Quantum-Safe Zero-Knowledge for Privacy

Gregor Seiler

IBM Research Europe

May 10, 2026

Transition to Quantum-Safe

Progress towards cryptographically relevant quantum computers has accelerated:

- Physical qubit quality now sufficient for quantum error correction
- Fault tolerant quantum computers by 2029 (200 logical qubits, 100M gates)
- Algorithmic advances: < 1200 logical qubits to break ECDLP

The web is transitioning on an accelerated schedule:

- Already $> 65\%$ of human-initiated traffic to cloudflare is encrypted with ML-KEM (Kyber)
- The web will be fully quantum-safe including authentication using ML-DSA (Dilithium) by 2029

What about privacy-preserving cryptography, e.g. EUDI Wallet?

Timely Example: Electronic Identities

Users get certificates for their **credentials** (name, date of birth, ...) from **issuer**

Privacy: Users don't want to be traceable when presenting their certificates

Stop gap: Batched issuance – each certificate is only presented once

Credential signatures

Certificates are **signatures** of credentials

Proper untraceability: Users prove knowledge of signature in zero-knowledge

Which combination of **signature scheme** and zero-knowledge **proof system**?

Possible choices for quantum-safe signature schemes

NIST standards:

- ML-DSA (Dilithium)
- FN-DSA (Falcon)
- SLH-DSA (Sphincs+)

On-ramp:

- FEAST
- MAYO

Unfortunately, no replacement for nice proof-friendly standard-model signatures such as BBS

Modifications for easier proofs?

Dilithium

$$\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{t}) \quad \mathsf{sig} = (\boldsymbol{c}, \boldsymbol{z})$$

Verification:

$$\boldsymbol{w} := \boldsymbol{A}\boldsymbol{z} - \boldsymbol{c}\boldsymbol{t}$$
$$\mathsf{H}(\boldsymbol{w}, \mathsf{pk}, \mathsf{msg}) \stackrel{?}{=} \boldsymbol{c}$$
$$\|\boldsymbol{z}\| \stackrel{?}{\leq} \beta$$

Not proof friendly: SHAKE as random oracle H, with fixed-weight output polynomial

Falcon

$$\mathsf{pk} = \boldsymbol{A} \quad \mathsf{sig} = \boldsymbol{s}$$

Verification:

$$\boldsymbol{A}\boldsymbol{s} \stackrel{?}{=} H(\mathsf{msg})$$
$$\|\boldsymbol{s}\| \stackrel{?}{\leq} \beta$$

Not proof friendly: SHAKE as random oracle H, with several KB output

Proof-Friendly Falcon Variant

Replace SHAKE-based H by proof-friendly random oracle, e.g.

- Poseidon
- Lattice-based hash function à la Swifft

Even better under MSIS_f assumption: Replace $H(\mathrm{msg})$ by $\boldsymbol{B}\vec{\mu}$ where $\vec{\mu} = \mathrm{bin}(\mathrm{msg})$,

$$\boldsymbol{As} = \boldsymbol{B}\vec{\mu}$$

Alternative to signatures: Merkle tree credentials [RWGM23]

The web transitions away from signatures to Merkle trees.
Maybe also electronic identities?

zk-creds:

- Issuers create Merkle trees with user certificates as leafs
- Issuers publish the roots and send membership paths to users
- Users prove membership in a recent tree

Simple to set-up proofs when hash function is proof friendly (i.e. lattice-based)

Short-lived validity that is favourable in practice

Proof of Possession

Certificates need to be bound to a **user device** by adding pk of secure element to the credentials

In presentations, need to prove device possession:

- Secure element generates PoP signature for random challenge message

Need to hide PoP public key:

- User proves in zero-knowledge that PoP signature verifies with respect to pk among credentials

PoP Signature choice

Signature scheme needs to be supported by secure elements

Hence, realistically, it has to be Dilithium

Friendly Dilithium PoP

Dilithium verification: $\mathsf{H}(\underbrace{\boldsymbol{Az} - \boldsymbol{ct}}_{\boldsymbol{w}}, \mathsf{pk}, \mathsf{msg}) \stackrel{?}{=} \boldsymbol{c}$

Observe:

- PoP message is public random challenge
- User can output $\boldsymbol{w} = \boldsymbol{Az} - \boldsymbol{ct}$

Simplify:

- Don't hash pk in random oracle
- Same $\boldsymbol{A}$ for all public keys

Easy proof:

- Publicly evaluate $\boldsymbol{c} = \mathsf{H}(\boldsymbol{w}, \mathsf{msg})$
- User proves knowledge of $\boldsymbol{z}$ s.t. $\boldsymbol{Az} = \boldsymbol{w} + \boldsymbol{ct}$, $\|\boldsymbol{z}\| \leq \beta$

Fixed $\boldsymbol{A}$ means it doesn't need to be part of witness, and no proof of correct expansion

Blueprint for efficient electronic identity system

- Merkle tree certificates with collision-resistant lattice hashes (Only SIS, don't need to be random oracles)
- Mild modification of Dilithium for simple PoP

Only need to prove lattice relations of the form

$$\boldsymbol{A}\boldsymbol{s} = \boldsymbol{t}, \|\boldsymbol{s}\| \leq \beta$$

Perfect for a lattice proof system!

Advantages of lattice proof systems

- Natively support proving **lattice relations** without overhead
- Have built-in support for **l2-norm** proofs (the "right" norm for lattice schemes)
- Share many **implementation details** with lattice schemes
- Have **smaller** proof sizes and **faster** prover runtimes than hash-based proof systems (STARKs)

Other options

- Use standard signature scheme for credential signature
- Use hash-based proof system (longfellow)
- Proof MAYO using (linear-size) VOLE-in-the-head proofs

Lattice Proof Construction

Algebraic Setup

Arithmetic takes place over power-of-two cyclotomic ring modulo rational prime

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$$

Allows for high-performance NTT-based multiplier implementations

But:

- Proof systems don't allow for fully-splitting primes q
- Residue fields ("NTT slots") need to be exponentially large for soundness
- Can for example split into prime factors of degree 4 when $q \approx 2^{32}$

Principal Relation

Parameterized by a **rank** n and a **multiplicity** r

Witness consists of r (polynomial) vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r$ of rank n that fulfill many **dot-product relations**

$$f^{(k)}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r) = \sum_{i,j} \boldsymbol{a}_{ij}^{(k)} \langle \boldsymbol{s}_i, \boldsymbol{s}_j \rangle + \sum_i \langle \boldsymbol{\varphi}_i^{(k)}, \boldsymbol{s}_i \rangle + \boldsymbol{b}^{(k)} = \boldsymbol{0},$$

and **norm constraints**

$$\|\boldsymbol{s}_i\|^2 \leq \beta_i^2$$

Polynomial evaluation constraint

Witness vectors $\boldsymbol{s}_i$ represent multivariate polynomial $\check{\boldsymbol{s}}(X) = \mathrm{MLE}[\boldsymbol{s}](X_1, \ldots, X_{\log n})$ that fulfills evaluation constraint

$$\check{\boldsymbol{s}}(x) = \boldsymbol{y}$$

Over cyclotomic ring $\mathcal{R}_q$:

- Useful for proving lattice relations via sumcheck

Over finite field $\mathbb{F}_{p^l}$:

- Allows for proving generic circuit representations such as R1CS, Plonkish, Air, CCS

Anatomy of a lattice proof

Main operations:

1. **Decompose** witness vectors
2. **Commit** to witness vectors
3. **Project** witness vectors
4. **Batch** integer dot-product relations
5. **Lift** batched integer relations to cyclotomic polynomial relations
6. **Aggregate** polynomial relations
7. **Prove** aggregated relation
8. **Amortize** witness vectors
9. **Recurse** proof system

Adding zero-knowledge:

- Add randomness to commitments to make them hiding
- Mask projections, liftings, amortized opening
- Perform rejection sampling

Ajtai Commitments

Commit to short witness vectors $\boldsymbol{s}_i$, $\|s\|_i \leq \beta_i$

$$\boldsymbol{t}_i = \mathrm{Com}(\boldsymbol{s}_i) = \boldsymbol{A}\boldsymbol{s}_i$$

for public uniformly random matrix $\boldsymbol{A} \in \mathcal{R}_q^{\kappa \times n}$

Computing high-rank linear maps over cyclotomic rings is extremely fast on modern CPUs: Vectorizable and massively parallel. Main reason for high performance of lattice proof systems

- NTT of matrices can be sampled / stored
- NTT of witness vector is amortized over rows of A
- Hence pointwise products dominate runtime
- Multimodular multiplier modulo small fully-splitting primes beats direct multiplier with partial NTT for large n

Merkle Tree Commitments

Linear size of commitment matrices is barrier for sublinear verification

- Can use tensor-structured matrices under vSIS assumption [CLM23]

Alternatively, multi-layered Merkle tree commitments with constant-size $\boldsymbol{A}$

$$\boldsymbol{A} \begin{pmatrix} g^{-1}(\boldsymbol{A}\boldsymbol{s}_i^t) \\ g^{-1}(\boldsymbol{A}\boldsymbol{s}_i^b) \end{pmatrix}$$

where g^{-1} decomposes wrt small base

Modular Johnson-Lindenstrauss Projection

Johnson-Lindenstrauss: Random linear maps from high to low dimensional Euclidean space preserve ℓ_2-norm up to small constants with high probability

Modular Johnson-Lindenstrauss: Even if projected vector is only short mod q this implies shortness (under mild conditions)

$$\vec{p} = \begin{pmatrix} -1 & 1 & 0 & -1 \\ 0 & 1 & -1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

- Verifier sends random matrices Π_i
- Prover sends projection $\vec{p} = \sum_i \Pi_i \vec{s_i} \in \mathbb{Z}_q^{256}$
- Verifier checks $\|\vec{p}\| \leq \sqrt{128}\beta$

For sublinear verification can be layered as in commitments

Amortization

Amortize over witness vectors $\boldsymbol{s}_i$

- Verifier sends short **challenge polynomials** $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_r$
- Prover sends **amortized opening**

$$\boldsymbol{z} = \boldsymbol{c}_1 \boldsymbol{s}_1 + \boldsymbol{c}_2 \boldsymbol{s}_2 + \cdots + \boldsymbol{c}_r \boldsymbol{s}_r$$

- Verifier checks

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{c}_1 \boldsymbol{t}_1 + \cdots + \boldsymbol{c}_r \boldsymbol{t}_r$$
$$\|\boldsymbol{z}\| \leq \gamma$$

- Coordinate-wise extraction gives weak openings for all commitments

Proof of aggregated dot-product relation

Two approaches for proving dot-product relation

$$\sum_{i,j} \boldsymbol{a}_{ij} \langle \boldsymbol{s}_i, \boldsymbol{s}_j \rangle + \sum_i \langle \boldsymbol{\varphi}_i, \boldsymbol{s}_i \rangle + \boldsymbol{b} = \boldsymbol{0},$$

Direct proof:

- Prover sends garbage terms $\boldsymbol{g}_{ij} = \langle \boldsymbol{s}_i, \boldsymbol{s}_j \rangle$, $\boldsymbol{h}_{ij} = \langle \boldsymbol{\varphi}_i, \boldsymbol{s}_j \rangle$ before knowing challenges $\boldsymbol{c}_i$
- Verifier checks

$$\langle \boldsymbol{z}, \boldsymbol{z} \rangle = \sum_{i,j} \boldsymbol{c}_{ij} \boldsymbol{g}_{ij} \qquad \langle \boldsymbol{\varphi}, \boldsymbol{z} \rangle = \sum_{i,j} \boldsymbol{c}_{ij} \boldsymbol{h}_{ij}$$

$$\sum_{i,j} \boldsymbol{a}_{ij} \boldsymbol{g}_{ij} + \sum_i \boldsymbol{h}_i + \boldsymbol{b} = \boldsymbol{0}$$

Second approach: Proof via sumcheck

- Write dot-product relation as sumcheck relation

$$\sum_{x,x'} \sum_{i,j} \boldsymbol{a}_{ij} eq_{ij}(x,x') \check{\boldsymbol{s}}(x) \check{\boldsymbol{s}} f(x') + \sum_{x} \sum_{i} eq_i(x) \check{\varphi}(x) \check{\boldsymbol{s}}(x) + \boldsymbol{b} = \boldsymbol{0}$$

- Reduce to polynomial evaluation via sumcheck protocol

Now efficient due to aggregation over subfields [KLNOT26]

Recursion

Verifier checks are all of dot-product or norm constraint type

- Last prover message can be proven with iteration of the protocol
- Size of other prover messages can be hidden by wrapping them in commitments and opening them in last message
- Fiat-Shamir hashes are computed in the clear

Decomposition in Rank

Before recursing the protocol, want to increase multiplicity and decrease rank

Decomposition in rank: Split vectors of rank n into r vectors of rank n/r:

$$\boldsymbol{z} = \boldsymbol{z}_1 \parallel \cdots \parallel \boldsymbol{z}_r$$

Quadratic term $\langle \boldsymbol{z}, \boldsymbol{z} \rangle$ transforms as

$$\langle \boldsymbol{z}, \boldsymbol{z} \rangle = \langle \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle + \cdots + \langle \boldsymbol{z}_r, \boldsymbol{z}_r \rangle$$

Decomposition in Width

Amortization blows up standard deviation due to multiplication by challenge polynomials; consequently, lattice parameters need to increase to retain SIS-hardness

Decomposition in width:

$$\boldsymbol{z} = \boldsymbol{z}_0 + b\boldsymbol{z}_1 \text{ with } \|\boldsymbol{z}_0\|_\infty \leq \frac{b}{2}$$

Quadratic term $\langle \boldsymbol{z}, \boldsymbol{z} \rangle$ transforms as

$$\langle \boldsymbol{z}, \boldsymbol{z} \rangle = \langle \boldsymbol{z}_0, \boldsymbol{z}_0 \rangle + 2b\langle \boldsymbol{z}_0, \boldsymbol{z}_1 \rangle + b^2 \langle \boldsymbol{z}_1, \boldsymbol{z}_1 \rangle$$

Matrix $(\boldsymbol{a}_{ij})$ was diagonal, now becomes tridiagonal

Overview of current concrete lattice proofs

LNP [LNP22]: Linear proof sizes, concretely very small proofs for small statements, zero-knowledge

LaBRADOR [BS23]: Polylog proof sizes, concretely smallest quantum-safe proofs for big statements ($\approx$ 50KB)

Greyhound [NS24]: Polynomial commitment scheme, square root verifier complexity

RoKoko [KLNOT26]: Polynomial commitment scheme, polylog verifier complexity

Current benchmarks

	2^{25}				2^{29}			
	size	comm	prove	verify	size	comm	prove	verify
Brakedown-PC	49′157 KB	36 s	3.21 s	0.703 s	181′948 KB	605 s	48.6 s	2.96 s
Ligero-PC	7′256 KB	83.9 s	3.13 s	0.338 s	28′631 KB	1590 s	51.6 s	1.57 s
FRI-PC	740 KB	168 s	185 s	0.041 s	—	—	—	—
WHIR	640 KB	105 s		5 ms	808 KB	1850 s		6.4 ms
Greyhound	47 KB	1.84 s	0.788 s	239 ms	52 KB	72 s	21.7 s	1.61 s
RoKoko	187 KB	0.71 s	0.94 s	6.85 ms	187 KB	11.17 s	4.04 s	9.5 ms

Unified proof system

From LNP: Masking and rejection sampling for zero-knowledge

From LaBRADOR: Outer commitments and advanced parameterization for small proofs

From Greyhound: Efficient polynomial evaluation

From RoKoko: Sumcheck reduction from lattice relations to polynomial evaluation for polylog verifier

Plus: Merkle tree commitments for poylog verifier under standard assumption

Should have best of everything:

- Support of lattice relations as well as finite-field polynomial evaluations
- Smallest quantum-safe proof sizes for small and big statements
- Fastest prover runtimes
- Millisecond verifier runtimes
- Zero-Knowledge

Summary and Discussion

We have several practical approaches to quantum-safe privacy protocols, in particular pq credentials

Turning them into production ready solutions is a huge amount of work:

- End-to-end demonstrator implementation
- More implementation research
- Standards
- Formal verification
- ...

Thank you!