Post-Quantum Cryptography in Practice:

Migration Strategies for Constrained and Embedded Systems

Kris Kwiatkowski
Staff Cryptography Architect
21 / 11 / 2025

NIST PQC standardization process

In 2016, NIST requested 2 types of asymmetric cryptosystems for:

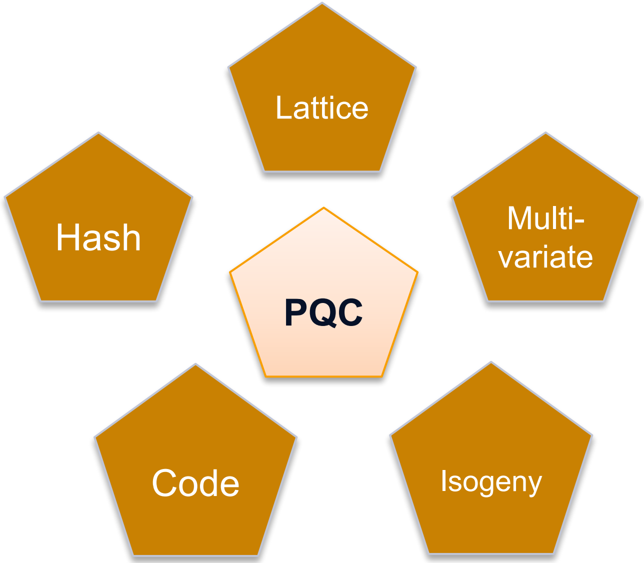
- Digital signature
- Key Encapsulation Mechanism (KEM) for key agreement

The three-round process with each round lasting for around 2 years

- **82** schemes were submitted, **69** candidates were accepted, 5 different categories, each representing a different underlying hard problem
- Round 1 ('18) - 64 accepted (19 digital signatures / 45 KEMs)
- Round 2 ('20) - 26 accepted (9 digital signatures / 17 KEMs)
- Round 3 ('23) - 4 schemes selected for standardization in '24/'25 (3 digital signature and 1 KEM scheme)

Different and more complicated than AES/SHA-3 standardization

- Larger problem space
- Must integrate well with comms / Internet protocols
- KEM is not a drop-in replacement for DH



New NIST Post-Quantum Standards

PQ SHIELD

Released in October ’20

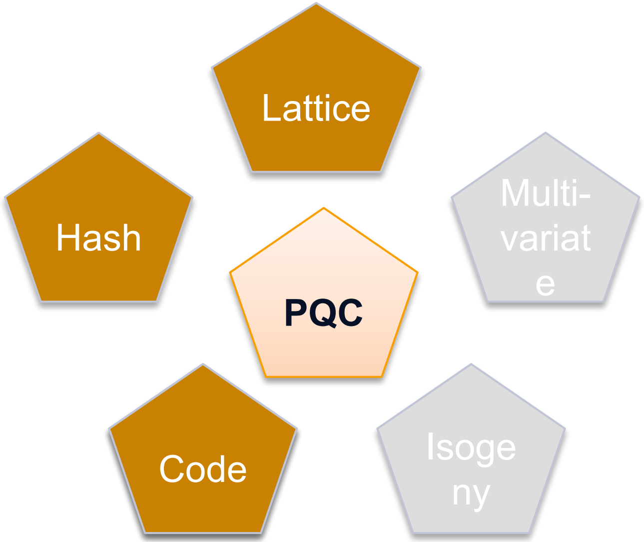
- SP800-208 - **LMS** (RFC8554) and **XMSS** (RFC 8391) . Stateful Hash-based Digital Signatures, standardized by IETF already in 2019. Part of CNSA 2.0 suite, to be used for software/firmware updates

Released in August ’24

- FIPS 203 - **ML-KEM** ("Kyber") for Key Establishment. Replaces EC Diffie-Hellman key exchange (example: TLS handshake) and RSA.
- FIPS 204 - **ML-DSA** ("Dilithium") for Signatures. Replaces {Ed,EC}DSA and RSA signatures in web authentication, PKI certificates.
- FIPS 205 - **SLH-DSA** ("SPHINCS+") Stateless Hash-based Digital Signature Algorithm. Likely to see use in "root of trust" applications

To be released

- FIPS 206 **FN-DSA** ("Falcon"), Nov 2025
- **HQC** - and additional KEM scheme selected in Round 4 of PQC



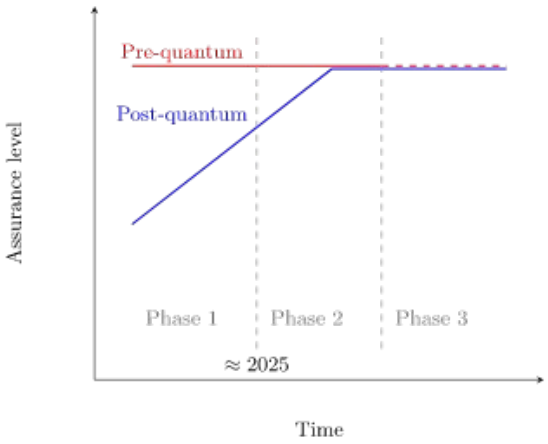
Need for compliance: it is irrelevant whether cryptographically relevant quantum computers are a threat to public-key crypto.

Hybrid Transition Path

PQ SHIELD

The industry expressed an interest in multi-step migration into post-quantum cryptography. ANSSI proposes 3-step process:

- **Phase** 1 (~2022): hybridization to provide some additional post-quantum *defense-in-depth* to the pre-quantum security assurance.
- **Phase 2** (~2025): hybridization to provide *post-quantum security assurance* while avoiding any pre-quantum security regression.
- **Phase 3** (~2030): optional standalone post-quantum cryptography.



NIST has approved hybrid key exchange as part of FIPS certification process.

- FIPS-approved cryptographic libraries increase the credibility of PQ schemes

SP800-56C rev2

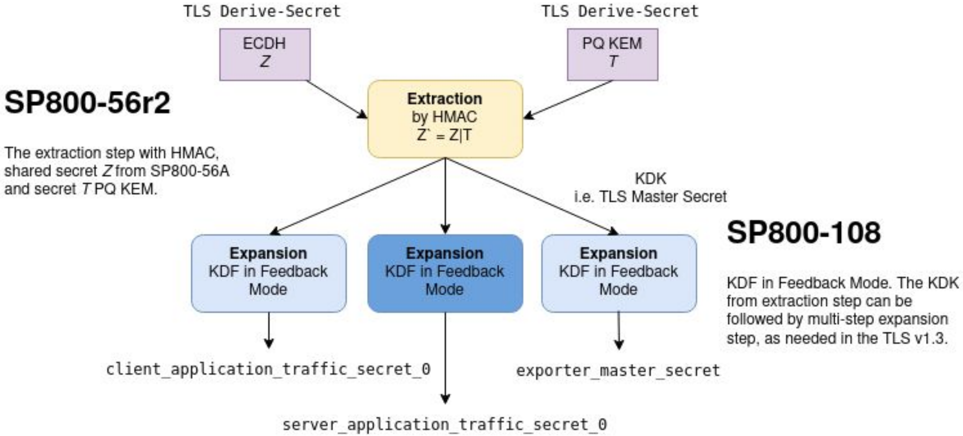
In addition to the currently **approved** techniques for the generation of the shared secret Z as specified in SP 800-56A and SP 800-56B, this Recommendation permits the use of a "hybrid" shared secret of the form $Z' = Z || T$, a concatenation consisting of a "standard" shared secret Z that was generated during the execution of a key-establishment scheme (as currently specified in [SP 800-56A] or [SP 800-56B]) followed by an auxiliary shared secret T that has been generated using some other method. The content, format, length, and method used to generate T must be known

Hybrid Transition Path

Example: *Key exchange in the TLS*

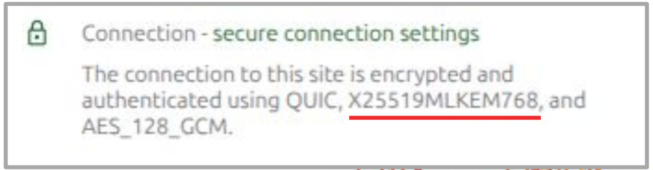
Hybrid key exchange in TLS v1.3

- The ***hybrid*** mode provided by HKDF
- ***Extract***: Combines output of two key agreements
- ***Expand***: Outputs symmetric key
- SP800-56Cr2 allows for FIPS-approved hybrid PQ TLS key exchange



Workgroup:	Transport Layer Security			
Internet-Draft:	draft-kwiatkowski-tls-ecdhe-mlkem-03			
Published:	25 December 2024			
Intended Status:	Informational			
Expires:	28 June 2025			
Authors:	K. Kwiatkowski *PQShield*	P. Kampanakis *AWS*	B. E. Westerbaan *Cloudflare*	D. Stebila *University of Waterloo*

Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3



Hybrid Transition Path

Authentication in the TLS

PQ SHIELD

Challenges

- Lack of drop-in replacement for RSA / ECDSA / EdDSA.
- The increased size leads to practical issues: e.g., larger certificate chains can trigger **extra network round-trips** or **compatibility failures** in middle-boxes.
- Non-obvious migration path for **hybrid** X.509 certificates.
- Complex logic required for both signing and verification when supporting classical + PQ algorithms.
- Nevertheless, standardization of these schemes for IETF protocols is ongoing.

```
Internet Engineering Task Force (IETF)                        J. Massimo
Request for Comments: 9881                                 P. Kampanakis
Category: Standards Track                                            AWS
ISSN: 2070-1721                                                S. Turner
                                                                   sn3rd
                                                        B. E. Westerbaan
                                                              Cloudflare
                                                            October 2025

  Internet X.509 Public Key Infrastructure -- Algorithm Identifiers for
      the Module-Lattice-Based Digital Signature Algorithm (ML-DSA)
```

```
LAMPS                                                       M. Ounsworth
Internet-Draft                                                   J. Gray
Intended status: Standards Track                                 Entrust
Expires: 2 May 2026                                              M. Pala
                                                              OpenCA Labs
                                                             J. Klaussner
                                                     Bundesdruckerei GmbH
                                                                S. Fluhrer
                                                             Cisco Systems
                                                           29 October 2025

      Composite ML-DSA for use in X.509 Public Key Infrastructure
                  draft-ietf-lamps-pq-composite-sigs-13
```

```
Network Working Group                                          N. Bindel
Internet-Draft                                                SandboxAQ
Intended status: Informational                                   B. Hale
Expires: 22 December 2025                      Naval Postgraduate School
                                                              D. Connolly
                                                                SandboxAQ
                                                               F. Driscoll
                                         UK National Cyber Security Centre
                                                              20 June 2025

                        Hybrid signature spectrums
             draft-ietf-pquip-hybrid-signature-spectrums-07
```

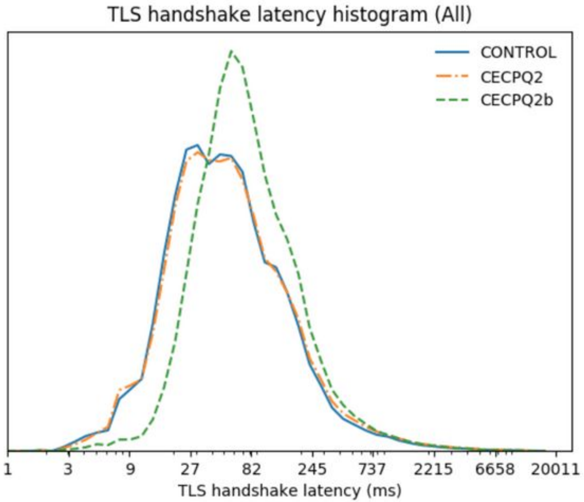
Why Drop-In Replacements Fail

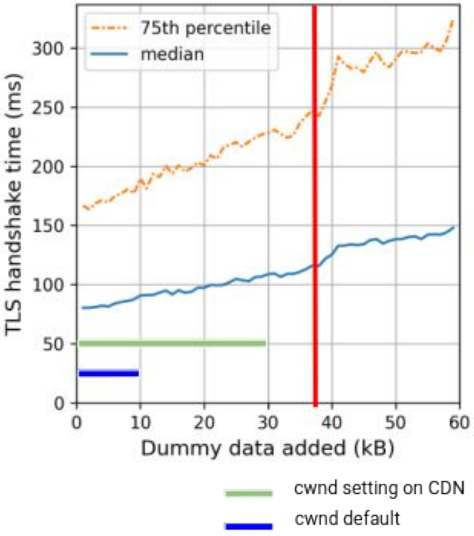
TLS key exchange:

- **CECPQ2** = X25519 + NTRU-HRSS, **CECPQ2b** = X25519 + SIKE, **CONTRL** = X25519
- CECPQ2 behaves almost the same as CONTROL group
- Moderate impact on key exchange

TLS authentication: 40KB is a breaking point.

- A single TLS session uses ~6 signatures and public keys.
- MLDSA would have a large impact on TLS handshake





* blog.cloudflare.com

System Level Costs

PQ SHIELD

- **Key agreement**
  - Public key / ciphertext: ~25x bigger
- **Digital signature** (MLDSA, general purpose)
  - Public key: ~40x bigger
  - Signature: ~35x bigger
- **Long-term support is critical**
  - HW can't be updated
  - PQC implementations are still evolving
  - Support for *brownfield* devices (legacy)

Key agreement

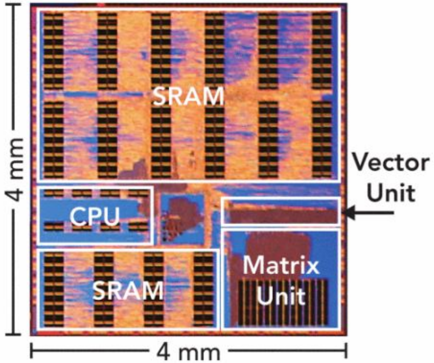
	Security*	**Public key**	**Ciphertext**	**Secret**
ECDH/p256	128	32 (x-only)	N/A	32
ML-KEM 512	128	800	768	32
ML-KEM 768	256	1184	1088	32
ML-KEM 1024	256	1568	1568	32

Digital signatures

	Security	**Public key**	**Signature**
ECDSA/p256	128	**32** (x-only)	64
LMS-SHA2-M32-H15-W8	256	52	1612
ML-DSA-65	192	**1952**	3309
SLH-DSA-SHA2-256s	256	64	**29792**

Deployment Considerations on IoT/Edge

- **Memory**: Typically 8-16KB RAM in constrained devices (for crypto)
- **Performance:** Optimizing memory may impact performance.
- **Energy consumption:** Important factor on battery operated devices
- Trade-off triangle:
  - **speed ↔ size ↔ energy**
  - only two can be optimized at once



Memory usage

	Keygen	Sign	Verify
EdDSA	7.5	7.5	3
ML-DSA-65 (small)	4.5	11.8	4.6
ML-DSA-65 (fast)	60.8*	68.8*	9.8*
ML-KEM-768 (small)	9	9.1	12.8
ML-KEM-768 (fast, x86)	19.6	19.9	26.7

Performance (10k cycles, Cortex-M4)

	Keygen	Sign or Encaps	Verify or Decaps
ML-DSA-65 (small)	3412*	24421*	5732*
ML-DSA-65 (fast)	2516*	6193*	2415*
ML-KEM-768 (portable C)	988*	1138*	1387*
ML-KEM-768 (small&fast)	644*	664*	714*

* Results of pqm4 library

Deployment Considerations on IoT/Edge

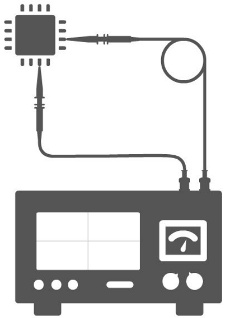
- **Why PQC on resource constrained devices is hard**
  - PQC is resource-intensive
  - Side-channel protections make PQC even harder to implement efficiently.
  - Lack of straightforward migration path

- Elliptic Curve Cryptography was a "***Swiss knife***" for most crypto applications
  - Small, fast, secure… all in one

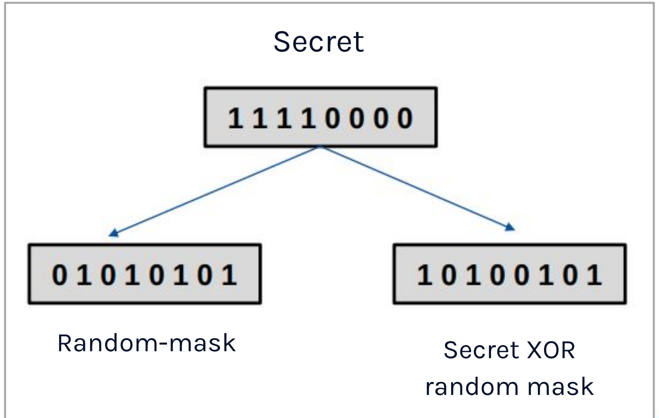
The challenge with post-quantum cryptography is to find the right **balance** between **application, scheme** and **implementation technique.**

Security-Critical Aspects

PQ SHIELD



- Side-channel leakage can expose information about the secret key used on the platform.
- Implementations for custom devices must be designed to avoid time-based side channels.
- Different operations may require varying amounts of **time** or **power**, and the same operation can show different **power** or **timing characteristics** depending on the processed data.
- As a result, an attacker **may infer which operations or data values** are being handled.



First order masking

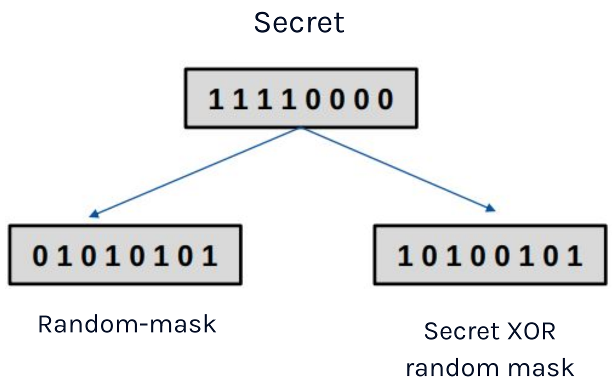
Security-Critical Aspects

PQ SHIELD

- Protection against more advanced side-channel attacks, like **Differential Power Analysis (DPA)**, requires the use of modern implementation techniques.

- Higher-order masked decapsulation of ML-KEM (from [BGR+21], measured on ARM Cortex-M4)

Masking order	d = 1	d = 2	d = 3
Performance	x3.5	x50	x130



First order masking

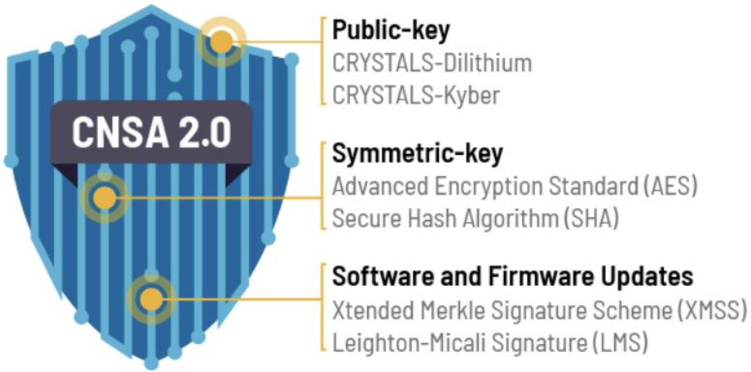
Use case

Consider the following theoretical use case for an ***embedded*** device that needs to exchange data with a cloud service.

- **The secure boot of the embedded device**
  - The firmware is signed with the hash-based signature
  - The signing is done on the HSM
  - Verification of the firmware must be fast => **LMS** (RFC8554)

- **Authentication**
  - Device uses mutual authentication to authenticate to the cloud service (e.g. TLS)
  - Signature size is important => **ML-DSA** (FIPS 204)

- **Key agreement**
  - To agree on symmetric encryption keys => **ML-KEM** (FIPS 203)



Lattice-based schemes

PQ SHIELD

ML-KEM
Key exchange scheme

- Replacement for ECDH/RSA key agreement
- Three security levels: ML-KEM-{512, 768, 1024}
- Implementations work on matrices of size kxk (k=2,3,4)
- Vectors are composed of polynomials of degree 255 with coefficients in a ring Z_q, $q=13 \cdot 2^8+1$ (**12-bit**)
- Produces full entropy shared secret. No need to apply KDF to get full entropy.
- IND-CCA2 security: ensures the confidentiality of the plaintext and resistance against chosen-ciphertext attacks (higher bar vs ECDH)

ML-DSA
Digital signature scheme

- Replacement for ECDSA/EdDSA
- Three security levels: ML-DSA-{44,65,87}
- The design follows the *Fiat-Shamir with* ***Aborts*** framework introduced by Lyubashevsky
- Implementations work on vectors of size kxl (kxl=4x4,6x5,8x7)
- Vectors composed of polynomials of degree 255 with coefficients in a ring Z_q, $q=2^{23} + 2^{13} + 1$ (**23-bit**)
- Uses uniformly-distributed random number sampling over small **integers** for computing coefficients in error vectors

ML-KEM and ML-DSA

Lattice-based, key encapsulation and digital signature algorithms

- Both schemes operate on a large matrix (*A*) of polynomials
  - **MLKEM-1024** uses matrix of 4x4 polynomials
    - Each polynomial has 256 coefficients, each 12-bit
    - Requires **8KB** of memory "just" for *A*
  - **MLDSA-87** uses matrix A of 8x7 polynomials
    - Each polynomial has 256 coefficients, each 23-bit
    - Requires **56KB** of memory "just" for *A*
  - Additional memory required by vectors r, e_1, e_2 and temporary results

Input : $pk = (\boldsymbol{t}, \boldsymbol{\rho})$, $\boldsymbol{m} \in \{0,1\}^{256}$, random $\boldsymbol{\mu} \in \{0,1\}^{256}$

Output: ciphertext (u, v)

$A \in R_q^{k \times k} \leftarrow sampleUniform(\boldsymbol{\rho})$

$r \in R_q^{k} \leftarrow sampleCBD^{\eta 1}(\boldsymbol{\mu})$

$e_1 \in R_q^{k}, e_2 \in R_q^{k} \leftarrow sampleCBD^{\eta 2}(\boldsymbol{\mu})$

$\boldsymbol{u} \leftarrow \mathbf{A^T r} + \boldsymbol{e_1}$

$\boldsymbol{v} \leftarrow t^T r + e_2 + \text{Encode}(m)$

$\boldsymbol{U'} = \text{Encode}(\mathbf{Compress}_q(\boldsymbol{u}, du))$,

$\boldsymbol{V'} = \text{Encode}(\mathbf{Compress}_q(\boldsymbol{v}, dv))$

Return ct = ($\boldsymbol{U'}$, $\boldsymbol{V'}$)

MLKEM.Encrypt

Optimization Techniques

PQ SHIELD

Operations (MLDSA)

- Signing and key generation are much larger than verification
- Seed must be expanded to large matrix (A)
- Vector (y) is expanded once per loop iteration
- Result of matrix (A) * vector (y) multiplication used twice

Input : $sk = (\boldsymbol{\rho}, \ldots)$, message $\boldsymbol{M} \in \{0,1\}^{256}$

Output: signature $\sigma = (z, c)$

$z \leftarrow \perp$, $K = 0$

$A \leftarrow ExpanA(\boldsymbol{\rho})$

while $z = \perp$ **do**

  $y \leftarrow S^{l}_{\gamma 1-1}(K, \ldots)$

  $\mathbf{w_1} \leftarrow HighBits(\boldsymbol{Ay}, 2\gamma_2)$

  $\mathbf{c} \in B_\tau \leftarrow H(M \,||\, \mathbf{w_1})$

  $\mathbf{z} \leftarrow y + \mathbf{c}s_1$

  if $||z||_\infty >= \gamma_1 - \beta$ or $||LowBits(\boldsymbol{Ay} - cs_2, 2\gamma_2)||_\infty >= \gamma_2 - \beta$

    $z \leftarrow \perp$

  $K \leftarrow K + 1$

done

Return $\sigma = (\boldsymbol{z}, \boldsymbol{c})$

MLDSA.Sign

Optimization Techniques

PQ SHIELD

Solutions

- Algorithm redesign to *lazy expansion*
  - Works only with a single value of *w*
- This uses 3 polynomials at max
- Reduces memory footprint from ~56KB to 4KB (ML-DSA)

$$w \begin{bmatrix} w_1 = a_{11} \cdot y_1 + \cdots \\ w_2 = a_{21} \cdot y_1 + \cdots \\ \cdots \\ w_k = a_{k1} \cdot y_1 + \cdots \end{bmatrix} = A \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1l} \\ a_{21} & a_{22} & \cdots & a_{2l} \\ \cdots & \cdots & \cdots & \cdots \\ a_{k1} & a_{k2} & \cdots & a_{kl} \end{bmatrix} \cdot y \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_l \end{bmatrix}$$

Memory footprint for MLDSA-87

	Keygen [KB]	Sign [KB]	Verify [KB]
Reference	97	123	93
Optimized	4.8	8.1	2.7
EdDSA	7.5	7.5	3

Input : $sk = (\boldsymbol{\rho}, \ldots)$, message $\boldsymbol{M} \in \{0, 1\}^{256}$

Output: signature $\sigma = (z, c)$

$z \leftarrow \perp$, $K = 0$

$A \leftarrow ExpanA(\boldsymbol{\rho})$

while $z = \perp$ **do**

  $y \leftarrow S^{l}_{\gamma 1-1}(K, \ldots)$

  $\mathbf{w_1} \leftarrow HighBits(\mathbf{Ay}, 2\gamma_2)$

  $\mathbf{c} \in B_{\tau} \leftarrow H(M \,||\, \mathbf{w_1})$

  $\mathbf{z} \leftarrow y + \mathbf{c}s_1$

  if $||z||_{\infty} >= \gamma_1 - \beta$ *or* $||LowBits(\mathbf{Ay} - cs_2, 2\gamma_2)||_{\infty} >= \gamma_2 - \beta$

    $z \leftarrow \perp$

  $K \leftarrow K + 1$

done

Return $\sigma = (\boldsymbol{z}, \boldsymbol{c})$

MLDSA.Sign

Optimization Techniques

PQ SHIELD

Solutions

- If more memory is available, expand only vector y *lazily*.
  - Useful for ML-KEM

$$w \begin{bmatrix} w_1 = a_{11} \cdot y_1 + \cdots \\ w_2 = a_{21} \cdot y_1 + \cdots \\ \cdots \\ w_k = a_{k1} \cdot y_1 + \cdots \end{bmatrix} = A \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1l} \\ a_{21} & a_{22} & \cdots & a_{2l} \\ \cdots & \cdots & \cdots & \cdots \\ a_{k1} & a_{k2} & \cdots & a_{kl} \end{bmatrix} \cdot y \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_l \end{bmatrix}$$

Memory footprint for MLDSA-87 (KB)

	Keygen	Sign	Verify
Reference	38	51	36
Optimized	4.8	8.1	2.7
EdDSA	7.5	7.5	3

Input : $sk = (\boldsymbol{\rho}, \ldots)$, message $\boldsymbol{M} \in \{0, 1\}^{256}$

Output: signature $\sigma = (z, c)$

$z \leftarrow \perp$, $K = 0$

$A \leftarrow ExpanA(\boldsymbol{\rho})$

while $z = \perp$ **do**

$y \leftarrow S^{l}_{\gamma 1\text{-}1}(K, \ldots)$

$\mathbf{w_1} \leftarrow HighBits(\mathbf{Ay}, 2\gamma_2)$

$\mathbf{c} \in B_\tau \leftarrow H(M \,||\, \mathbf{w_1})$

$\mathbf{z} \leftarrow y + \mathbf{c}s_1$

if $||z||_\infty >= \gamma_1 - \beta$ or $||LowBits(\mathbf{Ay} - cs_2, 2\gamma_2)||_\infty >= \gamma_2 - \beta$

$z \leftarrow \perp$

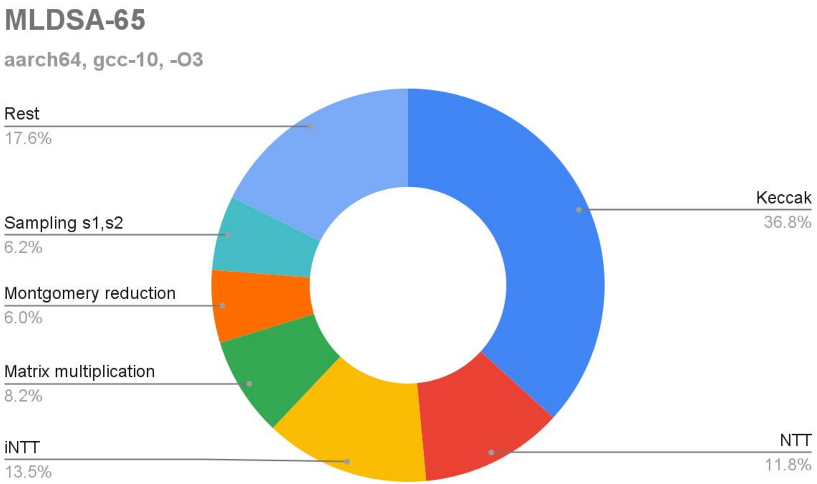
$K \leftarrow K + 1$

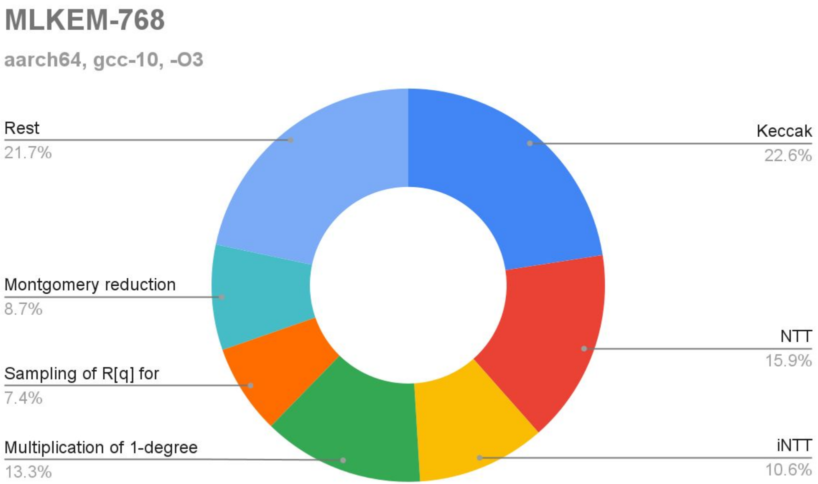
done

Return $\sigma = (\boldsymbol{z}, \boldsymbol{c})$

MLDSA.Sign

Analysis of hot-spots





Runtime determined by:

- SHA3/SHAKE, closer to 50% when implemented on smaller devices
- Polynomial arithmetic (NTT)

Keccak (SHA3/SHAKE)

Keccak (SHA3/SHAKE) is a main optimization target

- Expansion of matrix A is a big contributor to runtime
  - MLKEM-768: (3x3)x256 12-bit coefficients
  - MLDSA-65: (6x5)x256 23-bit coefficients
- Fast Keccak could speed up matrix A generation

HW-assisted implementations of SHA-3 possible today (ARM):

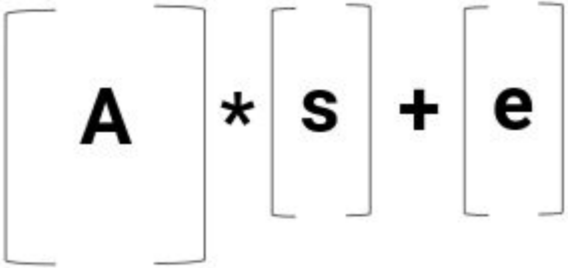
- Possibility to leverage BIC instruction from ARM ISA (A&~B) and ROR with barrel shifter

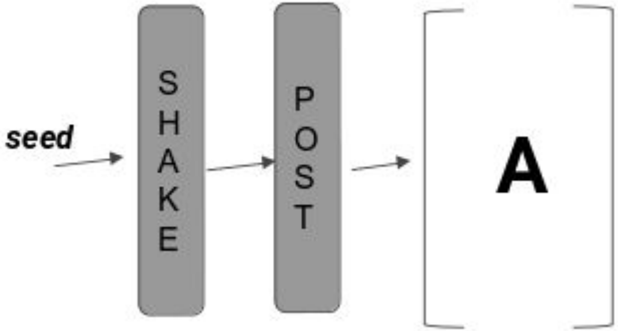
For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$, let

$$\mathbf{A}'[x,y,z] = \mathbf{A}[x,y,z] \oplus ((\mathbf{A}[(x+1) \bmod 5, y, z] \oplus 1) \cdot \mathbf{A}[(x+2) \bmod 5, y, z]).$$

- SIMD can be used to perform Keccak on multiple inputs in parallel. Research ongoing for ARM Cortex-M55 (Helium ISA)

HW-based SHA-3 accelerator to improve performance



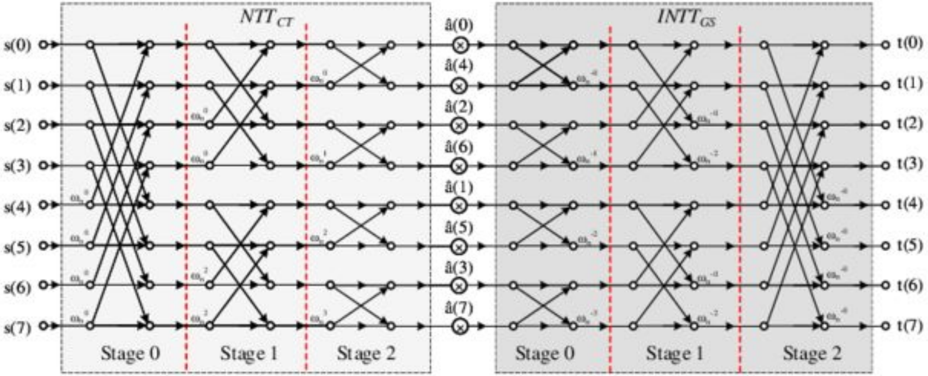


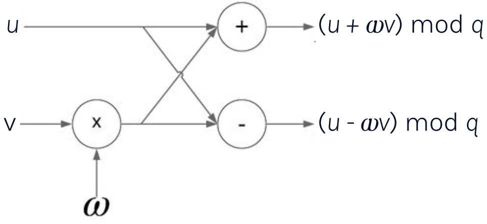
NTT

NTT = *Number Theoretic Transform*
(FFT in finite ring, usage similar as CRT in RSA)

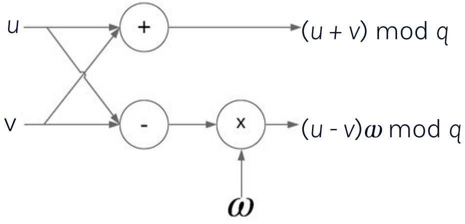
- Used in both MLKEM and MLDSA
- Complexity:
  - Transformation: O(*n* log*n*)
  - Multiplication : O(n)

- Polynomial arithmetic done in the NTT-domain
- $\mathbf{x} * \mathbf{y} = \mathrm{NTT}^{-1}(\mathrm{NTT}(\mathbf{x}) * \mathrm{NTT}(\mathbf{y}))$
- Example of usage in MLKEM:
  In theory, pubkey: $\mathbf{t} = \mathbf{As} + \mathbf{e}$
  In MLKEM :
  $\hat{\mathbf{t}} = \mathrm{NTT}(\mathbf{A}) * \mathrm{NTT}(\mathbf{s}) + \mathrm{NTT}(\mathbf{e})$





Cooley-Tukey Butterfly



Gentleman-Sande Butterfly

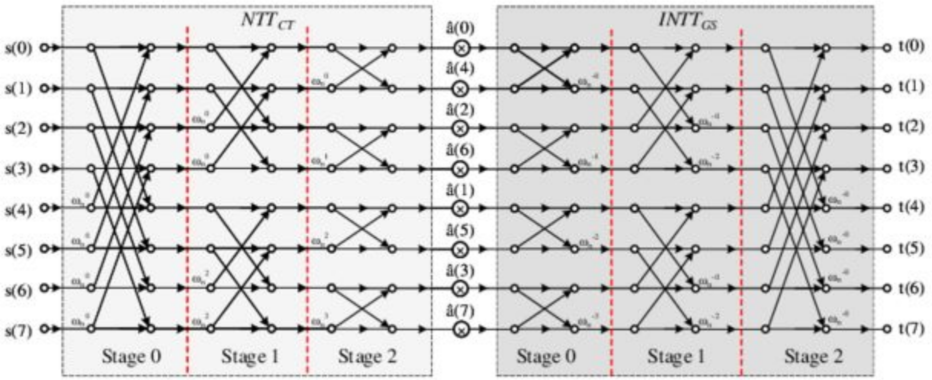
NTT

Scalar implementations (Cortex-M)

- Accumulate in double-width and reduce lazily, as late as possible [6],[3]
- Balance between different multiplication methods **Plantard**[7] or **Montgomery**

Vectorized implementations (Cortex-M55/85)

- Transform to the NTT domain is amenable to vectorization with SIMD type of parallel processing



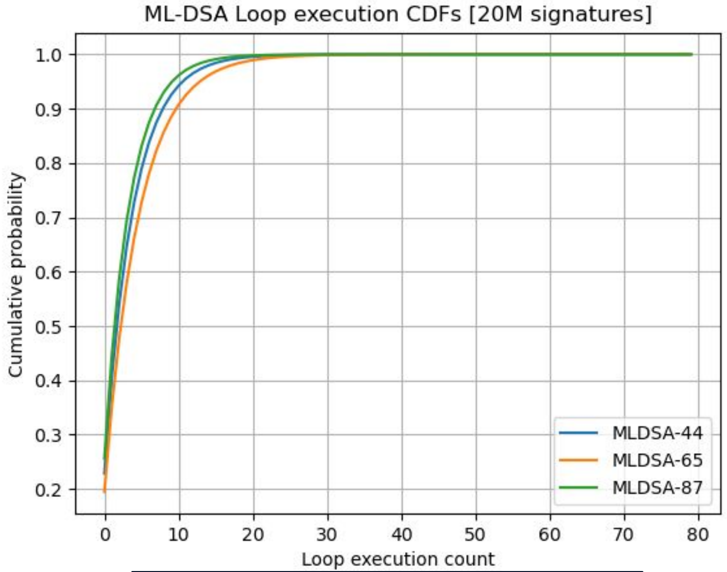
Common Pitfalls

ML-DSA

- The design follows the Fiat-Shamir with ***Aborts***
- Signing time is variable and depends on:
  - Public key
  - Message being signed
  - The random value generated during the signing
- The execution time of the algorithm is variable
- FIPS-204 provides the **expected** number of loops per parameter set, as well as guidance regarding max number of repetitions.

	Values assigned by each parameter set		
	ML-DSA-44	ML-DSA-65	ML-DSA-87
Repetitions (see explanation below)	4.25	5.1	3.85

- Worst-case runtime may be far from the expected value.



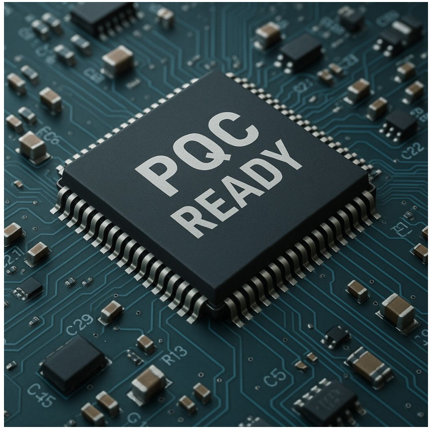
#	MLDSA-44	MLDSA-65	MLDSA-87
0	0.228676	0.194658	0.256148
1	0.405019	0.351457	0.446696
2	0.541106	0.477542	0.588357
3	0.646112	0.579195	0.693786
…			
10	0.942534	0.907480	0.961456
…			
83	1.0	1.0	1.0

Conclusions & Takeaways

- **Hybrid PQC is the realistic migration path today**
  - It enables quantum-safe key exchange without breaking existing systems.
  - Authentication remains costly - be deliberate about where you need it. Not every link needs hybrid auth now.
  - Pragmatic alternative - hash-based signatures are well-studied - good fit for firmware/code signing and some offline uses.

- **PQC on embedded devices is feasible only with careful engineering.**
  - RAM, flash, and energy constraints require optimized implementations.
  - Design for your bottleneck: Some operations need masking, while others can rely on constant-time logic.

- **PQC changes the cost model compared to ECC.**
  - Larger keys, heavy hashing, and masking requirements require system-level redesign.
  - Side-channel-secure implementations are much harder.



Thank you for your time

Questions?

References

During this presentation, I've used some ideas previously described in the following research papers:

- [1] Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography
- [2] Compact Dilithium Implementations on Cortex-M3 and Cortex-M4
- [3] Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1
- [4] Dilithium for Memory Constrained Devices
- [5] Hybrid scalar/vector implementations of Keccak and SPHINCS+ on AArch64
- [6] When to Barrett reduce in the inverse NTT
- [7] Improved Plantard Arithmetic for Lattice-based Cryptography
- [8] https://github.com/Emill/X25519-Cortex-M4
- [9] https://link.springer.com/chapter/10.1007/978-3-030-25283-0_6

Backup

ML-KEM

Lattice-based, key encapsulation mechanism

- Based on the hardness of lattice problems over module lattices*
- IND-CCA2 security: ensures the confidentiality of the plaintext and resistance to chosen-ciphertext attacks (higher bar vs ECDH)
- Produces full entropy shared secret
  - No need to apply KDF to get full entropy
  - Still may be needed, but for a different reason*

*** See "Binding" property in the IETF draft draft-ietf-pquip-pqc-engineers**

- Implementations work on vectors of size k (k=2,3,4)
- Three security levels:
  - ML-KEM-512, ML-KEM-768 and ML-KEM-1024
- Vectors represent polynomials of degree 255 with coefficients in a ring Z_q, $q=13\cdot2^8+1$ (12-bit)

Memory footprint for MLKEM-768

	ECDH/p256 (HW)	ML-KEM (SW)
RAM	1	x5
Timing	1	x4
Data transfer	1	x12

LMS: Leighton-Micali Signature scheme

Hash-based, stateful, signature scheme (NIST SP800-208)

Structure of the key

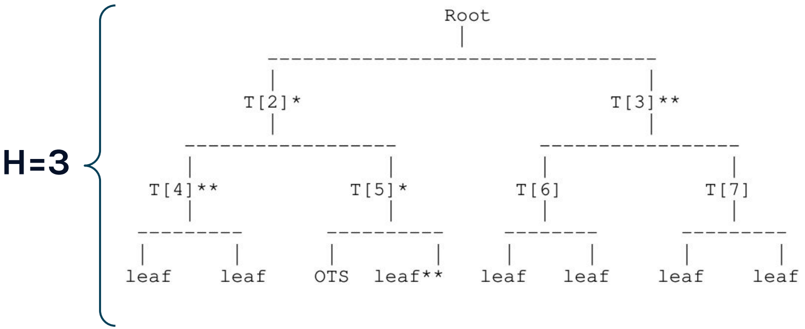
- Leaves represent a one-time event called LMOTS
- All *T[i]* are hash of two child leaves
- “Root” - a public key

Signing

- Message is signed with LMOTS secret key
- Authentication path: `leaf**, T[4]**, T[3]**`
- The signature includes index of the leaf

Verification

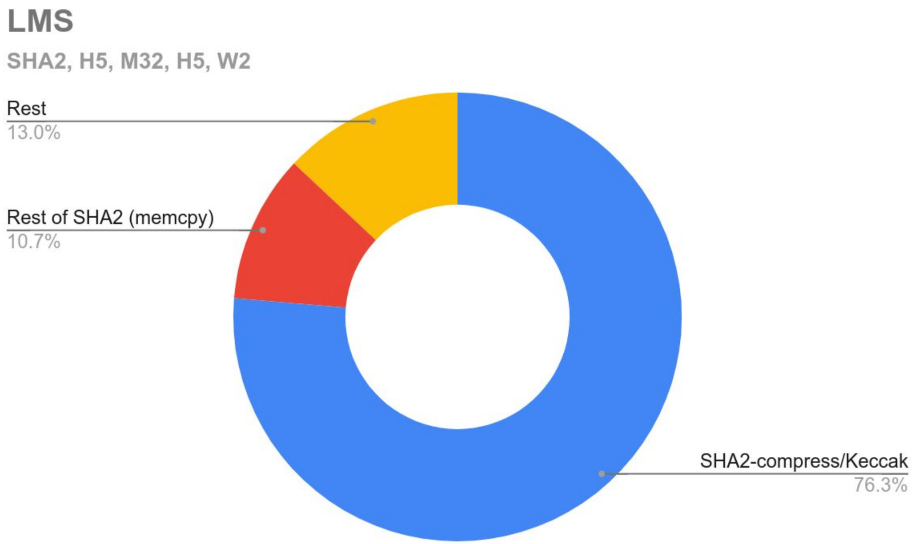
- LMOTS public key used to verify OTS part
- Hash of the authentication path
- Check if the result is the same as the `Root`



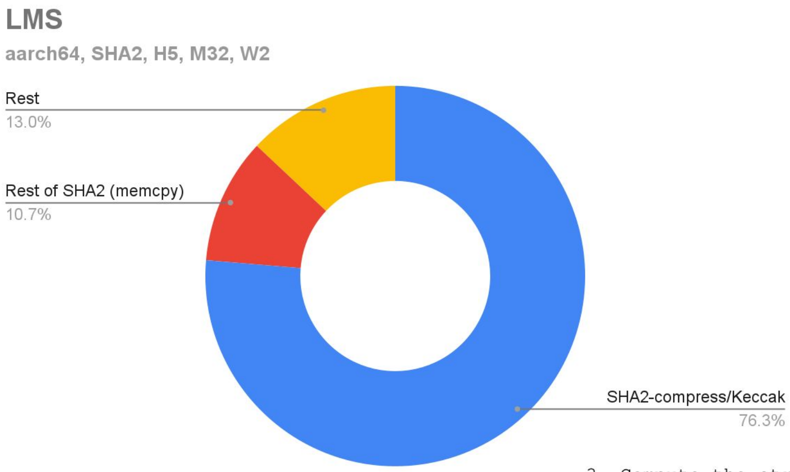
Number of signatures: $2^H = 8$

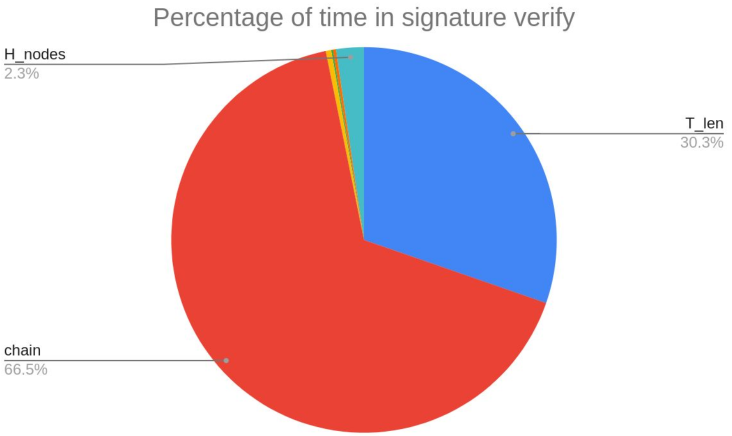
LMS performance

- Performance is largely dominated by runtime of the hash function
- A lot of operation on small chunks of memory



LMS performance






```
3. Compute the string Kc as follows:
   Q = H(I || u32str(q) || u16str(D_MESG) || C || message)
   for ( i = 0; i < p; i = i + 1 ) {
     a = coef(Q || Cksm(Q), i, w)
     tmp = y[i]
     for ( j = a; j < 2^w - 1; j = j + 1 ) {
       tmp = H(I || u32str(q) || u16str(i) || u8str(j) || tmp)
     }
     z[i] = tmp
   }
   Kc = H(I || u32str(q) || u16str(D_PBLC) ||
                                 z[0] || z[1] || ... || z[p-1])
```

LMS

Performance/size tradeoffs

- Large number of parametrizations (80)
- Can be instantiated with SHA2 or SHAKE256
- Number of signatures
- Operation runtime
- Signature size

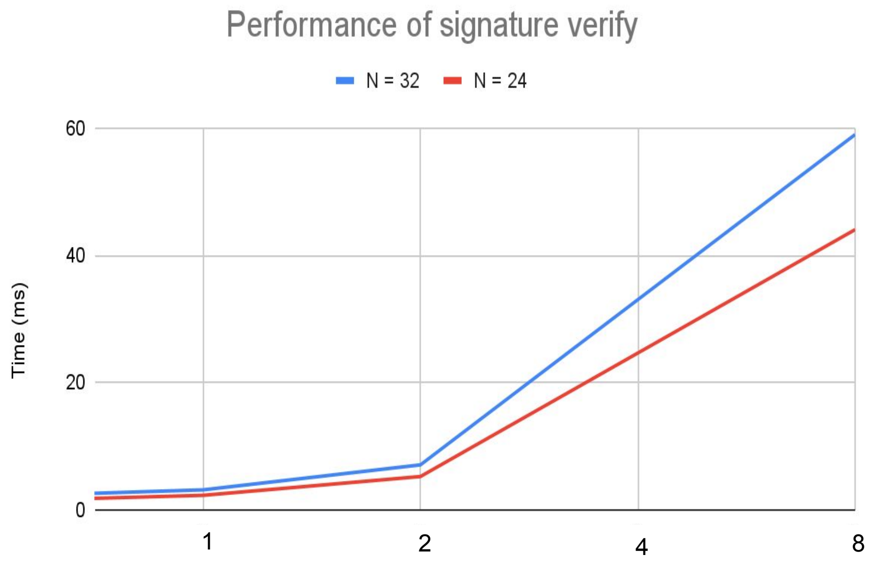
- Very fast verification
- Security based on the security of hash functions

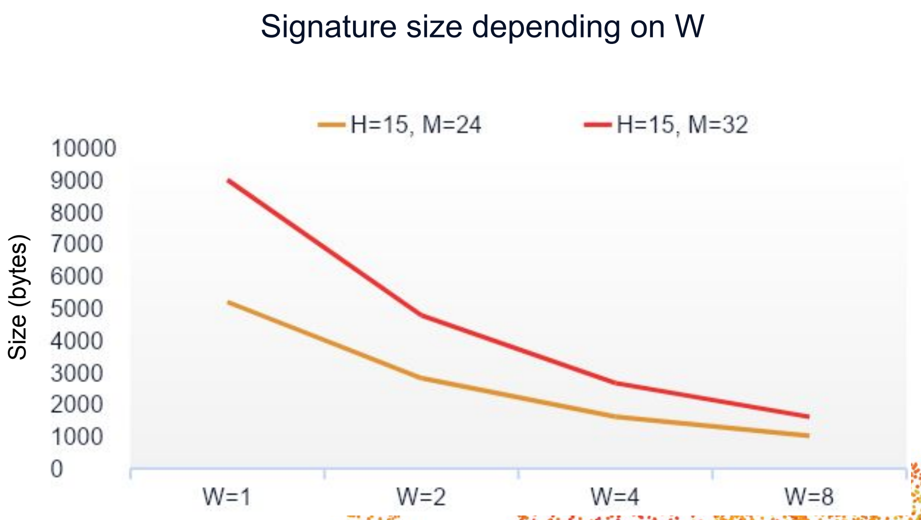
Pitfalls

- Stateful scheme
- Reuse of LMOTS key for signing two different messages compromises security guarantees
  - *Solution*: SLH-DSA (FIPS-205)
- Limited applicability (not suitable for generic use)
- Software implementations are not FIPS-approved.
- Slow and memory-hungry key generation and signing time (need to rebuild Merkle Tree)

Recommended by NSA in the **CNSA 2.0 for firmware signing.**

LMS parameterization

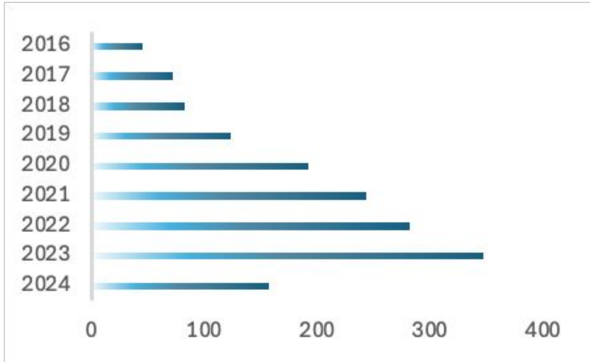




Support from cryptographic community

Academia

- PQC has been a very active research area in the past few decades
- Main contribution - design and cryptanalysis of the candidate schemes
- Number of PQC papers according to DBLP:



Industry / Protocol Standardization

- Feedback from deployments and experimentations
  - ’16, ’19, and ’21: Experimental TLS deployments of CECPQ1/2 schemes by Google and Cloudflare
  - ’23 Google enables Kyber in the Chrome browser
  - ’23 Signal updates X3DH protocol design to include PQ
  - ’24 Apple upgrades iMessage to use PQ3 protocol
  - ’24 Zoom announces rollout of PQC for E2EE
- IETF:
  - ’20 Hybrid-PQ TLS and IKEv2 start to be discussed
  - ’22 IETF starts PQC effort to integrate PQC in PKI
- NIST
  - ’23 NCCoE releases SP1800-38 describing migration to post-quantum