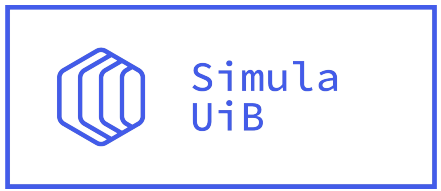
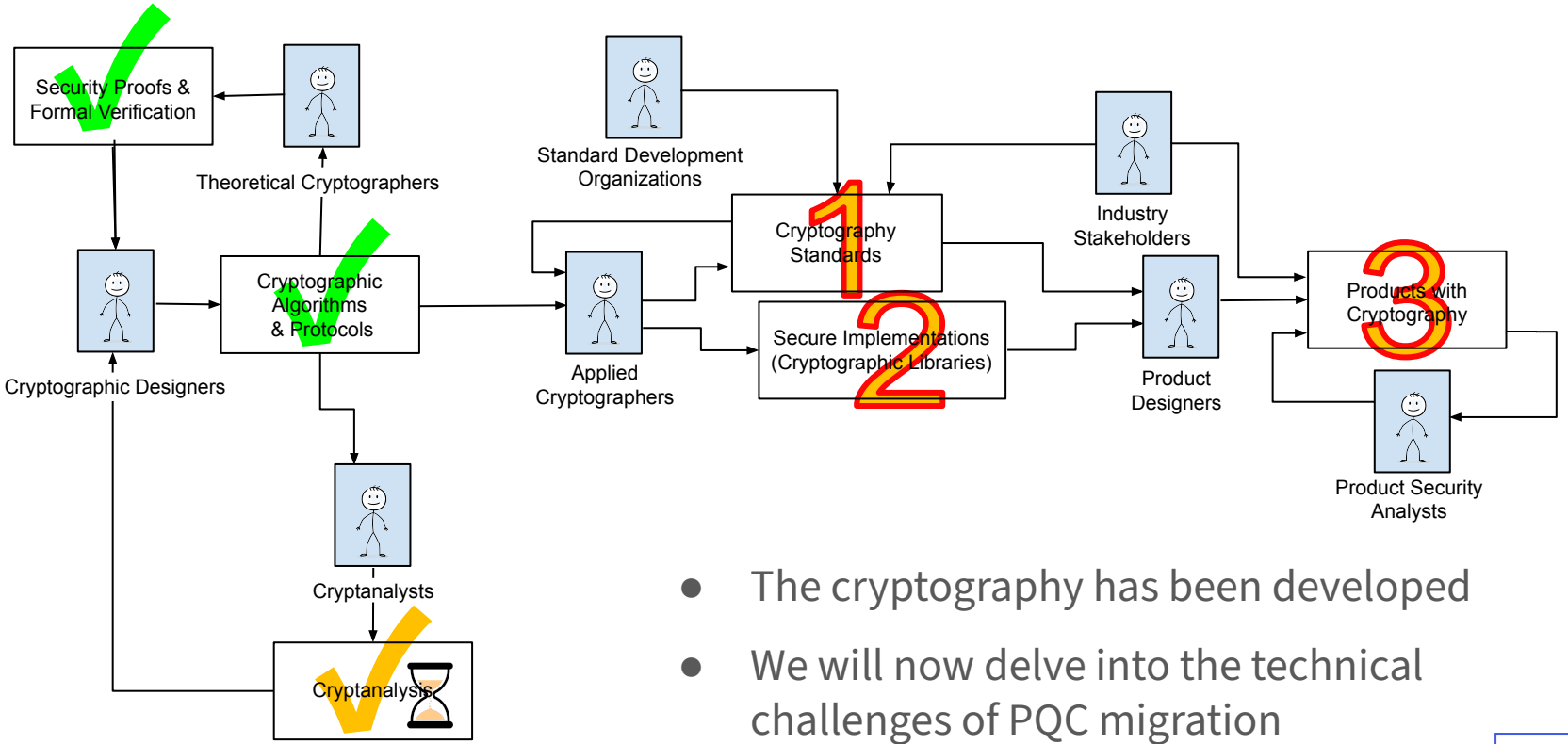
Part IV

Technical Deep Dive



PQC Migration from Theory to Practice



- The cryptography has been developed
- We will now delve into the technical challenges of PQC migration

Simula UiB

PQC Standards

Security levels of PQC algorithms

PQC algorithms come in variants designed to meet three security levels:

Level 1: breaking scheme is similar to breaking AES with **128-bit** key

Level 3: breaking scheme is similar to breaking AES with **192-bit** key

Level 5: breaking scheme is similar to breaking AES with **256-bit** key

Kyber = ML-KEM = CRYSTALS-Kyber

NIST standard FIPS 203

Lattice-based security,
Structured lattice

Three variants
ML-KEM-512, security level 1
ML-KEM-768, security level 3
ML-KEM-1024, security level 5

	ML-KEM-512	ML-KEM-768	ML-KEM-1024
Private key size	1632 bytes	2400 bytes	3168 bytes
Public key size	800 bytes	1132 bytes	1568 bytes
Ciphertext size	768 bytes	1088 bytes	1568 bytes

Recommended by NSM

FrodoKEM

ISO considering standardization
Recommended by BSI (Germany) and ANSSI (France)

Lattice-based security,
Unstructured lattice

Three variants
FrodoKEM-640, security level 1
FrodoKEM-976, security level 3
FrodoKEM-1344, security level 5

	FrodoKEM-640	FrodoKEM-976	FrodoKEM-1344
Private key size	19888 bytes	31296 bytes	43088 bytes
Public key size	9616 bytes	15632 bytes	21520 bytes
Ciphertext size	9752 bytes	15792 bytes	21696 bytes

Dilithium = CRYSTALS-Dilithium = ML-DSA

NIST standard FIPS 204

Lattice-based security,
Structured lattice

Three variants
ML-DSA44, security level 1
ML-DSA65, security level 3
ML-DSA87, security level 5

	ML-DSA44	ML-DSA65	ML-DSA87
Private key size	2560 bytes	4032 bytes	4896 bytes
Public key size	1312 bytes	1952 bytes	2592 bytes
Signature size	2420 bytes	3309 bytes	4627 bytes

Recommended by NSM

SPHINCS+ = SLH-DSA

NIST standard FIPS 205

Security based on hash functions

Three variants
SLH-DSA-128, security level 1
SLH-DSA-192, security level 3
SLH-DSA-256, security level 5

	SLH-DSA-128	SLH-DSA-192	SLH-DSA-256
Private key size	64 bytes	96 bytes	128 bytes
Public key size	32 bytes	48 bytes	64 bytes
Signature size (**s**mall signature)	7856 bytes	16224 bytes	29792 bytes
Signature size (**f**ast signing)	17088 bytes	35664 bytes	49856 bytes

FALCON = FN-DSA

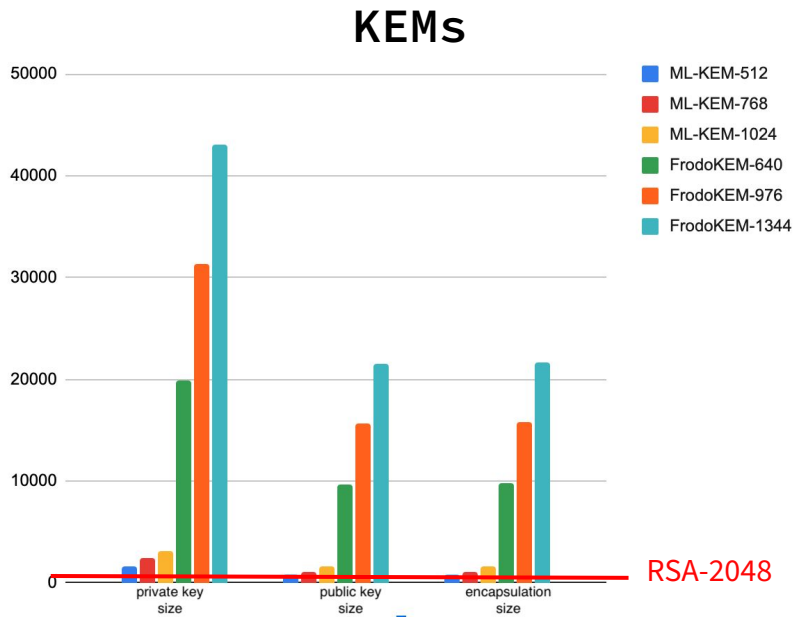
Soon: NIST standard
FIPS 206

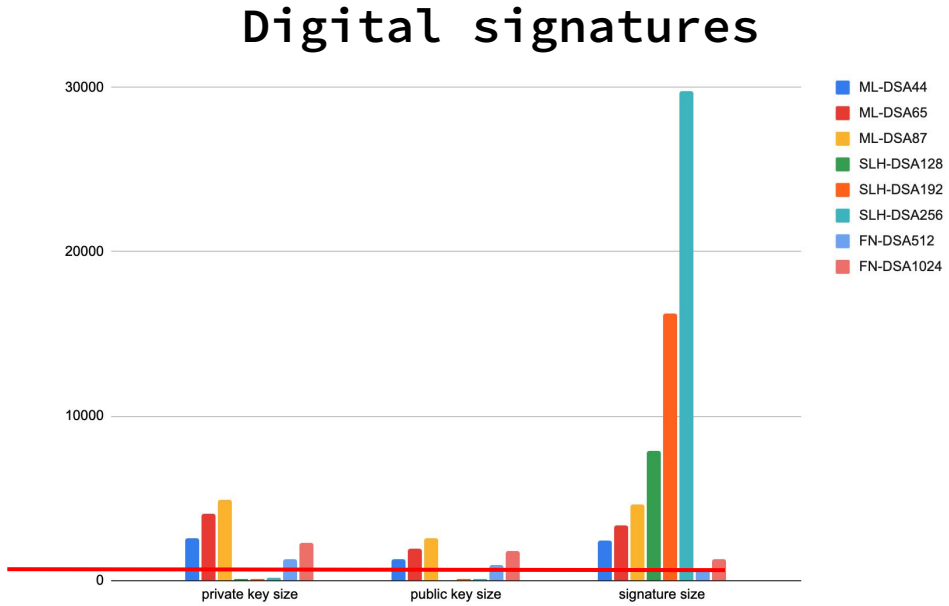
Lattice-based security,
structured lattices

Two variants
FN-DSA-512, security level 1
FN-DSA-1024, security level 5

	FN-DSA-512	FN-DSA-1024
Private key size	1281 bytes	2305 bytes
Public key size	897 bytes	1793 bytes
Signature size	666 bytes	1280 bytes

Comparisons of memory requirements





Simula UiB

Execution times for PQC

KEMs

	Encapsulation	Decapsulation
ML-KEM	Roughly the same (up to 2x slower)	Significantly faster
FrodoKEM	Faster in hardware, much slower in software	Faster in hardware, much slower in software

Digital signatures

	Signing	Verification
ML-DSA	Roughly the same	Much faster
SLH-DSA	Much slower	Much slower
FN-DSA	5-10x slower	Much faster

Ballpark execution times, compared to algorithms commonly used today
Execution times depend on many factors: hardware, compiler, RAM and cache, etc.

Source: https://postquantum.com/post-quantum/cryptography-pqc-nist/

Stateful vs stateless hash-based signatures

SLH-DSA: SLH = stateless hash

Hash-based digital signature schemes can be **stateless** or **stateful**

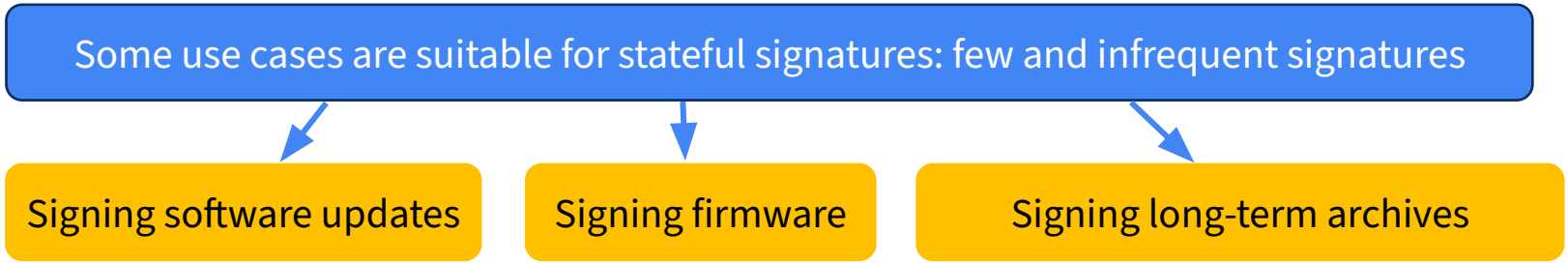
Stateless

- No need to keep track of generated signatures
- Easier to implement and use
- Larger signatures
- Slow execution

Stateful

- Must securely keep track of generated signatures
- Harder to use
- Smaller signatures
- Fast execution

Stateful signatures for special purposes



In such cases NSM recommends **XMSS** and **LMS**

Standardization recap

Several new PQC standards have been published already (by NIST)

- **Stateful signatures**: XMSS, LMS
- **Stateless signatures**: ML-DSA, SLH-DSA
- **Key encapsulation mechanism**: ML-KEM

and more are expected in the coming years

- **NIST:** FN-DSA (Falcon) and HQC-KEM (HQC)
- Further alternatives by **ISO** (Frodo, Classic McEliece) and various **Asian** countries

Fragmentation of PQC standards makes migration more complicated

Standardization percolation

The standardization effort does not stop at KEMs and Signatures

1. Protocol standards using KEMs (or PKE) and signatures (or PKI) need to allow the new PQC standards (and phase out RSA/ECC)
2. Sometimes the new and the old will have to live side-by-side (Hybrid)
3. Advanced cryptographic functionalities, for instance to enable eIDAS 2.0 digital wallets, relies on more than just KEMs and signatures
   (here the core cryptography is not PQC ready yet)

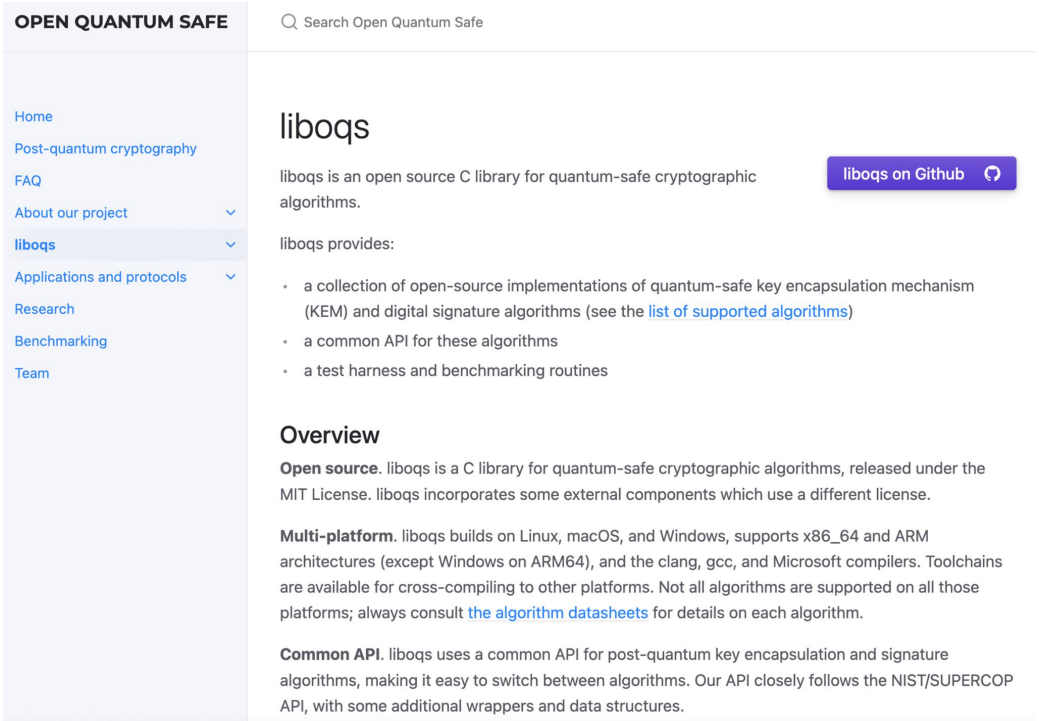
Implementation Issues

Cryptographic libraries implementing PQC

A number of popular cryptographic libraries already support the new PQC standards:

- liboqs
- OpenSSL (version 3.5.0 and later)
- Bouncy Castle
- wolfSSL
- Botan

Cryptographic libraries implementing PQC



see https://openquantumsafe.org/liboqs/

Update to modern systems

Dependencies on older external resources/implementations can cause issues

Transport Layer Security v1.3 **already** supports PQC
(not all TLS v1.3 implementations and configurations currently do though)

Transport Layer Security v1.2 will **not** support PQC
(and TLS v1.2 is still being used out there)

Similar story for Java, Oracle's JDK 24 (released 18-03-2025) first to support PQC

Morale: already invest in updating your environments where possible

How easy is migration?

General misconception: **replacing** classical cryptographic functionality in existing or new implementations is straightforward **by simply "dropping in"** post-quantum algorithms?

Real world deployment challenges posed by:

- parameter sizes (keys, ciphertext, signature sizes in bytes)
- performance in software and hardware
- implementation profile
- side-channel protection profile

Negatively affecting their use in protocols, due to unacceptable latency, bandwidth requirements, etc.

Sizes: PQC vs. Current RSA/ECC

(sizes in bytes)	ML-KEM-512	ML-KEM-768	ML-KEM-1024
Private key	1632	2400	3168
Public key	800	1132	1568
Ciphertext	768	1088	1568

PQC

(sizes in bytes)	ML-DSA44	ML-DSA65	ML-DSA87
Private key	2560	4032	4896
Public key	1312	1952	2592
Signature	2420	3309	4627

KEM: RSA-3072

- sk, pk, ctx: 384 bytes

Classical alternatives

Signature: ECDSA P-256

- sk: 32 bytes
- pk: 64 bytes
- sig: 64 bytes

PQC Misfits: PQC is considerably bigger than classical

Case study 1: ML-DSA for the Web PKI

Web PKI (Web public key infrastructure):

The system of certificates, certification authorities, root stores, and root programs working together to secure connections between web browsers and web sites.

Key enabler for *Transport Layer Security* (TLS 1.3)

https://daniel.fm/blog/posts/pqc-signatures-2024/

Simula UiB

Case study 1: ML-DSA for the Web PKI

David Adrian (Google) notes, when discussing the new NIST PQC standards

> Unfortunately, there has not been enough discussion about how what NIST has standardized is simply not good enough to deploy on the public web in most cases. We need better algorithms. Specifically, we need algorithms that use fewer bytes on the wire—a KEM that when embedded in a TLS ClientHello is still under one MTU, a signature that performs on par with ECDSA that is no larger than RSA-2048, and a sub-100 byte signature where we can optionally handle a larger public key.

One MTU is a Maximum Transmission Unit, ~1400 bytes
Go over, and your packets

- get fragmented over two underlying transport layer (TCP or UDP) packets
- causing latency increases

ML-KEM's Client Hello for TLS already causes a 4% latency increase (Cloudflare) even when using classical signatures!

Case study 1: ML-DSA for the Web PKI

Using PQC signatures is currently a complete no-go

The current breakdown of key and signature sizes in TLS is roughly:

- Root certificates often contain RSA keys, as do intermediate certificates. Root certificates are predistributed, and intermediates are provided by the server, alongside the leaft certificate. An RSA intermediate certificate has a 4096-bit (512 byte) signature, and a 2048-bit (256 byte) public key.
- An ECDSA leaf certificate has a 32-byte key and a 256-byte RSA signature from the intermediate.
- The handshake contains a 64-byte ECDSA signature.
- Each SCT contains a 64-byte ECDSA signature.

In total, this is 512 + 256 + 256 + 32 + 64 + 2*64 = 1,248 bytes of signatures and public keys in a normal TLS handshake for HTTPS. Of the winning signature algorithms from the first NIST PQC competition, ML-DSA (Dilithium) is the only signature algorithm that could be used in the context of TLS[2] and it has 1,312-byte public keys and 2,420-byte signatures. This means *a single ML-DSA public key is bigger than all of the 5 signatures and 2 public keys currently transmitted during an HTTPS connection*. In a direct "copy-and-replace" of current signature algorithms with ML-DSA, a TLS handshake would contain 5*2420 + 2*1312 = 14,724 bytes of signatures and public keys, an over 10x increase.

Case study 1: ML-DSA for the Web PKI

Adrian closes with…

> The best thing we could do to make the post-quantum transition more feasible is to come up with better algorithms that have performance characteristics no worse than RSA-2048. Specifically:
>
> 1. A post-quantum KEM that fits in a single MTU when combined with the rest of the TLS ClientHello
> 2. A 10,000x signing speed improvement and 100x verification speed improvement in SQISign (or a new, equivalent algorithm with these characteristics)
>
> To some extent, this may be yelling for the impossible. Unfortunately, using ML-DSA for the Web PKI in its current form is also impossible.

General consensus for TLS:
Prioritize migration for KEMs (confidentiality) over web PKI for authenticity

Case study 2: Big Tech Perspective

PQC Migration is a Joint Effort

Modern digital infrastructures are **complex with** many service **providers**

For main Internet protocols and services **migration options** will be available

Your **responsibility depends** on model: Infrastructure/Platform/ Software as a Service

Source: https://cloud.google.com/architecture/framework/security/shared-responsibility-shared-fate

Case study 2 (Big Tech): Amazon

Workstream 1: Inventory of existing systems, identification and development of new standards, testing, and migration planning. While the first set of algorithm standards has been published, there are additional standards to come that will define how PQC should be integrated in specific applications and protocols to ensure interoperability.

Workstream 2: Integration of PQC algorithms on public AWS endpoints to provide long-lived confidentiality of customer data transmitted to AWS.

Workstream 3: Integration of PQC signing algorithms into AWS cryptographic services to enable customers to deploy new post-quantum long-lived roots of trust to be used for functions such as software, firmware, and document signing.

Workstream 4: Integration of PQC signing algorithms into AWS services to enable the use of post-quantum signatures for session-based authentication such as server and client certificate validation.

Source: https://aws.amazon.com/blogs/security/aws-post-quantum-cryptography-migration-plan

Case study 2 (Big Tech): Google

Open Source Tink Cryptographic Library has C++ PQC implementations, low-level implementations forthcoming.

For Chrome, they implemented ML-KEM in Google’s cryptography library, BoringSSL

Seemingly awaiting better PQC signature standards

Researchers contribute to awareness, standardization, implementations, etc. See CISO perspectives

Source: Google's PQC Resources

Case study 2 (Big Tech): Microsoft

- Making PQC capabilities available for Windows Insiders, Canary Channel Build 27852 and higher, and Linux, SymCrypt-OpenSSL version 1.9.0.
- Collaborating with the IETF to develop and standardize quantum-safe authentication mechanisms
- Working on is helping support PQC algorithms within Microsoft Active Directory Certificate Services (ADCS)

Source:
https://techcommunity.microsoft.com/blog/microsoft-security-blog/post-quantum-cryptography-comes-to-windows-insiders-and-linux/4413803

Case study 3: ML-DSA for EMV

EMV: the standard for for chip-based card payments, commonly used in chip-and-PIN transactions, relies on cryptographic mechanisms for security of transactions.

During transactions, the card's chip will interact with the terminal and generate:

- an **online cryptogram**, used in most payment scenarios **for remote authentication** and verified by the card issuer. Based on either 3DES or AES.
- an **offline cryptogram**, used f**or local authentication** and verified by the terminal; here, EMV supports the use of **ECC and RSA**.

Case study 3: ML-DSA for EMV

For **offline** transactions, the payment card uses its private key to

- authenticate itself and the transaction data to the terminal
- by signing a challenge created by terminal

For contactless payment, **latency is crucial**!

- EMV contactless “latency budgets” is only a few hundred milliseconds (<300ms); otherwise the transaction will feel “laggy” and/or rejected

PQC Migration Challenge: Keep the latency low

Case study 3: ML-DSA for EMV

Latency in an EMV contactless transaction is primarily due to

- computation on the card
- data transmission between card and terminal
  - ECC signing (64 bytes signature): fast, latency <200ms
  - RSA-1984 (248 bytes)[1]: slower operations, latency may be just acceptable
  - ML-DSA-44 (2420 bytes): **inacceptable**

(Latency of terminal and backend processing is manageable)

EMV's risk assessment means PQC transition here is not deemed urgent

1: largest RSA size in the EMV standard
see https://www.emvco.com/knowledge-hub/quantum-computing-and-emv-chip-whats-the-threat/

Case study 4: PQC on NXP Microelectronics

Implemented PQC on a Cortex-M4 @ 200 MHz software-only (relevant for embedded systems)

Used pqm4 open source project for benchmarks

Compared against their own NXP PQC implementations

Also considered physical attacks

Case study 4: PQC on NXP Microelectronics

pqm4 benchmarks

Algorithm	PQC	Encaps	Decaps	SK	PK	CT
EC-P384	No	"Fast"	"Fast"	48 B	48 B	96 B
FIPS 203 (ML-KEM)	Yes	4 ms	4 ms	2 400 B	1 184 B	1 088 B

Algorithm	PQC	Sign	Verify	SK	PK	Sig
ECDSA-P384	No	"Fast"	"Fast"	48 B	48 B	96 B
FIPS 204 (ML-DSA)	Yes	31 ms	12 ms	4 032 B	1 952 B	3 309 B
FIPS 205 (SLH-DSA)***	Yes	77 s	68 ms	96 B	48 B	16 224 B
SP 800-20 (LMS/XMSS)	Yes	**(Stateful) 19 s	13 ms	48 B	48 B	1 860 B

Reproduced from: https://www.joppebos.com/presentations/Summerschool_PROACT_PQC.pdf

Case study 4: PQC on NXP Microelectronics

pqm4 benchmarks

		Runtime	RAM
Dilithium-2	Sign	19 ms	50 kB
	Verify	7 ms	11 kB
Dilithium-3	Sign	31 ms	69 kB
	Verify	12 ms	10 kB
Dilithium-5	Sign	42 ms	123 kB
	Verify	21 ms	12 kB

Won't fit on embedded system

NXP alternatives

Runtime	RAM
61 ms	5 kB
16 ms	3 kB
119 ms	7 kB
29 ms	3 kB
168 ms	8 kB
50 ms	3 kB

10x smaller
3~4x slower

Reproduced from: https://www.joppebos.com/presentations/Summerschool_PROACT_PQC.pdf

Case study 4: PQC on NXP Microelectronics

Setting

Embedded systems threat model includes

- Side-channel attacks
- Fault-injection attacks
- Invasive attacks

Standard implementations are almost always vulnerable, requiring countermeasures

Well understood for standard symmetric cryptography, RSA, and ECC

PQC Challenge

New PQC cryptosystems behave fundamentally different

- FO-transform for ML-KEM
- Gaussian sampling for ML-KEM/ML-DSA
- Floating points for Falcon
  (NIST standard pending)

Side-channel, fault-injection, invasive attacks and their countermeasures still **active research area**

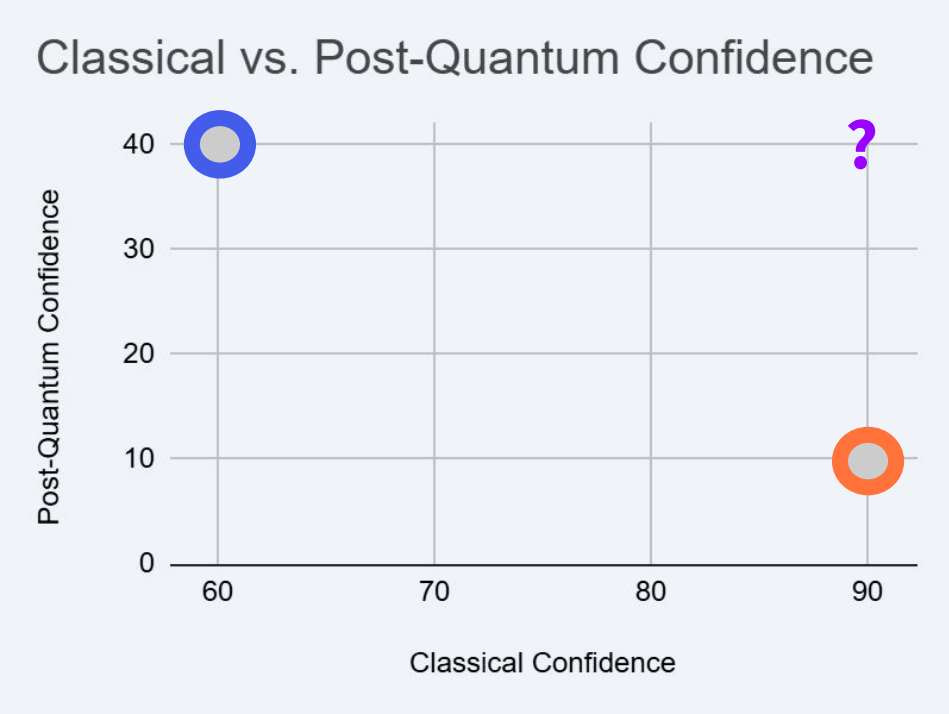
Agility needed for future-proofing

Reproduced from: https://www.joppebos.com/presentations/Summerschool_PROACT_PQC.pdf

Hybrid PQC

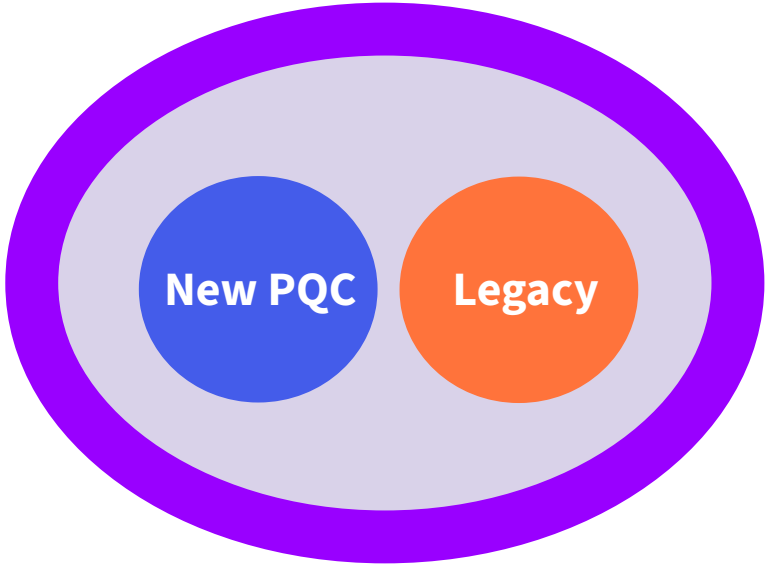
The PQC Conundrum

- Legacy PKC systems
  - 🚫 Not PQ secure nor compliant
  - ➕ High confidence against classical attacks
  - ➕ High assurance implementations
- Modern PQC systems
  - ➕ No known quantum attacks
  - 🤔 PQC's novelty creates uncertainty
    - Strength of hard problems
    - Assurance of implementation



Simula UiB

What is Hybrid PQC?



Hybrid Solution

💡 **Combine old** RSA/ECC **and new** PQC schemes so the result remains secure if at least one of its components does

⛔ **NOT** using **old** RSA/ECC schemes **and new** PQC schemes **side by side** for backwards compatibility

(becomes insecure as soon as one of its components breaks)

Simula UiB

Why use Hybrid PQC?

Pros

- **hedge against** a cryptographic or implementation **flaws** in one of the underlying component algorithms*
- accommodates **PQC** even **if** sector-specific requirements still **mandate legacy** algorithms

Cons

- **adds complexity** to architectures and implementations, *increasing costs and potentially security risks*
- typically **only** a **temporary** measure, *needing a second transition* to PQC only solution

*** NSM:** “There is a concern about possible future improvements in attacks against lattice-based cryptography, and there is a concern about potential vulnerabilities in immature implementations.”

CatKDF: A KEM Combiner

Key = HKDF(secret, label, f_context, length)

Where

- secret = psk || k1 || k2 || … || kn
- length is the desired output length
- label provides application separation
- f_context can bind to previous messages sent and received

What is it?

👍 ETSI standard

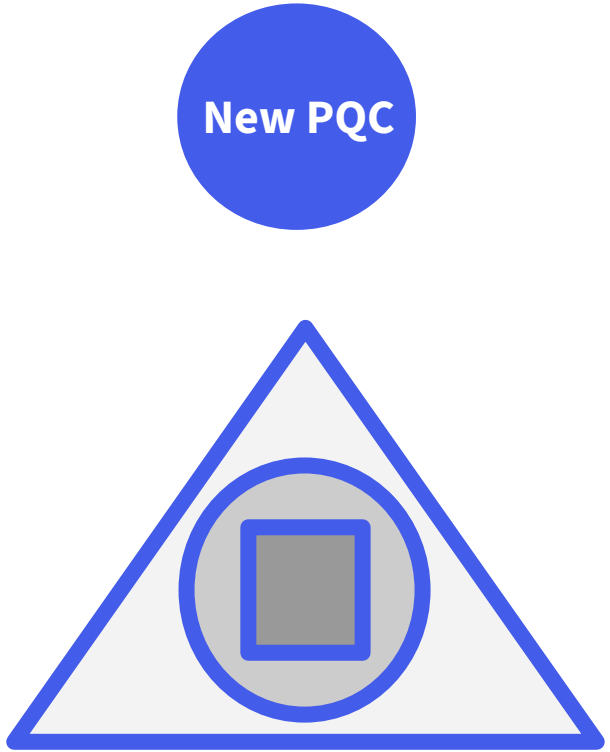
👍 **NSM** recommended

😕 Can also be used as AKE combiner

😟 Uses HKDF in a non-standard way

Source: https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf

The Complexity of Modern Cryptography



Simplistic view

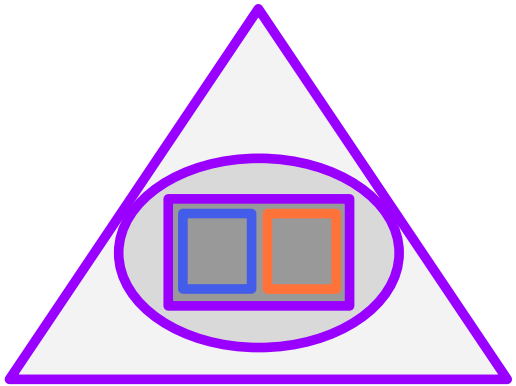
- Treating cryptography as a black-box

Realistic view

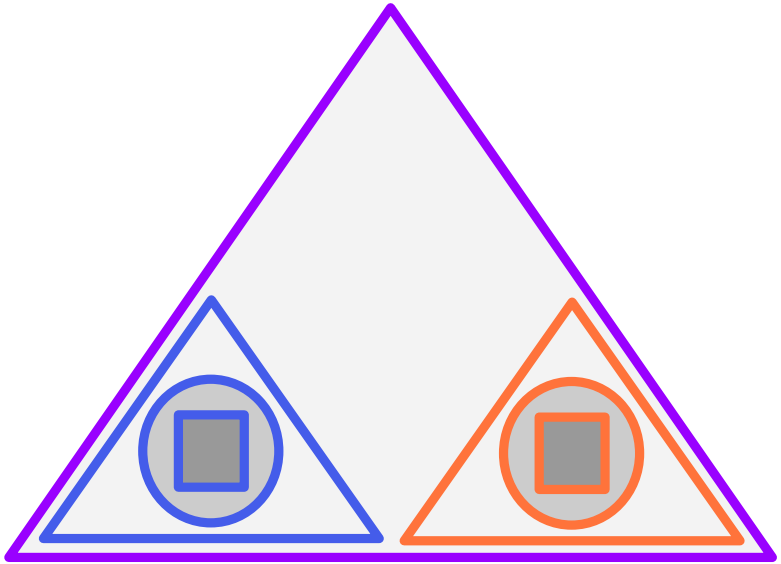
- *Inside* the box, there is another cryptographic primitive
- *Outside* the box, there is some cryptographic protocol

Curiously, modular design complicates hybrid PQC

Options for Hybrid PQC



Combine at the *lowest* level



Combine at the *highest* level

Choices for Hybrid PQC: Pros and Cons

Triangle represents a trade-off between competing goals, where combining at

- **lower levels** is more **efficient**
- **higher levels** is more **compliant** with standards
- **any level** is **challenging** w.r.t. **security** guarantees

Bespoke solutions *balance efficiency with security and compliance needs*

Simula UiB

Cryptographic Agility

What is Cryptographic Agility?

> ***Cryptographic Agility*** *is a measure of an organization's ability to adapt cryptographic solutions or algorithms (including their parameters and keys) quickly and efficiently in response to developments in cryptanalysis, emerging threats, technological advances, and/or vulnerabilities.*
>
> *It is also a design principle for implementing, updating, replacing, running, and adapting cryptography and related business processes and policies with no significant architectural changes, minimal disruption to business operations, and short transition time.*

"Building Cryptographic Agility in the Financial Sector", FS-ISAC,2024

Cryptographic agility allows you to

- Easily adapt and upgrade algorithms
- Migrate smoothly in the future
  (say from hybrid PQC to pure PQC)

Applying the concept of key management to the entire cryptographic ecosystem

Recommendations roundup

NSM Cryptographic Recommendations

Related to the new PQC standards (ML-KEM and ML-DSA), concern about

- possible future improvements in attacks against lattice-based cryptography
- potential vulnerabilities in immature implementations

recommend to use

- both a quantum-vulnerable and a quantum-resistant scheme in hybrid mode
- with rather conservative parameters to account for potential improvements in cryptanalysis

NSM Scheme Specific Recommendations

For confidentiality:

- ML-KEM-768 or ML-KEM-1024 using CatKDF as hybrid combiner

For authenticity:

- ML-DSA-65 or ML-DSA-87 using a simple hybrid combiner *if possible*
- SLH-DSA (no need for hybrid)

Possible for software signing / firmware updates:

- LMS and XMSS (no need for hybrid, but challenging to deploy securely)

Recommendations will be updated based on further NIST standards

Joppe Bos (NXP)'s Lessons

Lesson 1: scattered standards will be a problem

Lesson 2: urgency when to migrate depends on use case

Lesson 3: often size is a bigger issue than speed

Lesson 4: side-channels are a moving target

Lesson 5: migration is complicated → hybrid crypto

Source: https://www.joppebos.com/presentations/Summerschool_PROACT_PQC.pdf

Resources

Sector-specific Resources

Internet	https://wiki.ietf.org/group/sec/PQCAgility
Cloud	https://cloudsecurityalliance.org/research/working-groups/quantum-safe-security
Finance	https://www.fsisac.com/knowledge/pqc https://www.europol.europa.eu/about-europol/european-cybercrime-centre-ec3/qsff
Telecommunications	https://www.gsma.com/solutions-and-impact/technologies/security/post-quantum/