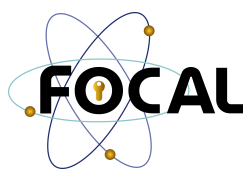
Full Secret Key Recovery of First-Order Masked ML-KEM implementation using Multiple Chosen-ciphertext Attacks

Souhayl BEN EL HAJ SOULAMI, Dr. Yann CONNAN,
Pf. Sylvain DUQUESNE

CASCADE 2026, *Regensburg, Germany*
March 30, 2026



Introduction

- ML-KEM selected by the NIST for standardisation
- Implementations can be vulnerable to side-channel attacks
- Unprofiled full secret key recovery attack on masked ML-KEM

Table of content

General description

- Lattice-based Key Exchange mechanism
- Hard Problem : Module-Learning-with-Error (MLWE)
- Elements (keys, ciphertexts etc...) are represented as polynomials or vectors of polynomials in $\mathbb{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ the set of polynomials of degree $N-1$ and of coefficients in $\mathbb{Z}_q$
- Polynomial multiplications are performed using `NTT`.

Secret key structure

- The secret is a vector of polynomials of coefficients with small values in $\mathscr{S} = [\![-\eta_1 ; \eta_1]\!]$ ($\eta_1 \in \{2, 3\}$)
- Coefficients follow $B(\frac{1}{2}, 2\eta_1)$ centered on 0.
- There are $256 \times k$ secret coefficients with $k \in \{2, 3, 4\}$
- Objective: recover all secret coefficients

Table: Distribution law for secret values when $\eta_1 = 2$ (`MLKEM768`)

i	-2	-1	0	1	2
$P(s = i)$	0.0625	0.25	0.3725	0.25	0.0625

Protocol description

Party 1		*Party* 2
$(pk, sk) \leftarrow \texttt{KEM.Keygen}()$	$\xrightarrow{\quad pk \quad}$	$\texttt{KEM.Encaps}(pk)$:
		$m' \leftarrow \mathcal{U}(\{0,1\}^{256})$
		$m \leftarrow \mathcal{H}(m')$
		$(K, r) = \mathcal{G}(m, \mathcal{H}(pk))$
	$\xleftarrow{\quad c \quad}$	$c = \texttt{Encrypt}(pk, m, r)$
$\texttt{KEM.Decaps}(sk, c)$:		$K = \texttt{KDF}(K, \mathcal{H}(c))$
$m' = \texttt{Decrypt}(sk, c)$		
$(K', r') = \mathcal{G}(m', \mathcal{H}(pk))$		
$c' = \texttt{Encrypt}(pk, m', r')$		
If $c = c'$ then $K = \texttt{KDF}(K', \mathcal{H}(c))$		
Else $K = \texttt{KDF}(z, \mathcal{H}(c))$		

Figure: ML-KEM shared key establishment protocol

Key decryption

Algorithm $\texttt{MLKEM.CPA.PKE.Dec}(s, c)$ (simplified)

1: $\mathbf{u} = \text{Decompress}_q(\text{Decode}_{d_u}(c_1), d_u)$ ▷ $\mathbf{u} \in \mathbb{R}_q^k$
2: $v = \text{Decompress}_q(\text{Decode}_{d_v}(c_2), d_v)$ ▷ $\mathbf{v} \in \mathbb{R}_q$
3: $\texttt{NTT}(\mathbf{s}) = \text{Decode}_{12}(s)$
4: $m = \text{Encode}_1(\text{Compress}_q(v - \texttt{INTT}(\texttt{NTT}(\mathbf{s}) \circ \texttt{NTT}(\mathbf{u}), 1))$ ▷ $m \in \mathcal{U}(\{0,1\}^{256})$
5: **return** m

- Since the secret key is manipulated in clear, it is vulnerable to attacks.

Masking countermeasure

Let $x \in \mathbb{Z}_q$, the idea is to split the variable to multiples shares and compute the algorithm in the shares instead.

- The sharing is refreshed at each execution with a random sampling.
- The masking can be either arithmetic or Boolean.
- The masking needs to be compatible with the operations we make.
- The computations dependent on the secret key should be masked throughout the entire algorithm.
- The strength of the masking depends on the number of shares used.

SECURE-IC THE SECURITY SCIENCE COMPANY

UNIVERSITÉ DE RENNES 1

Valeo

FOCAL

Masked decryption

In ML-KEM, we start with the arithmetic masking, so we attack at this operation.

- $\mathbf{X}_1 \leftarrow \mathcal{U}(\mathbb{R}_q^k)$
- $\mathbf{X}_2 = \mathtt{NTT}(\mathbf{s}) - \mathbf{X}_1$
- $\mathtt{INTT}(X_1 \circ \mathtt{NTT}(\mathbf{u}))$ and $\mathtt{INTT}(X_2 \circ \mathtt{NTT}(\mathbf{u}))$ $\triangleright\ \mathbf{s}^T \circ \mathbf{u}$

Adversary model

- We assume that the attacker has access to the Device Under Analysis and can run the `MLKEM.CCA.KEM.Decaps()` decapsulation algorithm of `ML-KEM`.
- The attacker can run the decapsulation function with the ciphertext of their choice as many times as they wish.
- The attacker is able to observe the power consumption of two intermediate variables during the execution of the algorithm.

Leakage model

Let x a variable. The leakage associated to this variable follows classically the Hamming Weight model:

$$l(x) = \omega(x) + n^{(t)}$$

- $\omega(x)$ is the Hamming weight of x.
- $n^{(t)}$ is the noise, we assume $n^{(t)} \leftarrow \mathcal{N}(0, \sigma)$.
- Valid assumption for microcontroler `STMF303` where $\sigma = 0.5$.

Distribution of $(\omega(x_1), \omega(x_2))$

- We first focus on single coefficient s of the secret key.
- The attacker is able to measure the consumption of $(\omega(x_1), \omega(x_2))$.
- It is possible to build the theoretical distribution of $(\omega(x_1), \omega(x_2))$ for all possible couples of values $x_1, x_2 \in \mathbb{Z}_q = 3,329$ and this for every value of s. This is only possible because we have small values of s and q.

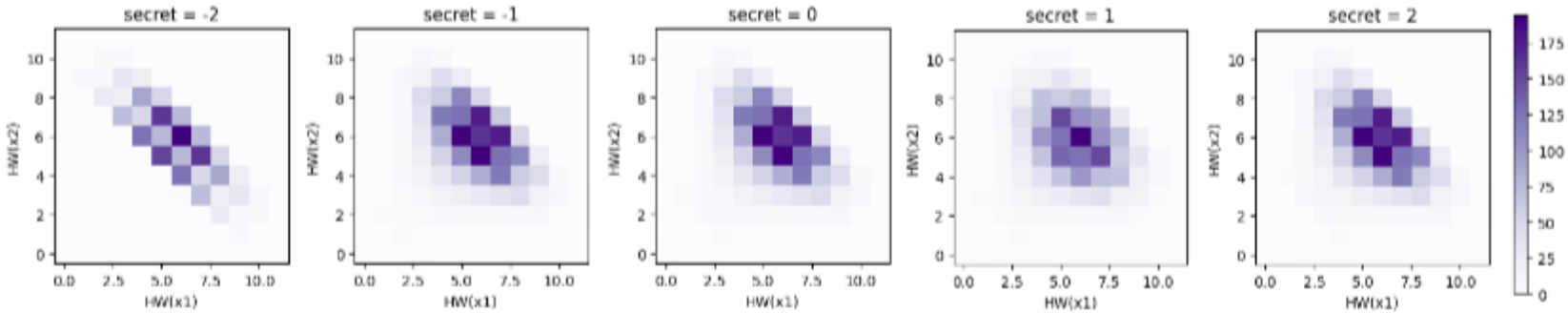


Fig. 3. Compound distribution $(\omega(x_1), \omega(x_2))_s$

- The distribution -2 and 1 are clearly distinct from the rest.

Product Leakage Distinguisher

- Let $l_1^{(t)}$ and $l_2^{(t)}$ be the leakages resulting from the manipulation of the first and the second shares at the execution number t and $l^{(t)} = l_1^{(t)} \cdot l_2^{(t)}$.
- While $l_1^{(t)}$ and $l_2^{(t)}$ are statistically independent from s, the combination of them is dependent on the secret.
- We use the Product Leakage Distinguisher as following:

$$\lim_{T\to\infty} \sum_{t=0}^{T} \frac{l^{(t)}}{T} = \mathbb{E}_s[\omega(X_1)\cdot\omega(X_2)] = \frac{\sum_{x_1\in\mathbb{Z}_q} \omega(x_1)\cdot\omega(s-x_1)}{q}$$

Therefore, $\sum_{t=0}^{T} \frac{l^{(t)}}{T}$ converges towards $\mathbb{E}_s[\omega(X_1)\cdot\omega(X_2)]$ when we increase the number of traces.

Recovering -1 and 2

- $\sum_{t=0}^{T} \frac{l^{(t)}}{T}$ distinguisher allows to distinguish -1 and 2 if provided with enough traces with noise taken into account.
- Recovering -1 and 2 isn't enough because the effort using BKZ remains substantial. (31% of the secret)

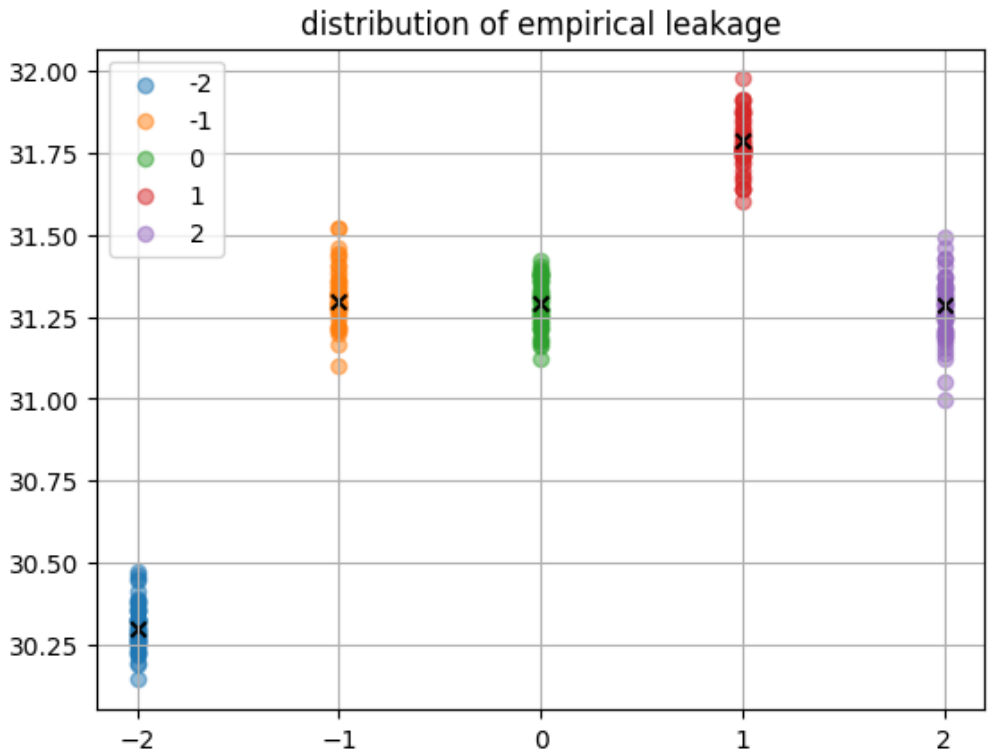


Figure: Empirical leakage of different secret values $T = 40{,}000$ and $\sigma = 0.5$

Improvement of this distinguisher

- We can choose ciphertexts to get other values of $a \in Z_q$ s.t.

$$as = \underbrace{ax_1}_{HW(ax_1)} + \underbrace{ax_2}_{HW(ax_2)}$$

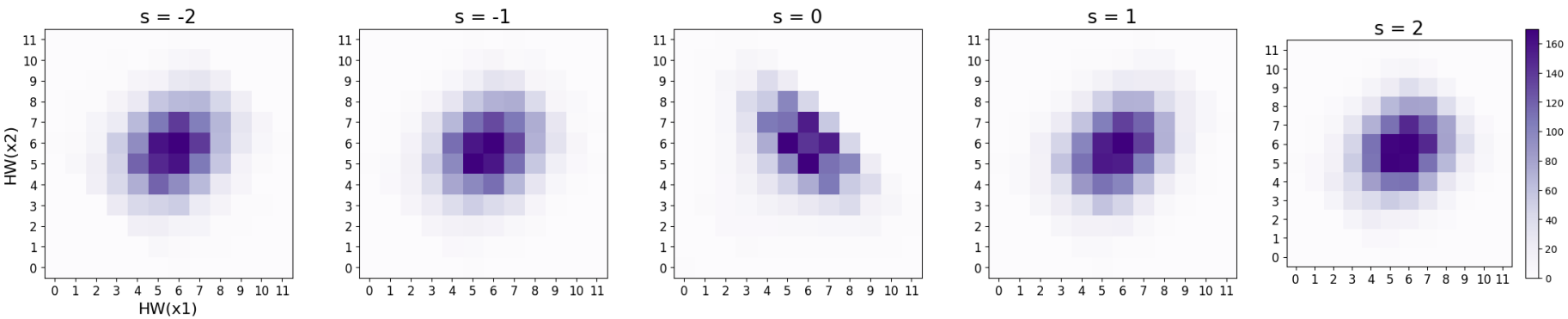


Figure: Compound distribution $(\omega(a \cdot x_1), \omega(a \cdot x_2))_s$ for $a = 1,109$

Multidimensional Product Leakage Distinguisher

- The distributions change according to the value of a as well. For $a = 1109$ for example, the distribution of 0 becomes apparent.
- We can continue so forth to recover -1 and 2. However, we can reduce the dimension by wisely choosing the values of a.

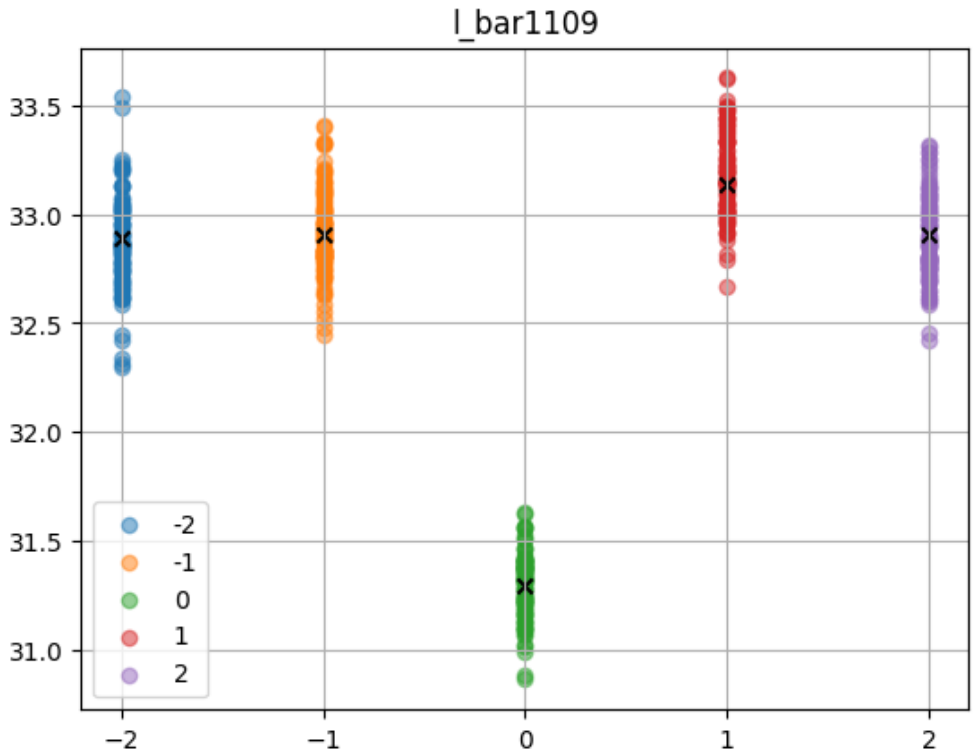
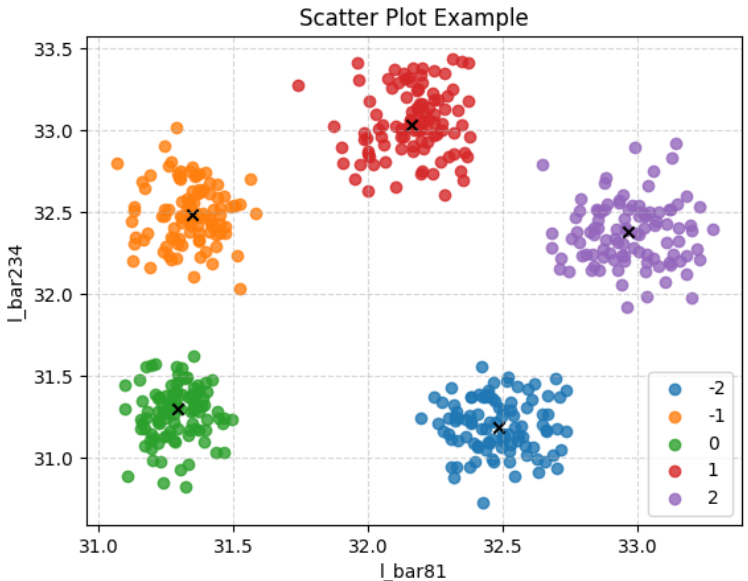


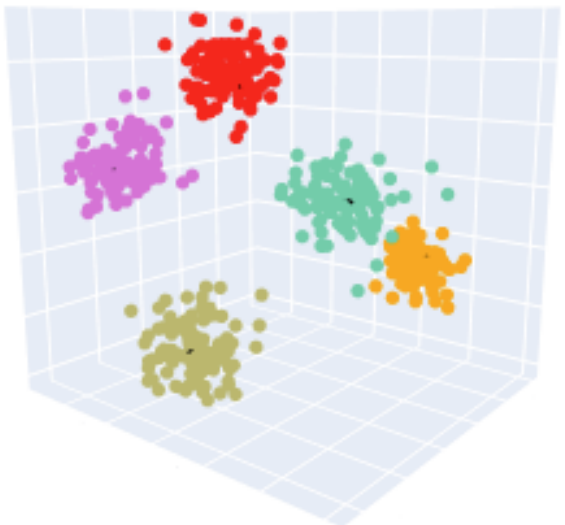
Figure: Compound distribution of $(\omega(a \cdot x_1), \omega(a \cdot x_2))_s$ for $a = 1109$

Leakage combination

- We compute $\mathbb{E}_s[\omega(a \cdot X_1) \cdot \omega(a \cdot (s - X_1))]$ for different couples of distinguishers and select the best for each dimension.



(a) Two dimensional representation with 2D-filter (81, 234)



(b) Three dimensional representation with 3D-filter (137, 322, 2672)

Figure: Representation of distributions in 2 and 3 dimensions

Advantages of the method

- New source of leakage to feed to the PLD.
- Does not need any sort of profiling.
- Parallelisable: we can attack 256 coefficients with the same set of traces.
- Extends to algorithms with small number of possible secrets and small fields.

Experimental results

- To reach a success rate of 95% on the entire algorithm, it is necessary to have a success rate of 99.99% on a single coefficient.

Table: Number of traces required for different noise levels to recover the entire key with more than 95% probability

Noise level sigma	0.5	1	2
SNR	10.7	2.67	0.67
Number of traces	42 000	54 000	75 000

Conclusion

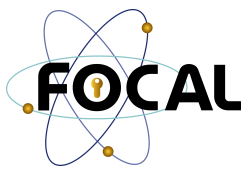
- Building secure implementations for post-quantum cryptography is still very challenging.
- We show a new methodology of performing unprofiled attack on `ML-KEM`.
- The attack can extend to `ML-DSA` as well.
- Extend the attack to higher-order masking.
- Perform the full attack on a real device running the masked `ML-KEM` implementation.

END

Thank You!

For your attention

Questions?



FOCAL

We shed lights on the FOCAL research project.