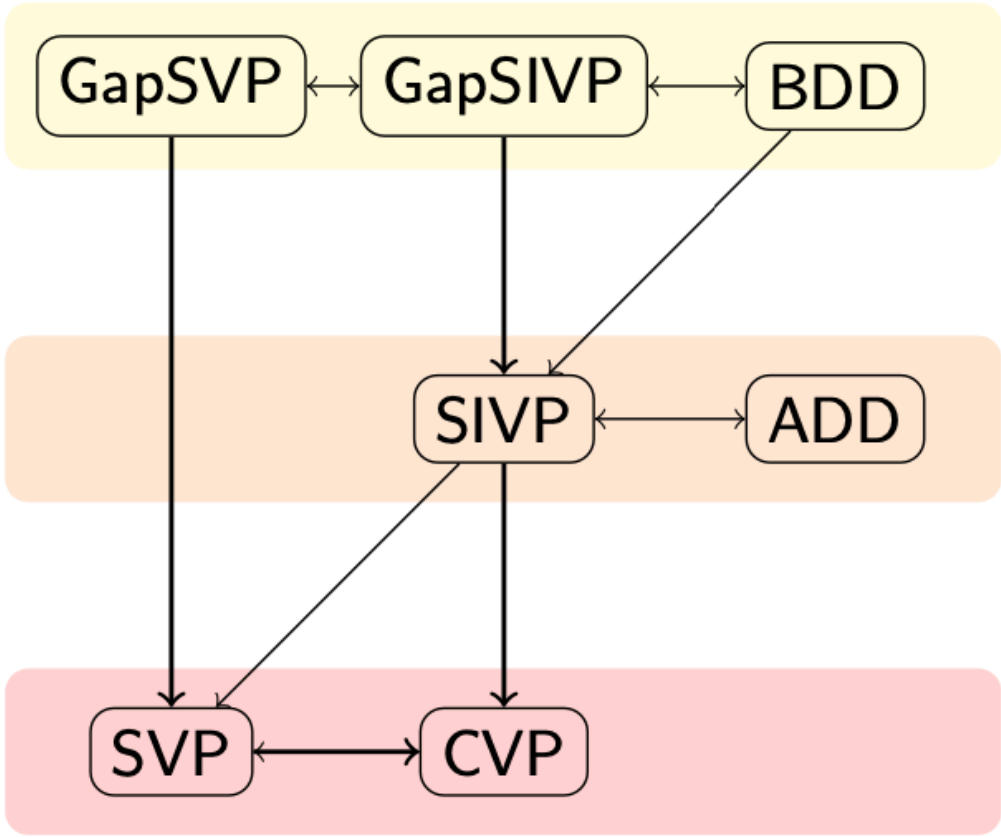
Definition

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

- 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

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



Random lattices in Cryptography

- Cryptography typically uses (random) lattices $\wedge$ such that $\wedge \subseteq \mathbb{Z}^d$ 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)

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

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

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

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

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

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

Theorem (R'05)

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

How are they related?

LWE and q-ary lattices

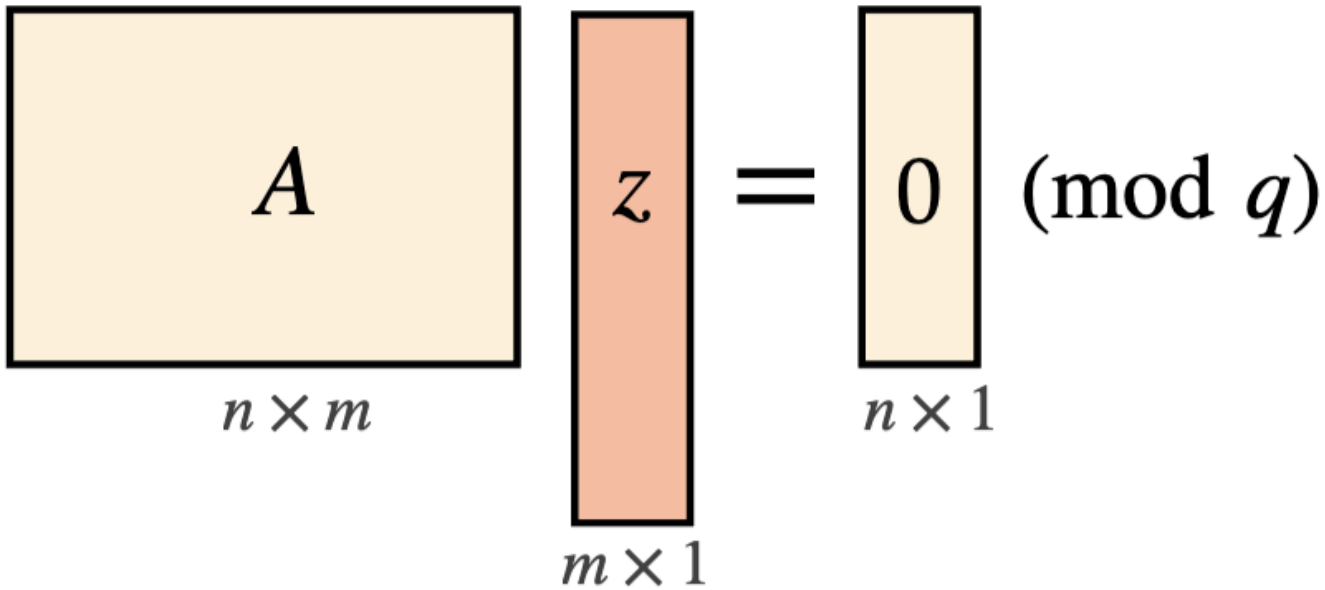
- If $\mathbf{e} = \mathbf{0}$, then $\mathbf{As} + \mathbf{e} = \mathbf{As} \in \Lambda\left(\mathbf{A}^t\right)$
- Same as CVP in random q-ary lattice $\Lambda\left(\mathbf{A}^t\right)$ with random target $\mathbf{t} = \mathbf{As} + \mathbf{e}$
- Usually $\mathbf{e}$ is shorter than $\frac{1}{2}\lambda_1\left(\Lambda\left(\mathbf{A}^T\right)\right)$, 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 $_{n,m,\beta}$ over $\mathcal{R}_q$ is defined as follows.

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



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 $H_A(z) = Az (\mathrm{mod}\, q)$.

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 $Az_1 = Az_2 (\mathrm{mod}\, q)$, whence $Az = 0 (\mathrm{mod}\, q)$ 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

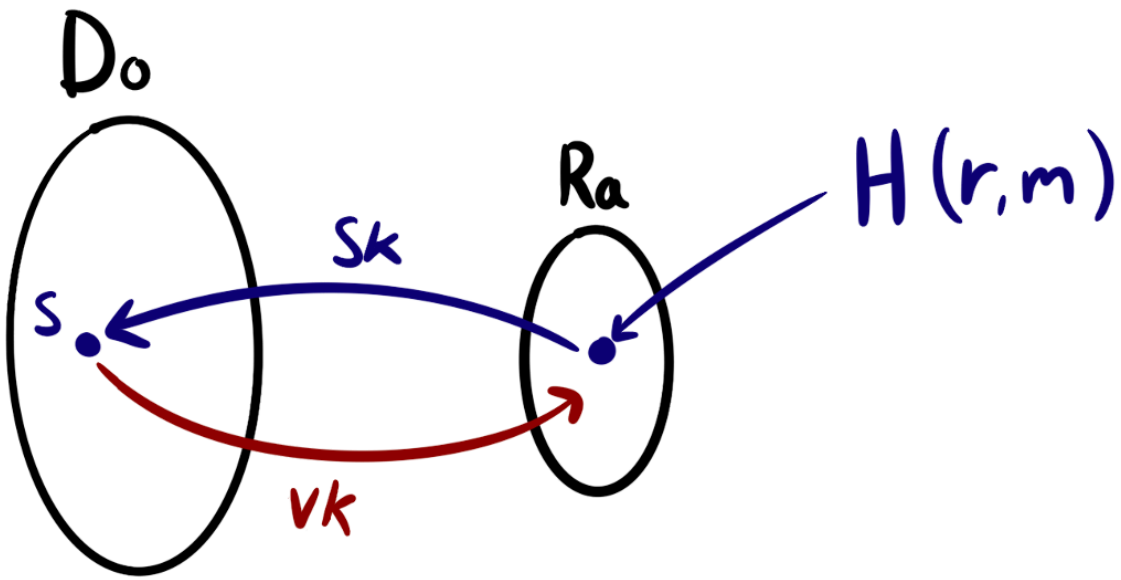
- The NTRU cryptosystem is parameterized by a certain polynomial ring $R = \mathbb{Z}[X]/(f(X))$, e.g., $f(X) = X^n - 1$ for a prime n or $f(X) = X^n + 1$ for an n that is a power of two, and a sufficiently large odd modulus q that defines the quotient ring $R_q = R/qR$.
- The public key is $h = 2g \cdot s^{-1} \in R_q$ 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 $y = \mathrm{H}(m)$ in the range of f, then outputs the signature $\sigma = f^{-1}(y)$.
- Verification consists of checking that $f(\sigma) = \mathrm{H}(m)$.



Recall RSA-FDH / BLS signature

GPV framework: lattice-based hash-and-sign

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

Falcon: GPV over NTRU lattice

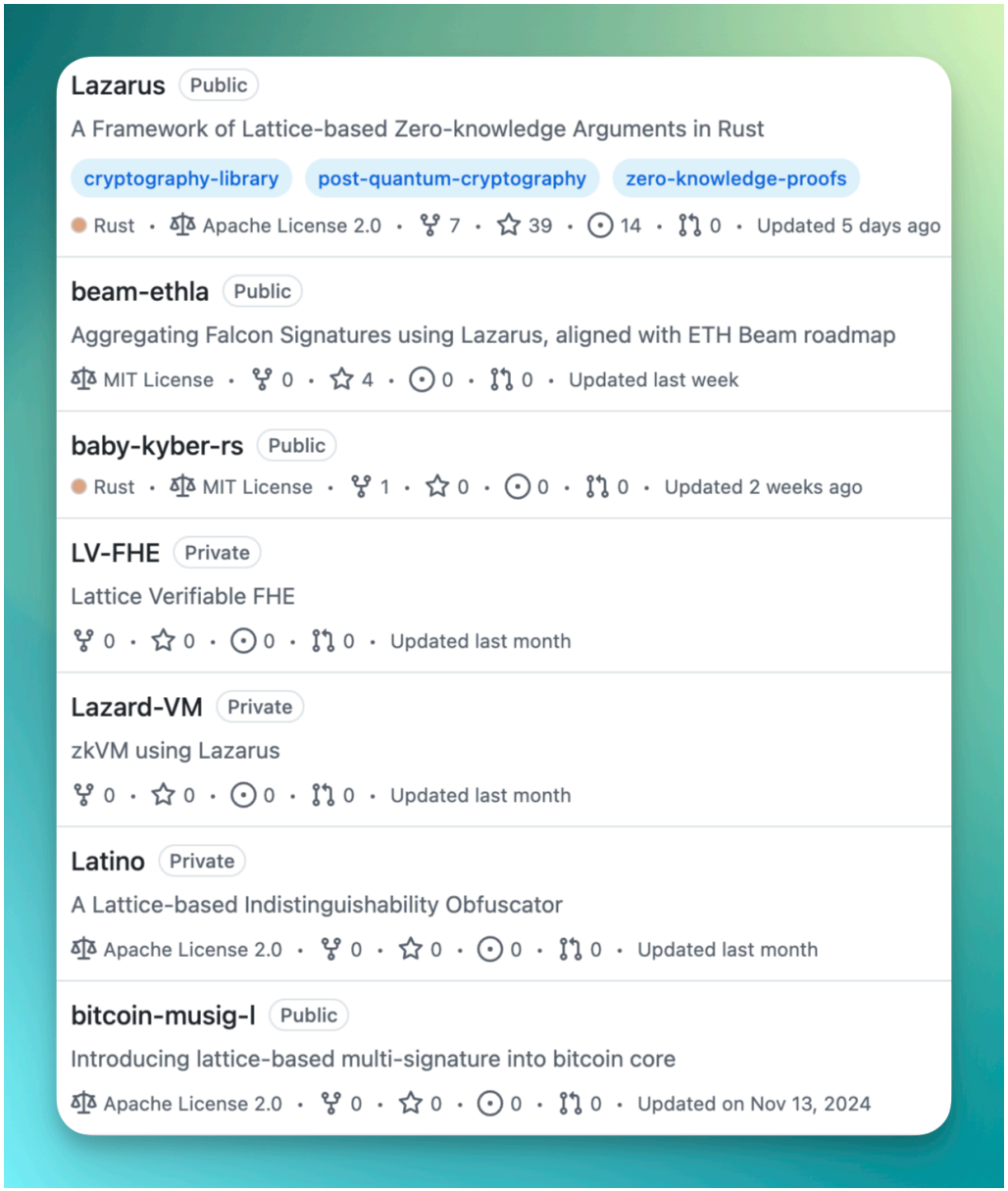
- $\mathrm{pk} = \boldsymbol{h} = g \cdot f^{-1} \bmod q \in \mathcal{R}_q$ defines an NTRU-module $\Lambda = \left\{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}^2 : \boldsymbol{u} + \boldsymbol{h}\boldsymbol{v} = \boldsymbol{0} \bmod q\right\}$
- sk contains a secret (short) basis of Λ that allows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

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

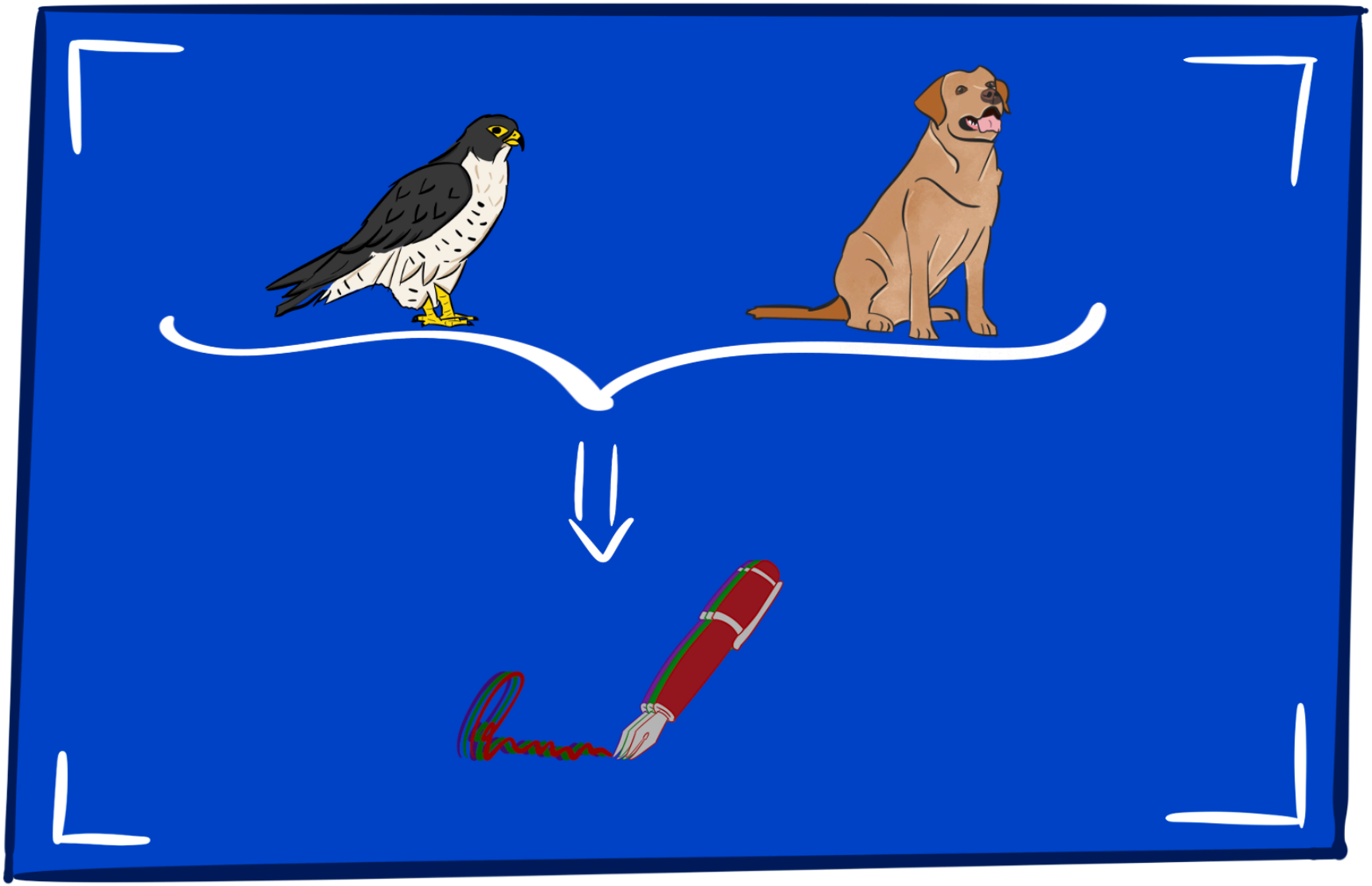
What is LaBRADOR

- 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 :)

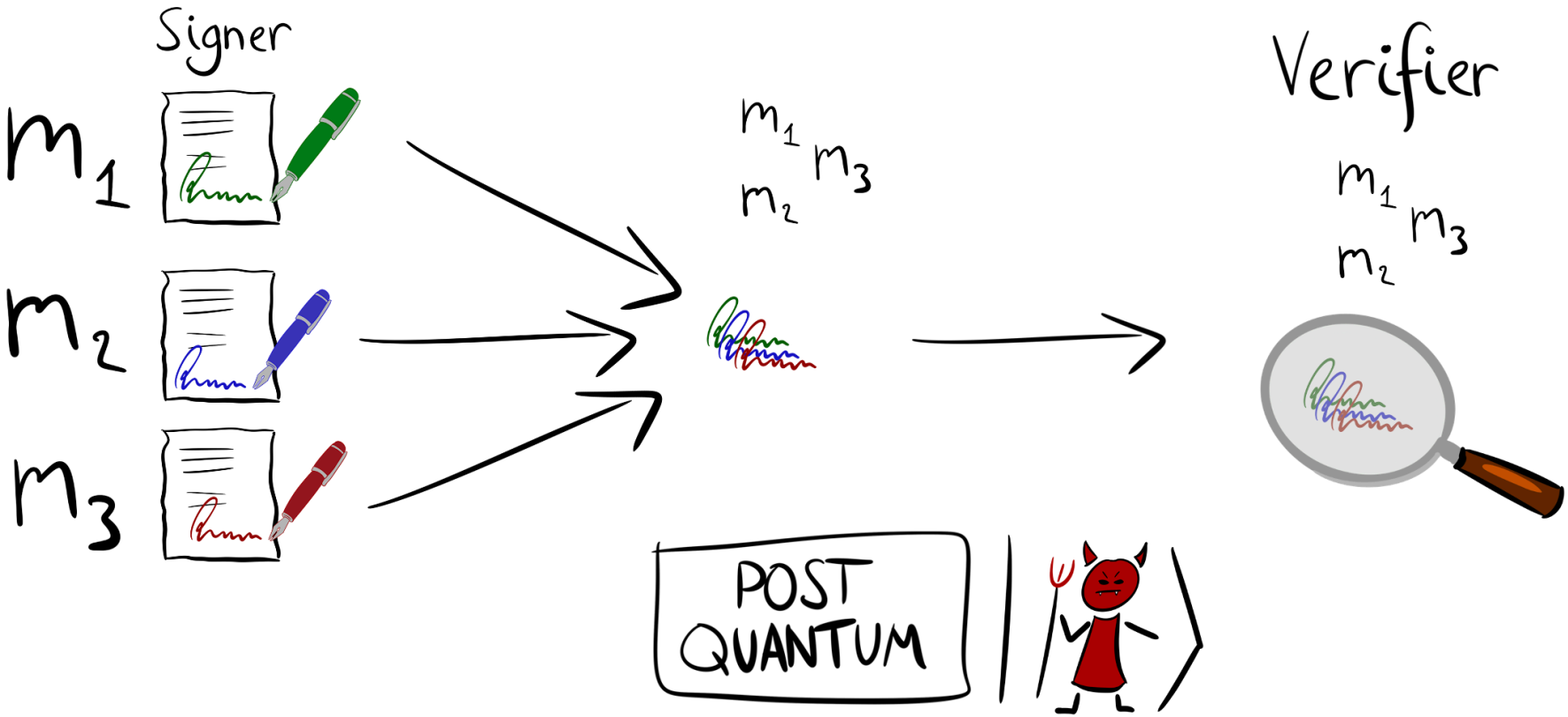
LaCOM


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

Aggregating Falcon Signatures with LaBRADOR



(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

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

Definition (Correctness)

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

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

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

Aggregate Signature: Correctness & Security

Definition (Unforgeability)

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

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

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

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$\mathcal{Q} := \varnothing$
$(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) \leftarrow \mathcal{A}^{\mathsf{OSign}}(\mathsf{pp}, \mathsf{pk})$
if $\mathsf{AggVer}(\{\mathsf{pk}_i, m_i\}_{i\in[N]}, \sigma_{\mathsf{agg}}) \wedge \exists i^* \in [N] : (\mathsf{pk}_{i^*} = \mathsf{pk} \wedge m_{i^*} \notin \mathcal{Q})$ **then**
    return 1
else
    return 0

$\mathsf{OSign}(m)$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return σ

Computing hash locally

Batch-proving knowledge of N Gaussian samples $(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2})_{i\in[N]}$ *and salts* $(r_i)_{i\in[N]}$ *meeting verification conditions w.r.t. a list* $(m_i, \boldsymbol{h}_i)_{i\in[N]}$ *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 $\boldsymbol{t}_i = \mathrm{H}(r_i, m_i)$ locally.

Adapting LaBRADOR for Aggregating Falcon Signatures

Principal relation of LaBRADOR:

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

- 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: $\forall i = 1, \ldots, N, \boldsymbol{s}_{i,1} + \boldsymbol{h}_i \boldsymbol{s}_{i,2} - \boldsymbol{t}_i = \boldsymbol{0} \bmod q$ and $\sum_{i=1}^{N} \sum_{j=1}^{2} \|\boldsymbol{s}_{i,j}\|_2^2 \leq N\beta^2$.

If LaBRADOR was instantiated over the ring used by Falcon then $(s_{i,1}, s_{i,2})_{i \in [N]}$ 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. $\rightarrow$ 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 λ and some corresponding constant C_1. For both Falcon parameter sets this is not satisfied. $\rightarrow$ 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). $\rightarrow$ 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 $\rightarrow$ Lifted to composite modulus for NTT [CHK+'22]
- see more in Greyhound [NS'24]

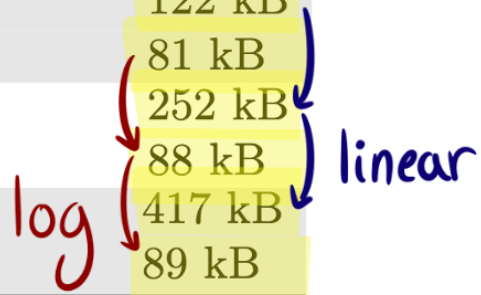
Estimates and Comparison

Single Falcon sig size $\approx$ 666 B

Aggr. Signature Scheme	# Sign. N	Sec. Param. λ	Life Cycle	$\lvert\sigma_{\text{agg}}\rvert$
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	∞	122 kB
Ours for Falcon-512 without salt	1024	121	∞	81 kB

Concatenation $\approx$ 682 KB , -82% [3]

Aggr. Signature Scheme	# Sign. N	Sec. Param. λ	$\lvert\sigma_{\text{agg}}\rvert$
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



[3] https://github.com/dfaranha/aggregate-falcon

Thanks!