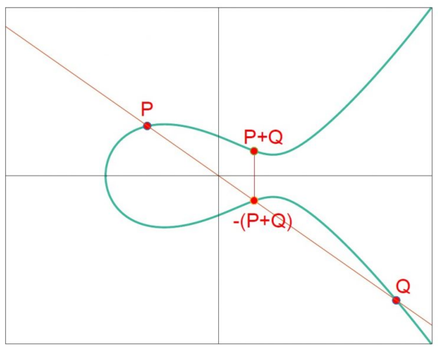
ML-DSA-OSH: An Efficient, Open-Source Hardware Implementation of ML-DSA

Quinten Norga Suparna Kundu Ingrid Verbauwhede

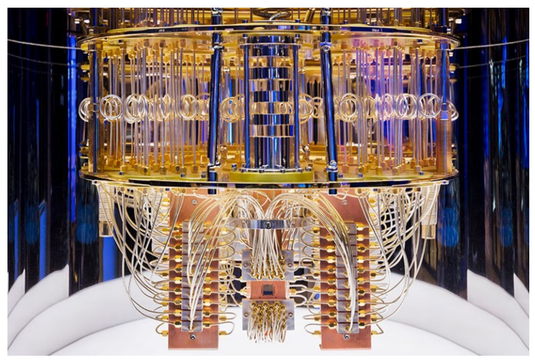
COSIC, KU Leuven

Post-Quantum Cryptography

Most of the widely deployed public-key cryptographic algorithms are based on **RSA** [RSA78] and elliptic-curve cryptography (**ECC**) [Mil86]. However, classical public-key cryptographic schemes are vulnerable against an adversary with large-scale quantum computers due to **Shor's algorithm** [Sho94].



Elliptic Curve Cryptography (ECC)



A Quantum Computer. *Image credit: IBM* [IB21].

Figure 1. A Cryptographically Relevant Quantum Computer (CRQC) would break classical public-key cryptography (e.g. ECC), motivating the need for Post-Quantum Cryptography (PQC).

Standardization

The National Institute of Standards and Technology (NIST) started organizing their Post-Quantum Cryptography (**PQC**) standardization procedure in 2016. As of early 2026, NIST's PQC standards portfolio contains a variety of Key Encapsulation Mechanisms (KEM) and Digital Signature Algorithms (**DSA**), based on different mathematical hard problems (Table 1).

Table 1. National Institute of Standards and Technology (NIST) PQC Standards, including lattice-based (green), code-based (orange) and hash-based (blue) algorithms.

KEM		DSA			
ML-KEM	HQC-KEM	ML-DSA	FN-DSA	SLH-DSA	...
FIPS 203	FIPS 207	**FIPS 204**	FIPS 206	FIPS 205	-
2024	*2027*	*2024*	*2026*	*2024*	-
[NIS24b]	-	[NIS24a]	-	[NIS24c]	-

Compared to `ML-KEM`, significantly less research has been performed on the implementation aspects of `ML-DSA`. Remarkably, there are only a handful of published hardware designs and no open-source hardware implementations of complete `ML-DSA`.

ML-DSA: Module-Lattice-Based Digital Signature Standard

`ML-DSA` is a post-quantum lattice-based digital signature algorithm (DSA) that the National Institute of Standards and Technology (NIST) recently standardized as FIPS 204. Industry and government are encouraged to migrate rapidly towards PQC, with the **real-world applications** of digital signature algorithms being enormous and responsible for ensuring **authenticity** and integrity of service providers, users, or data. Some examples:

- Transport Layer Security (**TLS**), to verify the identity of the server.
- **Blockchain** applications, to attain integrity of the transaction and non-repudiation.
- In the financial sector, for security, trust and maintaining integrity.
- In software libraries, firmware, or operating systems, to ensure integrity.
- Secure **email**, to ensure authenticity.
- In **healthcare** sector, providing privacy and integrity,

Differences with CRYSTALS-Dilithium (v3.0)

Although `ML-DSA` and `CRYSTALS-Dilithium` (v3.0) share most of the primitive components, there are some differences between them. Overall, the required modifications for migrating existing `CRYSTALS-Dilithium` implementations to match FIPS 204 are:

- **Parameters and Variables**: To ensure the properties beyond strong unforgeability [CDF+21], a 64-byte hash of the public key (tr) is added to the secret key and used for signing, instead of the message M directly.

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

- **Domain Separation**: ensure independence between different instantiations of some functions.

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

- **Hedged and Deterministic Signing**: scenarios in which **side-channel attacks and fault attacks** are possible should opt to implement the hedged variant, such that it is not easily guessed.

$$rnd \leftarrow \mathbb{B}^{32} \qquad \rho'' \leftarrow \mathtt{H}(K \,\|\, rnd \,\|\, \mu, 64)$$

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

We present an efficient, 'combined' open-source hardware (**OSH**) design of `ML-DSA`, based on a Dilithium implementation by Beckwith et al. [BNG21], which supports all `ML-DSA` operations (`KeyGen`, `SigGen` and `SigVer`) for all parameter sets (NIST II, III & V).

`https://github.com/KULeuven-COSIC/ML-DSA-OSH`

- **Operation Scheduling**: introduce **pre-computation** of critical variables in the signing rejection loop, reducing average signing latency by $16 - 36\%$.
- **Robustness**: support signing of a wider range of message lengths.
- **FIPS Compliance**: integrate all modifications/upgrades compared to `CRYSTALS`-Dilithium.

Performance Evaluation

All results are obtained using Xilinx Vivado 2021.1, targeting the ZCU102 Ultrascale+ platform Table 2 shows the resource utilization on FPGA. Table 3 provides the area and performance results of our designs and prior work for security levels II, III, and V.

Table 2. Resource utilization of `ML-DSA-OSH` and its sub-modules on FPGA.

Sub-module	LUTs	FFs	DSPs	BRAMs
Samplers $\{\mathbf{A}, \mathbf{s}, \mathbf{y}, c\}$	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**

As expected, our proposed modifications result in a **minor** ($\pm 1\%$) increase in **area overhead**, compared to the `CRYSTALS-Dilithium` (v3.1) version [BNG21]. Interestingly, our modifications and improvements result in a **reduction** of 16% - 36% in (average) clock cycles for the **signature generation** (NIST II, III and V) operation. A contributing factor is ensuring the multiplication $\hat{\mathbf{A}} \circ \mathtt{NTT}(\mathbf{y})$ starts as soon as both multiplicands are computed.

Table 3. Overview and comparison of PQC DSA hardware implementations.

Scheme		Platform	Latency [cc & μs] KeyGen	SigGen	SigVer	Area [LUT/FF/DSP/BRAM] (or mm^2 for ASIC)
`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)`	[BNG21]	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
	[ZZW+21]	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`	**Ours**	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

Summary

Our design is **highly flexible**, supporting all three security levels and digital signature routines. Our improved signature generation instruction scheduling results in signature generation latency of 106/143/160 μs for NIST II, III and V variants.

Future Work

We lay out a wide range of potential applications and directions of research, further enabled through `ML-DSA-OSH`, an open-source FPGA implementation of `ML-DSA`.

- Prototyping, evaluating or **optimizing** the implementation of `ML-DSA` and its sub-modules (NTT, Sampler, Lightweight Hash) for specific cryptographic applications.
- Evaluating side-channel and fault-injection **attacks** and allows for the direct integration **countermeasures** to prevent them (masking, shuffling, duplication), in the hardware domain.

References

[BNG21] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj.
High-performance hardware implementation of crystals-dilithium.
In *2021 International Conference on Field-Programmable Technology (ICFPT)*, pages 1–10, 2021.

[CDF+21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson.
Buffing signature schemes beyond unforgeability and the case of post-quantum signatures.
In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1696–1714, 2021.

[IB21] IBM and Philip Ball.
First quantum computer to pack 100 qubits enters crowded race.
Nature, 599:542, November 2021.

[Mil86] Victor S. Miller.
Use of elliptic curves in cryptography.
In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[NIS24a] NIST.
Module-lattice-based digital signature standard.
Technical report, U.S. Department of Commerce, Washington, D.C., 2024.

[NIS24b] NIST.
Module-lattice-based key-encapsulation mechanism standard.
Technical report, U.S. Department of Commerce, Washington, D.C., 2024.

[NIS24c] NIST.
Stateless hash-based digital signature standard.
Technical report, U.S. Department of Commerce, Washington, D.C., 2024.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Commun. ACM, 21(2):120–126, 1978.

[Sho94] Peter W. Shor.
Algorithms for quantum computation: Discrete logarithms and factoring.
In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.

[ZZW+21] Cankun Zhao, Neng Zhang, Hanning Wang, Bohan Yang, Wenping Zhu, Zhengdong Li, Min Zhu, Shouyi Yin, Shaojun Wei, and Leibo Liu.
A compact and high-performance hardware architecture for crystals-dilithium.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(1):270–295, Nov. 2021.

Resources and Funding

IACR ePrint

GitHub

Funded by the European Union
European Research Council
Established by the European Commission
BeQCI
Cyber Security Flanders
intel

https://www.esat.kuleuven.be/cosic/
COSIC, KU Leuven
quinten.norga@esat.kuleuven.be