Implementation Attacks on ML-KEM: Analysis, Practical Assessment, and Recommendations

Vitor Hugo Galhardo Moia, Kevin Pires Novaes, Vinicius de Paula Rossetto

Departamento de Computação Embarcada
Instituto de Pesquisas Eldorado – Campinas, SP – Brazil

{vitor.moia, kevin.novaes, vinicius.rossetto}@eldorado.org.br

Abstract. *ML-KEM is the NIST-standardized key encapsulation mechanism derived from CRYSTALS-Kyber, a post-quantum cryptographic algorithm designed to provide security against both classical and quantum computers. However, implementation attacks threaten the adoption of this algorithm, specially in constrained environments (e.g., IoT, OT/ICS, and other embedded applications). In this work, we present an analysis of Side-Channel and Fault Injection attacks, demonstrating which algorithm operations are targeted by adversaries, practical experiments to detect side-channel leakage, adversary capabilities required to perform attacks, and recommendations for securing implementations.*

1. Introduction

As quantum computers become increasingly a reality, so does the threat they pose to the security world. Although a cryptographically relevant quantum computer - i.e., a quantum computer capable of breaking current cryptographic algorithms - does not exist, advances in recent years suggest that this day may be closer than we think. Nevertheless, a threat is already looming over the digital world: the *"harvest now, decrypt later"* strategy, which is causing concern among companies and governments regarding their long-term security.

Although Post-Quantum Cryptography (PQC) provides a solution to this problem, migrating from traditional cryptography to PQC presents significant challenges. NIST has conducted a multi-round selection process to standardize algorithms designed to replace RSA and ECC [Alagic et al. 2022]. After several rounds of revisions and feedback, NIST has selected some algorithms to become standards for key encapsulation and digital signature. Among the selected algorithms so far is the Module-Lattice Key Encapsulation Mechanism (ML-KEM), which offers a balance between security and performance. However, while PQC algorithms are designed to resist attacks enabled by a quantum computer using Shor's algorithm, their real-world security depends not only on theoretical robustness, but also on the resilience of their implementation.

Many implementation attacks targeting ML-KEM and other NIST selected candidates were proposed in recent years. Surveys and literature reviews [Ravi et al. 2024a, Roy et al. 2024] describe how attacks are carried out, exploiting unintended information leakage (Side-Channel Attack (SCA)) or inducing computational errors (Fault Injection Attack (FIA)) rather than weaknesses in the underlying mathematics to reveal parts or the full secret key used to protect data. Despite considerable research, the literature remains vague about the real-world practicality of such attacks, the capabilities required by adversaries, and the defenses that should be employed to mitigate them. In this work, besides analyzing and characterizing ML-KEM against SCA and FIA attacks, we also discuss the

conditions and requirements for these attacks, demonstrate a practical experiment on an ARM Cortex-M4 microcontroller to identify side-channel leakage as a possible countermeasure, and examine additional techniques for securing real-world applications. As an additional contribution, we release the dataset used in our experiments to support reproducibility and further research. The dataset is available at: `https://huggingface.co/datasets/ai-eldorado/ML-KEM-SideChannel-Traces`.

2. The NIST Selected PQC KEM Algorithm

ML-KEM is the NIST-standardized version of the CRYSTALS-Kyber key encapsulation mechanism, a lattice-based cryptographic algorithm selected in NIST PQC competition [Alagic et al. 2022] and published as FIPS 203 [Alagic et al. 2024]. ML-KEM combines a Chosen-Plaintext Attack (CPA) secure Public-Key Encryption (PKE) scheme with a Chosen Ciphertext Attack (CCA) secure KEM construction. The PKE algorithm is based on the Module Learning with Errors (M-LWE) problem and provides semantic security against CPA, while CCA security is achieved by the Fujisaki–Okamoto transform. For a detailed explanation about the algorithm, see [Alagic et al. 2024].

ML-KEM consists of three main parts: **Key Generation**, which creates the public and private keys; **Encapsulation**, which generates a shared secret and its corresponding ciphertext using the public key; and **Decapsulation**, which recovers the shared secret from the ciphertext using the private key. Its primary use is for post-quantum key establishment in secure communications. In this use case, the algorithm can be the target of attacks to break its security to obtain the confidential data. Since attacking the theoretical foundation of the algorithm has not yet proven successful, adversaries instead target its implementation, where successful attacks have been achieved via SCA or FIA.

During **Key Generation**, adversaries can either steal the secret key directly or manipulate the seeds to produce predictable randomness and, thus, facilitate key recovery [Kannwischer et al. 2020, Ravi et al. 2023, Ravi et al. 2019]. Notice that the adversary has only one chance to perform the attack in this part of the algorithm (assuming long-term rather than ephemeral keys). During **Encapsulation**, adversaries can recover the shared secret through process leakage [Sim et al. 2020] or by manipulating the random values [Ravi et al. 2023]. Since the shared secret is used only for a single communication session and then discarded, the impact of an attack and the amount of data available to an adversary are both limited.

The **Decapsulation** process is the primary target of adversaries because it operates on the secret key. Consequently, any leakage during this process may enable adversaries to recover the secret key, allowing them to decrypt protected communications and impersonate the legitimate key holder. In this work, we focus our analysis on this process.

3. Threats to ML-KEM Decapsulation Process

We conduct a literature review on implementation attacks targeting ML-KEM decapsulation process and summarize our insights in Algorithm 1. This alg. outlines a simplified version of the process and maps state-of-the-art SCA/FIA research to its key underlying operations. For conciseness, we do not repeatedly map attacks to recurring operations.

The attacks explore side-channel leakage or induce faults during cryptographic operation. In the specific case of leakage detection, some techniques can be employed to

Algorithm 1 ML-KEM Decapsulation Process (Simplified Version).

Input: Secret key $sk = (s, pk, H(pk), z)$, ciphertext $c = (c_1, c_2)$
Output: Shared secret K

$m' \leftarrow \text{PKE.DECRYPT}(s, c)$
$(K', r) \leftarrow \text{G}(m' \parallel H(pk))$ ▷ In FIPS 203, G is defined as SHA3-512 and H as SHA3-256
$\bar{K} \leftarrow \text{J}(z \parallel c)$ ▷ In FIPS 203, J is defined as SHAKE256
$c' \leftarrow \text{PKE.ENCRYPT}(pk, m', r)$
if $c \neq c'$ **then** ▷ Attacks: [Hermelink et al. 2021, Xagawa et al. 2021, Qin et al. 2021]
    $K' \leftarrow \bar{K}$
end if
return K'

function PKE.DECRYPT($s, c = (c_1, c_2)$)
    $u \leftarrow \text{DECOMPRESS}(\text{DECODECIPHERTEXT}(c_1))$
    $v \leftarrow \text{DECOMPRESS}(\text{DECODECIPHERTEXT}(c_2))$
    $u' \leftarrow \text{NTT}(u)$
    $t \leftarrow \text{POINTWISEMULTIPLY}(s, u')$ ▷ Attacks: [Zhao et al. 2023, Meng et al. 2026]
    $t \leftarrow \text{INTT}(t)$ ▷ Attacks: [Xu et al. 2021, Primas et al. 2017]
    $m' \leftarrow v - t$ ▷ Attacks: [Zhao et al. 2023]
    $m \leftarrow \text{DECODEMESSAGE}(m')$ ▷ Attacks: [Wang and Dubrova 2023, Ravi et al. 2021]
end function

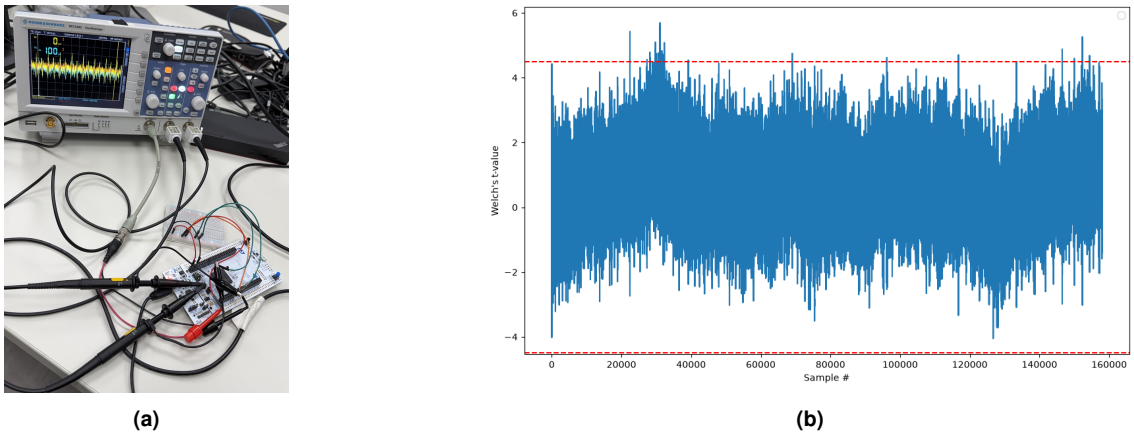
identify potential implementation leakages and guide the adoption of appropriate countermeasures. This can represent a first step toward developing secure solutions. In the next section, we perform experiments on ML-KEM to demonstrate the testing procedure.

4. Methodology for Testing Side-Channel Leakage

Analyzing an algorithm implementation for side-channel leakage can be done via the **Test Vector Leakage Assessment (TVLA)** [Gilbert Goodwill et al. 2011]. This is a methodology for detecting information leakage in cryptographic implementations through side-channel measurements. In general, this methodology requires that one collect many traces under two different conditions (e.g., fixed vs. random inputs), compute a *Welch's t-test* at every sample point, and analyze the results. If the absolute *t-value* exceeds a threshold (commonly $|t| > 4.5$), the implementation is considered to exhibit statistically detectable leakage. Notice that the TVLA is a leakage detection test, not a key-recovery attack. Passing the test does not prove an implementation is secure, and failing TVLA does not necessarily mean an attacker can recover the secret key. It indicates whether statistically significant leakage is detectable under the tested conditions.

For testing ML-KEM decapsulation, we consider as variables for TVLA the **secret key** and **ciphertext**. Our fixed set consists of multiple traces of the decapsulation process using the same secret key and the same ciphertext, whereas the random set also uses the same secret key but varies the ciphertext by generating a new one in each iteration.

We use for the experiments the *Rohde & Schwarz RTC1002 100Mhz* digital oscilloscope to capture power traces (using a shunt resistor) from a *STM32 Nucleo-L4R5ZI* board (ARM Cortex-M4), executing a Software C implementation of ML-KEM from the open-source PQM4 library [Kannwischer et al. 2019] (commit: *a24bb4b*), compiled using *arm-none-eabi-gcc*. For the experiments, we choose ML-KEM-768 (k=3) version and created the two sets for TVLA containing 1000 traces each (i.e., decapsulations). We develop a Python script to automate the tests with the oscilloscope and target board, used to send commands and receive power readings. The setup is shown in Figure 1a.



(a) (b)

Figure 1. (a) Equipment used for the experiments. (b) TVLA on ML-KEM-768 *DecodeMessage* function (*"poly_tomsg"*) from PQM4 (commit: *a24bb4b*).

5. Leakage Detection Procedure

[Teague 2022, Ravi et al. 2021] investigated the side-channel leakage of the decapsulation process in the pre-standardized version of ML-KEM (the Kyber NIST submission version) implemented in PQM4 [Kannwischer et al. 2019]. The study targeted the *DecodeMessage* function (line 16 of Algorithm 1) to recover the secret key. The *poly_tomsg()* function is responsible to perform this operation in PQM4 implementation. Since then, the code has undergone numerous changes, including modifications related to the standardization process. In addition, the target function of the attacks has also evolved.

In this work, we test the *poly_tomsg()* to verify whether the leakage persists in the recent and post-standardization implementation of the PQM4 library. To this end, we apply the TVLA on this particular operation. The results are presented in Figure 1b.

6. Results and Discussion

The experimental results provide evidence of first-order leakage, with several sample points exceeding the TVLA threshold (red line). Although the observed leakage is not exceptionally strong, the results indicate statistically significant differences between the two groups. However, TVLA alone does not demonstrate that the secret key can be recovered. Nevertheless, a more capable adversary may still compromise the current ML-KEM implementation in PQM4, as shown in [Wang and Dubrova 2023], where a power-based SCA attack recovered the secret key using manipulated ciphertexts and a cloned device for deep-learning training. In our setup, similar attack scenarios could be explored, including adversarial ciphertext manipulation to amplify leakage patterns during decapsulation and the use of collected traces for key recovery. This will be investigated in future work.

Based on the literature review, we conclude that implementation attacks against ML-KEM are difficult in practice. However, sufficiently capable adversaries may still be able to compromise implementations. SCA and FIA require special access to the target, specialized equipment, and, in most cases, access to a cloned device. A typical attack assumes the adversary has physical access to the target and can measure its power consumption or electromagnetic (EM) emissions during cryptographic operations. Besides, many attacks require profiling [Primas et al. 2017, Meng et al. 2026], in which the adver-

sary constructs a template using leakage from the target or a clone and uses the collected data to build a statistical or artificial intelligence model that supports the attack.

Most implementation attacks are intrusive. They require adversaries to input ciphertexts into the target to trigger decryption and capture side-channel leaks. This capability can be further divided based on the input provided by the adversary: (a) legitimate ciphertext, (b) random data (including EM or voltage faults), or (c) handcrafted ciphertext. Most attacks reported in the literature employ option (c). In the case of FIA, another limitation is the requirement to accurately position the equipment and inject the fault at the precise moment. Therefore, a high degree of precision is essential for successful fault injection [Zhao et al. 2023, Hermelink et al. 2021]. Another important requirement is the key setup scheme. There are two versions, static, with the same secret key being reused for multiple operations, and ephemeral, where a new secret key is created for every key exchange. This is an important requirement, since many attacks could be avoided in the restricted ephemeral scheme, only single-trace attacks would succeed.

Finally, SCA and FIA require special equipment. Setups found in the literature are composed of oscilloscope, low-noise amplifier, shunt resistor, pc station, and cloned device for power measurement, or a probe to collect EM emission instead of the shunt resistor. Some works use the ChipWhisperer toolchain to collect and perform the experiments instead of an oscilloscope. In the case of FIA, equipment to induce a fault, such as EM pulse generator is necessary. Observe that an adversary requires equivalent equipment to perform attacks.

7. Recommendations

Although the conditions to perform SCA and FIA attacks are not as trivial as for typical network attacks, where adversaries have fewer restrictions, it is paramount to be prepared and consider them in any threat modeling of real-world applications that may adopt PQC.

Attacks have been demonstrated against both unprotected and protected [Ravi et al. 2024b, Primas et al. 2017] implementations of ML-KEM. A single countermeasure is insufficient against adversaries with unlimited capabilities. Defending against SCA and FIA attacks requires a defense-in-depth approach.

To defend against SCA, besides applying tests such as the TVLA as suggested in this paper, we must also limit the adversary capabilities. As proposed in [Ravi et al. 2024a], detecting and avoiding the processing of handcrafted ciphertexts could avoid leakage of side-channel information by identifying ciphertexts composed of polynomials with most coefficients zeroed (this is one of the strategies used for adversaries). Masking and shuffling are the countermeasures most studied for protecting implementations. Masking refers to the split of a sensitive data or operations into multiple shares, which are combined only in the last step [Wang et al. 2023, Roy et al. 2024]. Shuffling [Roy et al. 2024] randomizes the order in which independent operations are performed, for instance, during the process of polynomials' coefficients. Noise Injection could also be another countermeasure, where noise is injected into power traces to make it more challenging to extract meaningful information [Roy et al. 2024]. On the other hand, countermeasures against FIA include redundancy, with the recomputation or duplication of critical operations, checksums/invariants, and error-detecting codes [Renita et al. 2025]. [Prokop and Peßl 2021] recommends CFI (control-flow in-

tegrity) to ensure that a sequence of executed instructions is correct and no fault has compromised any of these operations, leaving the algorithm vulnerable.

Finally, employing an ephemeral key scheme can help mitigate both types of attacks. Although this approach may not be feasible in some practical scenarios, implementing at least a key refresh mechanism after N operations with the key is necessary, where N should be chosen based on an analysis of attacks [Meng et al. 2026].

8. Conclusion

In this work, we present an analysis of implementation attacks against the PQC algorithm ML-KEM. We analyzed attacks from the literature and how they can affect different operations of the algorithm. We demonstrated a practical experiment showing how to detect leakage from power consumption during cryptographic operations using the TVLA methodology, and examined the requirements for adversaries to carry out attacks and appropriate countermeasures. Future research should complement the experiments performed by varying the conditions of TVLA (using handcrafted ciphertexts) and analyzing other algorithm operations, including the key generation and encapsulation processes.

9. Acknowledgment

This work has been totally funded by the project PQC Optimization for Critical Infrastructures supported by Centro Integrado de Segurança em Sistemas Avançados (CISSA), with financial resources from the PPI IoT/Manufatura 4.0 of the MCTI grant number 08/2-24, signed with EMBRAPII.

References

Alagic, G., Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.-K., Miller, C., et al. (2022). Status report on the third round of the nist post-quantum cryptography standardization process. *NIST*.

Alagic, G., Dang, Q., Moody, D., Robinson, A., Silberg, H., Smith-Tone, D., et al. (2024). Module-lattice-based key-encapsulation mechanism standard. *NIST Pubs*.

Gilbert Goodwill, B. J., Jaffe, J., Rohatgi, P., et al. (2011). A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136.

Hermelink, J., Pessl, P., and Pöppelmann, T. (2021). Fault-enabled chosen-ciphertext attacks on kyber. In *Inter. Conf. on Cryptology in India*, pages 311–334. Springer.

Kannwischer, M. J., Pessl, P., and Primas, R. (2020). Single-trace attacks on keccak. *Cryptology ePrint Archive*.

Kannwischer, M. J., Rijneveld, J., Schwabe, P., and Stoffelen, K. (2019). Pqm4: Post-quantum crypto library for the arm cortex-m4. Accessed: July. 12, 2026.

Meng, Y., Wang, B., Xing, Q., Wang, X., Zhou, D., and Liu, L. (2026). Deep learning based side-channel attack on polynomial multiplication in post-quantum cryptography. *ACM TECS*, 25(4):1–28.

Primas, R., Pessl, P., and Mangard, S. (2017). Single-trace side-channel attacks on masked lattice-based encryption. In *CHES*, pages 513–533. Springer.

Prokop, L. and Peßl, P. (2021). Fault attacks on cca-secure lattice kems. *TCHES*, 2021(2):37–60.

Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., and Ding, J. (2021). A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems. In *ASIACRYPT*, pages 92–121. Springer.

Ravi, P., Bhasin, S., Roy, S. S., and Chattopadhyay, A. (2021). On exploiting message leakage in (few) nist pqc candidates for practical message recovery attacks. *IEEE TIFS*, 17:684–699.

Ravi, P., Chattopadhyay, A., D'Anvers, J. P., and Baksi, A. (2024a). Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. *ACM TECS*, 23(2):1–54.

Ravi, P., Paiva, T., Jap, D., D'anvers, J.-P., and Bhasin, S. (2024b). Defeating low-cost countermeasures against side-channel attacks in lattice-based encryption. *TCHES*.

Ravi, P., Roy, D. B., Bhasin, S., Chattopadhyay, A., and Mukhopadhyay, D. (2019). Number "not used" once-practical fault attack on pqm4 implementations of nist candidates. In *COSADE*, pages 232–250. Springer.

Ravi, P., Yang, B., Bhasin, S., Zhang, F., and Chattopadhyay, A. (2023). Fiddling the twiddle constants-fault injection analysis of the number theoretic transform. *TCHES*, 2023(2):447–481.

Renita, J., Annadurai, S., et al. (2025). Side-channel and fault attacks on ml-kem: A survey of vulnerabilities and countermeasures. In *WISPNET*, pages 1–6. IEEE.

Roy, K. S., SL, S. D., Mishra, T. K., Hassan, M., and Hazarika, R. A. (2024). Analyzing crystals-kyber's susceptibility to side channel attacks: An empirical exploration.

Sim, B.-Y., Kwon, J., Lee, J., Kim, I.-J., Lee, T.-H., Han, J., Yoon, H., Cho, J., and Han, D.-G. (2020). Single-trace attacks on message encoding in lattice-based kems. *IEEE Access*, 8:183175–183191.

Teague, T. (2022). Side-channel analysis on post-quantum cryptography algorithms.

Wang, R., Brisfors, M., and Dubrova, E. (2023). A side-channel attack on a bitsliced higher-order masked crystals-kyber implementation. *Cryptology ePrint Archive*.

Wang, R. and Dubrova, E. (2023). A side-channel secret key recovery attack on crystals-kyber using k chosen ciphertexts. In *C2SI*, pages 109–128. Springer.

Xagawa, K., Ito, A., Ueno, R., Takahashi, J., and Homma, N. (2021). Fault-injection attacks against nist's post-quantum cryptography round 3 kem candidates. In *ASIACRYPT*, pages 33–61. Springer.

Xu, Z., Pemberton, O., Roy, S. S., Oswald, D., Yao, W., and Zheng, Z. (2021). Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Transactions on Computers*, 71(9):2163–2176.

Zhao, Y., Pan, S., Ma, H., Gao, Y., Song, X., He, J., and Jin, Y. (2023). Side channel security oriented evaluation and protection on hardware implementations of kyber. *IEEE TCAS-I*, 70(12):5025–5035.