Single-Trace Key Recovery Attacks on HQC Using Valid and Invalid Ciphertexts

Haiye Dong[2], Qian Guo[1], and Denis Nabokov[1]

[1] Dept. of Electrical and Information Technology, Lund University, Lund, Sweden
{qian.guo,denis.nabokov}@eit.lth.se
[2] Independent Researcher, Lund, Sweden
chelseadong202@gmail.com

Abstract. As the Hamming Quasi-Cyclic (HQC) cryptosystem was recently selected by NIST for standardization, a thorough evaluation of its implementation security is critical before its widespread deployment. This paper presents single-trace side-channel attacks that recover the full long-term secret key of HQC, experimentally evaluated on a protected Cortex-M4 implementation. We introduce two distinct attacks that significantly advance the state of the art: a passive attack that uniquely models key recovery as a moderate-density parity-check (MDPC) decoding problem from a single valid ciphertext, and an active chosen-ciphertext attack employing a new probing strategy on a linear combination of secret key components for significantly improved efficiency. Both attacks are enabled by a new information set decoding (ISD) variant that exploits soft side-channel information, a contribution of broader importance to code-based cryptography. Our experiments show that a single trace suffices for full key recovery under realistic conditions, effectively defeating countermeasures such as codeword masking for the first time. We also show that several existing defenses are ineffective against the new attacks.

Keywords: Post-Quantum Cryptography; Hamming Quasi-Cyclic (HQC); Side-Channel Analysis; Single-Trace Key Recovery

1 Introduction

The impending arrival of large-scale quantum computers poses a significant threat to our current public-key cryptographic infrastructure, which largely relies on the presumed hardness of integer factorization and the discrete logarithm problem. In response, the U.S. National Institute of Standards and Technology (NIST) initiated a comprehensive process to solicit, evaluate, and standardize quantum-resistant cryptographic algorithms. This multi-year effort has recently culminated in the selection of several schemes, including the Hamming Quasi-Cyclic (HQC) cryptosystem [11], a code-based Key Encapsulation Mechanism (KEM), which is designated for standardization.

As HQC transitions from a candidate to a global standard, the focus of the cryptographic community intensifies on its practical security. Beyond its theoretical foundations, a scheme's resilience in real-world deployments hinges on its resistance to implementation attacks. Side-channel analysis, which exploits physical leakages like power consumption or electromagnetic emissions, represents one of the most potent threats to cryptographic hardware and software. Therefore, a thorough and continuous evaluation of HQC's implementation security is not only critical for identifying potential vulnerabilities but also for developing robust and effective countermeasures before its widespread adoption.

Motivation. The implementation security of HQC has become a vibrant and fast-paced area of research with numerous side-channel and fault-injection attacks published, for instance [46,36,3,22,48,40,17,45,28,14,24,43,35,15,41,4,13,47,7,6,29]. These attacks exploit a wide array of physical leakages—including timing, cache-timing, power, and electromagnetic (EM) emissions—as well as complex microarchitectural behaviors. Methodologically, these attacks range from the direct exploitation of physical leakages to the construction of powerful mathematical oracles, such as plaintext-checking (PC) or decryption-failure (DF) oracles that leverage the decapsulation behavior of the cryptosystem.

The study of HQC's implementation security can be divided into two phases, mirroring the evolution of the scheme itself. The initial versions [1] of HQC used a concatenation of repetition and BCH codes. Following the demonstration of a decryption failure attack [18] in 2020, the proposal transitioned to a more efficient concatenated coding scheme of Reed-Muller and Reed-Solomon codes. For this newer version, the attack landscape has evolved rapidly; the number of required decapsulation queries, or traces, has plummeted from hundreds of thousands [17] in 2022 to as few as two in recent attacks [35,6].

Among the spectrum of side-channel threats, single-trace attacks are particularly devastating. Their ability to succeed with a single observation makes them an ideal tool for a practical adversary: they are exceptionally difficult to detect and thwart, bypassing countermeasures based on failure monitoring or rate limiting. However, the true impact of such an attack depends on its target. While recent research has demonstrated the feasibility of single-trace attacks for recovering the ephemeral shared key [15], the recovery of the long-term secret key represents a far more catastrophic failure. A compromised long-term key allows an adversary to impersonate users, decrypt future communications, and potentially decrypt all previously recorded traffic, leading to a total and retroactive security collapse.

Despite the recognized severity of single-trace attacks, their application to long-term key recovery in HQC remains an open problem. State-of-the-art practical attacks targeting the long-term secret key require a minimum of two side-channel traces [35,6]. Other single-trace investigations have been limited to simulations without experimental validation [4] or required a relatively high number of traces (83) on a real-world platform [29]. To close this critical gap, we propose two distinct single-trace attacks that achieve full secret key recovery on a Cortex-M4 platform. The first is a stealthy passive attack using a single valid ciphertext, while the second is an active chosen-ciphertext attack designed to succeed with significantly less leaked information than previously required.

1.1 Contributions

We present the first single-trace side-channel attacks that can achieve full secret key recovery on a protected implementation of the HQC KEM. Our attacks are built on three technical contributions:

- We introduce a novel passive side-channel attack (VC-KRA-MDPC) that models the key recovery process as a moderate-density parity-check (MDPC) decoding problem. This reframing significantly improves the information-theoretic efficiency of the attack, making it possible to recover the full secret key from a single side-channel trace of a valid HQC decapsulation.
- We design a new chosen-ciphertext attack (OT-FDA-CK) that enhances the capabilities of Full-Decryption (FD) attacks [6]. The attack employs a novel probing strategy that extracts information about a linear combination of secret key components. This method maximizes the information gained from each oracle query, enabling a single-trace attack on `hqc-1`.
- We develop a new information set decoding (ISD) variant specifically tailored to process soft information derived from side-channel leakages. This algorithm serves as the efficient recovery backend for both of our attacks, converting probabilistic data into the concrete secret key. The technique is general and may hold independent interest for other applications in side-channel analysis of code-based cryptosystems.

We demonstrate the practical viability of these contributions using power leakages measured from an HQC implementation with codeword masking on a Cortex-M4 microcontroller. Our experiments show that a single trace suffices for full key recovery on this target. Both attacks are inherently immune to simple countermeasures such as failure-detection mechanisms or ciphertext sanity checks, highlighting the limitations of current defenses.

1.2 New Techniques

Our single-trace attacks rely on three novel techniques that significantly improve the efficiency and practicality of side-channel analysis against HQC.

Key recovery as MDPC decoding. For our passive attack (VC-KRA-MDPC), we introduce a new conceptual framework that models the key recovery as a decoding problem for a MDPC code. We construct a parity check matrix from the public HQC parameters and define a codeword that concatenates the secret key components, x and y, with a corresponding syndrome vector. The noisy side-channel leakage obtained from the `expand_and_sum` operation provides a soft-information estimate of this syndrome [35]. We then apply an iterative belief propagation (BP) decoder, a well-studied area for MDPC codes [33,23,2], initializing it with priors based on the known sparsity of the secret key and the channel probabilities from the side-channel measurements. Crucially, the presence of many short cycles in the code's Tanner graph means the BP decoder's soft outputs are not accurate posterior probabilities. While the ranking of bit reliabilities remains highly informative, this imperfection is a primary reason why the attack requires a larger amount of initial side-channel information to succeed. This reframing is nonetheless substantially more information-theoretically efficient than prior methods, reducing the required side-channel information to a level where a single trace is sufficient for a full key recovery.

Combined probing for active attacks. For our active chosen-ciphertext attack (OT-FDA-CK), we develop a more efficient probing strategy that improves upon the state-of-the-art FD oracle attack [6]. Instead of probing the secret key components x and y in separate queries, our method crafts a single invalid ciphertext that manipulates the decapsulation process to depend on a linear combination of both. By carefully selecting the vectors r_1, r_2, and e, we create a perturbation of the form $X^l x + X^k y$ (where by default, $l = k = 0$), allowing us to extract information about both secret components simultaneously from the FD oracle's output. This combined probing approach extracts more information per query than the previous method in [6], eliminating the need for a second trace.

ISD with soft information. The final step in both of our attacks relies on a new ISD algorithm. It replaces random permutation with a targeted, probabilistic approach, using reliability scores to guide information set selection and employing weighted sampling to prioritize bit positions most likely to be zero. This specialized ISD is crucial for efficiently converting the noisy beliefs—whether they are the approximate scores from BP or the more precise ones from the FD oracle—into the final secret key. This method sees a more natural fit in code-based cryptography; in contrast, exploiting such soft information in lattice-based schemes, for instance by combining BP with lattice reduction as explored in [25,26], remains a more complex problem that warrants further investigation.

1.3 Comparison with Related Works

The field of implementation security for HQC has seen a rapid acceleration of results, summarized in Table 1. Our work significantly advances this state-of-the-art. In the domain of passive attacks using valid ciphertexts, previous works like Maillet et al. [29] and Paiva et al. [35] also target the `expand_and_sum` leakage. However, the algebraic approach of Maillet et al. requires the recovery of a full intermediate vector $v - u \cdot y$ (over 17 000 bits), leading to a practical requirement of **83 traces** on a Cortex-M4 platform. The method of Paiva et al. is more efficient but still requires **two traces**. In contrast, our method using MDPC decoding is information-theoretically superior, achieving a full key recovery from a **single trace** with only about 1 000 bits of information. This represents a more than 25-fold gain in efficiency over prior work in some high-noise scenarios[3].

[3] In an earlier version of their paper [35] (`https://eprint.iacr.org/archive/2023/1626/1697742959.pdf`), Paiva et al. reported they needed about 25 traces on average when the measured bit error rate was 0.35. Our method still succeeds with one trace.

Table 1: Key-recovery side-channel attacks for hqc-1 (Cortex-M4 results).

Attack	**Ciphertext Type**	**Oracle Type**	**Info (bits)**	**Traces**
Maillet et al., CRYPTO'25 [29]	Valid	$v - u \cdot y$ (`expand_and_sum`)	$\sim 17\,000$	83
Paiva et al., PQCrypto'25 [35]	Valid	$v - u \cdot y$ (`expand_and_sum`)	$\sim 7\,000$ per trace	2
VC-KRA-MDPC (Ours)	Valid	$v - u \cdot y$ (`expand_and_sum`)	$\sim 1\,000$	**1**
Dong & Guo, CHES'25 [6]	Invalid	Full-Decryption (FD)	~ 128 per query	2
OT-FDA-CK (Ours)	Invalid	Full-Decryption (FD)	~ 191	**1**

For active attacks using invalid ciphertexts, we improve upon the **two-trace** FD attack [6], where secret components are probed separately. Our novel combined probing strategy targets a linear combination of both components, extracting more information per query and enabling a full key recovery with just a **single trace**. Crucially, we are the first to demonstrate that these single-trace key-recovery attacks are effective against implementations protected by codeword masking. These contributions collectively establish our methods as the most efficient and robust side-channel attacks against HQC to date.

1.4 Organization

The remainder of this paper is organized as follows. Section 2 provides background on the HQC KEM. We then present our two new attacks: VC-KRA-MDPC, a passive method using valid ciphertexts (Section 3), and OT-FDA-CK, an active chosen-ciphertext approach (Section 4). We describe the soft-information ISD post-processing in Section 5 and validate our attacks experimentally in Section 6. Finally, we discuss implications in Section 7 and conclude in Section 8.

2 Preliminaries

Mathematical notations. Let n be a positive integer. We consider the polynomial ring $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$ defined over the binary field $\mathbb{F}_2$, where addition and subtraction are equivalent operations. An element $h \in \mathcal{R}$ can be interchangeably viewed as a polynomial or as its corresponding length-n binary vector, denoted $\mathbf{h}$ over $\mathbb{F}_2$. The number of non-zero coefficients in h is its Hamming weight ($w_{\mathrm{H}}(h)$).

The notation $\xleftarrow{\$} \mathcal{S}$ indicates a uniform random sampling from a given set $\mathcal{S}$. When sampling from the ring $\mathcal{R}$ with a fixed Hamming weight w, we denote the operation as $\xleftarrow{\$} (\mathcal{R}, w)$. For integers $a < b$, the set of integers from a to $b - 1$ is represented by $[a..b]$. Finally, a binary random variable X following the Bernoulli distribution with parameter p is denoted by $X \sim \mathsf{Ber}_p$. Such a variable assumes the value 1 with probability p and 0 with probability $1 - p$.

The Shannon entropy of a discrete random variable X with probability mass function $p(x)$ is $H(X) = -\sum_x p(x) \log_2 p(x)$. This simplifies to the binary entropy function, $H_b(p) = -p \log_2(p) - (1-p)\log_2(1-p)$, for a variable $X \sim \mathsf{Ber}_p$. The mutual information $I(X;Y)$ between two random variables X and Y measures their statistical dependence and is defined as $I(X;Y) = H(X) - H(X|Y)$.

2.1 Hamming Quasi-Cyclic (HQC)

Hamming Quasi-Cyclic (HQC) [11] is a code-based Key Encapsulation Mechanism (KEM) undergoing standardization by NIST, with its security rooted in the hardness of the quasi-cyclic syndrome decoding problem. The HQC construction follows a conventional methodology, starting with an IND-CPA secure Public Key Encryption (PKE) scheme, denoted HQC.CPAPKE, which is then elevated to an IND-CCA secure KEM, HQC.CCAKEM. This transformation leverages a modern instantiation of the Fujisaki-Okamoto (FO) transform [10], specifically the transform with implicit rejection as analyzed by Hofheinz, Hövelmanns, and Kiltz [27]. This section first outlines the core PKE scheme before detailing its extension to a CCA-secure KEM.

The core PKE scheme. The HQC.CPAPKE scheme comprises three fundamental algorithms: KeyGen, Enc, and Dec. All polynomial operations are performed in the ring $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$ for a prime n. Table 2 presents the three HQC parameter sets, hqc-1, hqc-3, and hqc-5, defined in [11].

KeyGen. The key generation algorithm samples a random polynomial $h \leftarrow\$ \mathcal{R}$ and two sparse polynomials $x, y \leftarrow\$ (\mathcal{R}, \omega)$, where $w_{\mathrm{H}}(x) = w_{\mathrm{H}}(y) = \omega$. The secret key is the pair $sk = (x, y)$, and the public key is $pk = (h, s)$, where the syndrome is computed as $s = x + h \cdot y$.

Enc. To encrypt a message m, the algorithm first deterministically generates three error vectors from a seed θ: $r_1, r_2 \in \mathcal{R}$ of weight ω_r, and $e \in \mathcal{R}$ of weight ω_e. The message m is encoded into a codeword $m\mathbf{G}$ using a linear code $\mathcal{C}$, which is realized as a concatenation of a shortened Reed-Solomon code and a duplicated Reed-Muller code. The ciphertext (u, v) is then computed as:

$$u = r_1 + h \cdot r_2 \quad \text{and} \quad v = m\mathbf{G} + \mathrm{Truncate}(s \cdot r_2 + e, n_1 n_2) \tag{1}$$

where $\mathrm{Truncate}(\cdot, n_1 n_2)$ reduces the polynomial to the length of the codeword space of $\mathcal{C}$.

Dec. Upon receiving a ciphertext (u, v), decryption involves decoding the value $V = v - \mathrm{Truncate}(u \cdot y, n_1 n_2)$. By substituting the expressions for u, v, and s, we derive the input to the decoder:

$$\begin{aligned} V &= m\mathbf{G} + \mathrm{Truncate}(s \cdot r_2 + e - u \cdot y, n_1 n_2) \\ &= m\mathbf{G} + \mathrm{Truncate}((x + h \cdot y) \cdot r_2 + e - (r_1 + h \cdot r_2) \cdot y, n_1 n_2) \\ &= m\mathbf{G} + \mathrm{Truncate}(x \cdot r_2 - r_1 \cdot y + e, n_1 n_2) \end{aligned}$$

The result is the original codeword perturbed by an effective error vector $e' = \mathrm{Truncate}(x \cdot r_2 - r_1 \cdot y + e, n_1 n_2)$. If $w_{\mathrm{H}}(e')$ is within the error-correction capacity of $\mathcal{C}$, the decoder recovers m successfully. Otherwise, a decryption failure occurs. Hereafter, the $\mathrm{Truncate}(\cdot, n_1 n_2)$ notation will be omitted for simplicity where the context is unambiguous.

Table 2: HQC parameter sets. The inner code is a duplicated Reed-Muller code derived from a base $[128, 8, 64]$ Reed-Muller code. Specifically, 3 duplications yield the $[384, 8, 192]$ code, while 5 duplications produce the $[640, 8, 320]$ code.

	RS-S			Duplicated RM				
Instance	n_1	k_1	d_{RS}	n_2	d_{RM}	n	ω	$\omega_r = \omega_e$
hqc-1	46	16	31	384	192	17669	66	75
hqc-3	56	24	33	640	320	35851	100	114
hqc-5	90	32	59	640	320	57637	131	149

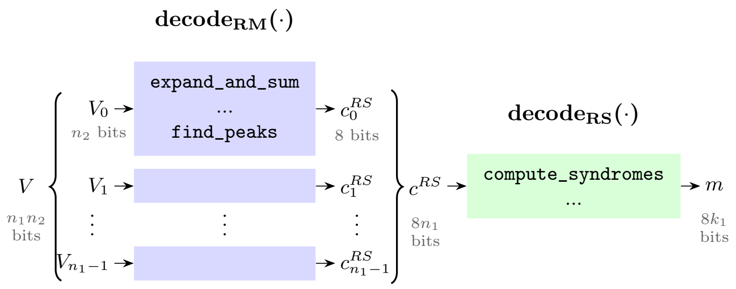


Fig. 1: Decoding of the concatenated Reed-Muller and Reed-Solomon codes.

Concatenated coding scheme in HQC. The error-correcting code $\mathcal{C}$ used in HQC is a two-layer concatenated code. The structure consists of an outer Reed-Solomon code and an inner duplicated Reed-Muller code.

The message $m \in \mathbb{F}_{2^8}^{k_1}$ is first encoded by the outer code, which is a shortened Reed-Solomon (RS) code with parameters $[n_1, k_1, d_{\mathrm{RS}}]$ over the field $\mathbb{F}_{2^8}$. This produces an intermediate codeword $m_1 \in \mathbb{F}_{2^8}^{n_1}$. Each of the n_1 bytes of this codeword is then individually encoded by the inner code. The inner code is a duplicated Reed-Muller (RM) code, derived from a base $[128, 8, 64]$ RM code. Depending on the parameter set, this base code is duplicated 3 or 5 times to yield a $[384, 8, 192]$ or $[640, 8, 320]$ code, respectively.

The final codeword $m\mathbf{G}$ is formed by concatenating the n_1 binary output blocks from the inner encoder, resulting in a vector of length $n_1 n_2$. The decoding process, illustrated in Figure 1, reverses this procedure: the received vector is split into n_1 blocks, each is decoded by the inner RM decoder to recover a byte, and the resulting sequence of bytes is then decoded by the outer RS decoder to recover the original message m.

HQC Key Encapsulation Mechanism (KEM). The construction of the IND-CCA secure KEM, referred to as HQC.CCAKEM, is detailed in Figure 2. This enhanced KEM variant builds upon HQC's public-key encryption (PKE)

```
**Input:** $ek_{\text{KEM}}$
**Output:** $K$, $c_{\text{KEM}} = (c_{\text{PKE}}, \text{salt})$
1: $m \leftarrow\!\!\$\ \mathbb{F}_2^{k_m}$
2: $\text{salt} \leftarrow\!\!\$\ \mathbb{F}_2^{128}$
3: $(K, \theta) \leftarrow \mathcal{G}(\mathcal{H}(ek_{\text{KEM}}) \parallel m \parallel \text{salt})$
4: $c_{\text{PKE}} \leftarrow \mathsf{HQC.CPAPKE.Encrypt}(ek_{\text{KEM}}, m, \theta)$
5: $c_{\text{KEM}} \leftarrow (c_{\text{PKE}}, \text{salt})$
```

(a) Encaps

```
**Input:** $dk_{\text{KEM}}$, $c_{\text{KEM}} = (c_{\text{PKE}}, \text{salt})$
**Output:** $K'$
1: $(ek_{\text{KEM}}, dk_{\text{PKE}}, \sigma, \_) \leftarrow dk_{\text{KEM}}$ ▷ *Parse key*
2: $m' \leftarrow \mathsf{HQC.CPAPKE.Decrypt}(dk_{\text{PKE}}, c_{\text{PKE}})$
3: $(K', \theta') \leftarrow \mathcal{G}(\mathcal{H}(ek_{\text{KEM}}) \parallel m' \parallel \text{salt})$
4: $c'_{\text{PKE}} \leftarrow \mathsf{HQC.CPAPKE.Encrypt}(ek_{\text{KEM}}, m', \theta')$
5: $c'_{\text{KEM}} \leftarrow (c'_{\text{PKE}}, \text{salt})$
6: **if** $c_{\text{KEM}} \neq c'_{\text{KEM}}$ or $m' = \perp$ **then**
7:     $K' \leftarrow \mathcal{J}(\mathcal{H}(ek_{\text{KEM}}) \parallel \sigma \parallel c_{\text{KEM}})$
8: **end if**
```

(b) Decaps

Fig. 2: HQC.CCAKEM.

core by integrating a suite of hash functions—$\mathcal{H}$, $\mathcal{G}$, and $\mathcal{J}$—to ensure security against active attackers.

A key feature of this construction is the de-randomization of the encryption process, which is achieved by using a seed θ. This seed is derived by applying the hash function $\mathcal{G}$ to a combination of the hashed public key $\mathcal{H}(ek_{\text{KEM}})$, a randomly generated message m, and a randomly chosen 128-bit salt. During the decapsulation stage, a crucial re-encryption step is performed to verify the integrity and authenticity of the received ciphertext. If this check fails, or if the initial decryption is unsuccessful, a separate hash function $\mathcal{J}$ is used to generate a pseudorandom key, preventing any information leakage to a potential attacker.

2.2 Codeword Masking Countermeasures

A prominent countermeasure against side-channel attacks on the decoding process is codeword masking [32], a technique also recommended for protecting HQC in [29]. The core principle is to mask the sensitive data processed by the decoder. This is achieved by first generating a random message mask, m', and encoding it into a corresponding codeword mask, c'. The decoder then processes the sum of the original ciphertext and this codeword mask. Due to the linearity of the code, the output is the original message masked by m', i.e., $m + m'$. The final plaintext, m, is retrieved by simply subtracting the random mask m'.

3 VC-KRA-MDPC: Single-Trace Key Recovery Attack with Valid Ciphertexts via MDPC Decoding

We present VC-KRA-MDPC, a novel single-trace attack that improves the side-channel attack of [35] on valid HQC ciphertexts targeting the `expand_and_sum` leakage. The original attack requires more information than can be obtained from a single trace. Our method overcomes this limitation by modeling the key-recovery problem as a moderate-density parity-check (MDPC) decoding problem. This approach makes the key recovery significantly more efficient and feasible with a single trace.

```c
void expand_and_sum(rm_expanded_cdw *dest, rm_codeword_t src[]) {
    // start with the first copy
    for (int32_t part = 0; part < 4; part++) {
        for (int32_t bit = 0; bit < 32; bit++) {
            (*dest)[part * 32 + bit] = src[0].u32[part]>>bit&1;}}
    // sum the rest of the copies
    for (int32_t copy = 1; copy < MULTIPLICITY; copy++) {
        for (int32_t part = 0; part < 4; part++) {
            for (int32_t bit = 0; bit < 32; bit++) {
                (*dest)[part * 32 + bit] += src[copy].u32[part]>>bit&1;}}}}
```

Listing 1: `expand_and_sum`

3.1 Threat Model

Our threat model considers a passive adversary aiming to recover the long-term secret key (x, y) of an HQC implementation. The attack is designed to be stealthy, requiring side-channel traces from a valid decapsulation.

The attacker operates under a standard gray-box, profiled model. This assumes physical access for power or EM monitoring and an offline profiling phase on a similar device to create leakage templates. The adversary has full knowledge of the HQC algorithm, public key, implementation details, and the intermediate random vectors r_1, r_2, and e. This is a practical assumption, as the adversary generates the ciphertext themselves. For example, Alice could recover Bob's key by measuring a single power trace of his device's decapsulation process.

3.2 Baseline Attack

Paiva et al. [35] exploit the leakage of the `expand_and_sum` operation during the Reed-Muller decoding process, where codeword in each Reed-Muller block is processed bit by bit. Let $V_i \in \mathbb{F}_2^{n_2}$ be the input vector to the i-th Reed-Muller decoder block for each i in $[0..n_1]$ as shown in Figure 1, representing a possibly corrupted codeword from the repeated Reed-Muller code used in HQC. Define the multiplicity as M and the block length $n_2 = 128M$.

As noted by Paiva et al. [35], the `expand_and_sum` operation shown in Listing 1 processes the input vector V_i (denoted as `src[]` in the listing) in two distinct steps to produce the output integer vector $a_i \in \mathbb{Z}^{128}$ (via the pointer `dest`). In the *initialization step*, the algorithm processes only the first 128-bit repetition within the input block V_i. Each bit from this initial repetition is extracted and its integer value (0 or 1) is assigned as the initial value for the corresponding element $a_i[j]$ in the output array. Following the initialization, the *accumulation step* processes the remaining $M-1$ repetitions within V_i. For each of these subsequent repetitions, the algorithm again extracts the bit at each position j. The integer value of this extracted bit (0 or 1) is then added to the existing value stored in the corresponding element $a_i[j]$.

Paiva et al. [35] use profiled side-channel attacks focusing on the power and EM leakages generated by these bit-wise manipulations. They are able to classify the value of each bit being processed, leading to the recovery of a noisy version of the intermediate codeword V_i, denoted z_{V_i}, with a bit error rate ρ_1. The channel can be modeled as a binary symmetric channel (BSC) with crossover probability ρ_1, representing the bit error rate: $P(z_{V_i}[j] \neq V_i[j]) = \rho_1$.

Their post-processing method requires aggregating information to construct likelihood vectors, which are then used to heuristically guide an algebraic solver. In their experiments, two traces are needed for full key recovery.

3.3 Key Recovery as an MDPC Decoding Problem with BP

Our core contribution is to model the HQC key recovery task as a decoding problem for a moderate-density parity-check (MDPC) code. This allows the application of iterative belief propagation (BP) algorithms to output soft information about the key bits from noisy side-channel observations.

MDPC codes [33] are a class of linear codes defined by a parity-check matrix that is denser than that of traditional low-density parity-check (LDPC) codes. While early proposals for MDPC-based cryptosystems often focused on low-complexity, hard-decision decoding algorithms like Gallager's bit-flipping [12], their structure is also amenable to soft-decision BP decoders. Unlike sparse LDPC matrices, the parity-check matrix of an MDPC code typically has row and column weights proportional to the square root of the code length.

Parity check matrix and codeword definition. We first construct the parity-check matrix $\mathbf{H}$ for our code. Given $\mathsf{vec} \in \mathbb{F}_2^n$, let $\mathrm{cir}(\mathsf{vec})$ denote the circulant matrix induced by vec. For the valid-ciphertext attack, the matrix $\mathbf{H}$ is formed by concatenating the circulant matrices derived from r_1 and r_2, two vectors known to the attacker. The final parity-check matrix $\mathbf{P}$ for our decoding problem is defined by appending the $n \times n$ identity matrix $\mathbf{I}_n$ to $\mathbf{H}$:

$$\mathbf{P} = [\mathbf{H}|\mathbf{I}_n] = [\mathrm{cir}(r_2)|\mathrm{cir}(r_1)|\mathbf{I}_n]. \tag{2}$$

Any codeword $\hat{c} \in \mathbb{F}_2^{3n}$ of this code must satisfy the parity-check equation $\mathbf{P}\hat{c}^T = \mathbf{0}^T \pmod 2$. We structure this codeword $\hat{c}$ to contain the secret key components and a corresponding syndrome vector, syn:

$$\hat{c} = [x|y|\mathsf{syn}]. \tag{3}$$

Substituting this structure into the parity-check equation directly yields the definition of the syndrome vector: $\mathsf{syn} = [x|y]\mathbf{H}^T \pmod 2$.

This framework allows us to use a noisy observation of the syndrome, z_{syn}, to recover soft information about the secret key components x and y. We employ the sum-product BP algorithm for this decoding task.

For hqc-1, the weights of r_1 and r_2 (75 each) result in an MDPC code with a row weight of 150 in its $\mathbf{H}$ matrix. Additionally, due to HQC's truncation, only the first $n_1 n_2$ rows of $\mathbf{P}$, corresponding to the observed portion of the syndrome, receive initial channel information during decoding. Henceforth, all remaining analysis considers this truncated $n_1 n_2 \times 3n$ parity-check matrix.

LLR initialization. The BP algorithm operates on log-likelihood ratios (LLRs), defined for a binary variable b given an observation z as $L(b|z) = \ln \frac{P(b=0|z)}{P(b=1|z)}$. We initialize the LLR for each bit of the codeword $\hat{c} = [x|y|\mathsf{syn}]$ as follows.

Prior LLR for secret bits. For the first $2n$ positions corresponding to the secret key bits of $[x|y]$, the prior probability of being 1 is $p_1 = \frac{\omega}{n}$ and being 0 is $p_0 = 1 - p_1$. The prior LLR for each secret bit position is thus:

$$L_{\text{prior}}(i) = \ln \frac{p_0}{p_1} = \ln \left(\frac{n - \omega}{\omega} \right). \tag{4}$$

Channel LLR for syndrome bits. For the last n bits corresponding to the syndrome, the initial LLR is determined by the noisy observations of each syndrome bit, z_{syn_i} for $i \in [0..n]$, obtained from the side-channel oracle (modeled as a BSC with bit error rate ρ_1). The channel LLR for each syndrome bit is calculated as:

$$L_{\mathsf{channel}}(i) = \ln \frac{P(z_{\mathsf{syn}_i}|\mathsf{syn}_i = 0)}{P(z_{\mathsf{syn}_i}|\mathsf{syn}_i = 1)} = (-1)^{z_{\mathsf{syn}_i}} \ln \left(\frac{1 - \rho_1}{\rho_1} \right). \tag{5}$$

These prior and channel LLRs are concatenated to form the initial LLR vector $L_{\mathsf{init}} \in \mathbb{R}^{3n}$.

Iterative BP and bits ranking. The sum-product BP algorithm is executed on the Tanner graph representation [44] of the parity check matrix $\mathbf{P}$. The algorithm takes $\mathbf{P}$ and L_{init} as input. It iteratively passes messages between variable and check nodes for a fixed number of iterations, Iter. This process refines the belief for each bit position in the codeword $\hat{c}$, ultimately yielding a final normalized posterior LLR vector, $L_{\mathsf{final}} \in \mathbb{R}^{3n}$ (details in Appendix B). The first $2n$ elements of L_{final}, which contain the posterior beliefs for the secret key bits x and y, serve as the soft-information input to our new ISD variant for the final key recovery in Section 5.

We rank the secret key bit indices $i \in [0..2n]$ based on their values in L_{final} from large to small. Indices with the largest values are considered most likely positions for zeros. We calculate the number of errors (i.e., actual ones) in the first n positions of the reordered key vector. To understand the performance of the BP algorithm, this statistic is calculated for various bit error rates ρ_1 and the number of iterations Iter. Figure 3 presents the distribution of this statistic for three error rate levels and Iter up to 5. The results reveal three key insights: first, even a single iteration is remarkably effective. Second, while further iterations yield improvements, the gains diminish rapidly, with performance largely stabilizing after three iterations. Third, this improvement with increased iteration is more pronounced for lower bit error rates. Consequently, we use the soft information after four iterations for the ISD algorithm.

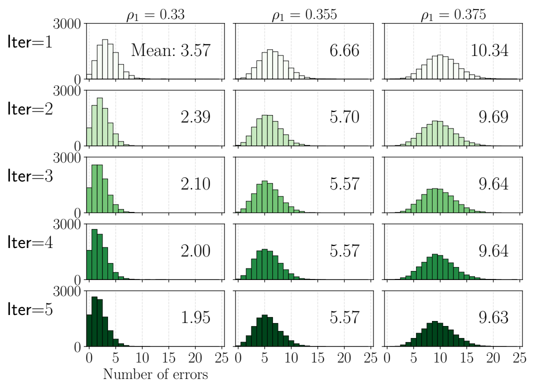


Fig. 3: Frequency plot of the number of errors in the first n positions of the reordered vector for hqc-1. Each plot is generated with 10 000 random key pairs.

4 OT-FDA-CK: Chosen-Ciphertext Attack with Invalid Ciphertexts Using Combined Key Components

We present OT-FDA-CK, a novel side-channel attack capable of recovering the long-term secret key of HQC. With a Full-Decryption (FD) oracle, the attack achieves full key recovery with a single chosen-ciphertext query, enabling a single-trace attack on the Cortex-M4 platform. Our work introduces a more efficient probing method targeting a combination of secret key components, advancing the OT-PCA framework [7] and improving upon the two-call FD oracle attack [6].

4.1 Threat Model

The adversary's objective is to recover the long-term secret key, (x, y), through an active chosen-ciphertext attack. In this scenario, the adversary submits a carefully crafted, invalid ciphertext to a device performing HQC decapsulation and measures the resulting physical side-channel leakages, such as power consumption or electromagnetic (EM) radiation.

The attack relies on the construction of a Full Decryption (FD) oracle, denoted $\mathcal{O}_{\text{FD}}$, from side-channel leakage. Unlike traditional binary PC and DF oracles—versatile tools effective against many post-quantum schemes and instantiated from diverse leakages such as timing, power, or cache behavior—the FD oracle, firstly proposed for HQC in [6], goes further. By exploiting the HQC decoder's parallel architecture, it simultaneously reveals all n_1 inner Reed-Muller decoding outputs in a single query. To model its behavior in a noisy environment, we define an inaccuracy parameter, ρ_2, as the probability that each Reed-Muller decoding from the FD oracle yields an incorrect output. The power of the FD oracle is the key element that allows the adversary to succeed with a single trace.

To build the FD oracle, our attack assumes a standard profiled side-channel model. For our experimental validation on the Cortex-M4, we presume the adversary uses a similar, or even identical, "clone" device to build the required leakage templates. We note, however, that this is a conservative assumption and not a fundamental requirement. As demonstrated in attacks on lattice-based cryptography [34,24], the clone-device assumption can often be relaxed, with attacks remaining potent in cross-device settings. Given that the side-channel leakage for constructing an FD oracle for HQC [6] is significantly stronger than for schemes like ML-KEM [24], the feasibility of such cross-device attacks is further reinforced.

4.2 State of the Art: Probing Key Components Individually

The OT-PCA framework [7,6] uses the publicly available HQC decoder to construct offline templates (OT) capable of extracting soft information about the secret key components, x and y, separately. The central idea is that the secret key acts as a sparse perturbation to a carefully chosen error vector e. The resulting ciphertext, when processed by the decoder, produces different outputs (e.g., "success" or "failure" in the PC oracle case [7]) that are correlated with the secret key's structure.

The framework comprises of an offline phase and an online attack phase. During the offline phase, error patterns are optimized to maximize the information gain from each decoder query via a genetic-style algorithm. Then, OTs are built for each optimized pattern. The process involves generating a large number of low-weight vectors sampled according to the secret key's distribution and categorizing them based on their decoding outcomes. Within each category, the empirical conditional probability of a secret bit position being one is estimated. The templates essentially map each possible decoder output to the probability of a secret bit being one at a given position [7,6].

During the actual attack, the adversary crafts ciphertexts by combining the optimized error vector e and strategically selected polynomials for r_1 and r_2 to probe the secret key parts x and y separately. After obtaining the decoder's output, the attacker consults the pre-computed OTs to gather information regarding key bit positions' likelihood of being one, which facilitates the final post-processing step.

Following the OT-PCA framework [7], we assume the adversary, during the actual attack, has access to pre-computed OTs and optimized error vectors or has constructed them beforehand. Generating these resources is a one-time low-cost pre-computation that falls outside the scope of the actual attack phase, as they relate to the HQC construction itself rather than a specific device or key.

4.3 New Attack Methodology: Probing Combined Key Components

We propose an enhanced OT-PCA variant that improves attack efficiency by simultaneously targeting both secret key components, x and y, within a single decoding query. This is achieved by choosing special forms of r_1 and r_2 that result in a ciphertext with both x and y components. Specifically, the attacker sets $r_1 = X^k$ and $r_2 = X^l$, with X^k and X^l polynomials with a Hamming weight of one. According to Equation (1), component u becomes $u = X^k + h \cdot X^l$, and v becomes $v = m\mathbf{G} + s \cdot X^l + e$. The input to the decoder, $v - u \cdot y$, is:

$$\begin{aligned} v - u \cdot y &= (m\mathbf{G} + s \cdot X^l + e) - (X^k + h \cdot X^l)y \\ &= m\mathbf{G} + e - X^k \cdot y + X^l(s - h \cdot y). \end{aligned} \tag{6}$$

As $s = x + h \cdot y$, the above can be written as:

$$v - u \cdot y = m\mathbf{G} + e - X^k \cdot y + X^l \cdot x. \tag{7}$$

Equation (7) shows the key innovation of our method: the ciphertext is influenced by both secret key parts, x and y. Specifically, x contributes through the shift l ($X^l \cdot x$), while y contributes through the shift k ($X^k \cdot y$). In contrast, previous approaches [41,7,6] set either r_1 or r_2 to 0 to include only one key component in the ciphertext, forcing each component to be probed individually.

With an FD oracle defined in [6] that outputs n_1 RM decoding results simultaneously, our attack proceeds by setting the shift parameters k and l to 0. The ciphertext in Equation (7) then simplifies to $v - u \cdot y = m\mathbf{G} + e - y + x$, where the perturbation to e is the combined secret $x + y$. We configure e by placing the same length-n_2 optimized error pattern ϵ in each of the n_1 inner RM blocks.

Since the inner RM decoders act on disjoint length-n_2 blocks of the input, their operations are independent; and the probability distribution for any block of $x + y$ is statistically the same. Consequently, our entire offline analysis can be simplified to focusing on a single, representative RM block.

Offline phase. The offline phase aims to identify optimal error patterns ϵ and to construct an OT for each. Both require querying the RM decoder with input $\epsilon + \xi$, where the vector ξ models a length-n_2 block of the combined secret $x + y$. We generate ξ by summing two independent samples, ξ_x and ξ_y, each drawn according to the distribution of a secret key block.

To find an optimal ϵ, we use a genetic algorithm with random starts similar to that in [6], with the goal of maximizing the output entropy of the RM decoder. This iterative algorithm generates mutations of the current best pattern by flipping one or two bits. Each mutation is evaluated by repeatedly querying the decoder to compute the empirical entropy of its outputs, given by $H = -\sum_{\mathsf{d} \in \mathsf{DO}} p_{\mathsf{d}} \log_2(p_{\mathsf{d}})$, where p_{d} is the empirical probability of decoding output d in the set of all obtained decoding outputs DO. The mutation yielding the highest entropy is selected to seed the next generation.

The resulting best error pattern for hqc-1 achieves an entropy of 4.15 bits. This increases the information extraction per FD oracle query to approximately 191 bits (46×4.15), a significant improvement over the 128 bits from the previous method in [6], which targets key components individually.

Following the OT-PCA framework [7], we then construct one template $\mathcal{T}_{\epsilon,m}$ for each optimized error pattern ϵ ($m = 0$ by default). To do this, we generate 100

Algorithm 1 FD attack with combined key components (online phase)

```
Input: $h,\ s,\ m\ (:= 0),\ salt$, scheme parameters $(n_1, n_2, \mathbf{G})$
       $\mathcal{E} = \{\epsilon_1, \ldots, \epsilon_t\}, \mathcal{T}_{\epsilon,m}$ for each $\epsilon \in \mathcal{E}$
Output: Reliability score vector $R$
 1: $R_\ell \leftarrow 1.0 \quad \forall \ell \in [0..n_1 n_2]$
 2: for each $\epsilon \in \mathcal{E}$ do
 3:     $e \leftarrow (\epsilon, \epsilon, \ldots, \epsilon)$                  ▷ $n_1$ blocks, each with length $n_2$
 4:     $r_1 \leftarrow X^0,\ r_2 \leftarrow X^0, u \leftarrow X^0 + h \cdot X^0$
 5:     $c \leftarrow \big(u,\ m\mathbf{G} + s \cdot X^0 + e,\ salt\big)$
 6:     Query oracle: $(o_0, \ldots, o_{n_1-1}) \leftarrow \mathcal{O}^{\rho_2}_{\mathrm{FD}}(c)$
 7:     for $k \in [0..n_1]$ do                  ▷ for each of the $n_1$ parallel outputs
 8:         for $j \in [0..n_2]$ do
 9:             $P \leftarrow \Pr(\xi_j = 1 \mid o_k)$ from $\mathcal{T}_{\epsilon,m}$
10:             $\ell \leftarrow j + k \cdot n_2$        ▷ map position $j$ in block $k$ to global index $\ell$
11:             $R_\ell \leftarrow R_\ell \cdot P$
12:         end for
13:     end for
14: end for
15: return $R$
```

million random ξ vectors, group them by their RM decoding outputs, and discard any groups with fewer than 50 000 samples. We get 37 groups for the best error pattern for hqc-1. For each group, we calculate the empirical probability of being one for each position in the length-n_2 block. The $\mathcal{T}_{\epsilon,m}$ maps each RM decoding output to a vector of conditional probabilities, where each entry represents the likelihood of a one at that bit position in one block of $x + y$.

Online phase. The online phase, shown in Algorithm 1, builds upon the FD attack in [6]. Our attack's core innovation is its ciphertext construction: setting specific values for r_1 and r_2 to probe the combination $x + y$ in a single query.

For each selected error pattern ϵ, a ciphertext is constructed and the FD oracle is queried; with a perfect oracle, a single ϵ (and thus a single query) suffices. The algorithm then uses the oracle's outputs to look up conditional probabilities in the pre-computed OTs. These probabilities are cumulatively multiplied into a reliability score vector, R, which contains the soft information on $x+y$. Crucially, the reliability score for each bit position of the combined secret $x + y$ is then assigned to the corresponding bit positions in both x and y. The scores are used to rank all secret key bit positions by their likelihood of being zero. This ranking serves as the input for the post-processing step described in Section 5. Figure 4 shows the distribution of the number of errors in the first n positions of the reordered secret key vector given different levels of oracle error probability ρ_2.

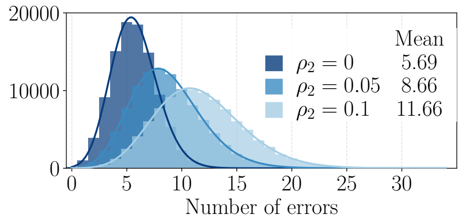


Fig. 4: Frequency plot of the number of errors in the first n positions of the reordered vector for different levels of oracle error probability ρ_2 from hqc-1. Each level is drawn with 100 000 random key pairs.

5 Post-Processing with a New ISD Variant

We next describe a new post-processing method to recover the secret vector with information set decoding (ISD). ISD algorithms are a fundamental tool for solving the decoding problem for random linear codes, a problem central to code-based cryptography. The foundational concept was introduced by Prange [38], with Stern's subsequent work [42] establishing the highly influential collision-based, meet-in-the-middle search strategy. This paradigm has been the basis for numerous modern improvements [30,5,31] and has recently been complemented by distinct approaches such as sieving-based ISD [21,8].

Moving to the context of side-channel analysis, the integration of soft information into ISD is critical. While there have been attempts to incorporate reliability information into Prange-style decoders, the substantially better performance of Stern's algorithm makes it the natural foundation for such work. Pessl and Mangard [37] pioneered this by adapting Stern's algorithm to use bit reliability, employing a column-swapping strategy analyzed with Markov chains. Building on this direction, we propose a new variant that also leverages soft information but through a different mechanism: weighted sampling within a constrained set of the most zero-like bit positions. We adopt this approach for its analytical simplicity, as the independence between trials facilitates a straightforward performance model, which we validate through simulation.

More advanced soft-decision decoders, such as SoftStern [19,20], also exist. These are one-pass algorithms that do not re-sample the information set; they allow for enumerating the most probable error patterns for the collision search and offer flexibility in memory usage. Combining their list generation with our iterative sampling is a promising direction for future work. Such a hybrid approach would require a dedicated evaluation, particularly to analyze the performance trade-offs under different memory constraints.

Problem formulation. The post-processing task is to solve for the low-weight secret key (x, y) given the public key and the soft information obtained from the previous phase. The core of this problem is the syndrome decoding equation derived from the public key, which can be expressed in matrix form as:

$$\begin{bmatrix}\mathbf{I}_n \ \mathrm{cir}(h)\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = s, \tag{8}$$

where $\mathrm{cir}(h)$ is the circulant matrix generated from the public polynomial h.

To solve this system, we adapt the general ISD framework of Finiasz and Sendrier [9]. This framework operates by selecting a subset of $n+\ell$ column positions (the information set) and transforming the corresponding submatrix of the full parity-check matrix into a quasi-standard form via Gaussian elimination:

$$\widetilde{\mathbf{H}} := \begin{pmatrix}\mathbf{Q}' & \mathbf{0}\\ \mathbf{Q}'' & \mathbf{I}_{n-\ell}\end{pmatrix}, \tag{9}$$

where $\mathbf{Q}'$ is an $\ell \times (n+\ell)$ matrix. The key input to our algorithm is a vector of reliability scores for the secret key (x, y), where each score indicates the likelihood of the corresponding bit being one.[4]. Our primary contribution is to leverage this soft information to guide the ISD process. Instead of randomly selecting an information set, our method first identifies a small candidate pool containing only the most reliable (likely zero) positions. The $n+\ell$ positions for the information set are then chosen from this constrained pool using a weighted sampling scheme.

5.1 A Modified ISD with Soft Information

Our algorithm proceeds in multiple independent trials. In each trial, an information set is first selected and processed to set up a collision search.

Information set selection. We begin by forming a candidate array of the $\phi \cdot n$ secret bit positions most likely to be zero ($1 < \phi < 2$), from which we will select the $n+\ell$ positions for the information set. This array is partitioned into two disjoint sets of equal size. To introduce randomness during partitioning, the array is first segmented into contiguous, non-overlapping blocks of four elements with each block containing an outer pair (bit indices 0 and 3) and an inner pair (indices 1 and 2). Then, one pair is randomly assigned to a first set and the other to a second set. This procedure is applied to all blocks, creating two disjoint sets of size $\frac{\phi \cdot n}{2}$ that form a complete partition of the candidate array. For each key, this partition is created only once and reused for all subsequent trials.

Each trial of the algorithm begins with a weighted sampling on the two sets, drawing $\frac{n+\ell}{2}$ bit positions from the first to form subset $\mathcal{S}_1$, and $\frac{n+\ell}{2}$ positions from the second to form subset $\mathcal{S}_2$. Together, these $n+\ell$ positions form the information set. The weighted sampling is to make sure that bits more likely to be zero are more prone to be selected. For OT-FDA-CK, we use the inverse of the conditional probability of being one for each bit position as the weight, which is directly obtained from the online attack phase. For VC-KRA-MDPC, we observe that the

[4] If only soft information for the $2n_1n_2$ truncated entries is available, the remaining positions are assigned the a priori probability ω/n.

probabilities of being one, inferred from the final normalized posterior LLRs in L_{final}, exhibit low variance, making direct use of these values akin to sampling from a uniform distribution. To amplify the underlying reliability ranking and make use of the good variation of L_{final}, we define the weight for bit position i using the transformation $(1.1 \cdot \max_j (L_{\mathsf{final}}(j)) - L_{\mathsf{final}}(i))^{-1}$.

The corresponding submatrix of the full parity-check matrix is then transformed via Gaussian elimination to yield a system with a partial syndrome of length ℓ. For an $n \times 2n$ parity-check matrix, the cost of this Gaussian elimination, denoted as C_{Gauss}, is $O(n^3)$. The concrete cost of a schoolbook implementation can be estimated as 2^{43}, 2^{46}, and 2^{48} for hqc-1, hqc-3, and hqc-5 respectively. The resulting partial syndrome becomes the target for the subsequent step.

Collision search. The search for a solution proceeds using a standard meet-in-the-middle approach to find a combination of two low-weight error patterns, one from each subset ($\mathcal{S}_1$ or $\mathcal{S}_2$), that solves the partial syndrome equation. Let the target weight in each subset be a small integer w_0.

First, a list L_1 is constructed by enumerating all possible error patterns of weight bounded to w_0 with support restricted to the positions in $\mathcal{S}_1$. For each pattern, the corresponding ℓ-bit partial syndrome is computed and stored, typically in a hash table for efficient lookup. Subsequently, we enumerate all error patterns of weight bounded to w_0 with support in $\mathcal{S}_2$. For each of these patterns, its partial syndrome s_2 is computed. We then check if the value $s_{\mathsf{tar}} - s_2$ exists in the hash table for L_1, where s_{tar} is the target partial syndrome from the Gaussian elimination step.

A successful collision identifies a candidate solution with a total weight of $\leq 2w_0$ within the information set. This candidate must then be verified against the full syndrome decoding problem. For our chosen parameters (e.g., $\ell = 51$ for hqc-1), the partial syndrome is sufficiently long to ensure that the cost of handling random collisions and verifying them is negligible compared to the cost of list generation and search, assuming an $O(1)$ memory access cost.

Success condition and complexity. A trial succeeds if it finds a low-weight key that passes all parity checks. Due to the design of HQC, this solution is guaranteed to be the unique secret key. In our simulations, since we know the true key, we can directly verify this success condition: a trial succeeds if the true secret key's partition across subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ has a weight of at most w_0 in each. The overall algorithm succeeds if any of its T independent trials succeeds.

The complexity of a single trial is dominated by the Gaussian elimination, list generation, and collision search. With a target error weight of w_0 in each partition, the cost can be approximated as:

$$C_{\mathsf{trial}} \approx C_{\mathsf{Gauss}} + 2 \cdot \binom{\frac{n+\ell}{2}}{w_0} \cdot w_0 \cdot \ell.$$

Here, C_{Gauss} is the cost of the initial Gaussian elimination. The second term represents the cost of the meet-in-the-middle search, where we generate two lists of size $\binom{(n+\ell)/2}{w_0}$ and the computation of each partial syndrome has a complexity proportional to $w_0\ell$.

The list size, $\binom{(n+\ell)/2}{w_0}$, determines the memory requirement. For hqc-1, where $n = 17\,669$ and we set $\ell = 51$, choosing $w_0 = 3$ leads to a list size of approximately $\binom{8860}{3} \approx 2^{36.5}$. Storing one of these lists may require hundreds of gigabytes of RAM, for an efficient implementation. The time complexity for this trial is dominated by the list operations, estimated at $2 \cdot 2^{36.5} \cdot w_0\ell \approx 2^{45}$ operations, which is slightly higher than the initial $C_{\mathsf{Gauss}} \approx 2^{43}$. For higher security levels, a smaller w_0 (e.g., $w_0 = 2$) may be necessary, or algorithms such as SoftStern [19,20] that allow adjusting the list size in a fine-grained manner.

The total complexity of the attack is therefore $T \cdot C_{\text{trial}}$. The number of trials an adversary can perform is ultimately limited by their computational budget. For example, with a total post-processing budget of 2^{60} operations, an adversary could execute approximately $T = 2^{60}/C_{\text{trial}} \approx 2^{15}$ trials.

5.2 Simulation Results

Parameter choice and simulated success rate. The algorithm's performance depends on two key parameters: the candidate pool size factor, ϕ, and the number of trials, T. The parameter ϕ governs a tradeoff between the number of errors in the candidate array and the sampling flexibility.

A small ϕ yields a candidate array nearly the same size as the $n+\ell$ positions to be sampled. While this pool contains the fewest total errors, it offers little sampling flexibility as the number of errors in the final subsets ($\mathcal{S}_1$ and $\mathcal{S}_2$) is nearly determined by the number of errors present in this constrained pool. Conversely, a larger ϕ introduces a vast number of additional zero positions into the pool albeit at the cost of also including some lower-ranked errors. It provides the crucial flexibility for the sampling process to "replace" an error from the top-ranked portion with a zero from the extended section. Ultimately, this flexibility can increase the probability of drawing two subsets with fewer errors, which is especially important when oracle accuracy is lower.

Figure 5 plots the simulated success rates of OT-FDA-CK with a perfect oracle and 10 000 random keys for two choices of ϕ, 1.1 versus 1.2, against the number of trials T up to 2^{15}. A significant advantage for the smaller ϕ value is evident when T is small. This performance gap narrows as the number of trials grows. In all remaining simulations, we set $\phi = 1.2$ by default.

Modeling the success rate. Our goal is to find the minimum T required to achieve a desired success rate (e.g., 95%). For a given key, as each trial is independent, the probability of success after T trials, P_{succ}, can be modeled by the one-shot success probability, P_{os}, as $P_{\mathsf{succ}} = 1 - (1 - P_{\mathsf{os}})^T$.

We first validate this theoretical model. To obtain an empirical estimate of P_{os} for a given key, we perform 10 000 independent tests, each with T bounded by 2^{15}. The value for P_{os} is calculated as the proportion of tests that succeed on the very first trial. Figure 6 shows this validation for three distinct random keys. In

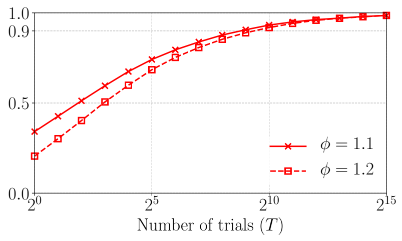


Fig. 5: Simulated success rates with different ϕ. x-axis in log scale.

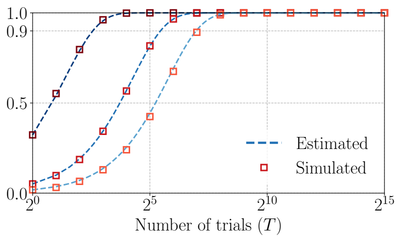


Fig. 6: Estimated vs. simulated success rates for three distinct keys. x-axis in log scale.

all three cases, the simulation success rates align closely with their corresponding curves from theoretical estimation, confirming the model's accuracy.

Analysis of the one-shot success probability. While the model holds for a single key, P_{os} varies across different keys. To predict the average success rate at a given T, we analyze the distribution of P_{os} using 1 000 random keys. For each key, P_{os} is calculated as the success rate over 100 000 single-trial tests ($T = 1$).

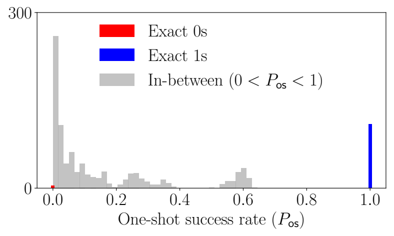
Figure 7(a) shows the histogram of P_{os} for VC-KRA-MDPC with $\rho_1 = 0.355$. The distribution is highly polarized. The BP decoder effectively solves "easy" key instances, pushing their success probability to nearly one within one trial. However, it has little impact on "harder" instances, which remain clustered at a low success probability. This creates a large gap in the distribution, with very few keys having a P_{os} between 0.6 and 1.0.

Figure 7(c) shows the histogram of P_{os} for OT-FDA-CK with a perfect oracle. In contrast, without the BP step, the data points are predominantly clustered at low success probabilities, and the remaining values spread continuously across the range. For both attacks, a guaranteed failure ($P_{\mathsf{os}} = 0$) is very rare, accounting for only 0.4% and 0.7% of keys, respectively.

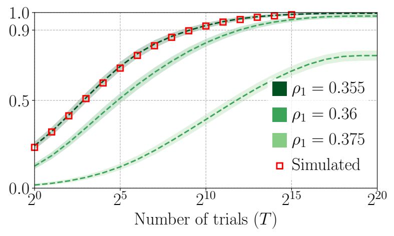
Estimating expected success rate. With the empirical distribution of P_{os}, we can estimate the expected success rate P_{succ}, which is given by the expectation of the success function: $P_{\mathsf{succ}} = \mathbb{E}[f(P_{\mathsf{os}})]$, where $f(p) = 1 - (1 - p)^T$.

We perform a bootstrap estimation to project the evolution of the expected success rates and its 95% confidence interval for T up to 2^{20}. Figure 7(b) and 7(d) show the estimated curves for the two attacks respectively. The estimation predicts that about $T = 2^{11}$ trials are required to achieve a 95% average success rate for VC-KRA-MDPC with $\rho_1 = 0.355$ and OT-FDA-CK with a perfect oracle. As shown in the figures, the estimation curves for these two specific configurations align closely with the success rates in simulation, validating the model.

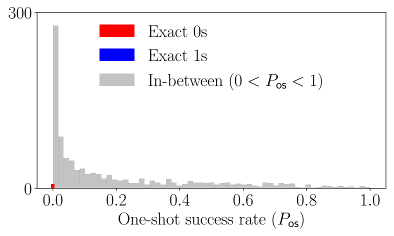
We can quantify our VC-KRA-MDPC attack from an information-theoretic perspective. Modeling the power leakage for each bit from `expand_and_sum` as a BSC channel with crossover probability ρ_1, a single trace provides $n_1 n_2 (1 -$



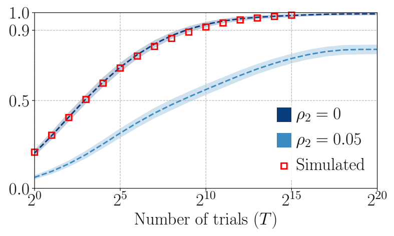
(a) VC-KRA-MDPC: Histogram of one-shot success rates (P_{os}) for 1 000 random keys with $\rho_1 = 0.355$.



(b) VC-KRA-MDPC: Estimated success rates (P_{succ}) using P_{os}. x-axis in log scale.



(c) OT-FDA-CK: Histogram of one-shot success rates (P_{os}) for 1 000 random keys with $\rho_2 = 0$.



(d) OT-FDA-CK: Estimated success rates (P_{succ}) using P_{os}. x-axis in log scale.

Fig. 7: Post-processing success rates for the two single-trace attacks on hqc-1.

$H_b(\rho_1)$) bits of information. For hqc-1, our error rate of $\rho_1 = 0.355$ yields approximately 1 090 bits. In contrast, the experiments in [35] report a lower error rate of 0.14—yielding roughly 7 300 bits per trace—yet their method requires two traces. This demonstrates our MDPC model's superior efficiency.

6 Validations

This section presents a practical, real-world validation of the single-trace attacks proposed in this paper. We evaluate the effectiveness of our attacks on a recent HQC implementation protected with codeword masking. Our evaluation follows a standard profiled side-channel attack methodology: first, we characterize the device's leakages and build templates on a clone device; subsequently, we use these templates in an attack phase to recover the secret key from a single power-consumption trace captured from the target.

Our experiments are based on the ChipWhisperer Husky toolkit, including the CW308 UFO board and the CW308T-STM32F405 target board with a 32-bit ARM Cortex-M4 CPU, operating at a frequency of 25 MHz. Power traces are sampled at 200 MHz. The Husky's hardware limits trace acquisition to approximately 131 000 samples per capture. To analyze longer functions, we performed multiple captures with identical inputs, adjusting the measurement offset for each. This sequence of captures is treated as a single trace, as no new information is learned beyond the extended observation window.

We target the most recent reference implementation of HQC [11] (commit from 26 Aug, 2025) for the `hqc-1` parameter set. The source code was compiled using the `arm-none-eabi-gcc` compiler with the `-O3` optimization level.

6.1 Targeting Codeword Masking with Valid Ciphertexts

Bypassing codeword masking. The codeword masking relies on the linearity of the underlying codes. A random message mask m' is generated, encoded into a codeword mask c', and added to the decoder input. If an attacker can recover m', the countermeasure is trivially defeated.

We recover m' by targeting the Reed-Solomon encoder, as shown in its simplified form in Listing 2. The authors of [15] demonstrated that the Galois field multiplication, `gf_mul`, leaks significant information about its first operand. In the reference HQC implementation, this first operand (`gate`) is directly derived from the secret message mask m', while the second operand is a known constant.

Following a standard profiling procedure, we collected 10 000 traces of the Reed-Solomon encoder for random messages. Each trace is split into k_1 segments according to the outer loop; each segment contains d_{RS} multiplications that share the same first operand. We use Fisher's linear discriminant analysis (LDA) to train a classifier that predicts the byte value of this first operand.

As input to the classifier, we use a reduced trace obtained via a standard t-test: for each bit b_i of the `gate` variable, we compute a list of points of interest (PoI) lst_i. We then form the union $\bigcup_i \mathrm{lst}_i$ of these bit-wise PoI lists and select the leftmost and rightmost points a and b from that union; the reduced trace is built from PoI equal to $[a..(b+1)]$. This procedure yields a single, compact window that covers all bit-specific informative samples while remaining suitable as an input to LDA. We train k_1 separate models in total, one per segment. For `hqc-1` this produces 310 000 traces per model, which we split 90%/10% into training and validation sets. The classifiers achieve an average accuracy of 0.9997 across all models for a *single* multiplication. Since each input is processed by d_{RS} independent multiplications, aggregating the per-multiplication predictions reduces the overall error probability to a negligible value. Finally, the mask m' can be recovered from the `gate` values using Gaussian elimination.

These results indicate that codeword masking does not provide meaningful protection. Even for a relatively modest single-multiplication success rate 0.765, the probability to fully recover m' with simple majority voting reaches 0.99. Therefore, the single-multiplication accuracy 0.9389 reported in [15] (the discrepancy with our result is attributable to differences in platform and measurement setup) is already sufficient to remove the mask with overwhelming probability.

Obtaining the leakage for VC-KRA-MDPC. With the mask m' recovered, we can compute the corresponding codeword mask $c' = m'\mathbf{G}$ and subtract it,

```c
void reed_solomon_encode(uint8_t *cdw, const uint8_t *msg) {
    memset(cdw, 0, PARAM_N1);
    for (i = 0; i < PARAM_K; ++i) {
        gate = msg[PARAM_K - 1 - i] ^ cwd[PARAM_N1 - PARAM_K - 1];
        for (j = 0; j < PARAM_G; ++j) {
            tmp[j] = gf_mul(gate, PARAM_RS_POLY[j]);
        }...}...}
```

Listing 2: `reed_solomon_encode` (simplified)

Table 3: Average accuracy of recovering bits from `expand_and_sum` for hqc-1.

Bit Range	0–127	128–255	256–383
Accuracy	0.9698	0.8144	0.7159

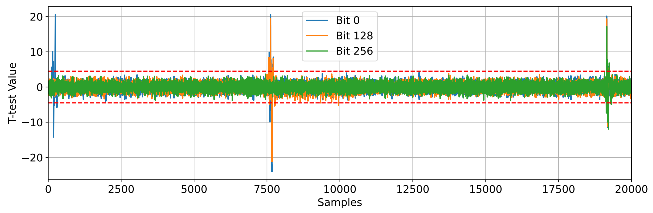


Fig. 8: The t-test value for output bit 0 of `expand_and_sum`. The latter part of the function is truncated as the t-test value is within red lines until the end of the function.

thereby exposing the original value $c = m\mathbf{G} + x \cdot r_2 + y \cdot r_1 + e$. The goal of VC-KRA-MDPC is to obtain a noisy observation of this value, $\tilde{c} \approx c + c'$, from which we can approximate the syndrome $\mathsf{syn} = x \cdot r_2 + y \cdot r_1$.

This leakage is sourced from the `expand_and_sum` function (shown in Listing 1) within the Reed-Muller decoder. The function's structure reveals that each output bit in the `dest` array depends on M distinct bits from the input `src` array, spaced 128 bits apart. This dependency is visualized in the t-test results shown in Figure 8, where the leakage related to the processing of bits 0, 128, and 256 for the first output bit is clearly visible.

Inspired by this structure, we built templates to recover the M-bit tuples that contribute to each output bit simultaneously, rather than recovering the codeword bit-by-bit. For each of 128 tuples, we trained an LDA model on 10 000 traces of the first call to `expand_and_sum` during decapsulation. The resulting bit-wise recovery accuracies are detailed in Table 3. As expected, the accuracy slowly decreases for later bits as there are fewer points in the trace leaking information about them.

Attack results. To evaluate the success rate of the full single-trace attack, we performed 100 independent trials. In each trial, a new key pair and a valid ci-

```
void compute_syndromes(uint16_t *syndromes, uint8_t *cdw) {
    for (size_t i = 0; i < 2 * PARAM_DELTA; ++i) {
        for (size_t j = 1; j < PARAM_N1; ++j) {
            syndromes[i] ^= gf_mul(cdw[j], alpha_ij_pow[i][j - 1]);}
        syndromes[i] ^= cdw[0];}}
```

Listing 3: `compute_syndromes`

phertext were generated. From a single power trace of the decapsulation, we first executed our attack on the Reed-Solomon encoder to recover the mask m', which was successful in all 100 trials. We then used our models to recover the masked noisy codeword $\tilde{c}$ from n_1 `expand_and_sum` functions. From this, we computed the Log-Likelihood Ratios (LLRs) for the syndrome vector, incorporating the bit-wise probabilities from our models.

This soft information was fed into the Belief Propagation (BP) decoder. For all 100 keys, after a single iteration of BP, there were zero errors remaining in the first n positions of the reordered key vector. A final Gaussian elimination step was sufficient to recover the full secret key, achieving a 100% success rate. This result confirms the practical viability of the VC-KRA-MDPC attack, even against a codeword-masked implementation.

6.2 Invalid Ciphertext Case

We validated the OT-FDA-CK attack against the HQC implementation protected by codeword masking. The countermeasure was bypassed by first recovering the random mask message m', as detailed in Section 6.1, and then constructing a FD oracle for the masked Reed-Muller outputs. The linearity of the Reed-Muller code allows us to subtract the known mask from the oracle's output to obtain the true decoded values.

The oracle targets the `gf_mul` operation within the `compute_syndromes` function (Listing 3). The first operand to `gf_mul` corresponds to the output of an inner Reed-Muller decoder. This byte leaks significantly, enabling near-perfect recovery. For hqc-1, each Reed-Muller output is used in 30 separate multiplications, providing ample leakage to create an oracle with no observed errors.

Our experiments on the Cortex-M4 platform confirm that we can perfectly recover the last 45 Reed-Muller decoding outputs, thus constructing a perfect *partial* FD oracle. This strong leakage source was previously identified in [6]. To assess its impact, we simulated the attack for hqc-1 over 10 000 random keys. With a 45-output oracle, we achieved a 90% key recovery rate after 2^{10} ISD trials and 98% after 2^{15} trials. This is only marginally less effective than an attack using a full 46-output oracle, with the performance gap diminishing as the number of trials increases (see Figure A.2 in the appendix for a direct comparison).

7 Discussion

7.1 On Attack Methods

Information-theoretical analysis. From an information-theoretic perspective, our proposed attacks demonstrate a significant leap in efficiency compared to previous works, requiring substantially fewer bits of information from the side channels to achieve full key recovery. This reduction is a key enabler for the success of our single-trace attacks.

For attacks using *valid ciphertexts*, the contrast is stark. The method by Maillet et al. [29] is the most demanding, requiring the recovery of $u \cdot y$, on the order of n bits, which is $\sim 17\,000$ bits for hqc-1. This high threshold explains the practical need for 83 traces in their experiment on the Cortex M4 platform. The attack by Paiva et al. [35] is more efficient but still requires approximately 7 000 bits per trace, totaling $\sim 14\,000$ bits for their two-trace attack. Our passive attack, VC-KRA-MDPC, radically lowers this requirement. By modeling the problem as MDPC decoding, we need only about 1 000 bits of information, making a single-trace recovery feasible.

In the realm of CCA, our active attack (OT-FDA-CK) is also superior. The original FD-oracle attack [6] required two oracle calls for hqc-1, with each revealing around 128 bits, meaning the side channel had to leak a minimum of 256 bits. In contrast, our improved OT-FDA-CK attack creates an FD oracle with an output entropy of approximately 191 bits. This indicates that our OT-FDA-CK method is likely the most efficient for scenarios involving protected implementations or high-noise environments where physical side-channel leakage is limited.

Combining leakage sources. The invalid ciphertext setting grants an attacker significantly more freedom, enabling a single-trace attack that is more powerful than either of our standalone attacks (VC-KRA-MDPC and OT-FDA-CK). While both attacks model the key recovery as a decoding problem on the vector $e' = e + r_1 \cdot y + r_2 \cdot x$ where e, r_1, and r_2 are known, the CCA adversary setting provides a critical advantage: the attacker can choose r_1 and r_2 to be extremely sparse (e.g., weight 1, 2, or 3).

This choice allows for a combination of leakages. Specifically, combining the noisy information from the `expand_and_sum` function provides a better estimation of e' than what the FD oracle provides alone, which in turn frames a fundamentally easier decoding problem. Also, the ability to select sparse r_1 and r_2 allows us to construct a significantly sparser LDPC code for post-processing. This is a marked improvement over the MDPC code derived from the denser, randomly generated vectors in the valid ciphertext attack. The transition to a sparser code improves the performance of BP algorithms, making the attack using combined leakage more robust to noise. Consequently, the attack is ideal for analyzing well-protected implementations or high-noise environments.

Alternative methods. In developing our OT-FDA-CK attack, we explored alternative probing strategies, such as increasing the Hamming weight of the intermediate polynomials r_1 and r_2. The hypothesis was that a larger, denser perturbation in the decoder's input vector ($v - u \cdot y = m\mathbf{G} + e + \sum_j X^{l_j} x - \sum_i X^{k_i} y$) would increase the chance of decoding to a faraway codeword. This, in turn, would increase the entropy of the Reed-Muller decoding output, thus yielding more information from the FD oracle.

However, our experiments showed that this method did not improve upon the version presented and, in fact, performed worse. A possible explanation is that an LDPC decoding problem with a denser parity-check matrix demands substantially more information to be effectively resolved.

Interestingly, we did observe that this method could be used to construct a more powerful Multi-Value Plaintext-Checking (MV-PC) oracle-based attack than the one presented in [6]. A detailed analysis of this improved MV-PC attack is beyond the scope of this paper and we leave its exploration for future research.

7.2 Countermeasures

The powerful single-trace attacks presented in this paper render many existing countermeasure strategies ineffective, highlighting an urgent need for more robust defenses. We analyze the shortcomings of two primary approaches: low-cost detection schemes and masking.

Low-cost, detection-based countermeasures. Low-cost countermeasures typically rely on detecting anomalous behavior, such as failing a sanity check on the ciphertext's structure (e.g., checking if it is too sparse or skewed) or observing an excessive number of decapsulation failures for a given key-pair. Upon detection, the device might refresh its secret key or shut down. However, these methods have known drawbacks. Besides being bypassable, as shown in the context of Kyber [39], they can introduce new vulnerabilities. Forcing a fallback to key generation, for instance, shifts the operational focus to a process that is inherently more vulnerable and costly to secure than decapsulation itself. Furthermore, an adversary could deliberately trigger repeated failures to mount a Denial-of-Service (DoS) attack, creating an additional attack surface.

Our single-trace attacks render such strategies largely ineffective. The passive attack using a valid ciphertext (VC-KRA-MDPC) is particularly stealthy, as it does not trigger decapsulation failures, making it invisible to these detection mechanisms. While the active, oracle-based attack (OT-FDA-CK) does induce errors, the detection-based approach fails to protect past communications. For instance, if a static key is used over an extended period (e.g., 30 days), an adversary can archive the encrypted traffic and mount the attack on the final day. A successful key recovery would retroactively compromise all data exchanged during that period. Our current chosen ciphertext selection in OT-FDA-CK also adds the public key h to u (s and e to v)—all of which are dense vectors in our setting—making simple sanity checks on ciphertext sparsity ineffective.

Masking countermeasures. Codeword masking has been suggested as a defense for the `expand_and_sum` operation in HQC [29]. However, our experimental results on the Cortex-M4—a crucial benchmark for embedded security—demonstrate that this countermeasure is insufficient to prevent our single-trace key recovery.

True protection against these attacks will likely require more granular, gate-level masking schemes that introduce more fresh randomness into sensitive computation. The high performance overhead of such schemes, however, remains a significant barrier. The development of efficient yet secure masking is therefore a critical direction for future research. This is especially pertinent given that even well-regarded first-order masking schemes for lattice-based cryptography have been broken with a small number of traces, even in a cross-device setting [24]. Since the side-channel leakage from HQC appears stronger than that from Kyber, the effectiveness of any proposed masking implementation must be rigorously evaluated against FD-based attacks like ours.

While recent work on masked HQC implementations [16] is a commendable step, public and open-source libraries are essential. The community requires rigorously-vetted implementations to properly evaluate the practical security of HQC before its widespread deployment as a standard.

8 Concluding Remarks

In this work, we have presented significant side-channel threats to HQC, a new NIST PQC KEM standard, driven by three methodological contributions and a validation using power consumption measurements taken with a ChipWhisperer device. Our methodological advances include: 1) a novel passive attack that uniquely reframes key recovery as a MDPC decoding problem; 2) a new active attack employing an efficient chosen-ciphertext strategy to probe a linear combination of secret key components; and 3) a new ISD variant that exploits reliability information, crucial for both attacks. We have validated these methods with a Cortex-M4 platform, demonstrating for the first time that even a codeword-masked implementation can be broken with a single trace. Critically, low-cost detection-based countermeasures are ineffective to both attacks.

This work opens several avenues for future research. First, given that our attacks defeat codeword masking, a critical area for future work is the development of more efficient and secure masking schemes. A second direction is the exploration of new leakage sources on various hardware platforms, such as FPGAs, to construct powerful oracles. Since the Full Decryption (FD) oracle for HQC is a recent development, understanding its practical feasibility and variations across different implementations is a critical next step. Finally, we highlight the need for a more rigorous, quantitative analysis of combined leakage attacks, especially when targeting state-of-the-art defenses such as the new masked implementation proposed in [16].

Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2021-04602 and 2023-03654), the Swedish Civil Contingencies Agency (grant number 2020-11632) the Crafoord Foundation, and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg

Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

References

1. Aguilar-Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Persichetti, E., Zémor, G.: HQC. Tech. rep., National Institute of Standards and Technology (2017), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions`
2. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Aguilar-Melchor, C., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.P., Zémor, G., Vasseur, V., Ghosh, S., Richter-Brokmann, J.: BIKE. Tech. rep., National Institute of Standards and Technology (2022), available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`
3. Băetu, C., Durak, F.B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S.: Misuse attacks on post-quantum cryptosystems. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 747–776. Springer, Cham, Switzerland, Darmstadt, Germany (May 19–23, 2019). https://doi.org/10.1007/978-3-030-17656-3_26
4. Baïsse, C., Moran, A., Goy, G., Maillard, J., Aragon, N., Gaborit, P., Lecomte, M., Loiseau, A.: Secret and shared keys recovery on hamming quasi-cyclic with sasca. Designs, Codes and Cryptography **93**, 2137–2157 (February 2025). https://doi.org/10.1007/s10623-025-01575-2, `https://link.springer.com/article/10.1007/s10623-025-01575-2`
5. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer Berlin Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_31
6. Dong, H., Guo, Q.: Multi-value plaintext-checking and full-decryption oracle-based attacks on hqc from offline templates. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**, 254–289 (2025). https://doi.org/10.46586/tches.v2025.i4.254-289, `https://tches.iacr.org/index.php/TCHES/article/view/12410`
7. Dong, H., Guo, Q.: Ot-pca: New key-recovery plaintext-checking oracle based side-channel attacks on hqc with offline templates. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(1), 251–274 (jan 2025). https://doi.org/10.46586/tches.v2025.i1.251-274, `https://tches.iacr.org/index.php/TCHES/article/view/11929`, date of Publication: 2024/12/09
8. Ducas, L., Esser, A., Etinski, S., Kirshanova, E.: Asymptotics and improvements of sieving for codes. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VII. LNCS, vol. 14657, pp. 151–180. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58754-2_6
9. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer Berlin Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009). https://doi.org/10.1007/978-3-642-10366-7_6

10. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_34
11. Gaborit, P., Aguilar-Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Persichetti, E., Zémor, G., Bos, J., Dion, A., Lacan, J., Robert, J.M., Véron, P., Barreto, P.L., Ghosh, S., Gueron, S., Güneysu, T., Misoczki, R., Richter-Brokmann, J., Sendrier, N., Tillich, J.P., Vasseur, V.: Hamming quasi-cyclic (hqc). `https://pqc-hqc.org` (aug 2025), `https://pqc-hqc.org`, specification document, version 2025/08/22
12. Gallager, R.G.: Low-Density Parity-Check Codes. No. 21 in M.I.T. Press Research Monograph Series, M.I.T. Press, Cambridge, MA (1963)
13. Gast, S., Weissteiner, H., Schröder, R.L., Gruss, D.: CounterSEVeillance: Performance-counter attacks on AMD SEV-SNP. In: NDSS 2025. The Internet Society, San Diego, CA, USA (Feb 24–28, 2025)
14. Goy, G., Loiseau, A., Gaborit, P.: A new key recovery side-channel attack on HQC with chosen ciphertext. In: Cheon, J.H., Johansson, T. (eds.) Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022. pp. 353–371. Springer, Cham, Switzerland, Virtual Event (Sep 28–30, 2022). https://doi.org/10.1007/978-3-031-17234-2_17
15. Goy, G., Maillard, J., Gaborit, P., Loiseau, A.: Single trace HQC shared key recovery with SASCA. IACR TCHES **2024**(2), 64–87 (2024). https://doi.org/10.46586/tches.v2024.i2.64-87
16. Goy, G., Spyropoulos, M., Aragon, N., Gaborit, P., Pacalet, R., Perion, F., Sauvage, L., Vigilant, D.: Side-channel sensitivity analysis on HQC: Towards a fully masked implementation. Cryptology ePrint Archive, Paper 2025/1344 (2025), `https://eprint.iacr.org/2025/1344`
17. Guo, Q., Hlauschek, C., Johansson, T., Lahr, N., Nilsson, A., Schröder, R.L.: Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE. IACR TCHES **2022**(3), 223–263 (2022). https://doi.org/10.46586/tches.v2022.i3.223-263
18. Guo, Q., Johansson, T.: A new decryption failure attack against HQC. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 353–382. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64837-4_12
19. Guo, Q., Johansson, T., Mårtensson, E., Stankovski, P.: Information set decoding with soft information and some cryptographic applications. In: 2017 IEEE International Symposium on Information Theory (ISIT). pp. 1793–1797. IEEE (2017)
20. Guo, Q., Johansson, T., Mårtensson, E., Wagner, P.S.: Some cryptanalytic and coding-theoretic applications of a soft stern algorithm. Advances in Mathematics of Communications **13**(4) (2019)
21. Guo, Q., Johansson, T., Nguyen, V.: A new sieving-style information-set decoding algorithm. IEEE Transactions on Information Theory (2024)
22. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 359–386. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_13
23. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) ASI-

ACRYPT 2016, Part I. LNCS, vol. 10031, pp. 789–815. Springer Berlin Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53887-6_29

24. Guo, Q., Nabokov, D., Nilsson, A., Johansson, T.: SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 203–236. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8730-6_7
25. Hermelink, J., Mårtensson, E., Samardjiska, S., Pessl, P., Rodosek, G.D.: Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. IACR TCHES **2023**(4), 287–317 (2023). https://doi.org/10.46586/tches.v2023.i4.287-317
26. Hermelink, J., Streit, S., Mårtensson, E., Petri, R.: A generic framework for side-channel attacks against LWE-based cryptosystems. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 3–32. Springer, Cham, Switzerland, Madrid, Spain (May 4–8, 2025). https://doi.org/10.1007/978-3-031-91101-9_1
27. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Cham, Switzerland, Baltimore, MD, USA (Nov 12–15, 2017). https://doi.org/10.1007/978-3-319-70500-2_12
28. Huang, S., Sim, R.Q., Chuengsatiansup, C., Guo, Q., Johansson, T.: Cache-timing attack against HQC. IACR TCHES **2023**(3), 136–163 (2023). https://doi.org/10.46586/tches.v2023.i3.136-163
29. Maillet, N., Nugier, C., Migliore, V., Deneuville, J.: Key recovery from side-channel power analysis attacks on non-simd hqc decryption. In: CRYPTO 2025. LNCS, vol. 16004, pp. 70–102. Springer (2025). https://doi.org/10.1007/978-3-032-01901-1_3
30. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer Berlin Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011). https://doi.org/10.1007/978-3-642-25385-0_6
31. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 203–228. Springer Berlin Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_9
32. Merli, D., Stumpf, F., Sigl, G.: Protecting PUF error correction by codeword masking. Cryptology ePrint Archive, Report 2013/334 (2013), https://eprint.iacr.org/2013/334
33. Misoczki, R., Tillich, J., Sendrier, N., Barreto, P.S.L.M.: Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In: Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013. pp. 2069–2073. IEEE (2013). https://doi.org/10.1109/ISIT.2013.6620590, https://doi.org/10.1109/ISIT.2013.6620590
34. Ngo, K., Dubrova, E., Guo, Q., Johansson, T.: A side-channel attack on a masked IND-CCA secure Saber KEM implementation. IACR TCHES **2021**(4), 676–707 (2021). https://doi.org/10.46586/tches.v2021.i4.676-707, https://tches.iacr.org/index.php/TCHES/article/view/9079
35. Paiva, T.B., Ravi, P., Jap, D., Bhasin, S., Das, S., Chattopadhyay, A.: Et tu, brute? Side-channel assisted chosen ciphertext attacks using valid cipher-

texts on HQC KEM. In: Niederhagen, R., Saarinen, M.J.O. (eds.) Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part II. pp. 294–321. Springer, Cham, Switzerland, Taipei, Taiwan (Apr 08–10, 2025). https://doi.org/10.1007/978-3-031-86602-9_11
36. Paiva, T.B., Terada, R.: A timing attack on the HQC encryption scheme. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 551–573. Springer, Cham, Switzerland, Waterloo, ON, Canada (Aug 12–16, 2019). https://doi.org/10.1007/978-3-030-38471-5_22
37. Pessl, P., Mangard, S.: Enhancing side-channel analysis of binary-field multiplication with bit reliability. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 255–270. Springer, Cham, Switzerland, San Francisco, CA, USA (Feb 29 – Mar 4, 2016). https://doi.org/10.1007/978-3-319-29485-8_15
38. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **IT-8**, 5–9 (1962)
39. Ravi, P., Paiva, T., Jap, D., D'Anvers, J.P., Bhasin, S.: Defeating low-cost countermeasures against side-channel attacks in lattice-based encryption A case study on Crystals-Kyber. IACR TCHES **2024**(2), 795–818 (2024). https://doi.org/10.46586/tches.v2024.i2.795-818
40. Schamberger, T., Holzbaur, L., Renner, J., Wachter-Zeh, A., Sigl, G.: A power side-channel attack on the reed-muller reed-solomon version of the HQC cryptosystem. In: Cheon, J.H., Johansson, T. (eds.) Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022. pp. 327–352. Springer, Cham, Switzerland, Virtual Event (Sep 28–30, 2022). https://doi.org/10.1007/978-3-031-17234-2_16
41. Schröder, R.L., Gast, S., Guo, Q.: Divide and surrender: Exploiting variable division instruction timing in HQC key recovery attacks. In: Balzarotti, D., Xu, W. (eds.) USENIX Security 2024. USENIX Association, Philadelphia, PA, USA (Aug 14–16, 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/schr%C3%B6der
42. Stern, J.: A method for finding codewords of small weight. In: International colloquium on coding theory and applications. pp. 106–113. Springer (1988)
43. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. IACR TCHES **2023**(3), 473–503 (2023). https://doi.org/10.46586/tches.v2023.i3.473-503
44. Tanner, R.M.: A recursive approach to low complexity codes. IEEE Transactions on Information Theory **27**(5) (1981)
45. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. IACR TCHES **2022**(1), 296–322 (2022). https://doi.org/10.46586/tches.v2022.i1.296-322
46. Wafo-Tapa, G., Bettaieb, S., Bidoux, L., Gaborit, P., Marcatel, E.: A practicable timing attack against HQC and its countermeasure. Cryptology ePrint Archive, Report 2019/909 (2019), https://eprint.iacr.org/2019/909
47. Weissteiner, H., Rauscher, F., Schröder, R.L., Juffinger, J., Gast, S., Wichelmann, J., Eisenbarth, T., Gruss, D., Fraunhofer, S., Fraunhofer Austria, V.: Teecorrelate: An information-preserving defense against performance-counter attacks on tees. In: USENIX Security 2025 (2025)
48. Xagawa, K., Ito, A., Ueno, R., Takahashi, J., Homma, N.: Fault-injection attacks against NIST's post-quantum cryptography round 3 KEM candidates. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 33–61. Springer, Cham, Switzerland, Singapore (Dec 6–10, 2021). https://doi.org/10.1007/978-3-030-92075-3_2

Auxiliary Supporting Material

A Additional Figures

Figure A.1. Figure A.1 shows the estimated success rates obtained from simulated P_{os} values for VC-KRA-MDPC. The results align with the expectation that $\phi = 1.1$ yields a higher success rate for a small number of trials (T). As T increases, however, the performance with $\phi = 1.2$ becomes comparable and can eventually be superior, especially when the bit error probability (ρ_1) becomes higher.

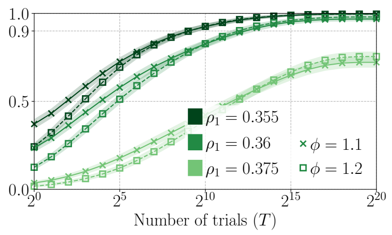


Fig. A.1: Estimated success rates (P_{succ}) using P_{os} via bootstrapping. x-axis in log scale.

Figure A.2. Figure A.2 presents a comparison of simulated success rates for the OT-FDA-CK attack, assuming a perfect Full-Decryption (FD) oracle. The simulation, conducted over 10 000 random keys, contrasts the scenario where the oracle provides all 46 Reed-Muller decoding outputs with a scenario where it provides only 45 outputs. The latter case corresponds to the experimental validation detailed in Section 6.2, where we demonstrate that a perfect oracle for the last 45 Reed-Muller outputs can be constructed by exploiting leakage from the `compute_syndrome` function in the Reed-Solomon decoder. The results show that the missing first Reed-Muller decoding output has only a minor impact on the attack's success rate. This effect is most pronounced for a small number of trials (T), while the performance gap between the two scenarios becomes negligible as T increases.

B Sum-Product BP Algorithm

This section provides a detailed description of the sum-product belief propagation (BP) algorithm used for decoding in our VC-KRA-MDPC attack, as introduced in Section 3.3. We use a normalized version to stabilize convergence. The

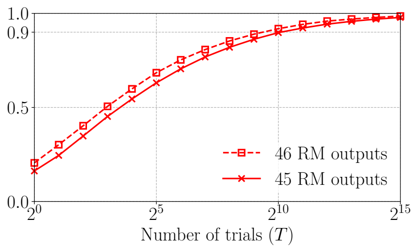


Fig. A.2: Impact of a partial oracle on OT-FDA-CK attack success rate. The plot compares simulated success rates using a perfect FD oracle with 46 RM outputs versus one with 45 RM outputs, averaged over 10 000 keys. x-axis in log scale.

algorithm operates on the Tanner graph representation [44] of the parity-check matrix $\mathbf{P} = [\mathbf{H}|\mathbf{I}_n]$, where $\mathbf{H} = [\text{cir}(r_2)|\text{cir}(r_1)]$. The goal is to compute the posterior log-likelihood ratios (LLRs) for the bits of the codeword $\hat{c} = [x|y|\mathsf{syn}]$ given the initial LLR vector, L_{init}.

Let v_i denote the i-th variable node (corresponding to the i-th bit of $\hat{c}$) and c_j denote the j-th check node (corresponding to the j-th row of $\mathbf{P}$). The algorithm iteratively passes messages between these nodes.

Let $L_{v_i \to c_j}$ denote the message from variable node v_i to check node c_j. This message represents the LLR of the variable node v_i based on information from all connected check nodes *except* c_j. Let $L_{c_j \to v_i}$ denote the message from check node c_j to variable node v_i. This message represents the extrinsic information about v_i provided by the parity-check equation at c_j, based on information from all other connected variable nodes. The algorithm proceeds as follows:

1. Initialization. For each variable node v_i and each adjacent check node c_j, the initial message is set to the LLR of the corresponding bit from L_{init}.

$$L^{(0)}_{v_i \to c_j} = L_{\mathsf{init}}(i),$$

where $L_{\mathsf{init}}(i)$ is defined by Equations (4) and (5) in the main text. The messages from check nodes to variable nodes are initialized to zero, i.e., $L^{(0)}_{c_j \to v_i} = 0$.

2. Iterative updates. For a fixed number of iterations, $t = 1, 2, \dots, \mathsf{Iter}$, the following messages are computed.

Check node to variable node update. The message from a check node c_j to a variable node v_i is calculated based on the incoming messages from all other variable nodes $v_{i'}$ connected to c_j (where $i' \neq i$). The update rule is:

$$L^{(t)}_{c_j \to v_i} = 2 \cdot \operatorname{atanh}\left(\prod_{i' \in \mathcal{N}(c_j) \setminus \{i\}} \tanh\left(\frac{L^{(t-1)}_{v_{i'} \to c_j}}{2} \right) \right),$$

where $\mathcal{N}(c_j)$ is the set of indices of variable nodes connected to check node c_j. This operation effectively computes the LLR for bit i assuming the parity-check equation at c_j is satisfied.

Variable node to check node update. The message from a variable node v_i to a check node c_j is the sum of its initial LLR and the incoming messages from all other check nodes $c_{j'}$ connected to v_i (where $j' \neq j$).

$$L^{(t)}_{v_i \to c_j} = L_{\mathsf{init}}(i) + \sum_{j' \in \mathcal{N}(v_i) \setminus \{j\}} L^{(t)}_{c_{j'} \to v_i},$$

where $\mathcal{N}(v_i)$ is the set of indices of check nodes connected to variable node v_i.

3. Normalized belief updates. In each iteration t, after receiving messages from all connected check nodes, the posterior LLR for each variable node is updated and normalized. This normalized belief, $L^{(t)}_{\mathsf{post}}(i)$, is calculated as:

$$L^{(t)}_{\mathsf{post}}(i) = \frac{L_{\mathsf{init}}(i) + \sum_{j \in \mathcal{N}(v_i)} L^{(t)}_{c_j \to v_i}}{|\mathcal{N}(v_i)| + 1}.$$

The final output of the algorithm, L_{final}, is the vector of posterior LLRs from the last iteration, i.e., $L_{\mathsf{final}}(i) = L^{(\mathsf{Iter})}_{\mathsf{post}}(i)$. This vector is then used as the soft-information input for the ISD-based post-processing step described in Section 5.

C Broad Applicability of the VC-KRA-MDPC Attack

Our new VC-KRA-MDPC attack has broad applicability. This attack exploits the same `expand_and_sum` leakage source identified by Maillet et al. in [29]. The authors of [29] claim their attack "targets standard Instruction Set Architectures (ARM T32, RISC-V, x86-64) and compiler optimization level" and works on both optimized (Non-SIMD) and reference implementations. Since our VC-KRA-MDPC attack targets the identical leakage source but employs a far more efficient key recovery approach, it inherits this wide-ranging applicability and significance.

As shown in Figure C.1, the VC-KRA-MDPC attack remains highly effective against hqc-3 and hqc-5, achieving high success probabilities even with a significant bit error rate of $\rho_1 = 0.35$.

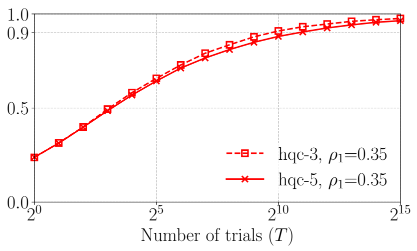


Fig. C.1: Simulated success rates for VC-KRA-MDPC on hqc-3 and hqc-5 with bit error rate $\rho_1 = 0.35$. Each series uses 10 000 keys and the number of trials T is bounded by 2^{15}. $w_0 = 3$. x-axis in log scale.