



DESIGN, AUTOMATION
AND TEST IN EUROPE

THE EUROPEAN EVENT FOR
ELECTRONIC SYSTEM DESIGN & TEST

20 – 22 APRIL 2026
VERONA, ITALY

PALAZZO DELLA GRAN GUARDIA



ML-DSA-OSH: An Efficient, Open-Source Hardware Implementation of ML-DSA

Quinten Norga, Suparna Kundu and Ingrid Verbauwhede



COSIC

COSIC, KU Leuven

KU LEUVEN

1. The Past

From CRYSTALS.Dilithium to ML-DSA.

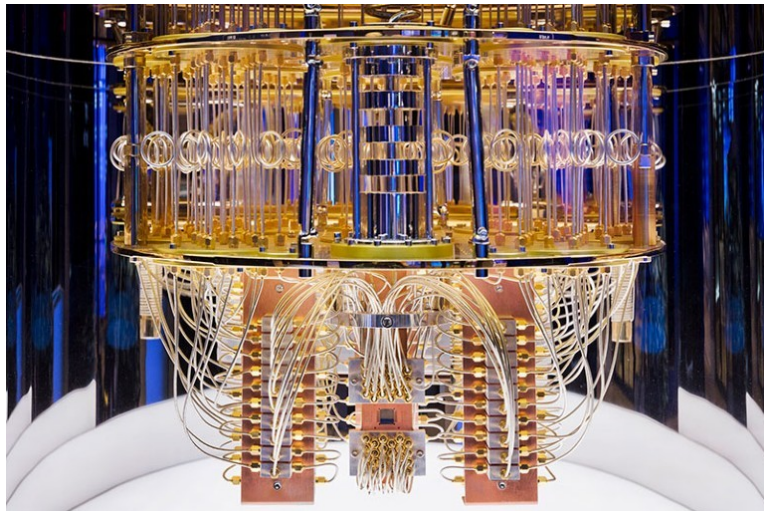
2. The Present

An Overview of ML-DSA-OSH.

3. The Future

Real-World Deployment of PQC.

Post-Quantum Cryptography



Source: doi.org/10.1126/science.adh2923

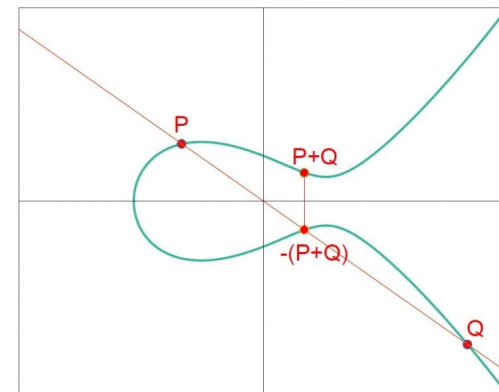
CRQ

A Cryptographically Relevant Quantum Computer

PKC

Public-Key Cryptography

Example:



ECC

Elliptic Curve Cryptography



National Institute of Standards and Technology

DSA

Digital Signature Algorithm

ML-DSA
FIPS 204
est. 2024

SLH-DSA
FIPS 205
est. 2024

FN-DSA
FIPS 206
est. 2026

...

KEM

Key Encapsulation Mechanism

ML-KEM
FIPS 203
est. 2024

HQC-KEM
FIPS 205
est. 2027



Lattice-based



Hash-based



Code-based

- Lattice-based DSA
- Selected in 2022
- Published in 2024 as ML-DSA (FIPS 204)
- Signing $\approx$ rejection sampling
 - Non-constant time
- Polynomial arithmetic, SHA-3 for hashing/XOF
- Challenging to implement securely & efficiently
 - Gap in literature: (open-source) hardware implementations

- Parameters & Variables

$$tr \leftarrow H(pk, \textcolor{red}{64})$$

$$\tilde{c} \leftarrow H(\mu, \text{w1Encode}(\mathbf{w}_1), \textcolor{red}{\lambda/4})$$

- Domain Separation

$$(\rho, \rho', K) \in \mathbb{B}^{32} \times \mathbb{B}^{64} \times \mathbb{B}^{32} \leftarrow H(\xi \parallel \textcolor{red}{\text{IntegerToBytes}(k, 1)} \parallel \textcolor{red}{\text{IntegerToBytes}(l, 1)}, 128)$$

- Hedged & Deterministic Signing

$$\textcolor{red}{rnd} \leftarrow \mathbb{B}^{32}$$

$$\textcolor{red}{rnd} \leftarrow \{0\}^{32}$$

$$\rho'' \leftarrow H(K \parallel \textcolor{red}{rnd} \parallel \mu, 64)$$

Efficient, Open-Source Hardware (RTL) Implementation of ML-DSA

Goal: address the lack of research into implementation aspects of ML-DSA

Enables:

- Prototyping optimizations
- Integration & validation of side-channel attacks/countermeasures

- *Unified* ML-DSA RTL implementation
 - Based on Crystals-Dilithium design [BeNgGa21]
- Supports KeyGen(), SigGen(), Verify()
 - NIST II, III & V parameter sets
- Reduce SigGen() latency by 16 - 36% vs. SoTA
 - Pre-computation & optimized operation scheduling

L. Beckwith, D. T. Nguyen, and K. Gaj, “High-performance hardware implementation of crystals-dilithium,” in *2021 International Conference on Field-Programmable Technology (ICFPT)*, 2021, pp. 1–10.

ZCU102 Ultrascale+

RESOURCE UTILIZATION OF ML-DSA AND ITS SUB-MODULES ON FPGA.

Sub-module	LUTs	FFs	DSPs	BRAMs
Samplers { A , s , y , c }	8,241	3,230	0	0
MakeHint & UseHint	8,837	3,772	0	0
PolyArith (NTT)	8,794	5,889	16	0
Keccak	18,678	13,394	0	0
(D)e(n)code & Decompose	6,889	1,368	0	0
Data Memory	2,317	0	0	29
Control & Other	1,186	1,656	0	0
Total	54,942	29,309	16	29

Performance Comparison

OVERVIEW AND COMPARISON OF PQC DSA HARDWARE IMPLEMENTATIONS.

Scheme	Platform	Latency [cc & μ s]			Area [LUT/FF/DSP/BRAM] (or mm^2 for ASIC)
		KeyGen	SigGen	SigVer	
Dilithium-II (v3.1)	A7 @ 163 MHz	18,761 115	76,613 470	19,687 121	27,433/10,681/45/15
	A7 @ 114.6 MHz	5,251 46	33,503 293	4,936 43	20,621/8,104/16/8
Dilithium-III (v3.1)	A7 @ 145 MHz	33,102 228	123,218 850	32,050 221	30,900/11,372/45/21
	A7 @ 109 MHz	7,871 72	53,577 491	7,610 70	23,378/9,079/8/20
Dilithium-V (v3.1)	A7 @ 140 MHz	50,982 363	145,192 1,042	52,712 377	44,653/13,814/45/31
	A7 @ 114.3 MHz	12,483 109	72,179 632	11,825 103	28,791/11,338/24/8
	US+ @ 391 MHz	63,200 161	113,900 291	67,900 173	13,975/6,845/35/4
ML-DSA-87	5nm @ 600 MHz	15,600 26	106,400 177	18,800 31	0.0366
Dilithium (v3.1)	US+ @ 256 MHz	4,875/8,291/14,037 19/32/55	29,876/49,437/55,070 117/193/215	6,582/9,724/13,642 26/38/53	53,907/28,435/16/29
	A7 @ 96.9 MHz	4,172/5,851/8,765 43/60/90	28,091/44,706/48,996 290/461/506	4,422/6,181/9,039 46/64/93	29,998/10,366/11/10
ML-DSA	US+ @ 230 MHz	4,872/8,291/14,033 21/36/61	24,283/32,800/37,592 106/143/160	6,637/9,730/14,642 29/42/64	54,942/29,309/16/29

ML-DSA-OSH enables:

- Optimizing sub-modules
- Architectural exploration
- Performance evaluation
- Prototyping side-channel & fault-injection attacks
- Evaluating effectiveness & cost of countermeasures
- ...

Thank You!

Questions?

See you in Verona!



IACR ePrint
2025/2337



GitHub
RTL Source Code