



NTNU

Norwegian University of
Science and Technology

QUANTUM-SAFE SIGNATURES

TTM4205 – Lecture 14

Tjerand Silde

09.10.2026

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

Reference Material

These slides are based on:

- ▶ The referred papers and standards in the slides
- ▶ JPA: parts of chapter 14
- ▶ DW: parts of chapter 14
- ▶ Lyubashevsky, *Basic Lattice Cryptography*,
<https://eprint.iacr.org/2024/1287>



Where This Fits

- ▶ **Lecture 13** replaced key exchange: LWE, and ML-KEM built on it
- ▶ **Today** we replace signatures: the same lattices, a second hard problem, and three quite different designs
- ▶ **Lecture 15** attacks the implementations of both

Recall: Cryptography Tomorrow

- ▶ Quantum computers are not better; they are different
- ▶ They will generally be worse, but do specific things better
- ▶ Shor's algorithm breaks every public key scheme based on factoring or discrete logarithms, in polynomial time
- ▶ Grover's algorithm only halves symmetric key lengths, in the worst case, so AES-128/256 and SHA-2/3 stay (for now)

Signatures Are the Harder Half

Key exchange is quantum-safe in your browser today. Signatures are not, and the reason is not mathematics but *deployment*.

A key exchange involves two parties who both upgrade at once. A signature is verified by parties who were not present when it was made: certificate chains, firmware, software updates, and hardware roots of trust that ship once and last a decade.

And the objects are larger. A quantum-safe certificate chain is ten times the size of today's, which collides with limits nobody thought about when TLS was designed.



Quantum-Secure Cryptography in Messaging Apps

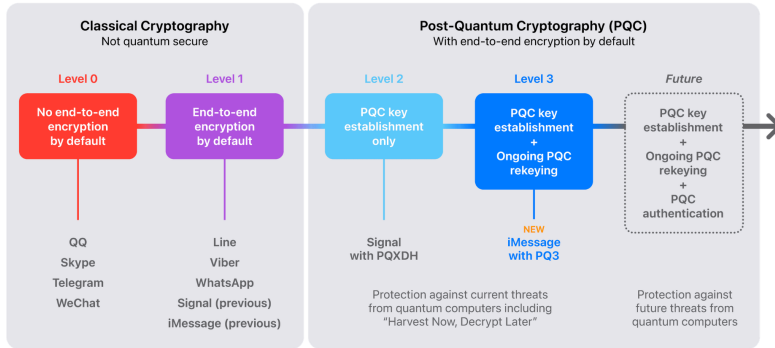


Figure: <https://security.apple.com/blog/imessage-pq3>

The NIST Standards

Standard	Algorithm	Origin	Status
FIPS 203	ML-KEM	CRYSTALS-Kyber	published August 2024
FIPS 204	ML-DSA	CRYSTALS-Dilithium	published August 2024
FIPS 205	SLH-DSA	SPHINCS+	published August 2024
FIPS 206	FN-DSA	FALCON	draft August 2025, final expected 2026–27
FIPS 207	HQC	HQC	selected March 2025, draft expected 2026

- ▶ ML = module lattice, SLH = stateless hash, FN = FFT over NTRU; KEM = key encapsulation, DSA = digital signature algorithm
- ▶ Three of the four signature-side standards are lattice-based. SLH-DSA is the deliberate exception.
- ▶ Separately, SP 800-208 (2020) standardises the *stateful* hash signatures XMSS and LMS

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

Recall: Computational Diffie-Hellman

Let $\mathbb{G}_p$ be a group of prime order p and g be a generator for $\mathbb{G}_p$.

Sample a, b uniformly at random from $\mathbb{Z}_p$. The Computational Diffie-Hellman problem is, given g, g^a and g^b , to find g^{ab} in $\mathbb{G}_p$.

Schnorr and (EC)DSA rest on the discrete logarithm problem, which is at least as hard; Shor solves all of them.

Recall: Learning With Errors

Definition

For positive integers n, m, q and $\beta < q$, write $[\beta] = \{-\beta, \dots, \beta\}$. The $\text{LWE}_{n,m,q,\beta}$ problem asks to distinguish:

1. $(A, As + e)$, where $A \leftarrow \mathbb{Z}_q^{m \times n}$, $s \leftarrow [\beta]^n$, $e \leftarrow [\beta]^m$
2. (A, u) , where $A \leftarrow \mathbb{Z}_q^{m \times n}$ and $u \leftarrow \mathbb{Z}_q^m$

Last lecture gave us encryption. For signatures we need a second problem.

New: Short Integer Solution (SIS)

Definition (Ajtai, 1996)

For positive integers n, m, q and $\beta < q$, the $\text{SIS}_{n,m,q,\beta}$ problem asks: given a random matrix $A \leftarrow \mathbb{Z}_q^{m \times n}$, the same shape as in LWE, find short vectors $s_1 \in [\beta]^n$ and $s_2 \in [\beta]^m$ such that

$$As_1 + s_2 = 0 \pmod{q}.$$

Without the shortness requirement this is trivial: pick any s_1 and set $s_2 = -As_1$. Asking for a *short* solution is what makes it a lattice problem: the solutions form a lattice, and we are asked for a short vector in it.

Where LWE asks you to *recover* or *distinguish* a short secret that someone planted, SIS asks you to *find* a short solution that merely exists.

SIS Hardness

The Short Integer Solution problem gets harder when...

- ▶ the dimension n gets larger
- ▶ the allowed solutions get smaller (smaller β)
- ▶ the modulus q gets larger

This is the opposite of LWE with respect to β and q . LWE is hard when the noise is large enough to hide the secret; SIS is hard when the solution has to be small enough to be rare.

Hardness of LWE and SIS

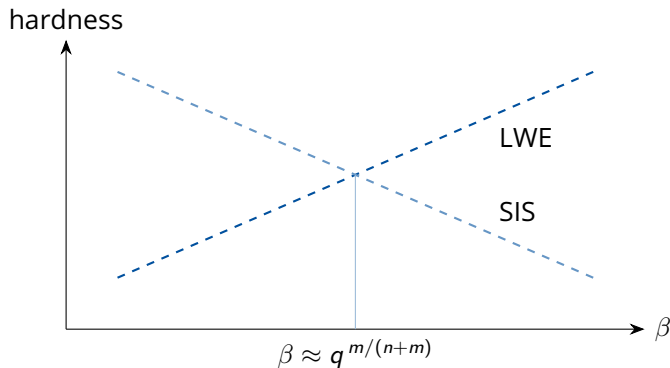


Figure: For fixed n, m, q : LWE gets harder and SIS easier as β grows. The lines only show the direction of the dependence, not concrete hardness.

Why a Signature Needs Both

A signature scheme has to hide the secret key (that is LWE: the public key is $t = As_1 + s_2$) *and* make forgery hard (that is SIS: a forger must produce a short vector satisfying a linear relation they did not construct).

ML-DSA's parameters therefore sit where both problems are hard at once: the sweet spot near the crossing of the two lines, with β small enough for SIS and large enough for LWE.

That is a narrower target than for encryption, and it is one reason lattice signatures are larger than lattice KEMs.

Basic Lattice Cryptography

The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA)

Vadim Lyubashevsky

IBM Research Europe, Zurich

vad@zurich.ibm.com

(Last updated: June 18, 2025)

Figure: <https://eprint.iacr.org/2024/1287.pdf>

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

Recall: Schnorr Signatures

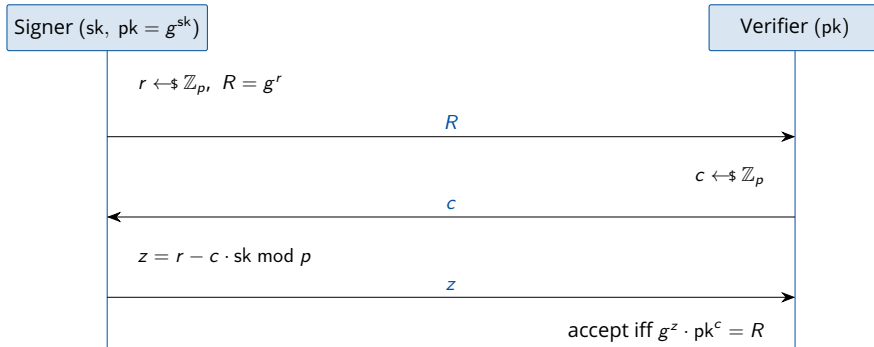


Figure: The Schnorr identification protocol. The signature scheme replaces the verifier's challenge by $c = H(pp, pk, m, R)$, which is the Fiat-Shamir transform.

Why Schnorr Leaks Nothing

The response $z = r - c \text{sk}$ is a uniformly random element of $\mathbb{Z}_p$, because r is: whatever sk is, adding a uniform r gives a uniform z .

So a transcript (R, c, z) can be produced by anyone *without* sk : pick c and z first, set $R = g^z \text{pk}^c$. Real and simulated transcripts have the same distribution, which is what “zero-knowledge” means here.

Over a lattice we lose this for free: the mask y has to be *short*, and a short mask plus $c s$ is not uniform. It is shifted towards $c s$, and that shift leaks the secret. Fixing this is the whole art of ML-DSA.

Fiat-Shamir with Aborts

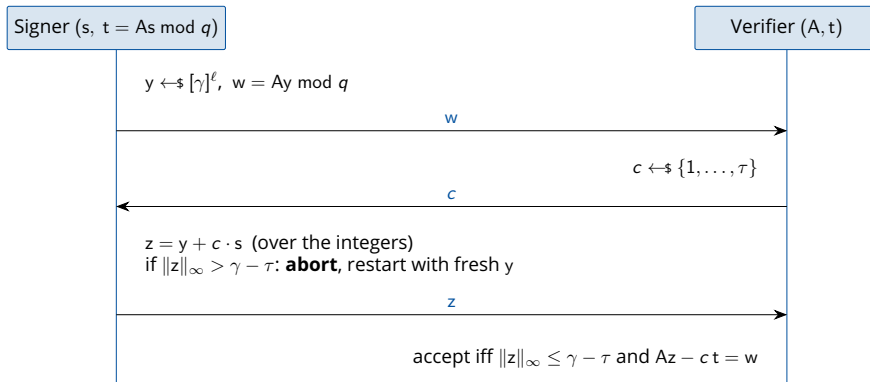


Figure: The toy ML-DSA of the weekly problem: $A \in \mathbb{Z}_q^{k \times \ell}$ with $\ell > k$, binary secret s , mask bound γ , challenge bound $\tau < \gamma$. Correctness: $Az - ct = Ay + cAs - cAs = w$.

Why the Abort

Each coordinate of $z = y + cs$ lies in $[-\gamma, \gamma]$ if $s_i = 0$ and in $[-\gamma + c, \gamma + c]$ if $s_i = 1$. The two ranges overlap except at the ends, and the *ends* give the secret away.

Rejecting every z with $\|z\|_\infty > \gamma - \tau$ cuts off exactly those ends. What survives is uniform on the box $[\gamma - \tau]^\ell$ *whatever s is*, so the transcript is simulatable again, and the Schnorr argument goes through.

The price is restarts: a coordinate is rejected with probability about τ/γ , so all ℓ survive with probability $(1 - \tau/\gamma)^\ell$. ML-DSA tunes γ so that this is roughly $1/4$, i.e. about four attempts per signature.

From the Toy to ML-DSA

- ▶ Everything lives in $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 8380417 = 2^{23} - 2^{13} + 1$, chosen for the NTT
- ▶ The key is $\mathbf{t} = A\mathbf{s}_1 + \mathbf{s}_2$ with *two* short secrets, an LWE sample rather than a bare SIS image, so the public key hides $\mathbf{s}$
- ▶ The challenge c is a polynomial with exactly τ coefficients equal to ± 1 , so $c\mathbf{s}$ stays short and c has $2^\tau \binom{256}{\tau}$ possible values
- ▶ A is expanded from a 32-byte seed, and the low bits of $\mathbf{t}$ are dropped from the public key; a *hint* in the signature lets the verifier repair the resulting rounding errors
- ▶ The abort also checks the low bits of $\mathbf{w} - c\mathbf{s}_2$, since those are what the hint would otherwise leak

ML-DSA Parameters

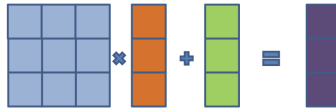
	(k, ℓ)	η	τ	γ_1	pk (B)	sk (B)	σ (B)	category
ML-DSA-44	(4, 4)	2	39	2^{17}	1312	2560	2420	2
ML-DSA-65	(6, 5)	4	49	2^{19}	1952	4032	3309	3
ML-DSA-87	(8, 7)	2	60	2^{19}	2592	4896	4627	5

- ▶ η bounds the secrets s_1, s_2 and γ_1 the mask y ; τ is the number of ± 1 in c
- ▶ Signatures are 2–5 kB, against 64 bytes for Ed25519: the mask must be much larger than the secret, and the whole mask is transmitted
- ▶ Signing is randomized by default (y from a fresh seed)

Visualization of ML-DSA



Public Key / Secret Key
Generation



$$\text{light green square} = H(\text{dark purple column}, \mu)$$



Sizes

Scheme	pk (B)	sk (B)	σ (B)	security
RSA-3072	384	384	384	classical 128
ECDSA-P256	64	32	64	classical 128
ML-DSA-44	1312	2560	2420	PQ category 2
ML-DSA-65	1952	4032	3309	PQ category 3
ML-DSA-87	2592	4896	4627	PQ category 5
FN-DSA-512	897	1281	666	PQ category 1
FN-DSA-1024	1793	2305	1280	PQ category 5
SLH-DSA-128s	32	64	7856	PQ category 1
SLH-DSA-128f	32	64	17088	PQ category 1

Category n means at least as hard as breaking the corresponding symmetric primitive: 1 = AES-128, 2 = SHA3-256 collisions, 3 = AES-192, 5 = AES-256.

Performance

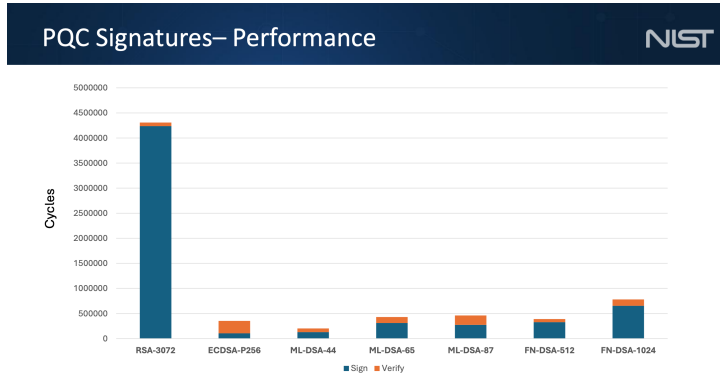


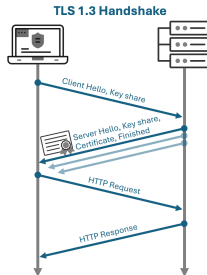
Figure: Cycles to sign and verify, from NIST. ML-DSA signs faster than RSA-3072 and verifies about as fast as ECDSA; FN-DSA verifies fastest but signs slowest.

Signatures in TLS

A bit much to chew?

NIST

- TLS & WebPKI Certificate Signatures
 - *Server Certificate*: 1 public key and signature, 2 SCT signatures
 - *Intermediate CA Certificate*: 1 public key and signature
 - *TLS Handshake*: 1 signature
 - ML-DSA-44 → **14,724 bytes**
 - Current Quantum-Vulnerable → **1,248 bytes**
- ML-KEM-768 key shares
 - Client → Server: 1,184 bytes
 - Server → Client: **1,088 bytes**
- Why does this matter?
 - *TCP initial congestion window* limits the first wave of messages
 - Typical default: **~14,600 bytes**
- Without protocol/implementation changes, this could slow web connection establishment



A Bit Much to Chew

A TLS 1.3 handshake carries five signatures and two public keys: the server certificate with two SCT signatures, the intermediate CA certificate, and the handshake signature itself.

Today that is about 1 248 bytes. With ML-DSA-44 throughout it is about 14 724 bytes, plus the ML-KEM-768 key shares.

The TCP initial congestion window, typically ten segments of about 1 460 bytes, limits what the server may send before the first acknowledgement to roughly 14 600 bytes. A quantum-safe certificate chain no longer fits in the first flight.

This is why the industry is looking at Merkle tree certificates, at leaving SCTs out of the chain, and at FN-DSA for the CA signatures.



Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps



Fast-Fourier Lattice-based
Compact Signatures over NTRU

FN-DSA (FALCON)

- ▶ Based on a different lattice assumption, NTRU, in the ring $\mathbb{Z}_q[X]/(X^n + 1)$ with $n = 512$ or 1024
- ▶ Smaller keys and signatures than ML-DSA, and the fastest verification of all
- ▶ Much more complex key generation and signing algorithms
- ▶ Very difficult to secure against side-channel attacks
- ▶ The draft standard, FIPS 206, was submitted in August 2025; the final version is expected in 2026 or 2027

Hash-and-Sign Instead of Fiat-Shamir

FN-DSA does not mask a secret. The secret key is a *short basis* of an NTRU lattice, and the public key a long basis of the same lattice.

To sign, hash the message to a random point c and use the short basis to find a lattice point close to it. The signature is the short difference $s = c - v$. Anyone can check that s is short and that $c - s$ is in the lattice; only the short basis makes finding such an s feasible.

The catch: the “find a close point” step must be *randomised*, sampling from a discrete Gaussian, or the signatures leak the short basis. That sampler uses floating-point arithmetic, and making it constant time is what held the standard back.

FIPS 205

Federal Information Processing Standards Publication

Stateless Hash-Based Digital Signature Standard

Category: Computer Security

Subcategory: Cryptography

SLH-DSA (SPHINCS+)

- ▶ Only based on the security of hash functions
- ▶ The most conservative design for signatures
- ▶ Small keys but large and slow signatures
- ▶ Great fit for certificates or code signing
- ▶ There exist more efficient *stateful* versions, XMSS and LMS
- ▶ Easy to implement securely in hardware

How a Hash-Based Signature Works

One-time signature (WOTS). The secret key is a set of random strings, the public key their hashes. To sign a message, reveal the preimages selected by the message digits. Each key may be used *once*: a second signature reveals enough preimages to forge.

Merkle tree. Generate 2^h one-time keys and hash their public keys pairwise up to a single root: that root is the public key. A signature is one WOTS signature plus the h sibling hashes on the path to the root.

Stateful vs stateless. XMSS and LMS make the signer remember which leaves were used; a lost counter is a broken key. SLH-DSA picks the leaf *pseudorandomly* from a huge hypertree and uses few-time signatures at the leaves, so an accidental reuse is harmless. That is what costs the 8–17 kB.

Three Designs, Three Trade-offs

	ML-DSA	FN-DSA	SLH-DSA
assumption	MLWE and MSIS	NTRU lattices	hash function only
paradigm	Fiat-Shamir with aborts	hash-and-sign, Gaussian sampling	Merkle trees of one-time keys
signature	2.4–4.6 kB	0.7–1.3 kB	8–50 kB
public key	1.3–2.6 kB	0.9–1.8 kB	32–64 B
signing	fast, integer only	slow, floating point	slow to very slow
implementation risk	moderate	high	low
best fit	general purpose	bandwidth-limited, e.g. CA keys	firmware, code signing, roots of trust

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

Additional Signatures - Round 1

Table 2. First-round digital signature candidates organized by category, with the candidates selected to advance to the second round bolded and in blue. The starred signature schemes MIRA and MiRitH merged to form a new candidate Mirath.

<u>Code-Based</u>	<u>Lattice-Based</u>	<u>MPC-in-the-Head</u>	<u>Multivariate</u>
CROSS	EagleSign	Biscuit	3WISE
Enhanced pqsigRM	EHTv4	MIRA*	DME-Sign
FuLeeca	HAETAE	MiRitH*	HPPC
LESS	HAWK	MQOM	MAYO
MEDS	HuFu	PERK	PROV
WAVE	Raccoon	RYDE	QR-UOV
	SQUIRRELS	SDitH	SNOVA
<u>Other</u>			TUOV
ALTEQ	<u>Symmetric-Based</u>	<u>Isogeny-Based</u>	UOV
eMLE-Sig 2.0	AlMer	SQIsign	VOX
KAZ-SIGN	Ascon-Sign		
PREON	FAEST		
Xifrat1-Sign.I	SPHINCS-alpha		

Figure: <https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8528.pdf>

Additional Signatures - Round 2

NIST accepted 14 candidate algorithms for the second round. Submission teams were allowed to make minor modifications and resubmit their updated packages, which had to meet the same requirements as the original submissions. One of the candidates was a merger of two first-round algorithms: Mirath (merged from MIRA and MiRitH). The complete updated specifications were posted on <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures> on March 4, 2025, for public review. Table 2 lists the second-round digital signature candidates organized by category. The candidates that were selected to advance to the third round are bold and in blue.

Table 2. Second-round digital signature candidates

<u>Code-Based</u>	<u>Lattice-Based</u>	<u>MPC-in-the-Head</u>	<u>Multivariate</u>
CROSS	HAWK	FAEST	UOV
LESS		Mirath (MIRA/MiRitH)	MAYO
		MQOM	QR-UOV
<u>Isogeny-Based</u>		PERK	SNOVA
SQIsign		RYDE	
		SDitH	

Figure: <https://nvlpubs.nist.gov/nistpubs/ir/2026/NIST.IR.8610.pdf>

Why a Second Competition

All three signature standards have a weak spot: ML-DSA and FN-DSA share the lattice assumption, and SLH-DSA is large. NIST opened a second call in 2022 for signatures that are *not* lattice-based, or that are much smaller.

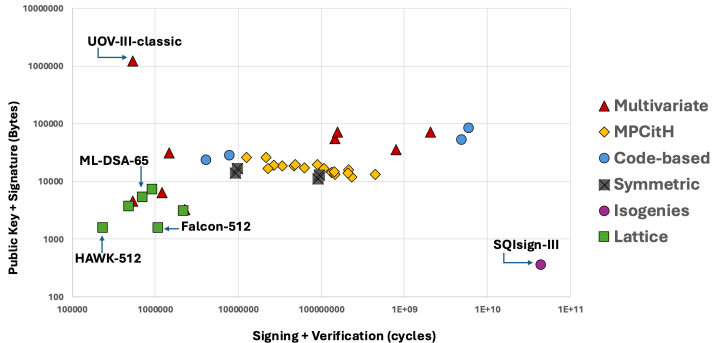
Of 40 accepted submissions, 14 advanced to round 2 in October 2024, and **nine to round 3 in May 2026**: multivariate (UOV, MAYO, QR-UOV, SNOVA), “MPC-in-the-head” from symmetric primitives or codes (FAEST, MQOM, SDitH), isogeny-based (SQIsign), and one lattice scheme, HAWK, **withdrawn in July 2026** after an attack. Eight remain.

The trade-offs are stark: SQIsign signs in tens of milliseconds for signatures of a few hundred bytes; the multivariate schemes pair tiny signatures with public keys of tens of kilobytes. Round 3 runs about two years.

Additional Signatures

Performance Summary (log scale)

NIST



What Round 3 Tells Us

Code-based signatures lost. CROSS, LESS, Mirath, PERK and RYDE were all dropped; SDitH is the one code-based survivor. Codes gave us HQC as a KEM, but for signatures they have not kept up.

Multivariate is back. Four of nine are UOV descendants, a 1999 design written off after Rainbow fell in 2022. Small signatures, public keys tens of KBs.

The lattice outsider fell. HAWK, kept because its lattice structure differed from ML-DSA and FN-DSA, was withdrawn by its designers in July 2026 after an attack exploiting an unused symmetry of that lattice halved its effective security ($2^{150} \rightarrow 2^{108}$ for HAWK-512). SQIsign is the only isogeny scheme left after SIDH fell in 2022: smallest signatures, slowest signing.

Contents

Quantum-Safe Cryptography

New Hardness Assumptions

ML-DSA (CRYSTALS-Dilithium)

FN-DSA (FALCON) and SLH-DSA (SPHINCS+)

Additional Signatures

The Next Steps

The Next Steps

- ▶ The FN-DSA and HQC standards must be completed
- ▶ The additional signature competition must be completed: round 3 ends around 2028
- ▶ ML-KEM is in TLS and SSH today; ML-DSA in certificates is still ahead
- ▶ Extensive research into side channels and implementation, next lecture
- ▶ New research designing other schemes from the same assumptions
- ▶ Update and upgrade existing protocols and systems to PQC

What to Remember

- ▶ Signatures need two problems: LWE to hide the key, SIS to prevent forgery
- ▶ ML-DSA is Schnorr over a lattice, with one addition: *abort* whenever the masked response would leak
- ▶ Without the abort the transcript is not simulatable and the secret leaks a few coordinates per signature
- ▶ FN-DSA trades size for a floating-point sampler; hard to implement safely
- ▶ SLH-DSA trades size for an assumption nobody doubts
- ▶ The deployment problem is bandwidth: the quantum-safe certificate chain does not fit in the first TCP flight

Questions?