Hardware/Software Co-Design of ML-DSA with Random Delay Insertion Against Correlation Power Analysis

Salvatore Coppa
DIEM
University of Salerno
Fisciano, Italy
scoppa@unisa.it

Ettore Napoli
DIEM
University of Salerno
Fisciano, Italy
enapoli@unisa.it

***Abstract*—Post-Quantum Cryptography (PQC) recently led to the standardization of the Module-Lattice-Based Digital Signature Algorithm (ML-DSA). Standard software implementations of ML-DSA on embedded devices are exposed to physical side-channel attacks, since the power consumption of the circuit depends on the processed secret data and allows key recovery via Correlation Power Analysis (CPA). In addition, protecting these implementations in software introduces large latency overheads. In this paper, the physical vulnerability of ML-DSA is considered focusing the study on embedded microcontroller and System-on-Chip (SoC) platforms. The computational cost of software countermeasures, validated via Test Vector Leakage Assessment (TVLA), is evaluated on both architectures. An Hardware/Software co-design implemented on a Zynq-7000 SoC is then proposed to overcome the software execution bottleneck. The vulnerable pointwise Montgomery multiplication is offloaded to an Advanced eXtensible Interface (AXI4-Full) hardware accelerator that integrates a 16-bit Linear Feedback Shift Register (LFSR) for Random Delay Insertion (RDI), preventing CPA trace alignment. Experimental results demonstrate a 13.4× speedup over the software-protected SoC implementation. The proposed architecture reduces the protected signing latency to 42.80 ms, achieving an execution time close to the 40.64 ms of the unprotected software implementation, while the dynamic power dissipated by the hardware accelerator in the programmable logic is limited to 33 mW.**

***Index Terms*—post-quantum cryptography, ML-DSA, side-channel analysis, correlation power analysis, hardware/software co-design, random delay insertion, System-on-Chip.**

I. Introduction

Public-key cryptographic schemes such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC) [1] rely on mathematical problems solvable by a quantum computer running Shor's algorithm, [2]. To address this threat, NIST standardized the Module-Lattice-based Digital Signature Algorithm (ML-DSA), derived from CRYSTALS-Dilithium, in Federal Information Processing Standards (FIPS) 204, [3]. The algorithm relies on the Fiat-Shamir with Aborts paradigm [4], extendable-output functions [5] to obtain compact keys, and the Number Theoretic Transform (NTT) [6] for fast polynomial arithmetic suitable for embedded systems [7].

When cryptographic algorithms run on MCU or System-on-Chip (SoC) devices, the switching activity of CMOS gates causes data dependent power dissipation [8]. Side-Channel Attacks (SCA), such as Correlation Power Analysis (CPA) [9], exploit these power variations to extract the secret key. Ref. [10] studied ML-DSA and identified the NTT pointwise Montgomery multiplication between the challenge polynomial and secret key as the main CPA target and demonstrated a nonprofiled CPA attack on an ARM Cortex-M4. It also showed that software masking and shuffling mitigated the vulnerability but required significant additional computational cost. Software countermeasure implementations confirm this performance cost. In [11], arithmetic and Boolean masking applied to Dilithium on a Cortex-M MCU required replacing the standard prime modulus with a power-of-two modulus to simplify masked bit-decomposition. In addition, as pointed out in [12], protecting the NTT against single-trace attacks via masking and shuffling on a Cortex-M4 increases signing time by up to 490%. Against nonprofiled multi-trace CPA, however, the forward NTT of the secret key does not interact with message-dependent inputs and exhibits an invariant leakage profile [10], isolating the pointwise multiplication as the vulnerable operation. Recently, Hardware/Software (HW/SW) co-design on heterogeneous SoCs, such as the Zynq-7000, has been proposed to accelerate Dilithium [13], [14]. However, existing architectures target only throughput optimization without hardware SCA protections. As shown in [15], unprotected FPGA implementations of ML-DSA are vulnerable to CPA due to deterministic register activity in the modular reduction circuit. While software Random Delay Insertion (RDI) is vulnerable to trace alignment frameworks [16], hardware RDI implemented in the synchronous state machine of an FPGA provides time-domain protection with low area overhead [17].

This paper studies the physical vulnerability of ML-DSA-44 (the NIST Security Level 2 parameter set featuring $(k, l) = (4, 4)$ polynomial matrix dimensions) and proposes an SCA protected HW/SW co-design for SoC architectures that avoids the latency penalty of software countermeasures. The results presented in the paper can be summarized as follows:

- profile ML-DSA-44 execution time on an ARM Cortex-M4 MCU (STM32G474RE) and quantify the physical leakage of the unprotected pointwise Montgomery multiplication via Test Vector Leakage Assessment (TVLA);
- implement arithmetic masking and coefficient shuffling [8], [11], [12] on the Cortex-M4 verified via TVLA ($|t| \leq 4.5$ where t is the Welch's t-test statistic quantifying first-order leakage confidence, [18]);
- evaluate the runtime overhead on the MCU (+485% per polynomial, +6.54% per signature) and on the 667 MHz ARM Cortex-A9 Processing System (PS) of a Zynq-7000 SoC (+1308%, 572.44 ms per signature);
- develop an HW/SW co-design on the Zynq-7000 SoC executing the pointwise multiplication via a 100 MHz AXI4-Full hardware accelerator in the Programmable Logic (PL);
- design the RDI protected HW accelerator that exploits a 16 bit LFSR RDI circuit (initial seed obtained by sampling the LSB of an on-chip ADC), to prevent CPA alignment. The design obtains 42.80 ms signing latency with 33 mW of dynamic power dissipation.

II. Algorithm Profiling and Vulnerability to SCA

To design an efficient and secure architecture, it is necessary to identify both the physical vulnerabilities of the algorithm and its computational bottlenecks.

During ML-DSA-44 signing, the response vector is computed over $R_q = \mathbb{Z}_q[X]/(X^{256}+1)$ ($q = 8\,380\,417$) as:

$$\mathbf{z} = \mathbf{y} + \text{INTT}(\hat{\mathbf{c}} \circ \hat{\mathbf{s}}_1) \pmod q \tag{1}$$

where INTT is the Inverse NTT and $\circ$ is the pointwise multiplication. From a physical security perspective, the critical point of failure is exactly this pointwise Montgomery multiplication $\hat{\mathbf{c}} \circ \hat{\mathbf{s}}_1$. During this operation, the processor combines the 23 bit secret key coefficients $\hat{\mathbf{s}}_1[i]$ with the known, message-dependent challenge $\hat{\mathbf{c}}[i]$. This interaction generates a data-dependent power consumption exploitable by a nonprofiled CPA. Indeed, [10] demonstrates that targeting the 64 bit intermediate products of this reduction on Cortex-M4 cores allows an attacker to recover the secret coefficients with only 157 traces.

Beyond physical vulnerabilities, the algorithm presents inherent computational challenges. Profiling the unprotected reference C implementation [7] on a bare-metal Cortex-M4 (STM32G474RE, 170 MHz) shows that while keypair generation (89.8 ms) and verification (95.5 ms) run in constant time, signing exhibits a variable mean latency of 282.20 ms ($\sigma = 166.90$ ms) across 500 executions due to the Fiat-Shamir with Aborts loop ($\mu = 3.29$ aborts). The overall execution runtime is heavily dominated by NTT conversions and the SHAKE128 extendable-output function (XOF) [5] used for pseudorandom matrix expansion.

To practically assess the vulnerability and evaluate the cost of standard defenses, Section III quantifies the physical leakage of the unprotected MCU and measures the latency overhead introduced by software countermeasures.

III. Software Countermeasures and Latency Bottleneck

A. Masking and Shuffling Implementation

Execution shuffling (temporal order randomization) and first-order arithmetic masking (data randomization) can mitigate pointwise multiplication leakage [8].

In this paper, the approaches studied in [8] have been implemented on MCU and SoC. The array with the 256 polynomial indices was pseudorandomly permuted before each multiplication using the Knuth-Yates algorithm [12] to desynchronize the power traces. In addition, the secret vector $\mathbf{s}_1$ was split into two additive shares over R_q, [11]:

$$\hat{\mathbf{s}}_1[i] = (\hat{\mathbf{s}}_1[i] - m_i) + m_i \pmod q. \tag{2}$$

To prevent early-recombination leakage in the ALU and Hamming distance overwrites on the memory bus, the partial Montgomery products were stored into two SRAM buffers, exploited by two independent INTT routines, and recombined only when adding $\mathbf{y}$.

B. Leakage Validation on Standard and Protected SW

To verify baseline vulnerability and countermeasure security, nonspecific Welch's t-test TVLA was run using 1500 fixed-key and 1500 random-key traces [18], [19]. Leakage validation was conducted on the STM32G474RE board, which provides a native current-sensing node [20]. To avoid adjacent instruction leakage overlap from internal voltage regulator filtering and eliminate Flash wait-state jitter, the Cortex-M4 clock was set to 16 MHz with the Flash Adaptive Real Time Accelerator disabled [21] and precomputed True Random Number Generator (TRNG) masks in SRAM. Current traces were acquired via a Hioki CT6711 probe (120 MHz bandwidth, 10 V/A gain) and sampled at 1 GS/s by a Tektronix MDO34 oscilloscope.

As shown in Fig. 1, the unprotected baseline confirms data-dependent leakage with a probability $> 99.999\%$, as 84.10% of the acquired samples exceed the ± 4.5 security threshold. Conversely, the protected firmware (Fig. 1, bottom) exhibits no sustained threshold violations associated with the cryptographic operation. The periodic spikes exceeding ± 4.5 in the protected trace (2.85% of points) occur at 1 ms intervals, corresponding to the 1 kHz ARM `SysTick` timer interrupt, whose spikes result from context stacking variations rather than cryptographic leakage.

The results presented for the MCU software implementation, being dependent on how the software handles data, are also valid for other software platforms and in particular for the Cortex-A9 implementation discussed in Section IV. In this paper, a direct TVLA evaluation on the Zynq-7000 board has not been conducted since, as demonstrated in [22], gathering power traces requires invasive PCB modifications.

C. Software Performance Bottleneck on MCU and SoC

To measure the deterministic cycle penalty of software countermeasures without Flash wait-state jitter, the protected pointwise multiplication was profiled at 16 MHz using the 32 bit Cortex-M4 Data Watchpoint and Trace cycle counter.

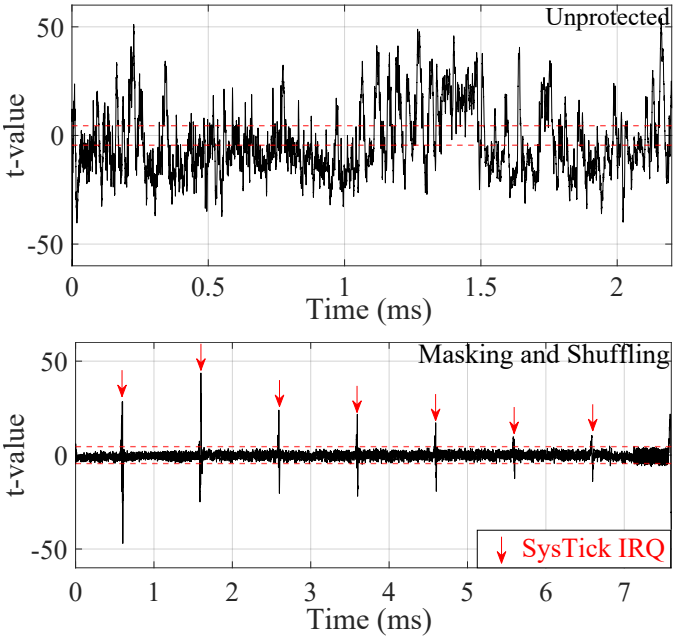

Fig. 1. TVLA evaluation on Cortex-M4 at 16 MHz (1500 fixed vs. 1500 random traces). Unprotected baseline (top): executes in 2.02 ms and exhibits data-dependent leakage with 84.10% of samples above the ± 4.5 threshold (red dashed lines). Protected with masking and shuffling (bottom): cryptographic leakage within the ± 4.5 threshold. The protected TVLA acquisition lasts ≈ 7.6 ms, which is shorter than the full 11.80 ms runtime in Section III-C because TRNG masks are precomputed in SRAM.

On a single 256-coefficient polynomial, the unprotected baseline takes 32 270 cycles (2.02 ms at 16 MHz). Execution shuffling requires 85 959 cycles (5.37 ms, +166%), arithmetic masking requires 132 235 cycles (8.26 ms, +310%), and combined protection requires 188 855 cycles (11.80 ms, +485%). At 170 MHz across 1500 signatures, since countermeasures are applied selectively to the CPA-vulnerable pointwise multiplication rather than to the full NTT as required against single-trace attacks (32%–490% signing overhead in [12]), and because the STM32G474RE includes a 48 MHz hardware TRNG supplying 32 bit masks with minimal CPU stalls, mean signing time increases from 282.20 ms to 300.65 ms (+6.54%).

The latency penalty is larger when ML-DSA is ported to an SoC in a software-only (PS only) setup lacking an hardware TRNG. On the 667 MHz ARM Cortex-A9 PS of a Zynq-7020 SoC, the unprotected baseline completes a signature in 40.64 ms (a $6.94\times$ speedup over the Cortex-M4). Evaluating the protected firmware on the Cortex-A9 with a dummy PRNG (`rand()`) shows that the CPU handles the arithmetic split and index permutation overhead efficiently (44.95 ms, +10.6%).

However, cryptographic security requires extracting physical entropy from a true random source. In the proposed system the entropy is extracted from the thermal noise of the LSB of the on-chip 12 bit Analog-to-Digital Converter (XADC) [23] (427 200 cycles at boot for a 256 bit seed) and expanded in software via a counter-mode SHAKE256 Cryptographically Secure PRNG (CSPRNG) across rejected trials. This continuous Keccak expansion saturates the Cortex-A9 CPU, increasing mean signing latency to 572.44 ms (142.03 ms to 3664.84 ms, +1308%), motivating an HW/SW co-design.

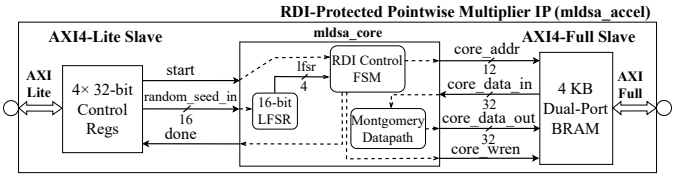

Fig. 2. Architecture of the proposed `mldsa_accel` IP core (100 MHz PL), illustrating the AXI4-Lite control interface, the RDI-protected `mldsa_core` arithmetic datapath, and the AXI4-Full Dual-Port BRAM interface.

IV. Hardware and Software Design and Random Delay Insertion

A. Accelerator Architecture

To overcome the software bottlenecks measured in Section III-C, the pointwise Montgomery multiplication ($\hat{\mathbf{c}} \circ \hat{\mathbf{s}}_1$) is executed using a custom Verilog IP (`mldsa_accel`) synthesized in the PL of the Zynq-7020 SoC at 100 MHz. Rather than synthesizing a full NTT processor, this partitioning isolates only the CPA sensitive operation into the PL, while the 667 MHz Cortex-A9 PS executes the ML-DSA-44 control flow, SHAKE hashing, and rejection checks.

As shown in Fig. 2, the accelerator is partitioned into three modules across a dual-bus AXI4 interface:

- **AXI4-Lite Control Slave:** Maps four 32 bit registers (`slv_reg0`–`slv_reg3`) at address `0x43C00000` for execution handshaking (`start` and `done`) and 16 bit seed injection (`random_seed_in[15:0]`).
- **AXI4-Full Memory Slave:** Infers a 4096 byte Dual-Port BRAM at `0x7AA00000` storing the 256-coefficient 32 bit operands (Op. A, B at `0x000`, `0x400`, Result C at `0x800`, 1 KB each). Port A interfaces with AXI4-Full. Port B connects to `mldsa_core` via a 12 bit byte address bus (`core_addr`), 32 bit data buses (`core_data_in`, `core_data_out`), and write-enable (`core_wren`).
- **Arithmetic Core (`mldsa_core`):** Implements a 13-state Finite State Machine (FSM) and a 4-stage pipelined Montgomery reduction datapath mapping the 32 bit signed multiplications ($q = 8\,380\,417$, $q^{-1} = 58\,728\,449 \bmod 2^{32}$) onto 9 DSP48 slices.

To optimize the 1 KB polynomial transfers between DDR memory and BRAM, a CPU driven AXI4-Full architecture was implemented instead of using a Central Direct Memory Access (CDMA) controller. The technique required configuring the attributes of the page of the translation lookaside buffer of the ARM Memory Management Unit for `0x7AA00000` as Normal Noncacheable memory (`0x14de2`). In this mode the L1/L2 caches are bypassed in hardware and `memcpy` calls are aggregated by the Cortex-A9 load/store unit into 16 AXI3 bursts (16 beats $\times$ 4 bytes). At 100 MHz PL clock, this CPU driven burst approach achieves a latency of 51.81 μs per polynomial, matching CDMA latency (51.85 μs) by eliminating CDMA descriptor setup and cache flushing overheads. In addition, removing the CDMA blocks reduces the PL footprint by 82% (from 4701 to 846 LUTs).

TABLE I
COMPARISON OF ZYNQ-7000 SOC AND ARTIX-7 ML-DSA IMPLEMENTATIONS

Work	Platform	Clk (PS/PL)	HW Scope / Defense	LUT / FF / DSP / BRAM	Sign Time	CPA Sec.
[13][a]	Zynq-7020	533 / 100 MHz	Full NTT / None	2620 / – / 16 / 6	15.60 ms	×
[14][b]	Zynq-7000 / A7	667 / 163 MHz	Full Dilithium / None	13975 / 6845 / 4 / 35	0.91 ms	×
This work (PS Base.)	Zynq-7020	667 / – MHz	None (Unprot. SW)	0 / 0 / 0 / 0	40.64 ms	×
This work (PS Prot.)[c]	Zynq-7020	667 / – MHz	None (SW Mask+Shuf)	0 / 0 / 0 / 0	572.44 ms	✓
This work (IP Core)[d]	**Zynq-7020**	**667 / 100 MHz**	**Pointwise / HW RDI**	**386 / 405 / 9 / 4**	**42.80 ms**	✓
This work (Full PL)[e]	**Zynq-7020**	**667 / 100 MHz**	**Pointwise / HW RDI**	**872 / 862 / 9 / 4**	**42.80 ms**	✓

[a]Dilithium $(k, l) = (5, 4)$ (Impl. II). [b]Full-HW accelerator on Artix-7 / Zynq-7010 (best-case 1st iter. Sign: 607.9×10^3 PS cycles at 667 MHz) [14].
[c]Driven by XADC + SW SHAKE256 CSPRNG. [d]Standalone `mldsa_accel` IP core. [e]Complete PL wrapper. ×: Vulnerable, ✓: Architecturally protected (by-design).

B. Fibonacci Linear Feedback Shift Register Delay Insertion

In its unprotected configuration, `mldsa_core` processes each coefficient in 10 clock cycles. Including the final FSM transition, the polynomial reduction requires a strictly deterministic 2561 cycles (25.61 μs at 100 MHz). As shown in [15], [22], such clock-synchronous, deterministic FPGA execution allows power traces to align across acquisitions, exposing the datapath registers to CPA. To break this alignment while avoiding the area overhead of data masking and its dependence on a CSPRNG, an hardware RDI countermeasure is embedded into the proposed synchronous FSM of `mldsa_core`.

The time domain defense is driven by a maximal length 16 bit Fibonacci LFSR configured with the primitive feedback polynomial $P(x) = x^{16} + x^{14} + x^{13} + x^{11} + 1$. For each polynomial multiplication, the Cortex-A9 PS extracts 2 bytes of thermal noise from the XADC to seed the LFSR (`random_seed_in`), bypassing the SHAKE256 expansion bottleneck. Shifting at 100 MHz, the LFSR feeds a 4 bit delay counter (`delay_cnt`) initialized with `random_seed_in[3:0]` at `start` and updated with `lfsr[7:4]` during the `S_NEXT` state of each coefficient iteration.

Before reading the operands of coefficient $i \in [0, 255]$, the FSM enters a stall state (`S_DUMMY`) for a uniformly distributed duration of 0 to 15 clock cycles ($\mu = 7.5$ cycles/coefficient). During `S_DUMMY`, `core_wren` is de-asserted and registers hold their state, while the 100 MHz clock tree and the shifting LFSR sustain dynamic CMOS switching activity. Combined with the Zynq package low-pass filtering [22], this prevents dummy cycle isolation. In behavioral RTL simulation with test seed `0xA5A5`, the core completed the 256-coefficient reduction in 4673 clock cycles, injecting 2112 dummy cycles (8.25 cycles/coefficient). Because delays are cumulative, the total temporal displacement follows a Gaussian distribution ($\mu = 1920$ clock cycles, $\sigma \approx 73.75$ cycles). Unlike software RDI [16], where dummy instruction loops exhibit distinct power amplitudes that allow wavelet-based removal, this cycle-accurate cumulative jitter propagates through the processing, preventing trace realignment and neutralizing CPA [17].

C. Performance Recovery and State-of-the-Art Comparison

At the IP level (100 MHz PL), enabling RDI increases the mean execution time of a single pointwise multiplication from 51.81 μs to 74.00 μs (+22.19 μs). Across 1500 signatures on the Zynq-7020 SoC (667 MHz PS, 100 MHz PL), the RDI protected HW/SW co-design achieves a mean ML-DSA-44 signing latency of 42.80 ms (18.73 ms to 211.01 ms under rejection variance). Compared to the 572.44 ms required by the software masked implementation on the Cortex-A9 PS, the proposed architecture provides a $13.4\times$ speedup, reducing the security latency overhead from +1308% to +5.31% relative to the unprotected software baseline (40.64 ms).

Tab. I summarizes the Zynq-7020 resource utilization, including LUTs, Flip-Flops (FFs), DSPs, and BRAMs, alongside the operating frequencies, and signing latencies. The design runs at 100 MHz with a Slack of +0.765 ns, dissipating 33 mW of dynamic PL power. The RDI logic adds only 26 LUTs and 20 FFs over the unprotected IP (360 LUTs, 385 FFs). While [13] and [14] achieve lower signing latencies by accelerating the full NTT or the complete Dilithium (standardized as ML-DSA) protocol in hardware without SCA protections, unshielded FPGA reduction datapaths remain vulnerable to CPA key recovery within 10 000 traces [15]. Conversely, offloading only the CPA vulnerable pointwise multiplication at 100 MHz reduces PL LUT occupation by 66.7% compared to [13] (872 vs. 2620 LUTs) and by 93.8% compared to [14] (13 975 LUTs), providing architectural protection against CPA with a +5.31% signing latency overhead.

V. CONCLUSION

This paper evaluated ML-DSA-44 physical vulnerabilities and proposed an SCA protected HW/SW co-design on a Zynq-7020 SoC.

Baseline TVLA on a Cortex-M4 confirmed severe leakage during pointwise Montgomery multiplication, with 84.10% of samples exceeding the ± 4.5 threshold. While software masking and shuffling successfully suppressed this leakage, they resulted in a +485% polynomial overhead. Furthermore, porting these defenses to the 667 MHz Cortex-A9 PS inflated total signing latency to 572.44 ms (+1308%) due to continuous SHAKE256 CSPRNG expansion.

To overcome this bottleneck, the co-design replaces software masking and shuffling by offloading the vulnerable operation to a 100 MHz AXI4-Full PL accelerator. By integrating a 16 bit LFSR for Random Delay Insertion, the architecture injects cumulative Gaussian jitter to prevent CPA trace alignment. This hardware protection requires only 26 LUTs and 20 FFs, obtaining mean signing latency of 42.80 ms that is close to the 40.64 ms of the unprotected software implementation.

REFERENCES

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed., Global Edition. Harlow, U.K.: Pearson Education, 2023.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[3] National Institute of Standards and Technology, "FIPS 204: Module-lattice-based digital signature standard," U.S. Dept. of Commerce, Gaithersburg, MD, USA, Tech. Rep., Aug. 2024.

[4] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Dec. 2009, pp. 598–616.

[5] National Institute of Standards and Technology, "FIPS 202: SHA-3 standard: Permutation-based hash and extendable-output functions," U.S. Dept. of Commerce, Gaithersburg, MD, USA, Tech. Rep., Aug. 2015.

[6] A. Satriawan, I. Syafalni, R. Mareta, I. Anshori, W. Shalannanda, and A. Barra, "Conceptual review on number theoretic transform and comprehensive review on its implementations," *IEEE Access*, vol. 11, pp. 70288–70316, 2023.

[7] PQ-CRYSTALS Team, "CRYSTALS-Dilithium: Official reference implementation," GitHub repository. [Online]. Available: https://github.com/pq-crystals/dilithium (accessed Oct. 2024).

[8] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer, 2007.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES)*, Aug. 2004, pp. 16–29.

[10] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing, "An efficient non-profiled side-channel attack on the CRYSTALS-Dilithium post-quantum signature," in *Proc. IEEE 39th Int. Conf. Comput. Design (ICCD)*, Oct. 2021, pp. 583–590.

[11] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque, "Masking Dilithium: Efficient implementation and side-channel evaluation," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur. (ACNS)*, Jun. 2019, pp. 344–362.

[12] P. Ravi, R. Poussier, S. Bhasin, and A. Chattopadhyay, "On configurable SCA countermeasures against single trace attacks for the NTT: A performance evaluation study over Kyber and Dilithium on the ARM Cortex-M4," in *Proc. Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng. (SPACE)*, Dec. 2020, pp. 123–146.

[13] Z. Zhou, D. He, Z. Liu, M. Luo, and K.-K. R. Choo, "A software/hardware co-design of Crystals-Dilithium signature scheme," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, no. 2, pp. 1–21, Jun. 2021, Art. no. 11.

[14] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, "Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 8, pp. 3234–3243, Aug. 2023.

[15] M. Karabulut and R. Azarderakhsh, "Efficient CPA attack on hardware implementation of ML-DSA in post-quantum root of trust," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2025, pp. 392–402.

[16] F. Zhang, X. Dong, B. Yang, Y. Zhou, and K. Ren, "A systematic evaluation of wavelet-based attack framework on random delay countermeasures," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1407–1422, 2020.

[17] Y. Lu, M. P. O'Neill, and J. V. McCanny, "FPGA implementation and analysis of random delay insertion countermeasure against DPA," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2008, pp. 201–208.

[18] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *Proc. NIST Non-Invasive Attack Testing Workshop (NIAT)*, Sep. 2011, pp. 115–136.

[19] A. Jayasena, E. Andrews, and P. Mishra, "TVLA*: Test vector leakage assessment on hardware implementations of asymmetric cryptography algorithms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 9, pp. 1269–1279, Sep. 2023.

[20] *UM2505 User Manual: STM32G4 Nucleo-64 Boards (MB1367)*, Rev. 4, STMicroelectronics, Feb. 2021.

[21] *RM0440 Reference Manual: STM32G4 Series Advanced Arm-Based 32-Bit MCUs*, Rev. 8, STMicroelectronics, Feb. 2024.

[22] P. Socha, J. Brejník, and M. Bartík, "Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board," in *Proc. 7th Mediterranean Conf. Embed. Comput. (MECO)*, Jun. 2018, pp. 1–4.

[23] *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)*, Rev. 1.11, Advanced Micro Devices (AMD), 2022.