



NTNU

Norwegian University of
Science and Technology

QUANTUM-SAFE ENCRYPTION

TTM4205 – Lecture 13

Tjerand Silde

02.10.2026

Contents

Quantum-Safe Cryptography

Lattices and Learning With Errors

ML-KEM (CRYSTALS-Kyber)

Summary

Contents

Quantum-Safe Cryptography

Lattices and Learning With Errors

ML-KEM (CRYSTALS-Kyber)

Summary

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

- ▶ **Lectures 9–12** attacked implementations of RSA, ECC and AES
- ▶ **Today** the algorithms themselves are replaced: a quantum computer breaks RSA and ECC, and we need encryption that survives it
- ▶ **Lecture 14** does the same for digital signature schemes
- ▶ **Lecture 15** attacks the implementations of the new schemes



Cryptography Today

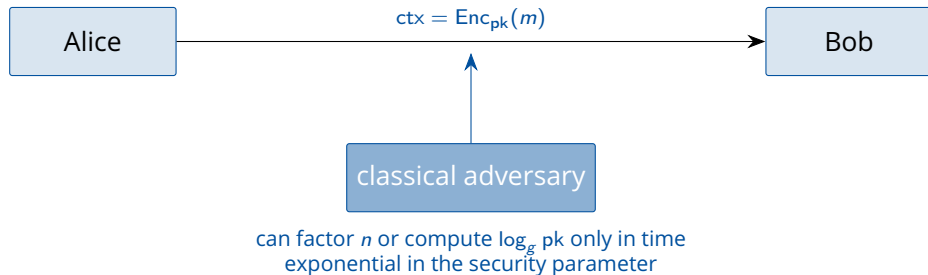


Figure: Alice encrypts to Bob's public key pk and the adversary sees the ciphertext. RSA, DH and DSA are secure against adversaries running on a classical computer.

Cryptography Tomorrow

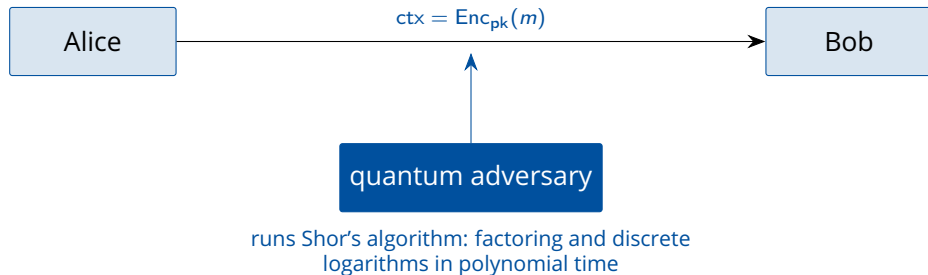


Figure: A quantum computer is outside the classical model of computation. The adversary's toolbox changes; the protocols we run must change with it.

Cryptography Tomorrow

Shor's quantum algorithm (1994) factors integers and computes discrete logarithms in polynomial time, given a large enough quantum computer. This breaks RSA, (EC)DH, (EC)DSA and every other scheme relying on these assumptions. To achieve long-term secrecy, there is an urgent need to replace those algorithms with something that can withstand quantum computers.



What a Quantum Computer Breaks, and What It Does Not

Shor: factoring and discrete logarithms, in any group, in polynomial time. RSA, DH, DSA, ECDH, ECDSA, EdDSA: all gone, at every key size.

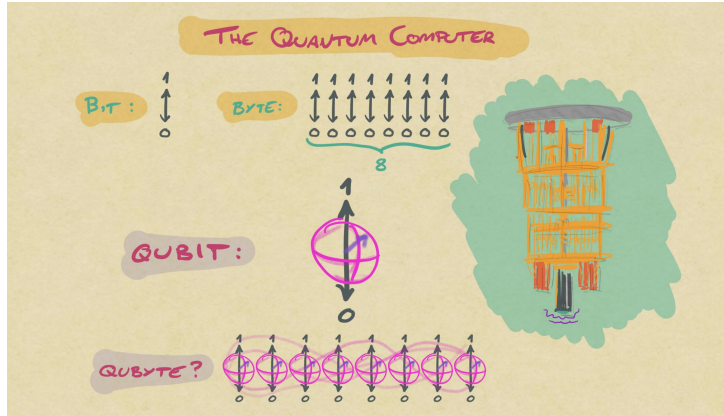
Grover: unstructured search over N items in about $\sqrt{N}$ steps instead of N . A k -bit key costs $2^{k/2}$ instead of 2^k , so AES-128 would drop to 64-bit security.

In practice the 2^{64} steps must run *sequentially* on a fault-tolerant machine, and the speed-up does not parallelize, so NIST and most analyses judge AES-128 and SHA-256 to remain safe. Symmetric crypto therefore keeps its current parameters; only in the worst case moves to AES-256 and longer hash outputs.

The fix for public key crypto is not: we need *new mathematical problems* that neither Shor nor anything else quantum is known to solve.



Quantum Computers



Quantum Computers

- ▶ Quantum computers are not better; they are different
- ▶ They will generally be worse, but do specific things better
- ▶ In theory, they can break public key encryption and digital signatures based on factoring and discrete log assumptions
- ▶ There are many recent developments in quantum computing

Quantum Computers

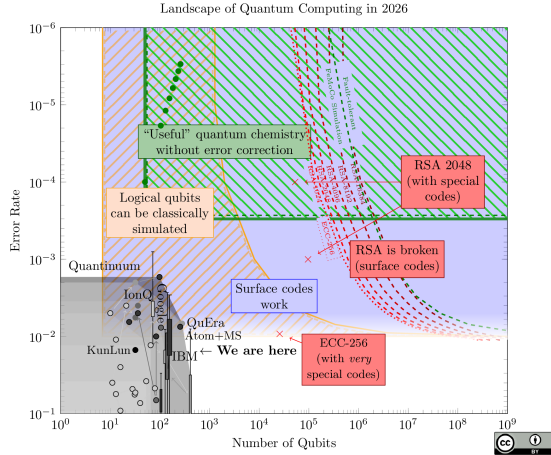


Figure: https://sam-jaques.appspot.com/quantum_landscape

Factoring RSA

How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA

June 9, 2025

Planning the transition to quantum-safe cryptosystems requires understanding the cost of quantum attacks on vulnerable cryptosystems. In Gidney+Ekerå 2019, I co-published an estimate stating that 2048 bit RSA integers could be factored in eight hours by a quantum computer with 20 million noisy qubits. In this paper, I substantially reduce the number of qubits required. I estimate that a 2048 bit RSA integer could be factored in less than a week by a quantum computer with less than a million noisy qubits. I make the same assumptions as in 2019: a square grid of qubits with nearest neighbor connections, a uniform gate error rate of 0.1%, a surface code cycle time of 1 microsecond, and a control system reaction time of 10 microseconds.

Figure: <https://arxiv.org/pdf/2505.15917>



The Best Known Attacks

- ▶ **Gidney and Ekerå (2019):** RSA-2048 in 8 hours with 20 million noisy qubits, the long-standing reference estimate. <https://arxiv.org/abs/1905.09749>
- ▶ **Gidney (2025):** the same with *under one million* noisy qubits in less than a week, through better error correction and arithmetic. <https://arxiv.org/abs/2505.15917>
- ▶ **Chevignard, Fouque, Schrottenloher (2024):** RSA-2048 with about 1 700 logical qubits, trading qubits for gates. <https://eprint.iacr.org/2024/222>
- ▶ **Regev (2023):** a multidimensional factoring algorithm with asymptotically fewer gates than Shor. <https://arxiv.org/abs/2308.06572>
- ▶ **Litinski (2023), Gouzien et al. (2023):** 256-bit elliptic curves fall to far smaller machines than RSA, $\approx 2\,300$ logical qubits. <https://arxiv.org/abs/2306.08585>

How Big Must the Machine Be?

Shor's algorithm needs *logical* qubits that are error-free for the whole computation. Physical qubits are noisy, so each logical qubit is encoded into many physical ones.

In 2019 the estimate for RSA-2048 was about 20 million noisy qubits running for eight hours. Gidney's 2025 estimate cuts this to *under one million* noisy qubits running for less than a week.

The largest machines today have on the order of a thousand physical qubits, and no fault-tolerant logical qubit at scale.

The gap is large, but it has shrunk by a factor of twenty in six years through better *algorithms*, not (only) better hardware. This cannot be ignored.



Harvest Now, Decrypt Later

Traffic encrypted with RSA or ECDH today can be recorded now and decrypted when a quantum computer arrives.

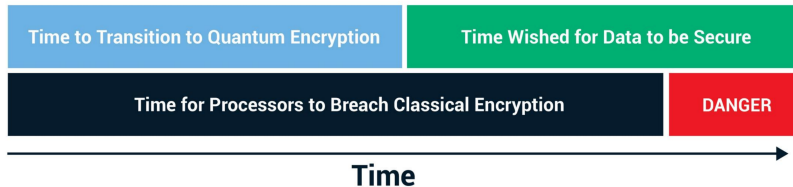
Mosca's inequality. With x the years the data must stay secret, y the years the migration takes, and z the years until a relevant quantum computer exists, the data is already at risk if $x + y > z$.

For state secrets x is decades, and y has been a decade for every global migration so far. Whatever you believe about z , this is why the migration started in 2016.



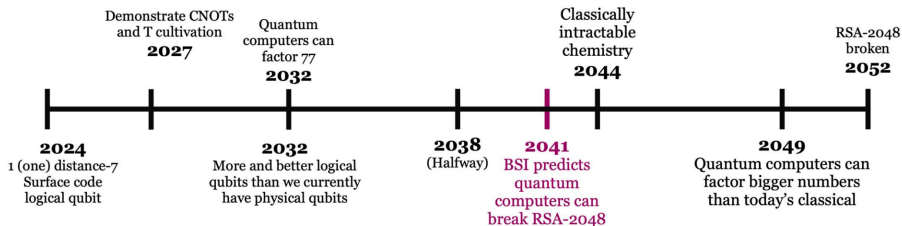
Harvest Now, Decrypt Later

Urgency: Mosca's Inequality

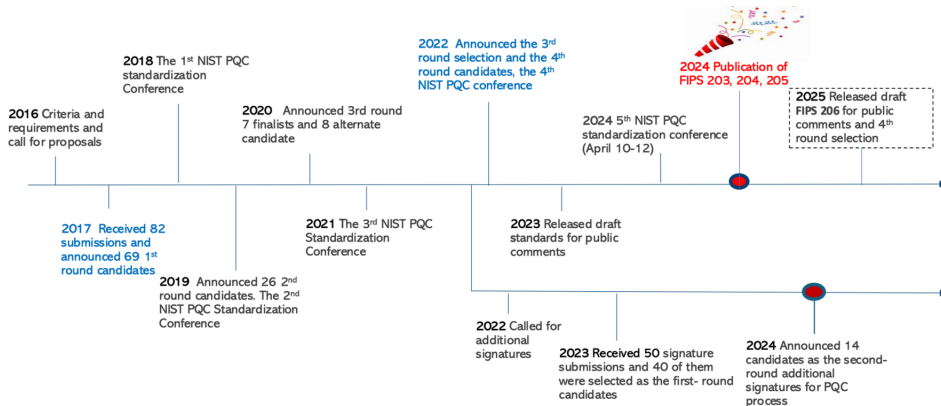


Don't wait - upgrade your encryption now!

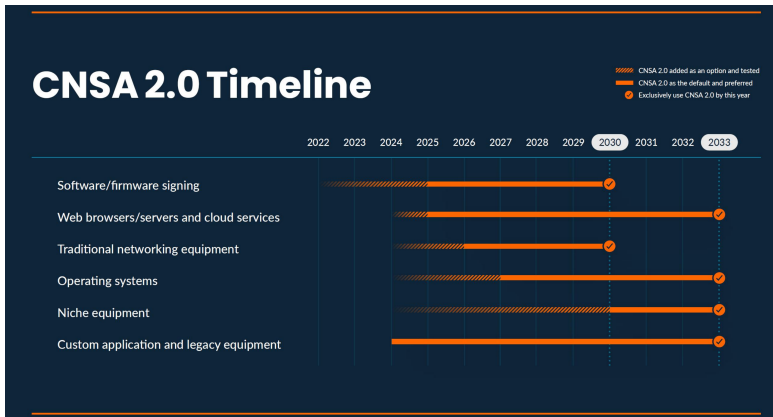
BSI Timeline



NIST Timeline



NSA Timeline



The Deadlines in One Place

- ▶ **NIST**, 2016: competition opens; 2022: Kyber, Dilithium, Falcon, SPHINCS+ selected; **August 2024**: FIPS 203, 204, 205 published
- ▶ **NIST IR 8547**: RSA and ECC at 112-bit security *deprecated* after 2030 and *disallowed* after 2035
- ▶ **NSA CNSA 2.0**: quantum-safe algorithms preferred now, exclusive use for national security systems by 2035, with earlier dates for software signing and web browsers
- ▶ **BSI and NSM**: recommend hybrid deployments today, and warn that the migration itself takes years

None of these dates is a prediction of when a quantum computer exists. They are the dates by which the *migration* has to be finished.

Quantum-resistant schemes

- Establishing a shared secret between two parties:

<i>Scheme</i>	<i>Status</i>
ML-KEM	D

ML-KEM must be used in hybrid mode with a quantum-vulnerable key establishment scheme using an appropriate KEM combiner. The recommended parameter sets are ML-KEM-768 and ML-KEM-1024.

Figure: <https://nsm.no/regelverk-og-hjelp/veiledere-og-handboker/kryptografi/ske-anbefalinger-en-veileder-fra-nsm>

Why Hybrid?

A *hybrid* key exchange runs a classical scheme and a quantum-safe scheme side by side and combines both shared secrets with a key derivation function.

The classical part protects against the new scheme turning out to be broken: the lattice assumptions are twenty years old, not fifty, and the implementations are new.

The quantum-safe part protects against the quantum computer. An adversary has to break *both*.

The cost is bandwidth and one more code path. Every large deployment so far has judged that worth it.



Deployed Today

- ▶ **TLS:** Chrome, Firefox and Cloudflare negotiate X25519MLKEM768 by default; well over half of Cloudflare's human traffic is already quantum-safe encrypted
- ▶ **Signal:** PQXDH (2023) adds ML-KEM to the initial key agreement, and the later SPQR ratchet adds it to the ongoing session
- ▶ **iMessage:** PQ3 (2024) rekeys with ML-KEM continuously during a conversation
- ▶ **SSH:** OpenSSH defaults to a hybrid ML-KEM key exchange, and GitHub has enabled it

All of these are *key exchange*. Quantum-safe *signatures* are much further behind, for reasons we get to next lecture.



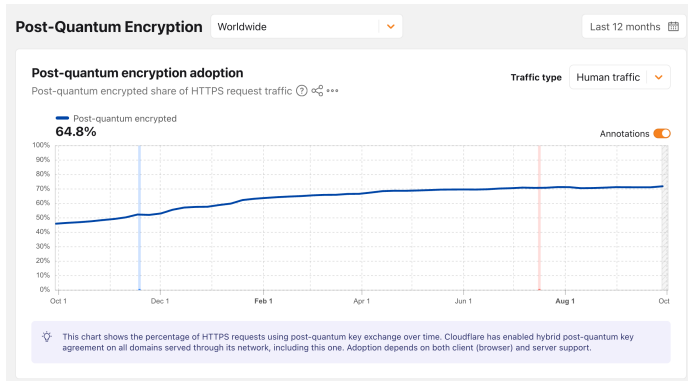


Figure: Share of human HTTPS traffic to Cloudflare using post-quantum key exchange, October 2025 to October 2026. <https://radar.cloudflare.com/post-quantum>



On essentially all domains served (1) through **Cloudflare**, including this one, **we have enabled** hybrid post-quantum key agreement. We are also **rolling out support** for post-quantum key agreement for connection from Cloudflare to origins (3). Check out our blog post **the state of the post-quantum Internet** for more context.

You are using *X25519MLKEM768* which is **post-quantum secure**.

Deployed key agreements

Available with TLSv1.3 including HTTP/3 (QUIC)

Key agreement	TLS identifier
<i>X25519MLKEM768</i>	0x11ec (recommended)
<i>X25519Kyber768Draft00</i>	0x6399 (obsolete), 0xfe31

Figure: <https://pq.cloudflareresearch.com>



Quantum Resistance and the Signal Protocol

ehrenkret on 19 Sep 2023

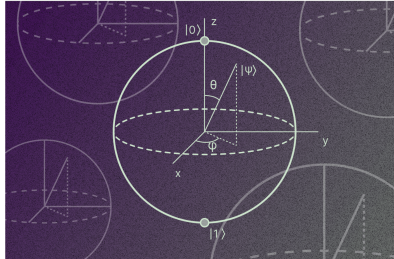


Figure: <https://signal.org/blog/pqxdh>

Quantum-Secure Cryptography in Messaging Apps

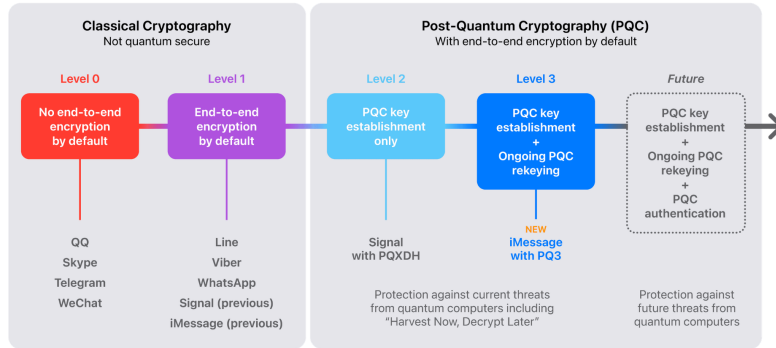


Figure: <https://security.apple.com/blog/imessage-pq3>

What Has to Change

No change	Drop-in replacement	Protocol redesign	Lattices only
Symmetric crypto: AES, SHA-2, SHA-3, HMAC <i>Done: double the key size</i>	Public key encryption, key exchange, signatures <i>Standardised 2024, deploying now</i>	Zero-knowledge proofs, distributed privacy, blockchain privacy <i>Recent designs near optimal, implementation starting</i>	Fully homomorphic encryption, some obfuscation <i>Deployment starting</i>

Adapted from Lyubashevsky. The second column is this lecture and the next.

Contents

Quantum-Safe Cryptography

Lattices and Learning With Errors

ML-KEM (CRYSTALS-Kyber)

Summary

Recall: Decisional 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, c uniformly at random from $\mathbb{Z}_p$. The Decisional Diffie–Hellman problem is to distinguish the two cases:

1. (g, g^a, g^b, g^{ab})
2. (g, g^a, g^b, g^c)

This is what ElGamal and ECDH rest on, and it is exactly what Shor breaks.

A Lattice

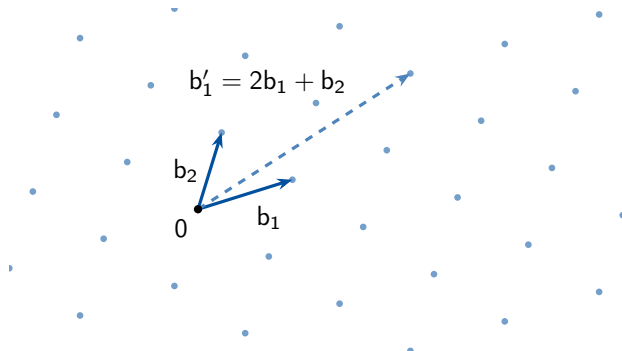


Figure: A lattice is the set of all integer combinations of a basis $b_1, \dots, b_n$. Many bases generate the same lattice; a *short* basis makes problems easy, a long skewed one makes them hard.

Two Problems on Lattices

Shortest Vector Problem (SVP). Given a basis, find the shortest non-zero lattice vector.

Closest Vector Problem (CVP). Given a basis and a target point t , find the lattice vector closest to t .

Both are easy in dimension 2 and believed exponentially hard in dimension n , for classical *and* quantum computers: the best algorithms, lattice reduction such as LLL and BKZ, cost $2^{\Theta(n)}$.

What we actually use is a problem that *looks like linear algebra* but hides a CVP instance inside.

Learning With Errors (LWE)

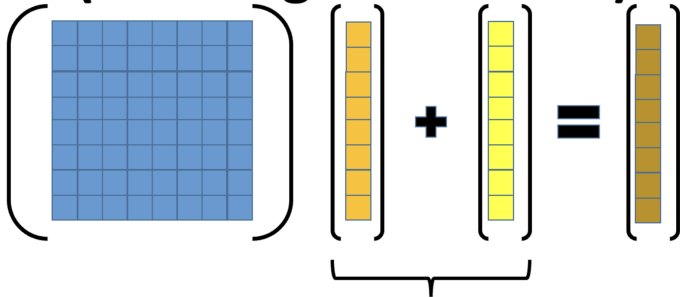
Definition (Regev, 2005)

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

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 $u \leftarrow \$ \mathbb{Z}_q^m$

The *search* version asks to recover s from $(A, As + e)$. Without e this is Gaussian elimination; with it, the best attacks are lattice reduction.

(Learning With Errors)



Small coefficients to enforce uniqueness

Why the Noise Matters

Each row of $b = As + e$ is a linear equation in s that is *almost* right: correct up to a small error.

With $m \geq n$ exact equations, Gaussian elimination returns s in $O(n^3)$ steps. One small error per equation, and elimination amplifies the errors until the result is uniform garbage.

Geometrically, As ranges over a lattice, and b is a point close to it. Finding s is finding the closest lattice point: a CVP instance in dimension n .



LWE Hardness

The Learning With Errors problem gets harder when...

- ▶ the dimension n gets larger
- ▶ the secret and noise values get larger (larger β)
- ▶ the modulus q gets smaller

What governs the hardness is essentially n and the ratio q/β : the noise has to be large relative to the modulus, or the lattice point is too easy to find. Correctness, as we will see, wants the opposite.

From LWE to Module-LWE

Plain LWE keys are large: the public key is an $m \times n$ matrix over $\mathbb{Z}_q$, hundreds of kilobytes for $n \approx 700$.

ML-KEM and ML-DSA replace $\mathbb{Z}_q$ by the ring $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$: one ring element packs 256 integers, and A becomes a $k \times k$ matrix of ring elements with $k \in \{2, 3, 4\}$. This is *Module-LWE*.

Multiplication in R_q uses the Number Theoretic Transform, the modular FFT, in $O(n \log n)$; $q = 3329$ is chosen so that it exists. The assumption has the same shape as LWE, and the structure buys a factor of 256 in size and speed.

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

Lattices and Learning With Errors

ML-KEM (CRYSTALS-Kyber)

Summary

FIPS 203

Federal Information Processing Standards Publication

Module-Lattice-Based Key-Encapsulation Mechanism Standard

Category: Computer Security

Subcategory: Cryptography

Figure: <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf>

Defining a KEM

Key Encapsulation Mechanism

A KEM is a triple of algorithms with public key space $\mathcal{PK}$, secret key space $\mathcal{SK}$, ciphertext space $\mathcal{C}$ and shared key space $\mathcal{K}$:

$\text{KGen}() \leftarrow \$ (\text{pk}, \text{sk})$ randomized; outputs a key pair.

$\text{Encaps}(\text{pk}) \leftarrow \$ (K, \text{ctx})$ randomized; outputs a shared key $K \in \mathcal{K}$ and a ciphertext $\text{ctx} \in \mathcal{C}$ that encapsulates it.

$\text{Decaps}(\text{sk}, \text{ctx}) \rightarrow K \text{ or } \perp$ deterministic; recovers K from ctx , or rejects.

Unlike public key encryption, the sender does not choose the message: the KEM chooses a random K . That is all a key exchange needs, and it is simpler to make secure.

Recall: ElGamal

Let $\mathbb{G}_p$ be a group of prime order p with generator g , public parameters $\text{pp} = (\mathbb{G}_p, g, p)$, secret key $\text{sk} \leftarrow \$ \mathbb{Z}_p$ and public key $\text{pk} = g^{\text{sk}}$.

The ElGamal encryption scheme, with message $m \in \mathbb{G}_p$, works as follows:

Enc: Sample $x \leftarrow \$ \mathbb{Z}_p$ and output $X = g^x$ and $Y = \text{pk}^x \cdot m$.

Dec: From (X, Y) recover $m = Y \cdot X^{-\text{sk}}$.

Keep the shape in mind: fresh randomness x , a mask pk^x only the key holder can recompute, and the message hidden under it.

LWE Encryption: Key Generation and Encryption

Let q be a prime, n the dimension, β a small noise bound, and $\langle \cdot, \cdot \rangle$ the inner product.

KGen: Sample $A \leftarrow \mathbb{Z}_q^{n \times n}$ and short $s, e \leftarrow [\beta]^n$. Compute $b = As + e \bmod q$ and output $pk = (A, b)$, $sk = s$.

Enc: To encrypt $m \in \{0, 1\}$, sample short $r, e_1 \leftarrow [\beta]^n$ and $e_2 \leftarrow [\beta]$, and output

$$ctx = (u, v) = \left(A^T r + e_1, \langle b, r \rangle + e_2 + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \right) \bmod q.$$

LWE Encryption: Decryption

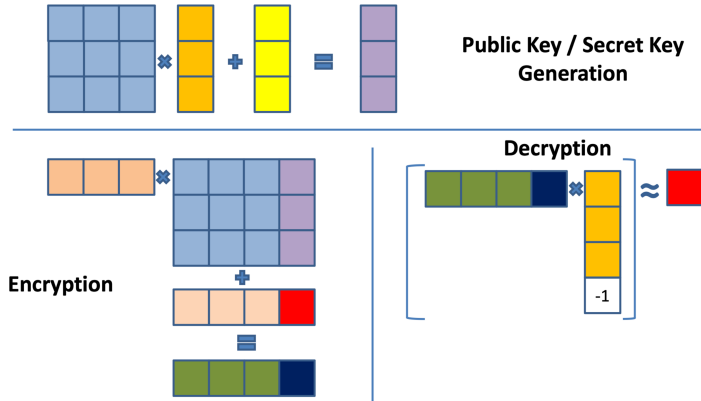
Dec: Compute $w = v - \langle u, s \rangle \bmod q$, taken in $(-q/2, q/2]$. Output $m = 0$ if $|w| < q/4$ and $m = 1$ otherwise.

Why it works: substituting $b = As + e$ and $u = A^T r + e_1$, every term with A cancels, because $\langle As, r \rangle = \langle s, A^T r \rangle$:

$$w = \underbrace{\langle e, r \rangle + e_2 - \langle e_1, s \rangle}_{\varepsilon} + m \left\lfloor \frac{q}{2} \right\rfloor.$$

Since $|\varepsilon| \leq 2n\beta^2 + \beta$, decryption is correct whenever $2n\beta^2 + \beta < q/4$. That is the design tension: security wants β large relative to q , correctness wants it small.

The Same Scheme, in a Picture



ML-KEM Parameters

	k	η_1	η_2	pk (B)	ctx (B)	failure prob.
ML-KEM-512	2	3	2	800	768	2^{-139}
ML-KEM-768	3	2	2	1184	1088	2^{-164}
ML-KEM-1024	4	2	2	1568	1568	2^{-174}

- ▶ Always $q = 3329$ and $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$; the security level is set by k alone
- ▶ η_1, η_2 are the noise bounds for keys and for encryption; β in our toy scheme
- ▶ Ciphertexts are *compressed* by dropping low bits, which adds a rounding term to the noise
- ▶ The worst-case correctness bound is replaced by a failure probability so small no adversary ever sees a failure

From Encryption to a KEM

The scheme above is only IND-CPA: given a decryption oracle, an adversary flips bits of ctx and learns about s from whether decryption still succeeds.

ML-KEM applies the **Fujisaki-Okamoto transform**: encrypt a random seed m and derive all encryption randomness (r, e_1, e_2) from $H(m)$, so encryption is deterministic given m .

Decapsulation decrypts to m' , *re-encrypts* it, and compares with the received ciphertext; any tampering fails the comparison.

On failure ML-KEM returns not $\perp$ but a pseudorandom key derived from a secret and the ciphertext (*implicit rejection*): no failure oracle, the padding-oracle lesson built in.

Classical vs Quantum-Safe Key Exchange

Algorithm	PQ	key share (bytes)		operations / s	
		client	server	client	server
ML-KEM-512	yes	800	768	50 000	100 000
ML-KEM-768	yes	1184	1088	31 000	70 000
X25519	no	32	32	17 000	17 000

ML-KEM is *faster* than elliptic curve Diffie–Hellman on the same hardware. What it costs is bandwidth: about $35\times$ the bytes on the wire.

X-Wing: Hybrid Key Exchange

X-Wing private key (2464 bytes):

ML-KEM-768 private key (2400 bytes)	X25519 private key (32 bytes)	X25519 public key (32 bytes)
--	----------------------------------	---------------------------------

X-Wing public key (1216 bytes):

ML-KEM-768 public key (1184 bytes)	X25519 public key (32 bytes)
---------------------------------------	---------------------------------

X-Wing ciphertext (1120 bytes):

ML-KEM-768 ciphertext (1088 bytes)	X25519 ciphertext (32 bytes)
---------------------------------------	---------------------------------

X-Wing shared key (32 bytes):

SHA3-256	$\left(\begin{array}{ c c c c c } \hline \backslash ./ \\ ./ \backslash \\ \hline \end{array} \right)$ (6 bytes)	ML-KEM-768 shared key (32 bytes)	X25519 shared key (32 bytes)	X25519 ciphertext (32 bytes)	X25519 public key (32 bytes)
----------	--	--	------------------------------------	------------------------------------	------------------------------------

Figure: <https://cic.iacr.org/p/1/1/21/pdf>

X-Wing: Hybrid Key Exchange

X-Wing combines **X25519** and **ML-KEM-768** into a single KEM. Both run independently, and the shared key is

$$K = \text{SHA3-256}(\text{label} \parallel K_{\text{ML-KEM}} \parallel K_{\text{X25519}} \parallel \text{ctx}_{\text{X25519}} \parallel \text{pk}_{\text{X25519}}).$$

The classical transcript goes into the hash so that the result is a secure KEM even if ML-KEM were completely broken; the ML-KEM ciphertext does not need to, because ML-KEM already binds its key to its ciphertext.

This is the design behind the TLS group your browser uses today, and it is a good example of a combiner that is exactly as complicated as it has to be.

NIST Selects HQC as Fifth Algorithm for Post-Quantum Encryption

March 11, 2025

- NIST has chosen a new algorithm for post-quantum encryption called HQC, which will serve as a backup for ML-KEM, the main algorithm for general encryption.
- HQC is based on different math than ML-KEM, which could be important if a weakness were discovered in ML-KEM.
- NIST plans to issue a draft standard incorporating the HQC algorithm in about a year, with a finalized standard expected in 2027.



MEDIA CONTACT

301-975-2762

ORGANIZATIONS

Information Technology Laboratory
Computer Security Division
Cryptographic Technology Group

RELATED NEWS

NIST Releases First 3 Finalized Post-Quantum Encryption Standards

Figure: <https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption>

HQC: A Backup From a Different Family

Every NIST KEM standard so far rests on lattices. If a structural weakness in module lattices were found, all of them would fall together.

HQC (Hamming Quasi-Cyclic) is a *code-based* KEM: the hard problem is decoding a random linear code, and is unrelated to lattices.

NIST selected it in March 2025 as a backup to ML-KEM. Its keys and ciphertexts are a few kilobytes, several times larger than ML-KEM's, so it will not replace ML-KEM, only insure it. A draft standard is expected in 2026.

Contents

Quantum-Safe Cryptography

Lattices and Learning With Errors

ML-KEM (CRYSTALS-Kyber)

Summary

What to Remember

- ▶ Shor breaks every factoring and discrete-log scheme; Grover only halves symmetric key length
- ▶ The deadline is secrecy lifetime plus migration time, not the arrival of the machine
- ▶ LWE is linear algebra with small noise: without it Gaussian elimination, with it a lattice problem
- ▶ The scheme is ElGamal's shape over a lattice; correctness needs $2n\beta^2 + \beta < q/4$, security needs β large relative to q
- ▶ ML-KEM adds ring structure for size and the FO transform for CCA security
- ▶ Deploy it hybrid, and deploy it now

Questions?