NTNU | Norwegian University of Science and Technology

QUANTUM-SAFE CRYPTOGRAPHY

TTM4135 - Lecture 12

Tjerand Silde

16.02.2026

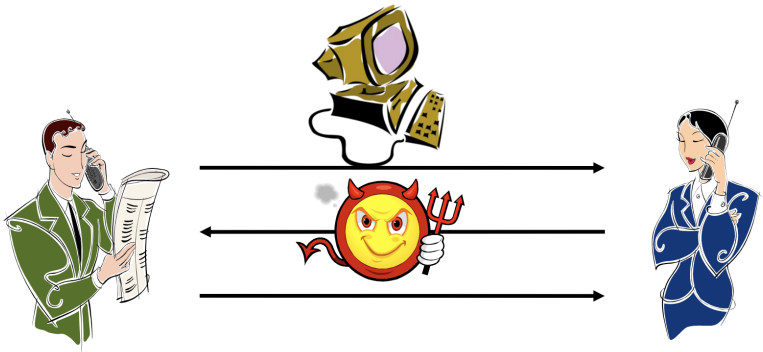
Motivation

- So far, we have looked at public key cryptographic primitives based on the hardness of factoring and discrete logarithm assumptions
- These assumptions can (in theory) be broken by Shor's algorithm on quantum computers, and quantum computing has seen recent advances
- Symmetric key encryption and hash functions seem to be fine; in the worst case, we need to double keys and outputs due to Grover's algorithm
- We need to standardize new Key Encapsulation Mechanisms (KEM) and Digital Signatures (DS) from quantum-safe assumptions, e.g., lattices
- There are also other KEM and DS standards that are not covered in this lecture, and there is an ongoing additional DS competition as we speak

Contents

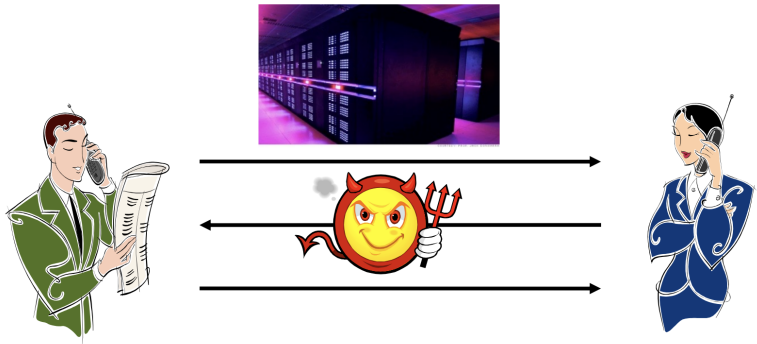
Cryptography Today

Allows for secure communication in the presence of malicious parties



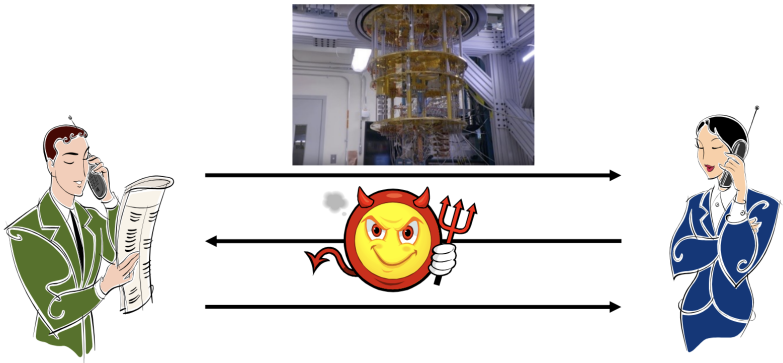
Cryptography Today

Large increase in the adversary's computing power
requires only a small increase in the key size



Cryptography Tomorrow

A quantum computer is outside the classical model of computation for efficiency purposes

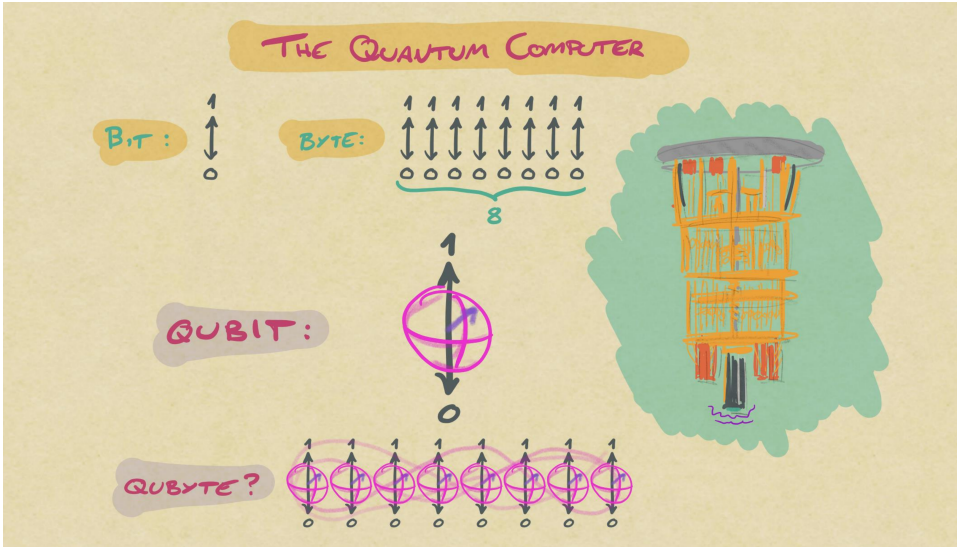


Cryptography Tomorrow

Shor's quantum algorithm can factorize integers and compute discrete logs essentially as fast as using them, given a large quantum computer. This will break RSA, (EC)DH, (EC)DSA schemes and others relying these assumptions. To achieve future secrecy, there is an urgent need to replace those algorithms.

Grover's quantum algorithm can improve exhaustive search by a square root factor, potentially speeding up key recovery and finding hash collisions.

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

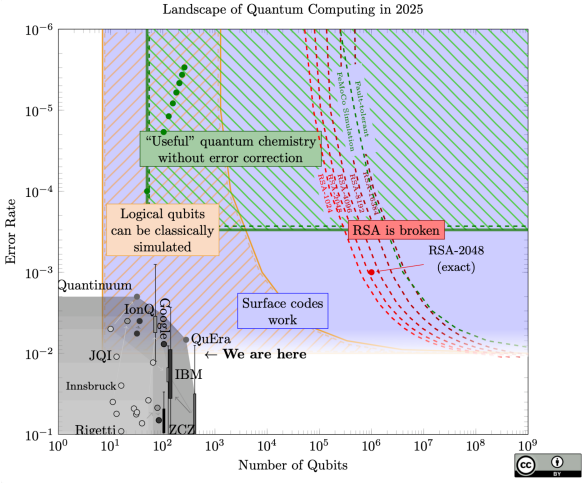


Figure: https://sam-jaques.appspot.com/quantum_landscape_2025

Factoring RSA

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

Craig Gidney

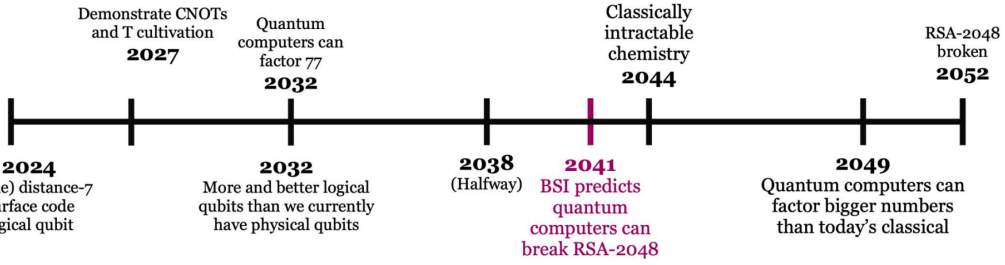
Google Quantum AI, Santa Barbara, California 93117, USA

June 9, 2025

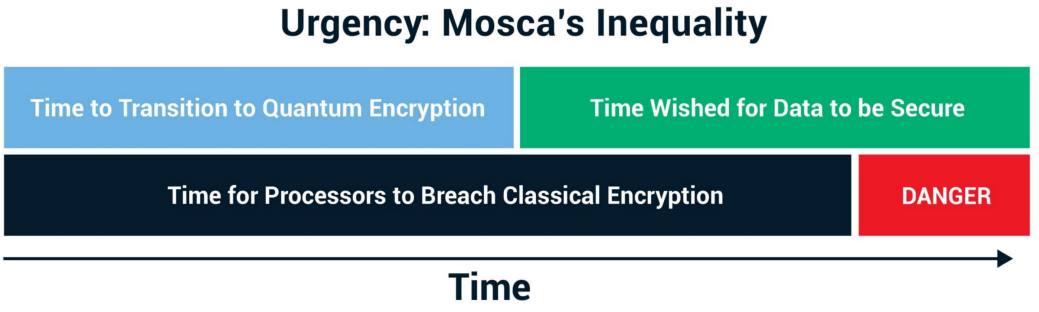
Planning the transition to quantum-safe cryptosystems requires understanding the cost of quantum attacks on vulnerable cryptosystems. In Gidney+Ekerå 2019, I co-published an estimate stating that 2048 bit RSA integers could be factored in eight hours by a quantum computer with 20 million noisy qubits. 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

BSI Timeline

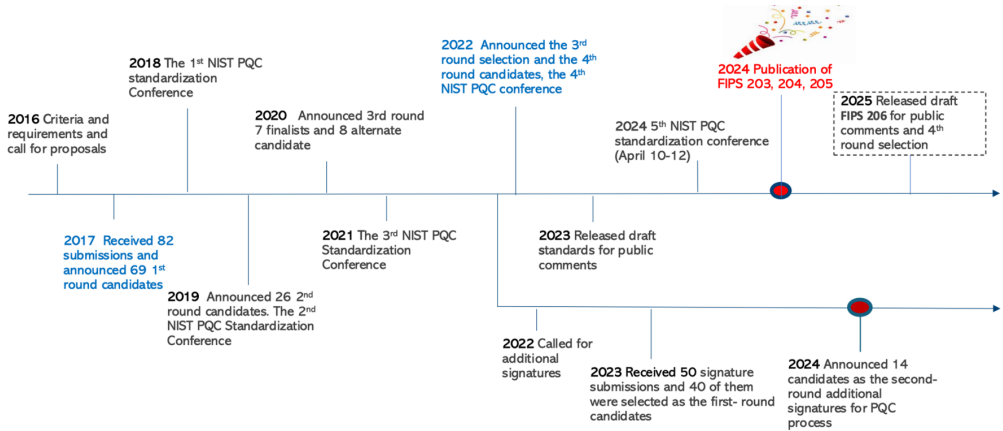


Harvest now, decrypt later

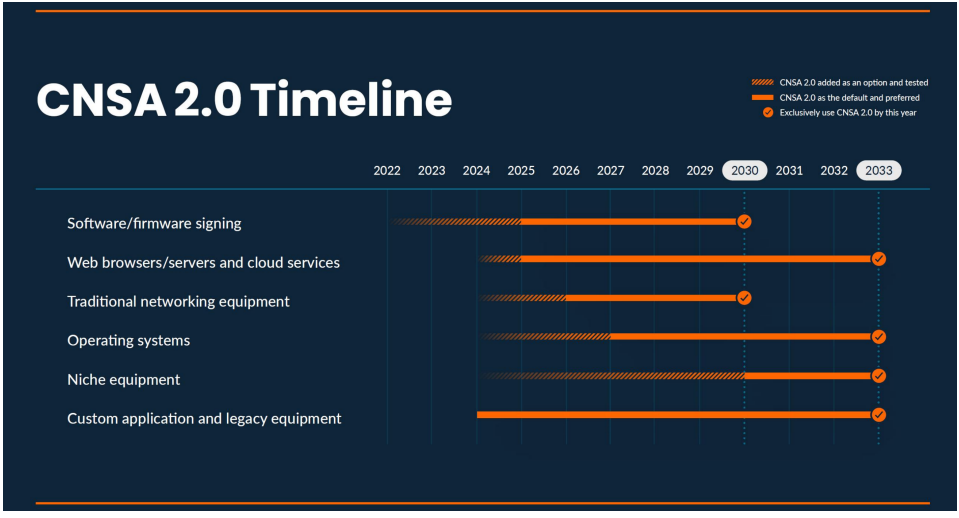


Don't wait - upgrade your encryption now!

NIST Timeline



NSA Timeline



Hybrid

Quantum-resistant schemes

- Establishing a shared secret between two parties:

Scheme	*Status*
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/kryptografiske-anbefalinger-en-veileder-fra-nsm

Cloudflare

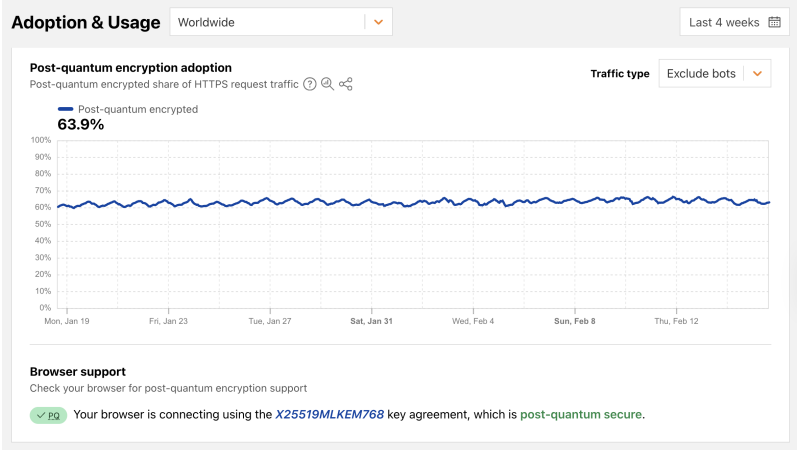


Figure: https://radar.cloudflare.com/adoption-and-usage#post-quantum-encryption-adoption

Signal

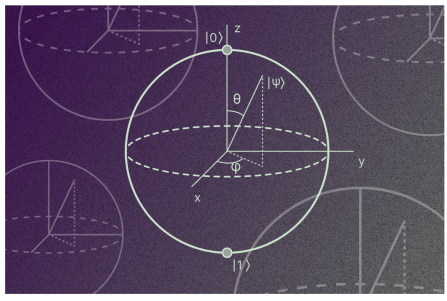


Figure: https://signal.org/blog/pqxdh

iMessage

Quantum-Secure Cryptography in Messaging Apps

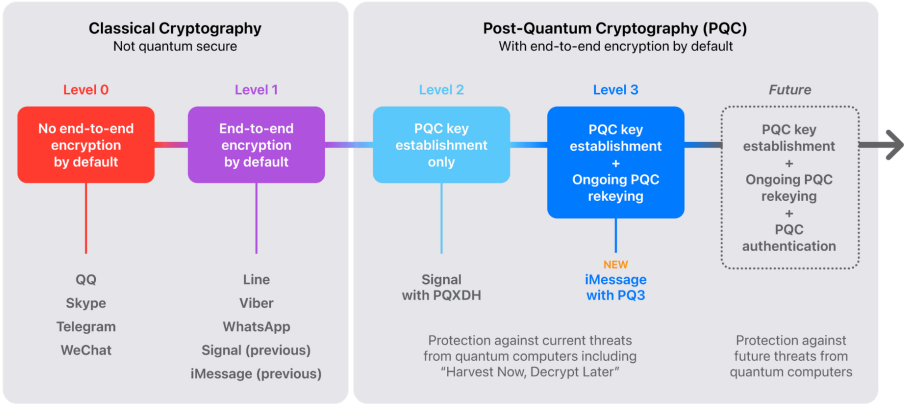


Figure: https://security.apple.com/blog/imessage-pq3

Crypto Categories

No Changes Necessary

Symmetric Cryptography:
- AES
- SHA-256 / SHA-3
- HMAC
- etc.

Done.

Almost Drop-in Replacements

NIST standardizations:
- Public Key Encryption
- Key Exchange
- Digital Signatures

A few other things:
- Identity-Based Encryption

Almost standards. Ready for deployment.

Serious Alterations of Protocols Required

Advanced Primitives:
- Zero-Knowledge Proofs
- Distributed Privacy
- Many blockchain privacy applications

Lots of recent progress on design. Near-optimality has just been achieved for certain primitives. Implementation starting at ZRL.

Can Only Be Done with Lattice Cryptography

- Fully-Homomorphic Encryption (FHE) - computation over encrypted data
- Some Obfuscation (still unclear if it can be efficient or have any useful applications)

Implementation / deployment of FHE at Haifa.

Contents

Recall: Decisional Diffie-Hellman

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

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

$$(g, g^a, g^b, g^{ab})$$

$$(g, g^a, g^b, g^c)$$

New: Learning With Errors (LWE)

Definition 1. For positive integers m, n, q, and $\beta < q$, the $\mathsf{LWE}_{n,m,q,\beta}$ problem asks to distinguish between the following two distributions:

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

LWE Hardness

The Learning With Errors problem gets harder when...

- the dimension gets larger
- the secret values gets larger
- the modulus gets smaller

Recall: Computational Diffie-Hellman

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

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 $\mathbb{G}$.

New: Short Integer Solution (SIS)

Definition 4. For positive integers m, n, q, and $\beta < q$, the $\mathsf{SIS}_{n,m,q,\beta}$ problem asks to find, for a randomly-chosen matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, vectors $\mathbf{s}_1 \in [\beta]^m$ and $\mathbf{s}_2 \in [\beta]^n$ such that $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{0} \pmod{q}$.

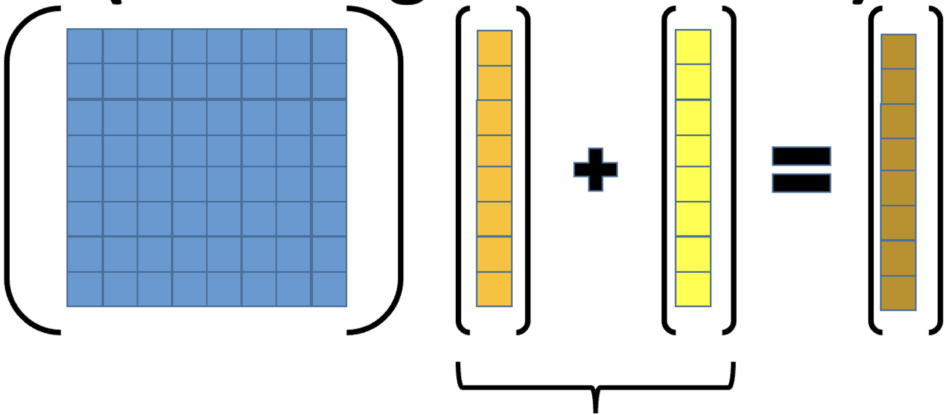
SIS Hardness

The Short Integer Solution problem gets harder when...

- the dimension gets larger
- the secret values gets smaller
- the modulus gets larger

This is opposite of LWE with respect to norms!

(Learning With Errors)



Hardness of LWE and SIS

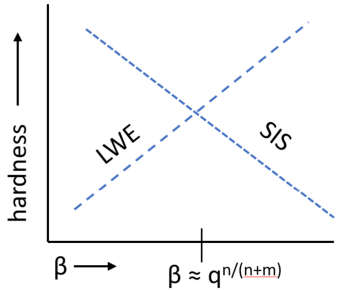


Figure 2: The hardness of $\mathsf{LWE}_{n,m,q,\beta}$ and $\mathsf{SIS}_{n,m,q,\beta}$ for fixed n, m, q, and varying β. The lines are not meant to describe the concrete hardness of these problems, but rather to illustrate the dependence of the hardness of these problems on β. The intersection point is approximately at $\beta = q^{n/(n+m)}$.

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

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

Definition 1 (Key Encapsulation Mechanism (KEM))**.** A key encapsulation mechanism is a triple of algorithms KEM $= \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ with public keyspace $\mathcal{PK}$, private keyspace $\mathcal{SK}$, ciphertext space $\mathcal{C}$, and shared keyspace $\mathcal{K}$. The triple of algorithms is defined as:

- $\mathrm{KEM.}\mathsf{KeyGen}() \xrightarrow{\$} (sk, pk)$ Randomized algorithm that outputs a secret (private) key $sk \in \mathcal{SK}$, and a public key $pk \in \mathcal{PK}$.
- $\mathrm{KEM.}\mathsf{Enc}(pk) \xrightarrow{\$} (k, c)$ Randomized algorithm that, given a public key $pk \in \mathcal{PK}$, outputs a shared key $k \in \mathcal{K}$, and a ciphertext $c \in \mathcal{C}$.
- $\mathrm{KEM.}\mathsf{Dec}(c, sk) \rightarrow y \in \{k, \perp\}$ Deterministic algorithm that, given a secret key, $sk \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$, returns the shared key $k \in \mathcal{K}$. In case of rejection, this algorithm returns $\perp$.

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

Recall: ElGamal

Let $\mathbb{G}$ be a group of prime order p and g be a generator for $\mathbb{G}$. Denote by pp the public parameters $(\mathbb{G}, g, p)$.

Let the secret key $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$ be sampled uniformly at random, and let the public key be $\mathsf{pk} = g^{\mathsf{sk}}$, where pk is made public.

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

Enc : Sample uniform $x \leftarrow\!\!\$\ \mathbb{Z}_p$ and encrypt as $X = g^x$ and $Y = \mathsf{pk}^x \cdot m$.

Dec : Decrypt the ciphertext (X, Y) to get the message $m = Y \cdot X^{-\mathsf{sk}}$.

ML-KEM KGen and Enc

$$\mathsf{sk}: \mathbf{s} \leftarrow [\beta]^m, \ \mathsf{pk}: (\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times m}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}_1), \text{ where } \mathbf{e}_1 \leftarrow [\beta]^m. \tag{6}$$

To encrypt a message $\mu \in \{0,1\}$, the encryptor chooses $\mathbf{r}, \mathbf{e}_2 \leftarrow [\beta]^m$ and $e_3 \leftarrow [\beta]$, and outputs

$$\left(\mathbf{u}^T = \mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T, v = \mathbf{r}^T\mathbf{t} + e_3 + \left\lceil \frac{q}{2} \right\rfloor \mu\right). \tag{7}$$

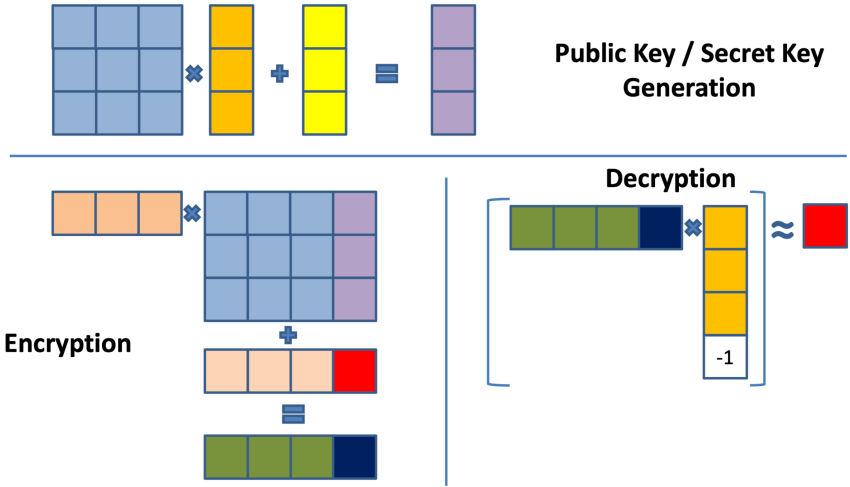
ML-KEM Dec

To decrypt, one computes $v - \mathbf{u}^{\tilde{T}}\mathbf{s}$. But rather than this cleanly giving us the message μ as in (4), we instead obtain

$$v - \mathbf{u}^T\mathbf{s} = \mathbf{r}^T(\mathbf{A}\mathbf{s} + \mathbf{e}_1) + e_3 + \frac{q}{2}\mu - \left(\mathbf{r}^T\mathbf{A} + \mathbf{e}_2^T\right)\mathbf{s} \tag{8}$$

$$= \mathbf{r}^T\mathbf{e}_1 + e_3 + \frac{q}{2}\mu - \mathbf{e}_2^T\mathbf{s} \tag{9}$$

Visualization of ML-KEM

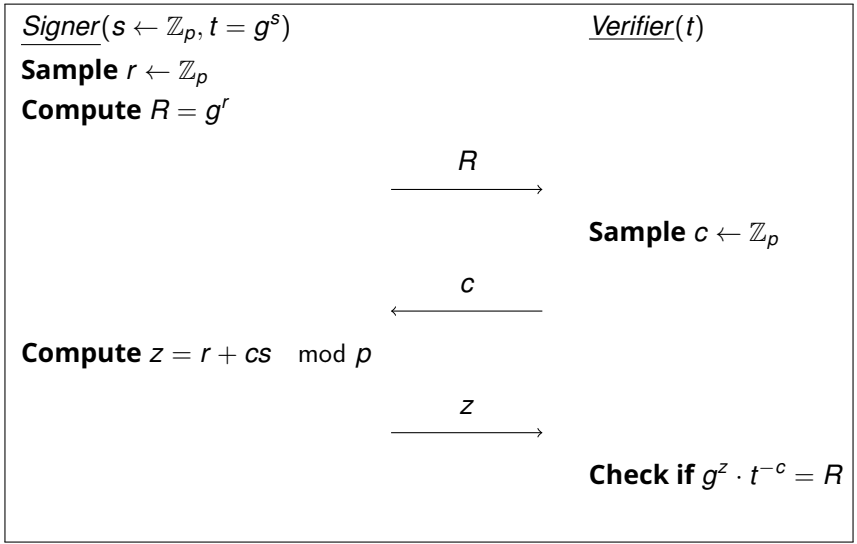


Classical vs Quantum-Safe Key-Exchange

		Size keyshares(in bytes)		Ops/sec (higher is better)	
Algorithm	PQ	**Client**	**Server**	**Client**	**Server**
Kyber512	✅	800	768	50,000	100,000
Kyber768	✅	1,184	1,088	31,000	70,000
X25519	❌	32	32	17,000	17,000

Contents

Recall: Schnorr Signatures (interactive)

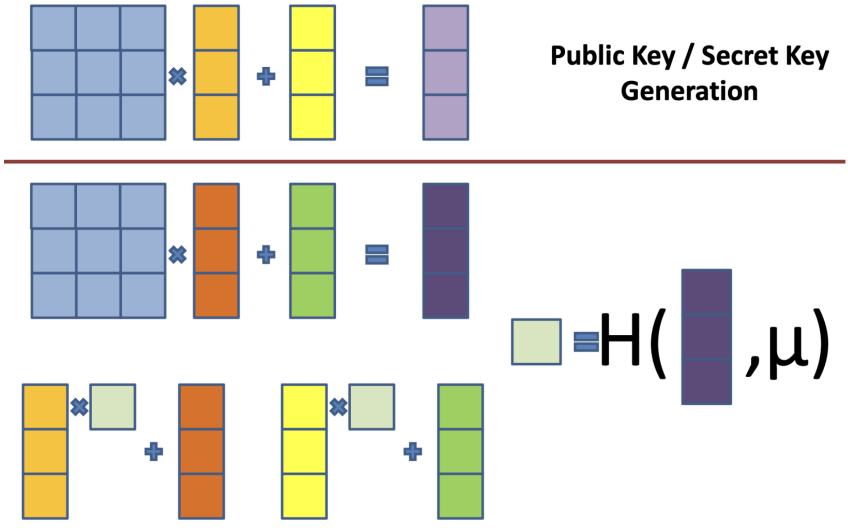


ML-DSA (interactive)

Private information: $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_{q,f}^n$

Prover		Verifier
$\mathbf{y}_1 \leftarrow [\gamma + \bar{\beta}]^m$ $\mathbf{y}_2 \leftarrow [\gamma + \bar{\beta}]^n$, $\mathbf{w} := \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$		
	$\xrightarrow{\mathbf{w}}$	
		$c \leftarrow \mathcal{C}$
	$\xleftarrow{c}$	
$\mathbf{z}_1 := c\mathbf{s}_1 + \mathbf{y}_1$ $\mathbf{z}_2 := c\mathbf{s}_2 + \mathbf{y}_2$ if $\mathbf{z}_1 \notin [\bar{\beta}]^m$ or $\mathbf{z}_2 \notin [\bar{\beta}]^n$ then $(\mathbf{z}_1, \mathbf{z}_2) := \perp$		
	$\xrightarrow{(\mathbf{z}_1, \mathbf{z}_2)}$	
		Accept iff $\mathbf{z}_1 \in [\bar{\beta}]^m$ and $\mathbf{z}_2 \in [\bar{\beta}]^n$ and $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{w}$

Visualization of ML-DSA



Schnorr vs ML-DSA

- Schnorr is based on DLOG, ML-DSA on LWE and SIS
- Schnorr has uniform challenges, ML-DSA has small
- ML-DSA aborts if the norm of z is too big
- ML-DSA require norm checks of the signature

Sizes

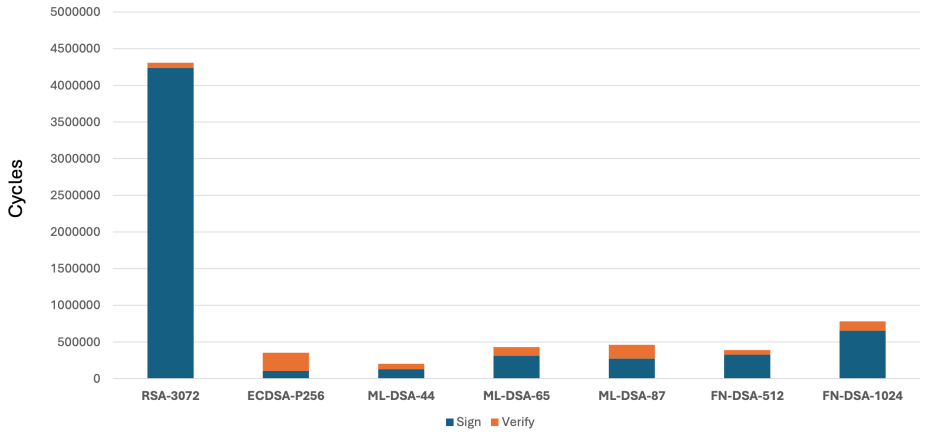
PQC Key and Signature Sizes

NIST

Scheme	Public Key (bytes)	Private Key (bytes)	Signature (bytes)	Security Level
RSA-3072	**384**	**384**	**384**	**Classical-128**
ECDSA-P256	**64**	**32**	**256**	**Classical-128**
ML-DSA-44 (Dilithium2)	**1312**	**2528**	**2420**	**PQC Category 2** *(SHA3-256)*
ML-DSA-65 (Dilithium3)	**1952**	**4000**	**3293**	**PQC Category 3** *(AES-192)*
ML-DSA-87 (Dilithium5)	**2592**	**4864**	**4595**	**PQC Category 5** *(AES-256)*
FN-DSA-512 (Falcon512)	**897**	**7553**	**666**	**PQC Category 1** *(AES-128)*
FN-DSA-1024 (Falcon1024)	**1793**	**13953**	**1280**	**PQC Category 5** *(AES-256)*

Performance

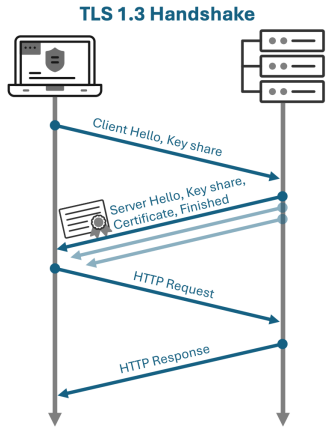
PQC Signatures– Performance



Signatures in TLS

A bit much to chew?

- 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



Questions?