IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–24. DOI:XXXXXXXX

OT-PCA: New Key-Recovery Plaintext-Checking Oracle Based Side-Channel Attacks on HQC with Offline Templates

Haiyue Dong[1] and Qian Guo[2]

[1] Independent Researcher, Lund, Sweden, chelseadong202@gmail.com
[2] Lund University, Lund, Sweden, qian.guo@eit.lth.se

Abstract. In this paper, we introduce OT-PCA, a novel approach for conducting Plaintext-Checking (PC) oracle based side-channel attacks, specifically designed for Hamming Quasi-Cyclic (HQC). By calling the publicly accessible HQC decoder, we build offline templates that enable efficient extraction of soft information for hundreds of secret positions with just a single PC oracle call. Our method addresses critical challenges in optimizing key-related information extraction, including maximizing decryption output entropy and ensuring error pattern independence, through the use of genetic-style algorithms.

Extensive simulations demonstrate that our new attack method significantly reduces the required number of oracle calls, achieving a 2.4-fold decrease for hqc-128 and even greater reductions for hqc-192 and hqc-256 compared to current state-of-the-art methods. Notably, the attack shows strong resilience against inaccuracy in the PC oracle—when the oracle accuracy decreases to 95%, the reduction factor in oracle call requirements increases to 7.6 for hqc-128.

Lastly, a real-world evaluation conducted using power analysis on a platform with an ARM Cortex-M4 microcontroller validates the practical applicability and effectiveness of our approach.

Keywords: Code-based cryptography · NIST post-quantum cryptography standardization · HQC · Side-channel attacks · KEM

1 Introduction

Post-Quantum Cryptography (PQC) has rapidly become an essential area in modern cybersecurity, aimed at mitigating the significant threats posed by quantum computers to existing cryptographic infrastructures. These infrastructures largely depend on the computational challenges associated with factoring and discrete logarithms—both of which are susceptible to quantum attacks. In response, the National Institute of Standards and Technology (NIST) initiated a project in 2016 to identify and standardize quantum-resistant public-key cryptographic algorithms, focusing primarily on Key Encapsulation Mechanisms (KEM) and digital signatures. As part of this initiative, the lattice-based CRYSTALS-Kyber has been adopted as the primary KEM standard [SAB+22]. Nevertheless, NIST continues to explore a diverse array of algorithms to ensure a secure and resilient cryptographic infrastructure. This includes evaluating code-based KEMs such as Classic McEliece [ABC+22], HQC [AAB+22], and BIKE [ABB+22], as well as the isogeny-based SIKE [JAC+22], which has been compromised.

Among these, Hamming Quasi-Cyclic (HQC) emerges as a particularly promising candidate. NIST reports [ACD+22] have highlighted that both BIKE and HQC, built on structured codes, are apt for general-purpose KEM applications not dependent on lattice

structures. NIST has indicated that it may choose one of these two for standardization after the fourth round of evaluation. Given this context, it is crucial to thoroughly assess HQC's security credentials and develop robust implementation strategies for various real-world platforms to ensure its effectiveness and reliability in a post-quantum world.

During the initial rounds of NIST's evaluation, the implementation security of HQC has been extensively studied, as documented in significant research efforts [WTBB+19, PT19, BDH+19, GJN20, XIU+21, SHR+22, GHJ+22, HSC+23, GLG22, GNNJ23, PRJB23, GMGL24, SGG24]. Among the various threats, Side-Channel Attacks (SCA) have emerged as particularly critical, necessitating thorough investigations to ensure secure implementations. These attacks exploit indirect data, such as timing differences, power consumption, or electromagnetic leaks, to extract sensitive information.

This paper focuses on a pivotal class of side-channel attacks in post-quantum cryptography, namely key-recovery Plaintext-Checking (PC) oracle based side-channel attacks targeting HQC. The objective of these attacks is to recover the security key of HQC by constructing a PC oracle through side-channel measurements. This PC oracle is capable of determining whether a specific ciphertext c corresponds to a predetermined message m, thus facilitating the detection of decryption failures. Such an oracle can bypass Chosen Ciphertext Attack (CCA) secure transformations like the Fujisaki-Okamoto (FO) transformation [FO99], compromising the security of Indistinguishability under Chosen Ciphertext Attack (IND-CCA) secure KEMs.

These attacks are highly effective, exploiting a range of side-channel vulnerabilities as shown in studies on timing attacks [GJN20, GHJ+22], power and electromagnetic (EM) attacks [RRCB20, UXT+22, SHR+22], and micro-architecture attacks [WPH+22, CWS+24, SGG24], including cache timing attacks [HSC+23]. They can leverage both substantial and minimal leakages, from strong power/EM signals [UXT+22] to subtle timing variations in high-performance CPUs [SGG24].

Beyond identifying new sources of side-channel leakages for constructing a PC oracle, a primary research challenge lies in enhancing attack efficiency by minimizing the number of required PC oracle calls. Each oracle call translates to one or several side-channel measurements, which often become bottlenecks due to real-time interaction limitations with the target system. These limitations are imposed by system constraints, security measures, or other operational factors.

Earlier versions of HQC, prior to the third round of the NIST PQC competition, utilized a tensor product code that combined repetition codes with BCH codes. Certain parameter sets of these versions were susceptible to decryption failure attacks, as noted in [GJ20]. From the October 2020 release onward, HQC has shifted to using concatenated codes that integrate Reed Solomon (RS) and Reed Muller (RM) codes. This shift introduces new challenges for PC oracle based attacks because the decoding properties of Reed Solomon and Reed Muller codes are more complex than those of tensor product codes.

Recent advancements [GHJ+22, SHR+22, HSC+23, GNNJ23, SGG24] have significantly reduced the number of required PC oracle calls for the latest version of HQC. From the initial report at CHES 2022 [GHJ+22], which required over 800 000 calls for hqc-128, subsequent works have progressively lowered this figure: to nine thousand at ASIACRYPT 2023 [GNNJ23], and down to one thousand at USENIX Security 2024 [SGG24]. These reductions mark substantial progress. Yet, there remains the question of whether further reductions are feasible. Additionally, this research strives to develop algorithms optimized for practical applications, particularly in settings where constructing a perfectly accurate PC oracle is challenging, such as in complex environments or when robust security measures are in place.

Contributions We introduce OT-PCA, a novel approach to key-recovery PC oracle based side-channel attacks on HQC. We summarize the key contributions as follows.

Our initial conceptual contribution is a novel method for efficiently extracting soft information representing the likelihood of entries in an HQC key block being non-zero. This method leverages the publicly available decoder to construct "offline templates" (OTs). Unlike traditional template attacks targeting specific devices, our OTs exploit the interaction in the decoding process between predefined error patterns and sparse vectors sampled from HQC's secret key distribution. Specifically, adding a sampled sparse vector to a given error pattern may cause bit flips in the latter, changing the likelihood of decryption successes or failures. By summarizing the sparse vectors leading to each decryption result, we empirically estimate position-wise probabilities, representing the probability of each position in the sparse vector being non-zero conditional on the decryption result. During the online attack phase, when the decryption result of a new unknown sparse vector—the key block—is observed, we can thus infer the probability of each of the positions within the key block being non-zero using the estimated position-wise empirical probabilities, i.e., offline templates.

This method enables the extraction of substantial soft information for hundreds of secret entries with a single oracle call, significantly enhancing the efficiency of PC oracle based side-channel attacks on HQC for full key recovery.

Our second major contribution tackles the challenge of identifying error patterns for template construction that maximize information extraction during subsequent attack stages. The most critical element of this process is the first optimization problem—finding error patterns that maximize the entropy of the decryption outcomes. By ensuring that each decryption output—success or failure—is approximately equally probable, our method effectively enhances the information gain from each future decryption instance. In the paper, we advance our methodology by employing a set of error patterns instead of an individual pattern, thereby leveraging the collective information they provide. To ensure the patterns in the set are as independent as possible, we solve the second optimization problem—finding the set of error patterns that maximize the sum of pairwise Hamming distances. We develop simple genetic-style algorithms to efficiently solve these two problems, refining the error patterns used in constructing offline templates.

To summarize, the complete OT-PCA process begins with selecting several error patterns and constructing offline templates without calling the PC oracle. Subsequent online attack steps involve generating chosen ciphertexts, calling the PC oracle to obtain decryption outputs, and collecting secret information by combining these outputs with the offline templates. This process is further enhanced by a new, straightforward Information Set Decoding (ISD) algorithm that leverages soft information for efficient key recovery.

Our simulation results demonstrate the high efficiency of this novel attack. A comparative analysis with previous methods using a perfect oracle (accuracy of 1) is shown in Table 1, where our method significantly reduces the number of required PC oracle calls. For hqc-128, our approach achieves a 2.4-fold reduction in calls compared to the state-of-the-art [SGG24]. For hqc-192 and hqc-256, the reductions are even more substantial, with improvements of 87 and 137 times, respectively, compared to [SHR+22]. These results mark major advancements over existing techniques, notably, the work [SGG24], which struggles with hqc-192.

Furthermore, with a less precise oracle (accuracy of 0.95), the improvement factor for hqc-128 increases to 7.6 compared to the previous best approach [SGG24]. This demonstrates the robustness and efficiency of our attack under challenging conditions, such as exploiting subtle microarchitectural timing leakages [SGG24] or targeting masked implementations [UXT+22], where constructing a perfect PC oracle is difficult.

The OT-PCA method represents a significant advancement in PC oracle based attacks (PCA) on HQC, with potential applications beyond side-channel attacks to other domains such as misuse attacks [BDH+19, HV20] and fault-injection attacks [XIU+21, HPP21], whenever a PC oracle can be constructed.

Table 1: Comparison of our attack with previous works: number of perfect PC oracle calls required for key recovery attacks against HQC. The symbol '—' indicates that the number is not provided in the corresponding work.

Work	hqc-128	hqc-192	hqc-256
GHJLNS22 [GHJ+22]	>800 000	—	—
SHRWS22 [SHR+22]	1 152×46	1 920×56	1 920×90
HSCGJ23 [HSC+23]	>50 000	—	—
SCA-LDPC [GNNJ23]	9 000	—	—
Zero-Tester [SGG24]	1 094	—	—
Our work	10×46	22×56	14×90

Lastly, we perform a real-world evaluation of our attack using power analysis on a platform with an ARM Cortex-M4 32-bit RISC core, targeting the PQClean implementation of hqc-128. The attacks were conducted across 120 different runs, each with a newly generated key. The outcomes from these real-world tests match our simulation results, demonstrating the practical effectiveness of our approach.

The source code is available at https://github.com/ot-pca/ot-pca.

Organization The rest of the paper is organized as follows. Section 2 provides the necessary background on HQC and an overview of PC oracle based side-channel attacks on HQC, including their threat model and general approaches for recovering the secret key from the constructed PC oracle. Section 3 details the new attack methodology. In Section 4, we present simulation results for various accuracy levels of the assumed PC oracle and validate these findings on a real-world platform. This is followed by a discussion of the gains and potential extensions of the new attack methods in Section 5. Finally, in Section 6, we conclude the paper and suggest future research directions.

2 Preliminaries

In this section, we provide the essential background on Hamming Quasi-Cyclic (HQC) codes and PC oracle based side-channel attacks on HQC. We begin by explaining the notations used throughout this paper.

2.1 Mathematical Notations

Denote the binary finite field $\mathbb{F}_2$, where the operation $+$ is equivalent to the operation $-$. We work in HQC within the polynomial ring $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$, where n is a positive integer. A polynomial h within $\mathcal{R}$ can also be viewed as a row vector $\mathbf{h}$ of length n, in a vector space over $\mathbb{F}_2$, so we mix the notation if there is no ambiguity. The notion $w_{\mathrm{H}}(h)$ indicates the Hamming weight of h, i.e., the number of non-zero entries in the vector. The notation $[a..b]$ represents an index set starting from a and ending at $b-1$, where a and b are two positive integers with $a < b$. Sampling operations are denoted as $\leftarrow\!\$(\mathcal{S})$ for uniform random sampling from a set $\mathcal{S}$, and $\leftarrow\!\$(\mathcal{R}, w)$ for sampling with the constraint that the elements have a Hamming weight of w. The Bernoulli distribution, denoted by Ber_p, assigns the value 1 with probability p and the value 0 with probability $1-p$.

2.2 Overview of Hamming Quasi-Cyclic (HQC)

Hamming Quasi-Cyclic (HQC) [AAB+22] is a prominent code-based KEM that secures communication by relying on the difficulty of decoding random quasi-cyclic codes in the Hamming metric. Similar to other NIST post-quantum KEM proposals, HQC begins

Table 2: HQC parameter sets as defined in [AAB+22]. The base Reed Muller code used is the first-order $[128, 8, 64]$ Reed Muller code.

	RS-S			Duplicated RM						
Instance	n_1	k_1	d_{RS}	Mult.	n_2	d_{RM}	$n_1 n_2$	n	ω	$\omega_r = \omega_e$
hqc-128	46	16	31	3	384	192	17664	17669	66	75
hqc-192	56	24	33	5	640	320	35840	35851	100	114
hqc-256	90	32	49	5	640	320	57600	57637	131	149

with an IND-CPA (Indistinguishability under Chosen Plaintext Attack) secure Public Key Encryption (PKE) scheme (HQC.CPAPKE) and transitions to an IND-CCA secure KEM (HQC.CCAKEM) through a CCA transformation. In HQC, this transformation employs the Hofheinz-Hövelmanns-Kiltz (HHK) method [HHK17].

2.2.1 The Public Key Encryption (PKE) Core in HQC

The PKE core in HQC consists of three procedures: PKE.KeyGen for key generation, PKE.Encrypt for encryption, and PKE.Decrypt for decryption.

Three Procedures in HQC.CPAPKE During the key generation phase (PKE.KeyGen), the scheme samples three polynomials: $h \leftarrow\$ \mathcal{R}$, $x \leftarrow\$ (\mathcal{R}, \omega)$, and $y \leftarrow\$ (\mathcal{R}, \omega)$. Consequently, we know that $w_{\text{H}}(x) = w_{\text{H}}(y) = \omega$. These polynomials define the secret key as (x, y) and the public key as $(h, s = x + h \cdot y)$.

The encryption (PKE.Encrypt) process begins by generating polynomials $r_1 \leftarrow\$ (\mathcal{R}, \omega_r)$, $r_2 \leftarrow\$ (\mathcal{R}, \omega_r)$, and $e \leftarrow\$ (\mathcal{R}, \omega_e)$. This sampling process can be made deterministic by initializing the pseudo-random number generator (PRNG) with a seed θ. These components are then used to construct the ciphertext (u, v), where $u = r_1 + h \cdot r_2$ and $v = m\mathbf{G} + s \cdot r_2 + e$. The matrix $\mathbf{G}$ represents a generator matrix for the linear code $\mathcal{C}$. In HQC, $\mathcal{C}$ is specifically designed as a concatenated code that combines Reed Muller and Reed Solomon codes.

The decryption procedure (PKE.Decrypt) utilizes a decoder $\mathcal{C}.\text{Decode}(v - u \cdot y)$. Given that $s = x + h \cdot y$, we have:

$$\begin{aligned} v - u \cdot y &= m\mathbf{G} + s \cdot r_2 + e - (r_1 + h \cdot r_2) \cdot y \\ &= m\mathbf{G} + e', \end{aligned}$$

where

$$e' = x \cdot r_2 - r_1 \cdot y + e.$$

The decoder is designed with a specific error-correcting capability. If the Hamming weight $w_{\text{H}}(e')$ is sufficiently small, the decoder can correct the error, successfully recovering the encrypted message m. However, if the Hamming weight exceeds this capability, the decoder may fail, resulting in a decryption error.

Concatenated Code Construction in HQC The latest HQC specification uses a concatenated Reed Muller code and Reed Solomon code. This process encodes an input message m into a codeword $m\mathbf{G}$, where $\mathbf{G}$ is a publicly known generator matrix. The matrix $\mathbf{G}$ is defined in the space $\mathbb{F}_2^{k_m \times n_1 n_2}$, with k_m set to $8k_1$.

The encoding process proceeds as follows: First, the message $m \in \mathbb{F}_{2^8}^{k_1}$ is encoded into $m_1 \in \mathbb{F}_{2^8}^{n_1}$ using the outer $[n_1, k_1, d_{\text{RS}}]$ Reed Solomon code. Each byte $m_{1,i}$, where $0 \leq i < n_1$, is then encoded by the inner duplicated Reed Muller code into $\bar{m}_{1,i} \in \mathbb{F}_2^{n_2}$. The resulting codeword is $m\mathbf{G} = (\bar{m}_{1,0}, \ldots, \bar{m}_{1,n_1-1})$ with a dimension of $n_1 n_2$. Since computations in HQC are performed in the ambient space $\mathbb{F}_2^n$, any excess bits beyond $n_1 n_2$ are truncated when necessary.

The input to the function $\mathcal{C}$.Decode is the vector $V = v - u \cdot y$, where $V \in \mathbb{F}_2^{n_1 n_2}$. As shown in the decoding process outlined in Figure 1, V is segmented into n_1 distinct blocks, labeled as $V = (V_0, \ldots, V_{n_1-1})$, with each block V_i within $\mathbb{F}_2^{n_2}$ and defined as an 'inner block' for $0 \leq i < n_1$. Each of these blocks V_i undergoes decoding via the internal duplicated Reed Muller decoder, resulting in outputs c_i^{RS} each within $\mathbb{F}_2^8$ for indices $0 \leq i < n_1$. These outputs are subsequently concatenated to form a sequence of $8n_1$ bits, represented as $c^{\mathrm{RS}} = (c_0^{\mathrm{RS}}, \ldots, c_{n_1-1}^{\mathrm{RS}})$, with each c_i^{RS} termed an 'internal codeword'. This sequence c^{RS}, now a noisy codeword of the external Reed Solomon code, is decoded into k_1 elements within $\mathbb{F}_{256}$ and subsequently translated into a message of k_1 bytes.

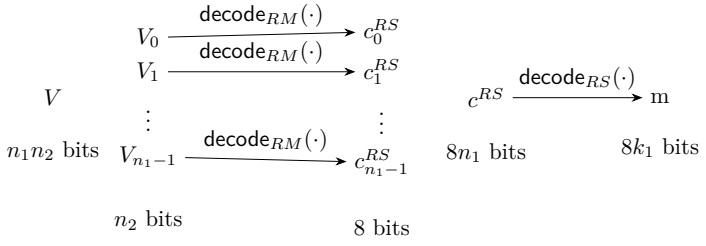


Figure 1: Decoding of concatenated Reed Muller and Reed Solomon codes

The three HQC parameter sets, hqc-128, hqc-192, and hqc-256, are presented in Table 2.

2.2.2 Key Encapsulation Mechanism (KEM) in HQC

The enhanced KEM variant of HQC, extending its PKE core, incorporates two crucial hash functions, $\mathcal{G}$ and $\mathcal{K}$. The process for constructing the IND-CCA secure KEM, denoted as HQC.CCAKEM, is illustrated in Figure 2. This construction includes a de-randomization of the encryption process via a seed θ, which is derived by hashing the message m, the public key, and a random 128-bit salt. The decapsulation phase features a re-encryption step designed to validate the authenticity of the ciphertext.

Require: pk
Ensure: K, $c = (u, v)$, salt
1: $m \leftarrow\$ (\mathbb{F}_2^{k_m})$
2: salt $\leftarrow\$ (\mathbb{F}_2^{128})$
3: $\theta \leftarrow \mathcal{G}(m \| \mathsf{pk} \| \text{salt})$
4: $c \leftarrow$PKE.Encrypt (pk, m, θ)
5: $K \leftarrow \mathcal{K}(m, c)$

(a) KEM.Encaps

Require: sk $= (x, y)$, $c = (u, v)$, salt
Ensure: K
1: $m' \leftarrow$PKE.Decrypt (sk, c)
2: $\theta' \leftarrow \mathcal{G}(m' \| pk \| \text{salt})$
3: $c' \leftarrow$PKE.Encrypt $(\mathsf{pk}, m', \theta')$
4: **if** $m' = \perp \ \vee \ c \neq c'$ **then**
5: $\quad K \leftarrow \mathcal{K}(\sigma, c)$
6: **else**
7: $\quad K \leftarrow \mathcal{K}(m', c)$
8: **end if**

(b) KEM.Decaps

Figure 2: HQC.CCAKEM

2.3 Overview of PC Oracle Based Side-Channel Attacks on HQC

We describe the threat model of PC oracle based side-channel attacks (SCA) on the HQC KEM and provide an overview of the key recovery process.

The Threat Model This paper focuses on side-channel-assisted chosen-ciphertext attacks targeting the decapsulation procedure of the HQC KEM. In our model, an attacker can choose both the ciphertext c and the message vector m, feeding them into the decapsulation algorithm on a specific device. The attacker leverages measurable side-channel leakages from this device.

Central to our threat model is the construction of a PC oracle, denoted as $\mathcal{O}^{\rho}_{\mathrm{PC}}$. This oracle [BDH+19] utilizes side-channel leakage to assess whether $\mathrm{PKE.Decrypt}(sk, c) = m$. The oracle's accuracy, ρ, indicates the probability of a correct decision, with $1 - \rho$ representing the probability of an error.

The capability to construct such a PC oracle from side-channel leakage allows the attacker to exploit various types of side-channel leakages reported in the literature. This includes, but is not limited to, timing attacks [GJN20, GHJ+22], power and electromagnetic (EM) attacks [RRCB20, UXT+22, SHR+22, SCZ+23], and micro-architecture attacks [WPH+22, CWS+24, SGG24] such as cache timing attacks [HSC+23]. Such attacks demonstrate the vulnerability of various post-quantum KEMs, including those based on lattice, coding, and isogeny theories (cf., [UXT+22]).

This study assumes a general PC oracle, allowing for broad applicability of our new attack methodology beyond the specific power leakages detailed in Section 4.2. Consequently, our approach is adaptable to multiple forms of side-channel leakages once such a PC oracle can be constructed.

Moreover, the PC oracle based chosen ciphertext attacks are extensible to fault-injection attacks [XIU+21, HPP21], including techniques like RowHammer [MKB+24], enhancing the applicability of the newly developed method.

Overview of HQC Key Recovery Methods In the attack model, the attacker primarily gains information on decryption failures through a PC oracle constructed from side-channel leakages. To achieve full key recovery, the attacker must correlate this information with the hidden secret keys y and x.

Research (e.g., [GHJ+22, HSC+23, GNNJ23, SGG24]) reveals a strategic approach by attackers: the careful selection of polynomials m, r_1, r_2, and e to craft ciphertexts (u, v) with specific properties essential for extracting secret keys. Central to this approach is the dependency of decryption outcomes on the input to the decoder, $v - u \cdot y$. This approach ensures that decoding results are closely linked to the secret vector y, and indirectly to vector x, through the public parameter h and the relationship $s = x + h \cdot y$.

These attack strategies assume that sufficient errors have already been introduced to push the outer Reed Solomon decoder to its limit for correct decoding. Consequently, corrupting even a single additional Reed Muller block results in a decryption failure.

3 The New Attack

In this section, we present the new attack methodology, OT-PCA, which significantly reduces the number of oracle calls required in a PC oracle based attack against HQC. This reduction is achieved by constructing offline templates using the publicly available decoder for the concatenated code. We begin with a brief description of the key novel ideas behind this approach and then provide a detailed explanation of each step in the OT-PCA method.

3.1 Novel Idea

The Basic Setting Huang et al. [HSC+23] pioneered a new strategy to reduce online PC oracle queries by leveraging the publicly available decoding function $\mathcal{C}.\mathrm{Decode}(\cdot)$. This approach was further refined in the study [SGG24] presented at Usenix Security 2024. In

these studies, the attacker strategically selects r_1 as a polynomial with a Hamming weight of one, specifically X^k, where k is an integer, while setting r_2 to zero. This manipulation yields the ciphertext components $u = X^k$ and $v = m\mathbf{G} + e$. The polynomial fed to the decoding function $\mathcal{C}$.Decode$(\cdot)$ during the decapsulation is

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

Alternatively, by setting $r_2 = X^k$ and r_1 to zero, a different ciphertext configuration emerges: $u = X^k \cdot h$ and $v = m\mathbf{G} + s \cdot X^k + e$. The polynomial fed to the decoding function $\mathcal{C}$.Decode$(\cdot)$ during the decapsulation is

$$v - u \cdot y = m\mathbf{G} + s \cdot X^k + e - X^k \cdot h \cdot y = m\mathbf{G} + x \cdot X^k + e,$$

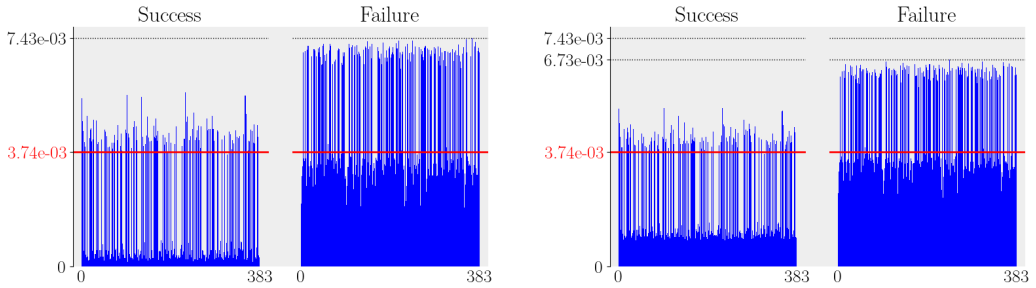
assuming the relationship $s = x + h \cdot y$. With these chosen r_1 and r_2, the attacker can ascertain information about one length-n_2 block of the secret polynomials x and y by designing specific forms of e. The chosen e should have $\delta = \lfloor \frac{d_{\text{RS}}}{2} \rfloor$ blocks flipped to all ones, to ensure the outer Reed Solomon decoding depends on the decoding output of the target Reed Muller block. The decoding output of the target Reed Muller block is sensitive to the values of certain positions in the secret polynomials x or y, since they act as a perturbation of the vector e and together they are the input to the decoder.

Without loss of generality, the attacker can place the target length-n_2 block as the first block from position 0 to 383 by choosing a suitable value of k for X^k. For instance, by choosing $k = -384$, the block with the index set [384..768] is shifted to the first block starting with position 0. A default setting for m is 0.

Constructing Offline Templates Building upon the basic settings proposed in [HSC+23] and [SGG24], we design a more efficient approach for selecting the first length-n_2 block of e and extracting secret information from the constructed PC oracle. The key idea is that for a given vector e, the decryption result depends on the overall error of $e + X^k \cdot y$ (or $e + X^k \cdot x$). The vector $X^k \cdot y$ (or $X^k \cdot x$) is sparse and adding it to e causes flips in e at positions where the corresponding positions in $X^k \cdot y$ (or $X^k \cdot x$) are one. A flip in certain positions will increase the weight of the vector fed to the decoder, thereby increasing the likelihood of a failure. Conversely, a flip in other positions can decrease the weight and thus the likelihood of failure. Therefore, conditional on the received decryption result, either 'Success' or 'Failure', the probability of a certain position in the first block of $X^k \cdot y$ (or $X^k \cdot x$) being one can be either higher or lower than the a-priori probability. This soft information can be obtained by calling the publicly available decoding function $\mathcal{C}$.Decode$(\cdot)$ and can be exploited for efficient full key recovery.

We begin by outlining the construction of the template given a selected error pattern e and a message m, denoted as $\mathcal{T}_{e,m}$. Specifically, we generate 1 000 000 low-weight vectors $\mathbf{u}$ of length n_2, sampled according to their frequency in the secret key distribution. Initially, each vector element has a probability $p_a = \frac{\omega}{n}$ of being one, which for hqc-128, is approximately 3.74×10^{-3}. The vector sum $m\mathbf{G}_{\text{RM}} + e + \mathbf{u}$, where $\mathbf{G}_{\text{RM}}$ is a generator matrix of the Reed Muller code, is then input into the Reed Muller decoder to derive the decoding output, 'Success' or 'Failure', representing whether the decoder outputs the selected message m. To this output, we add a Boolean variable sampled from a Bernoulli distribution $\text{Ber}_{1-\rho}$ to emulate the behavior of the imprecise PC oracle $\mathcal{O}^{\rho}_{\text{PC}}$, which exhibits a decision error probability of $1 - \rho$.

Given knowledge of the sampled vector $\mathbf{u}$, we can estimate the empirical conditional probability $\Pr(\mathbf{u}^i = 1 \mid \mathcal{O}^{\rho}_{\text{PC}})$ for $i \in [0..n_2]$. This represents the likelihood that the ith position in the block is one, conditioned on the Reed Muller decoding outcome. To determine this probability, we categorize $\mathbf{u}$ into two groups based on the decoding results. We then compute the frequency of the ith position being one within each group and normalize this by the total count of vectors in the respective group. This probability acts as an offline template $\mathcal{T}_{e,m}$, providing soft information crucial for further analysis.



(a) $\rho = 1$ (b) $\rho = 0.9$

Figure 3: Constructed offline templates for hqc-128 using a chosen error vector e_0 with the message set to 0. The x-axis represents the position within an Reed Muller block, and the y-axis represents the simulated empirical probability.

Example of Constructed Offline Templates for hqc-128 Figure 3 presents two offline templates for hqc-128, derived from a specific error vector e_0 with the message set to 0. Figure 3a illustrates the case using a perfect oracle ($\rho = 1$), and Figure 3b displays results from an oracle with 90% accuracy ($\mathcal{O}_{\mathrm{PC}}^{0.9}$). In both graphs, the red horizontal line indicates the a-priori probability $p_a \approx 3.74 \times 10^{-3}$, reflecting the initial likelihood of any position being one before decoding outputs are known.

The x-axis of each graph denotes the position within a Reed Muller block, while the y-axis measures the simulated empirical conditional probability of a bit being 1 given the decryption outcome. The decryption outcomes are labeled 'Success' or 'Failure'. With such decryption information, as shown in Figure 3, many positions are significantly more likely to be zero or one. Notably, the comparison between the two panels highlights that a perfect oracle discloses more information, as evidenced by the greater deviations of the conditional probabilities from the a-priori probability p_a. This illustrates a reduced number of required PC oracle calls when utilizing a more accurate PC oracle.

Maximizing Extracted Information by Selecting Appropriate Error Patterns In the previous example, we demonstrated that the empirical conditional probability $\Pr(\mathbf{u}^i = 1 \mid \mathcal{O}_{\mathrm{PC}}^{\rho})$ for $i \in [0..n_2]$ diverges from the a-priori probability p_a once decryption results are disclosed. The attacker selects a suitable message m, compiles a list of t error patterns $\mathcal{E} = \{e_1, e_2, \ldots, e_t\}$, collects the corresponding decryption outputs $\mathbf{o} = (o_1, o_2, \ldots, o_t)$, and performs a posteriori estimation by calculating:

$$\Pr(\mathbf{u}^i = 1 \mid \mathbf{o}) \ \propto \prod_{j=1}^{t} \Pr(\mathbf{u}^i = 1 \mid o_j), \tag{1}$$

assuming independence among decryption outputs. Since our objective is to rank these probabilities for the secret bits, we focus on calculating the right-hand side of Equation 1. The probability $\Pr(\mathbf{u}^i = 1 \mid o_j)$ can be read from the t constructed offline templates $\mathcal{T}_{e_j,m}$, for $i \in [0..n_2]$ and $j \in [1..t+1]$.

A key challenge is determining which error patterns to choose in our attacks, which involves solving two optimization problems.

- **Maximizing Decryption Output Entropy:** Firstly, the PC oracle reveals a Boolean output that can be viewed as a binary random variable. Our goal is to make this random variable as uniform as possible, achieving an entropy close to 1. The binary entropy function $H(p_d)$ is defined as $H(p_d) = -p_d \log_2(p_d) - (1 - p_d) \log_2(1 - p_d)$, where p_d is the probability of decryption failures. By maximizing the

entropy, we ensure that the information obtained from one oracle call is maximized, in accordance with information theory.

- **Making the Errors Different:** Secondly, to maximize information gain from multiple oracle queries, we aim to maximize the sum of Hamming distances:

$$\sum_{1 \le i < j \le t} w_{\mathrm{H}}\left(e_i + e_j\right).$$

To address the first optimization problem, we implement a heuristic algorithm inspired by mutation fuzzers. Our objective is to fine-tune an error pattern such that the Reed Muller decoding result yields uniformly distributed outputs after introducing minor perturbations determined by the secret distribution of y.

We begin with an initial error pattern e_{ini} of dimension n_2, which is randomly sampled to meet the condition:

$$\lfloor \theta_0 \cdot n_2 \rfloor \le \mathrm{wt}(e_{ini}) \le \lfloor \theta_1 \cdot n_2 \rfloor,$$

where $\mathrm{wt}(e_{ini})$ denotes the Hamming weight of e_{ini}, and θ_0 and θ_1 are parameters specific to the target HQC parameter sets. For hqc-128, we choose $\theta_0 = 0.42$ and $\theta_1 = 0.44$, while for hqc-192 and hqc-256, we select $\theta_0 = 0.435$ and $\theta_1 = 0.460$. These parameters are determined experimentally to ensure that the algorithm starts with an effective pattern, allowing it to quickly reach a near-optimal solution.

Next, we employ a simple genetic-style algorithm to refine the error pattern. During the mutation phase, the error pattern is dynamically adjusted by flipping one or two bits, based on the evaluation function $\mathcal{F}(\cdot)$, which calculates the binary entropy function of the output distribution from the Reed Muller decoder. The input to the Reed Muller decoder is the current binary error pattern $\hat{e}$ XORed with a perturbation vector that follows the HQC secret key distribution.

If the entropy is below a specified threshold, two bits are flipped to increase search diversity; otherwise, a single bit is flipped for finer adjustments. Each mutated pattern at iteration k is denoted as $\hat{e}^{(k)}$. The optimization criterion is applied as follows:

$$\hat{e}^{(k+1)} = \underset{\hat{e}' \in \mathrm{Mutate}(\hat{e}^{(k)})}{\arg\max} \mathcal{F}(\hat{e}'),$$

where $\mathrm{Mutate}(\cdot)$ represents the bit-flipping operation.

The algorithm iteratively updates and evaluates each generation, refining the error pattern to maximize fitness until reaching the predetermined number of generations. This process is executed concurrently across hundreds of threads, yielding a long list of optimized e_i. After this procedure, we retain a list of the patterns with binary entropy greater than a threshold, which is set to be 0.999 in our experiments.

The second optimization goal is to identify a set of t vectors from the list that maximize the sum of pairwise Hamming distances. A genetic algorithm with multiple random initializations is applied. Specifically, the algorithm starts with a randomly selected set of t vectors e_i and employs a genetic algorithm that iteratively evolves this set. In each generation, mutations are produced by randomly substituting one vector from the set with a non-duplicate from the list. The set demonstrating the highest fitness is selected as the parent set for the subsequent generation. This process continues until a predefined number of generations is reached, ultimately yielding the optimal set of vectors. To robustly explore the solution space and avoid local optima, we repeat the entire process across numerous random initializations, each independently seeking the optimal combination of vectors. The final output is the set of t vectors that demonstrates the maximum sum of pairwise Hamming distances observed across all iterations.

While the second optimization goal is maintained, it provides only marginal gains. This minimal impact is due to the fact that the two randomly selected elements, e_i and

e_j, already possess a relatively large Hamming distance following the first optimization. Therefore, the additional benefits from this subsequent optimization step are minor.

The offline optimization phase is independent of a specific attack instance. Notably, if these pre-computed error vectors were publicly accessible, they could facilitate the launch of an OT-PCA attack by any adversary.

3.2 Online Attack

Following the ciphertext selection method described in Section 3.1, we feed the selected ciphertexts to the constructed PC oracle to obtain decryption results. The pseudocodes of the online attack phase for this new PC oracle based attack are provided in Figure 4.

In this attack, we define an integer $k = -384 \times i$ for $i \in [0..n_1]$ to shift each of the Reed Muller blocks to the first block. For each shift, we obtain t measurements for the n_2 secret entries in each block. After n_1 shifts, we gather soft information for the first $n_1 \cdot n_2$ secret entries of x and y, respectively. By concatenating these $2n_1 \cdot n_2$ secret entries, we form a new vector $\mathbf{z}$. Consequently, the total number of required online PC oracle calls is $2 \cdot t \cdot n_1$.

Using the constructed offline template, we rank the conditional probability $\Pr(\mathbf{z}_i = 1 \mid \mathbf{o})$ by computing the product $\prod_{i=1}^{t} \Pr(\mathbf{z}_i = 1 \mid o_i)$, for $0 \leq i < 2n_1n_2$. This product provides soft reliability information about the secret entries. A lower value of this product indicates a higher likelihood that the corresponding secret entry is 0, based on the decryption outputs from the PC oracle.

3.3 Full Key Recovery from ISD with Soft Information

During the online attack phase, we collect the reliability information for $2n_1n_2$ positions of the secret (x, y). This enables us to select the most reliable $n + l$ positions for performing Information Set Decoding (ISD) with soft information, where l is a small positive integer. We ensure that the allowable post-processing cost is comparable to Gaussian elimination, facilitating a fair comparison with [SGG24]. We propose a modified Stern-style algorithm with simplified complexity estimation. Alternatively, for improved performance, one might consider adopting a more advanced ISD algorithm with soft information, such as the SoftStern algorithm described in [GJMW19].

For an $n \times 2n$ parity-check matrix, the cost of Gaussian elimination, denoted as C_{Gauss}, is $\mathsf{O}(n^3)$. According to an analysis in [Meu13], the concrete cost of a simple implementation can be estimated as 2^{43}, 2^{46}, and 2^{48} for hqc-128, hqc-192, and hqc-256, respectively.

Quasi-Standard Form We describe a new but simple approach to exploit the soft information to accelerate the ISD algorithm. Suppose we have a system of linear equations

$$\begin{bmatrix} I_n & \text{rot}(h) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = s, \tag{2}$$

which is obtained from the public key $s = x + h \cdot y$, where $\text{rot}(h)$ represents the circulant matrix induced by h. In this matrix, h is placed as the first column, and each subsequent column is a cyclic shift of the previous one.

Finiasz and Sendrier in [FS09] first proposed a general ISD framework—choosing a permutation of the positions and transforming the matrix $\begin{bmatrix} I_n & \text{rot}(h) \end{bmatrix}$ into a quasi-standard form

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

where $\mathbf{Q}'$ is a $\ell \times (n + \ell)$ matrix, $\mathbf{Q}''$ is a $(n - \ell) \times (n + \ell)$ matrix, and $\mathbf{0}$ is a $\ell \times (n - \ell)$ zero matrix.

Algorithm 1: New PC Oracle Based Attack on HQC.

Data: h from the public key, a suitable message m and salt, the t error patterns $\mathcal{E} = \{e_1, e_2, \ldots, e_t\}$, the constructed offline templates $\mathcal{T}_{e_i,m}$ for the selected m and e_i, for $i \in [1..t+1]$, and scheme parameters

Result: Reliability information

Reliability information $R_{i,j} \leftarrow 1.0\ \forall i \in \{\text{“X”, “Y”}\},\ j \in [0..n_1n_2]$
for $i_{tmp} \in [0..n_1]$ **do**
  $k = -384 \times i_{tmp}$
  for $i \in \{\text{“X”, “Y”}\}$ **do**
    for pattern $\in$ the error patterns $\mathcal{E}$ **do**
      Configure the error e by flipping $\delta = \lfloor \frac{d_{\text{RS}}}{2} \rfloor$ Reed Muller code blocks, ensuring that the corruption of the first block results in a Reed Solomon decoder failure.
      set the first block of e to the used pattern
      if $i =$ “X” **then**
        $r_2 \leftarrow X^k$
        $u \leftarrow X^k \cdot h$
      else if $i =$ “Y” **then**
        $r_2 \leftarrow 0$
        $u \leftarrow X^k$
      end
      construct ciphertext c as $(u, m\mathbf{G} + s \cdot r_2 + e, \text{salt})$
      $\mathsf{OutputPC} \leftarrow \mathcal{O}^{\rho}_{\text{PC}}(c)$
      for $j \in [0..n_2]$ **do**
        $R_{i,j-k} = R_{i,j-k} \times \Pr(\mathbf{u}^j = 1 \mid \mathsf{OutputPC})$
        This probability $\Pr(\mathbf{u}^j = 1 \mid \mathsf{OutputPC})$ is derived from the constructed offline template $\mathcal{T}_{\mathsf{pattern},m}$ for the selected m and pattern.
      end
    end
  end
end
return Reliability information $R_{i,j}\ \forall i \in \{\text{“X”, “Y”}\},\ j \in [0..n_1n_2]$

Figure 4: The online phase of the new OT-PCA attack on HQC.

New ISD Exploiting Soft Information Instead of randomly choosing $n + \ell$ positions to form the matrix $\mathbf{Q}'$ as in standard ISD algorithms [Meu13], we leverage the soft information obtained from the online decryption oracle calls. Specifically, we reorder the index set $[0..2n_1n_2]$ in increasing order according to the probability of the secret entries being 1 and select the top $n + \ell$ positions with the smallest probabilities.

Given that the top positions can have very low probabilities of being 1, we introduce two algorithm parameters, T_0 and T_1. We draw T_1 positions from the T_0 positions with the smallest probabilities of being 1, and assume that these T_1 positions are all zero. These positions are then removed from $\widetilde{\mathbf{H}}$. After removing the selected T_1 positions from the $n + \ell$ positions with the smallest probabilities, we denote the resulting index set as $\mathcal{I}$.

Next, we adopt a collision procedure that splits the index set $\mathcal{I}$ into two parts of equal size $\frac{n+\ell-T_1}{2}$, denoted by $\mathcal{I}_A$ and $\mathcal{I}_B$, respectively. The parameter ℓ is picked to ensure that $n + \ell - T_1$ is an even number. We then build two lists to enumerate bit patterns of length $\frac{n+\ell-T_1}{2}$ with Hamming weight bounded by w_0. We include the syndrome information in one list to find collisions that meet the partial syndrome of ℓ positions in

the Finiasz-Sendrier framework. To ensure that the overall complexity of the new ISD is not much larger than one Gaussian elimination, we only consider w_0 values of 2 or 3. The complexity of generating the two lists is thus

$$C_{\text{list}} = 2w_0 \cdot \ell \cdot \binom{\frac{n+\ell-T_1}{2}}{w_0}. \tag{3}$$

The collision procedure will output a list of potential candidates that need to be verified to ensure that Equation 2 is satisfied. We choose ℓ to be sufficiently large, making the cost of this verification step negligible.

Summary Our method involves performing one Gaussian elimination process and n_{iter} inner rounds, with n_{iter} set to 16 by default. The dominant part of the complexity is $\max\{C_{\text{Gauss}}, n_{iter} \cdot C_{\text{list}}\}$. The concrete cost will be calculated in Section 4.1, where experimental parameters determine the success probability. When w_0 is set to 3, the added cost is comparable to Gaussian elimination; when w_0 is set to 2, the added cost is significantly lower than one Gaussian elimination. For the $n+\ell$ positions with the smallest probabilities of being 1, the algorithm will succeed if, during one of the n_{iter} inner rounds, both (1) the T_1 positions set to zero are error-free, and (2) no more than w_0 errors occur in the positions in the index sets $\mathcal{I}_A$ and $\mathcal{I}_B$, respectively. We denote this success probability as P_{success}. The suitable values for parameters T_0 and T_1 are determined empirically.

4 Results

In this section, we present the results of our attack methodology, obtained from extensive simulations and real-world evaluations. The results are divided into two subsections: the first details the success probability of the attack obtained through simulations, considering a general PC oracle constructed from side-channel measurements with varying levels of accuracy. The second describes the real-world evaluation using power traces from a platform equipped with an ARM Cortex-M4 core.

4.1 Simulation Results

We begin with the simulation results, assuming a general PC oracle that can be constructed from side-channel measurements. Extensive simulations were conducted to evaluate the performance of the new attack with 10 000 random keys tested in each instance. These results are organized into two parts: one focusing on hqc-128, and the other on hqc-192 and hqc-256. Given that hqc-128 has been the primary focus in previous research, we conduct a detailed investigation of its attack performance across oracle accuracies of 1, 0.995, 0.95, and 0.9. This comprehensive analysis allows us to predict the performance of our methodology in various scenarios where an accurate PC oracle is difficult to obtain. For hqc-192 and hqc-256, we limit our study to a PC oracle accuracy of 1. While this study primarily focuses on hqc-128, there are no inherent limitations preventing the application of our methodology to hqc-192 and hqc-256 with less accurate oracles.

4.1.1 Targeting hqc-128

In our attack, we aim to shift the ones in the secret key towards the end of the vector after reordering based on the soft information obtained. Non-zero values within the first n positions of this reordered sequence are considered errors. Figure 5 shows the simulated number of errors for different oracle accuracies ρ across various numbers of PC oracle calls per length-n_2 block. Consequently, the total number of PC oracle calls needs to be multiplied by a factor of $2n_1$, which is 92 for hqc-128.

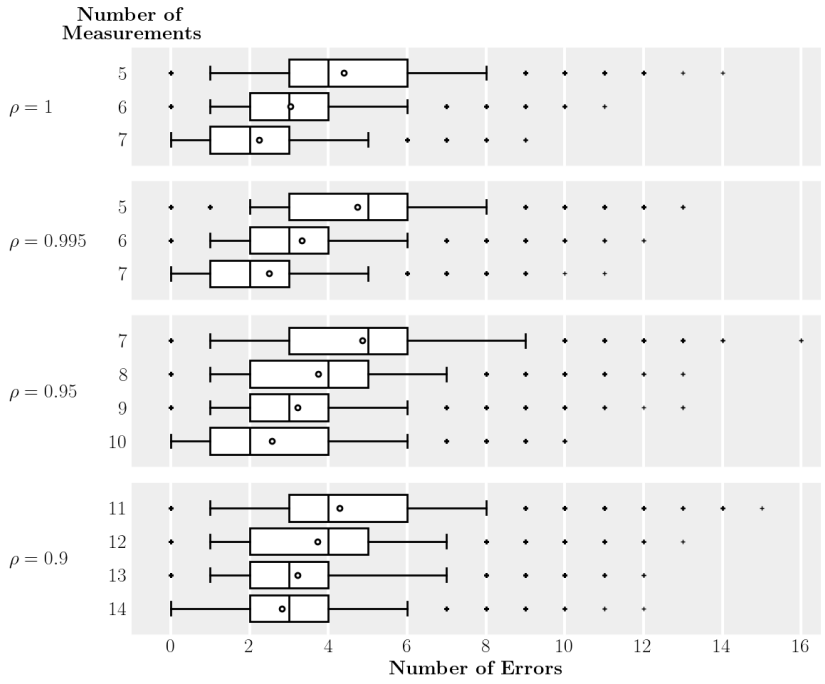


Figure 5: Number of measurements vs. number of errors for hqc-128 with various oracle accuracies, based on 10 000 random keys tested in each scenario. The number of errors in the first n positions of the reordered $\mathbf{z}$ vector decreases as the number of side-channel measurements per length-n_2 block increases. The whiskers represent the 5th and 95th percentiles. The small circles denote the mean number of errors, which are close to the median values indicated by the central line in each box.

It is evident that the number of simulated errors in the first n positions of the reordered vector $\mathbf{z}$ decreases as the number of side-channel measurements increases. We observed in simulation that when the mean error numbers are no more than 5, we achieve a success rate P_{success} higher than 50% when $w_0 = 3$.

Post-Processing Complexity vs. Success Rate In the post-processing phase for hqc-128, we apply our new ISD algorithm with parameters $\ell = 51$, $T_0 = 12\,000$, $T_1 = 10\,000$, and $n_{iter} = 16$. Consequently, the cost of $n_{iter} \cdot C_{\text{list}}$ is approximately 2^{45} for $w_0 = 3$, and 2^{34} for $w_0 = 2$. The list sizes for the two lists in the collision search are 2^{33} for $w_0 = 3$ and 2^{23} for $w_0 = 2$. The simulated success probability P_{success} is presented in Table 3. We observe that the success probability increases drastically as the number of PC oracle calls per length-n_2 block, denoted by t, increases.

Although ideally templates should be tailored to specific oracle precisions, our experiments demonstrate the robustness of a single template constructed using a perfect oracle. This template exhibits consistent effectiveness across various accuracy levels with minimal performance degradation. For instance, when applied to oracles with $\rho = 0.9$, $\omega_0 = 3$, and t ranging from 11 to 14, the success probabilities remained consistently high (65.95%, 75.18%, 82.97%, and 87.02% respectively), closely aligning with the results from Table 3.

Comparison with Previous Works Table 4 compares our results with the state-of-the-art as represented by SCA-LDPC [GNNJ23] and Zero-Tester [SGG24], over a range of oracle accuracies ρ. Both studies report the median number of oracle calls, so for a fair comparison,

Table 3: Simulated success rates P_{success} for hqc-128, illustrating the influence of oracle accuracy (ρ) and the number of PC oracle calls per length-n_2 block (t). The parameter w_0 is defined in Section 3.3. 10 000 random keys are tested in each scenario.

ρ	t	$w_0 = 2$	$w_0 = 3$
1	5	43.42%	65.10%
	6	70.18%	83.53%
	7	85.45%	92.60%
0.995	5	36.65%	59.25%
	6	65.10%	80.61%
	7	82.13%	91.80%
0.95	7	36.03%	57.45%
	8	57.48%	76.27%
	9	68.23%	83.72%
	10	80.12%	90.02%
0.9	11	48.44%	67.58%
	12	58.79%	76.40%
	13	68.05%	83.37%
	14	75.60%	87.55%

we report the number of oracle calls required to achieve a success rate (P_{success}) greater than 50%, indicating that at least half of the attack instances are successful.

Our approach demonstrates remarkable improvements, consistently requiring fewer oracle calls across all tested accuracies, thereby showing substantial efficiencies. The improvement factor is particularly striking, reaching as high as 58.8 compared to SCA-LDPC and 7.6 against Zero-Tester as the oracle accuracy decreases.

Furthermore, our method's performance advantage becomes more pronounced with decreasing oracle accuracy, showing its effectiveness in challenging scenarios. This robustness is essential for practical applications, especially in security-sensitive contexts where effective countermeasures are deployed and perfect oracles are rare.

Table 4: Comparison of median oracle calls. For our work, we report the required number of oracle calls needed to achieve a success rate greater than 50%, indicating that at least half of the attack instances are successful. Our results are compared with SCA-LDPC [GNNJ23] and Zero-Tester [SGG24] for various oracle accuracies ρ. Ratio A shows the ratio of oracle calls for SCA-LDPC to our method, and Ratio B shows the ratio for Zero-Tester to our method, highlighting our approach's efficiency.

ρ	1	0.995	0.95	0.9
SCA-LDPC [GNNJ23]	9 000	18 000	35 250	59 500
Zero-Tester [SGG24]	1 094	2 246	4 922	6 951
Our Work	5×92	5×92	7×92	11×92
Ratio A	19.6	39.1	54.7	58.8
Ratio B	2.4	4.9	7.6	6.9

4.1.2 Targeting hqc-192 and hqc-256

Figure 6 shows the number of errors from simulations for hqc-192 and hqc-256. This analysis varies the number of PC oracle calls per length-n_2 block, assuming a perfect PC oracle has been constructed. For these scenarios, the total number of PC oracle calls also

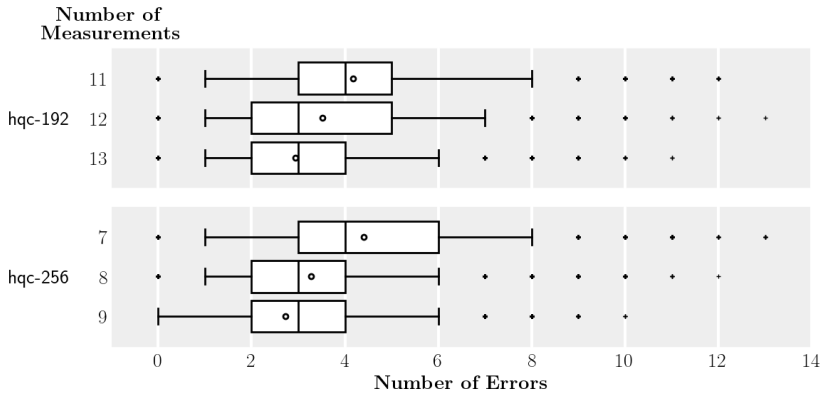


Figure 6: Number of measurements vs. number of errors for hqc-192 and hqc-256 with a perfect oracle, based on 10 000 random keys tested in each scenario. The number of errors in the first n positions of the reordered $\mathbf{z}$ vector decreases as the number of side-channel measurements per length-n_2 block increases. The whiskers represent the 5th and 95th percentiles. The small circles denote the mean number of errors, which are close to the median values indicated by the central line in each box.

Table 5: Simulated success rates P_{success} for hqc-192 and hqc-256, assuming a perfect PC oracle. The number of PC oracle calls per length-n_2 block is denoted by t. Parameter w_0 is detailed in Section 3.3. 10 000 random keys are tested in each scenario.

	t	$w_0 = 2$	$w_0 = 3$
hqc-192	11	47.90%	68.38%
	12	61.93%	78.06%
	13	72.47%	84.81%
hqc-256	7	49.45%	73.01%
	8	70.93%	87.51%
	9	81.37%	92.47%

needs to be adjusted by a factor of $2n_1$, which translates to 112 for hqc-192 and 180 for hqc-256. Similar to hqc-128, the number of simulated errors in the initial n positions of the reordered secret vector decreases steadily as the number of measurements increases.

Post-Processing Complexity vs. Success Rate During the post-processing phase for hqc-192, we implement our novel ISD algorithm with parameters $\ell = 61$, $T_0 = 24\,000$, $T_1 = 20\,000$, and $n_{iter} = 16$. For hqc-256, the parameters are adjusted to $\ell = 61$, $T_0 = 40\,000$, $T_1 = 30\,000$, and $n_{iter} = 16$. This phase reveals that the added computational cost for hqc-192, calculated as $n_{iter} \cdot C_{\text{list}}$, reaches approximately 2^{49} when $w_0 = 3$, and drops to 2^{37} with $w_0 = 2$. The sizes of the two lists used in the collision search are 2^{36} and 2^{25} for $w_0 = 3$ and $w_0 = 2$, respectively. In contrast, for hqc-256, these costs increase to about 2^{51} for $w_0 = 3$ and 2^{38} for $w_0 = 2$, with the corresponding list sizes at 2^{39} and 2^{27}. Tabel 5 showcases the simulated success rates, P_{success}. Mirroring trends observed with hqc-128, there is a marked increase in success probability as the number of PC oracle calls per length-n_2 block (t) increases.

Comparison with Previous Works In the context of key-recovery PC oracle based side-channel attacks against hqc-192 and hqc-256, the fewest number of required PC oracle calls were documented by Schamberger et al. [SHR+22], which is 1920×56 for hqc-192

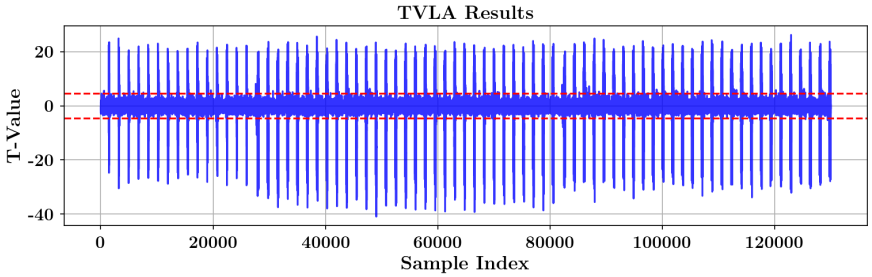
and 1920×90 for hqc-256. Our OT-PCA method achieves significant reductions with only minor post-processing costs—22×56 for hqc-192 and 14×90 for hqc-256. These reductions translate to remarkable factors of 87 for hqc-192 and 137 for hqc-256, demonstrating a major improvement in efficiency.

Additionally, while Schröder et al. in [SGG24] have noted challenges in applying the Zero-Tester method to hqc-192, our methodology, as discussed in Section 5.1, provides robust performance for hqc-192.

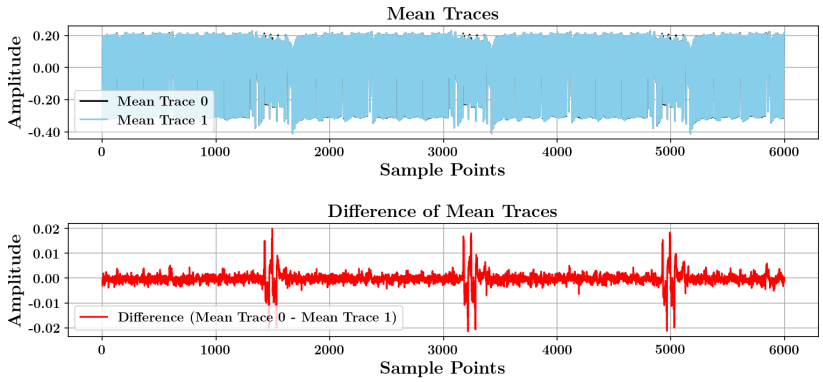
4.2 Real-World Validation

We demonstrate the feasibility and effectiveness of our newly proposed attack methodology by conducting a power analysis attack on a real-world platform.

Experimental Setup Our experimental setup consists of the ChipWhisperer Husky, the CW313 base-board, and an ATSAM4S2A microcontroller target with an ARM Cortex-M4 32-bit RISC core. The operating frequency is set to 24 MHz, with a sampling rate of 24 MS/sec. We target the PQClean implementation of hqc-128. Specifically, we incorporate trigger functions to capture the waveform during the execution of the PQCLEAN_HQC128_CLEAN_code_encode function, which is a crucial part in the re-encryption phase of HQC decapsulation KEM.Decaps. This function calculates the encoding $m'\mathbf{G}$ for the decrypted message m'. The implementation is compiled using arm-none-eabi-gcc with the highest optimization level -O3.



(a) TVLA leakage analysis: The red line represents the t-test threshold at ±4.5.



(b) Mean traces and the difference of mean traces.

Figure 7: Leakage analysis for the encoding function during re-encryption in KEM.Decaps. Each figure was plotted using data from 200 power traces per group. Only partial traces are shown, starting after 300 000 clock cycles. Figure 7a includes a total of 130 000 consecutive data points, while Figure 7b includes 6 000 consecutive data points.

Table 6: CNN hyperparameters (adapted from [UXT+22])

	Operator	Activation function	Batch normalization	Pooling	Stride
Conv1	conv1d (3)	SELU	Yes	Avg (2)	2
Conv2	conv1d (3)	SELU	Yes	Avg (2)	2
Conv3	conv1d (3)	SELU	Yes	Avg (2)	2
Conv4	conv1d (3)	SELU	Yes	Avg (2)	2
Conv5	conv1d (3)	SELU	Yes	Avg (2)	2
Conv6	conv1d (3)	SELU	Yes	Avg (2)	2
FLT	flatten	-	-	-	-
FC1	dense	SELU	No	No	-
FC2	dense	SELU	No	No	-
FC3	dense	Sigmoid	No	No	-

Leakage Detection Previous studies have identified several sources of power leakage that could be utilized to construct the required PC oracle for HQC. Notable examples include power side-channel leakages during hash function computation in the re-encryption process, as reported in [UXT+22], and from the Reed Solomon decoder, as detailed in [SHR+22]. Our research focuses on power traces from the Reed Muller Reed Solomon encoding procedures during the re-encryption of HQC's decapsulation KEM.Decaps phase. This extensive computation phase, primarily dependent on the input m', provides significant leakages, making it ideal for constructing a PC oracle to ascertain decryption outcomes.

In our experiments, the complete trace of this Reed Muller Reed Solomon encoding process during HQC re-encryption spans over 800 000 clock cycles. We analyze 130 000 data points starting from the 300 000th timestamp for each trace. To pinpoint leakage and identify the most significant leakage points, we conduct the Test Vector Leakage Assessment (TVLA) on 200 traces of decryption successes (decrypting to all zeros) and 200 traces of decryption failures. The TVLA results are illustrated in Figure 7a, with a detailed analysis of the initial 6 000 data points presented in Figure 7b to highlight the trace differences. We select sample positions exhibiting t-values greater than 16 in the TVLA test as the strongest leakage points, resulting in 1 280 points per trace.

Experimental Results We employ a convolutional neural network (CNN) architecture, as outlined in Table 6, to train a model capable of classifying whether decryption is successful. This model is adapted from [UXT+22], with adjustments made to the input size accordingly. We capture traces and retain only the 1 280 data points corresponding to positions that exhibit a t-value greater than 16 in our TVLA test, resulting in an input vector of dimension 1 280.

During the training phase, we collect traces from 80 runs of the attack, each using a different newly generated key pair and seven predefined patterns (i.e., $t = 7$). Each key pair produces 644 traces, calculated as $46 \times 7 \times 2$, so we collect a total of 51 520 traces. We separate the dataset into training, validation, and testing sets consisting of 30 418, 10 138, and 10 138 traces, respectively. To ensure balanced samples between the groups of decryption success and decryption failure, some traces are excluded based on the decrypted messages as ground truth. We train the model over 100 epochs. The trained model achieves an accuracy of 1 on the test set, indicating no classification errors on the 10 138 test traces. The model training is performed on a MacBook Pro laptop with a 10-core M1 Pro CPU, a 14-core built-in GPU, and 32GB of RAM.

We then apply the trained model to 77 280 traces collected from 120 newly generated random key pairs to ascertain decryption outcomes. Using the classification results, we employ the offline template designed for a perfect oracle ($\rho = 1$) to compute the conditional probabilities of each secret entry being 1, similar to our earlier simulations. Subsequently, the bits are ranked based on these probabilities. The average number of errors and the success rates of the attack are presented in Table 7, alongside the simulation results. The

Table 7: Comparison of simulation and real-world validation with seven PC oracle calls per length-n_2 block ($t = 7$) for hqc-128. The validation includes 120 real-world tests, each conducted with a different newly generated key pair.

	Mean Number of Errors	Success Rate (P_{success})	
		$w_0 = 2$	$w_0 = 3$
Simulation ($\rho = 1$)	2.25	85.45%	92.60%
Real-world validation	2.35	84.17%	92.50%

results from the real-world validation and the simulations align perfectly.

5 Discussion

In this section, we discuss the gains of our attack methodology compared to the state-of-the-art Zero-Tester method, and its potential use in constructing multi-value or parallel PC oracle based side-channel attacks on HQC.

5.1 More on Comparison with the Zero-Tester Method

In Section 4.1, we demonstrated that our new method requires significantly fewer oracle calls compared to the previous best, the Zero-Tester method [SGG24]. Additionally, our method is more robust to the inaccuracy in the constructed PC oracle.

Another notable advantage of our approach over the Zero-Tester method is its ability to address a limitation discussed in Section 6.3 of [SGG24]. The effectiveness of the Zero-Tester method depends on identifying long sequences of consecutive zero entries between two ones in the secret polynomials x and y. Even with an increased number of side-channel measurements, the Zero-Tester method is effective for only a small proportion (approximately 15%) of keys for hqc-192, where the likelihood of finding an all-zero block of length n_2 within the secret vectors x and y is low.

As shown in Figure 8, our method effectively reduces the number of decision errors in the first n positions of the reordered $\mathbf{z}$ vector as the number of side-channel measurements increases for hqc-192. The x-axis represents the number of measurements per Reed Muller block (t). Detailed information on sample complexity, post-processing complexity, and the probability of full key recovery for hqc-192 with an oracle accuracy of $\rho = 1$ is provided in Section 4.1.

5.2 Extension to Multi-value or Parallel PC oracle

Rajendran et al. [RRD+23] and Tanaka et al. [TUX+23] were the first to propose side-channel attacks based on multi-value or parallel PC oracles. In [RRD+23], the oracle is referred to as a parallel PC oracle, while in [TUX+23], it is termed a multi-value oracle. The concept behind these oracles is that, in specific scenarios such as unprotected software implementations on low-end platforms, power or EM leakages are substantial enough to construct a multi-value classifier. This allows a multi-value or parallel PC oracle to leak more than one bit of information, thereby significantly reducing the number of required traces. While these attacks have limited applications, they are more powerful than binary PC oracle based attacks when a multi-value or parallel PC oracle can be built. As noted by Tanaka et al. in [TUX+23], constructing such attacks for HQC remains an open problem.

The new OT-PCA method can be extended to address this open problem by creating different offline templates for more than two decrypted messages. Preliminary investigations have demonstrated gains compared to binary PC oracle based side-channel attacks. However, extensive experiments are necessary to evaluate and optimize these concrete

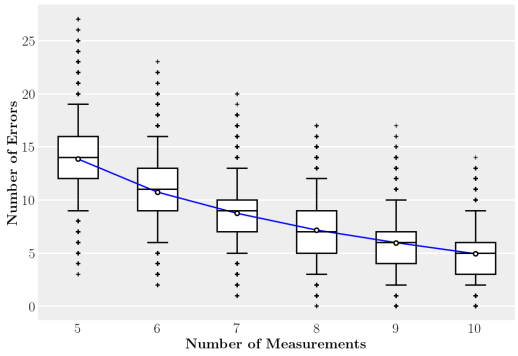


Figure 8: The number of errors in the first n positions of the reordered $\mathbf{z}$ vector decreases as the number of side-channel measurements per length-n_2 block increases for hqc-192 with an oracle accuracy of $\rho = 1$. The plot is based on testing 10 000 random keys for each scenario. The whiskers represent the 5th and 95th percentiles. The blue line connects the mean number of errors.

performance gains. We leave this research problem for future work because multi-value or parallel PC oracles are important oracles that deserve further and separate investigation.

6 Concluding Remarks

We have introduced OT-PCA, a novel key-recovery PC oracle based side-channel attack against HQC, utilizing templates constructed from offline access to the publicly available decoding function of Reed Muller codes. These templates provide valuable soft information for the post-processing stage. Our method requires minimal post-processing costs, comparable to Gaussian elimination. Simulation results show that OT-PCA significantly reduces the required number of PC oracle calls compared to state-of-the-art attacks. Furthermore, this new attack is more robust to inaccuracy in the PC oracle and performs significantly better than the method in [SGG24] on hqc-192. The effectiveness of this approach has also been validated through real-world tests using power traces on a ChipWhisperer Husky platform equipped with an ARM Cortex-M4 CPU, confirming the simulation results and showcasing the practical relevance of our method.

Countermeasures like constant-time implementation, masking, or shuffling can make constructing accurate PC oracles difficult or infeasible, thus hardening the attack. Future work includes a comprehensive evaluation of multi-value or parallel PC oracle based side-channel attacks using the OT-PCA method on HQC to understand its efficiency and potential limitations. Additionally, evaluating software PC oracle based side-channel attacks, such as GoFetch [CWS+24], against HQC will be valuable. Last, extending the OT-PCA method to Kyber and other lattice-based cryptographic schemes will provide insights into the implementation security of various lattice-based proposals.

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 Swedish Foundation for Strategic Research (grant number RIT17-0005), 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

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[ABB+22] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar-Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[ACD+22] Gorjan Alagic, David Cooper, Quynh Dang, Thinh Dang, John M. Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A. Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Daniel Apon. Status report on the third round of the nist post-quantum cryptography standardization process, 2022-07-05 04:07:00 2022.

[BDH+19] Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse attacks on post-quantum cryptosystems. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 747–776. Springer, Cham, May 2019.

[CWS+24] Boru Chen, Yingchen Wang, Pradyumna Shome, Christopher W. Fletcher, David Kohlbrenner, Riccardo Paccagnella, and Daniel Genkin. Gofetch: Breaking constant-time cryptographic implementations using data memory-dependent prefetchers. In *USENIX Security*, 2024.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999.

[FS09] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, Berlin, Heidelberg, December 2009.

[GHJ+22] Qian Guo, Clemens Hlauschek, Thomas Johansson, Norman Lahr, Alexander Nilsson, and Robin Leander Schröder. Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE. *IACR TCHES*, 2022(3):223–263, 2022.

[GJ20] Qian Guo and Thomas Johansson. A new decryption failure attack against HQC. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 353–382. Springer, Cham, December 2020.

[GJMW19] Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski Wagner. Some cryptanalytic and coding-theoretic applications of a soft stern algorithm. *Advances in Mathematics of Communications*, 13(4), 2019.

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 359–386. Springer, Cham, August 2020.

[GLG22] Guillaume Goy, Antoine Loiseau, and Philippe Gaborit. A new key recovery side-channel attack on HQC with chosen ciphertext. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 353–371. Springer, 2022.

[GMGL24] Guillaume Goy, Julien Maillard, Philippe Gaborit, and Antoine Loiseau. Single trace HQC shared key recovery with SASCA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):64–87, 2024.

[GNNJ23] Qian Guo, Denis Nabokov, Alexander Nilsson, and Thomas Johansson. SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 203–236. Springer, Singapore, December 2023.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017.

[HPP21] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-enabled chosen-ciphertext attacks on kyber. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, volume 13143 of *Lecture Notes in Computer Science*, pages 311–334. Springer, 2021.

[HSC+23] Senyang Huang, Rui Qi Sim, Chitchanok Chuengsatiansup, Qian Guo, and Thomas Johansson. Cache-timing attack against HQC. *IACR TCHES*, 2023(3):136–163, 2023.

[HV20] Loïs Huguenin-Dumittan and Serge Vaudenay. Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20International Conference on Applied Cryptography and Network Security, Part I*, volume 12146 of *LNCS*, pages 208–227. Springer, Cham, October 2020.

[JAC+22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[Meu13] Alexander Meurer. *A coding-theoretic approach to cryptanalysis*. PhD thesis, Verlag nicht ermittelbar, 2013.

[MKB+24] Puja Mondal, Suparna Kundu, Sarani Bhattacharya, Angshuman Karmakar, and Ingrid Verbauwhede. A practical key-recovery attack on LWE-based key-encapsulation mechanism schemes using rowhammer. In Christina Pöpper and Lejla Batina, editors, *ACNS 24International Conference on Applied Cryptography and Network Security, Part III*, volume 14585 of *LNCS*, pages 271–300. Springer, Cham, March 2024.

[PRJB23] Thales Paiva, Prasanna Ravi, Dirmanto Jap, and Shivam Bhasin. Et tu, brute? SCA assisted CCA using valid ciphertexts - A case study on HQC KEM. Cryptology ePrint Archive, Report 2023/1626, 2023.

[PT19] Thales Bandiera Paiva and Routo Terada. A timing attack on the HQC encryption scheme. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 551–573. Springer, Cham, August 2019.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR TCHES*, 2020(3):307–335, 2020.

[RRD+23] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the limits of generic side-channel attacks on LWE-based KEMs - parallel PC oracle attacks on Kyber KEM and beyond. *IACR TCHES*, 2023(2):418–446, 2023.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SCZ+23] Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. Find the bad apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of Kyber. *IACR TCHES*, 2023(1):89–112, 2023.

[SGG24] Robin Leander Schröder, Stefan Gast, and Qian Guo. Divide and surrender: Exploiting variable division instruction timing in HQC key recovery attacks. In *USENIX Security*. USENIX Association, 2024.

[SHR+22] Thomas Schamberger, Lukas Holzbaur, Julian Renner, Antonia Wachter-Zeh, and Georg Sigl. A Power Side-Channel Attack on the Reed-Muller Reed-Solomon Version of the HQC Cryptosystem. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*,

volume 13512 of *Lecture Notes in Computer Science*, pages 327–352. Springer, 2022.

[TUX+23] Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. *IACR TCHES*, 2023(3):473–503, 2023.

[UXT+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. *IACR TCHES*, 2022(1):296–322, 2022.

[WPH+22] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 679–697. USENIX Association, August 2022.

[WTBB+19] Guillaume Wafo-Tapa, Slim Bettaieb, Loic Bidoux, Philippe Gaborit, and Etienne Marcatel. A practicable timing attack against HQC and its countermeasure. Cryptology ePrint Archive, Report 2019/909, 2019.

[XIU+21] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-injection attacks against NIST's post-quantum cryptography round 3 KEM candidates. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 33–61. Springer, Cham, December 2021.