IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2023, No. 2, pp. 418–446. DOI:10.46586/tches.v2023.i2.418-446

Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond

Gokulnath Rajendran[2], Prasanna Ravi[1,2], Jan-Pieter D'Anvers[3], Shivam Bhasin[1] and Anupam Chattopadhyay[1,2]

[1] Temasek Laboratories, Nanyang Technological University, Singapore
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3] imec-COSIC KU Leuven, Kasteelpark Arenberg 10 - bus 2452, 3001 Leuven, Belgium
gokulnat002@e.ntu.edu.sg prasanna.ravi@ntu.edu.sg
janpieter.danvers@esat.kuleuven.be sbhasin@ntu.edu.sg anupam@ntu.edu.sg

Abstract. In this work, we propose generic and novel adaptations to the binary Plaintext-Checking (PC) oracle based side-channel attacks for Kyber KEM. These attacks operate in a chosen-ciphertext setting, and are fairly generic and easy to mount on a given target, as the attacker requires very minimal information about the target device. However, these attacks have an inherent disadvantage of requiring a few thousand traces to perform full key recovery. This is due to the fact that these attacks typically work by recovering a single bit of information about the secret key per query/trace. In this respect, we propose novel *parallel PC oracle* based side-channel attacks, which are capable of recovering a generic P number of bits of information about the secret key in a single query/trace. We propose novel techniques to build chosen-ciphertexts so as to efficiently realize a parallel PC oracle for Kyber KEM. We also build a multi-class classifier, which is capable of realizing a practical side-channel based parallel PC oracle with very high success rate. We experimentally validated the proposed attacks (upto $P = 10$) on the fastest implementation of unprotected Kyber KEM in the *pqm4* library. Our experiments yielded improvements in the range of 2.89× and 7.65× in the number of queries, compared to state-of-the-art binary PC oracle attacks, while arbitrarily higher improvements are possible for a motivated attacker, given the generic nature of the proposed attacks. We further conduct a thorough study on applicability to different scenarios, based on the presence/absence of a clone device, and also partial key recovery. Finally, we also show that the proposed attacks are able to achieve the lowest number of queries for key recovery, even for implementations protected with low-cost countermeasures such as shuffling. Our work therefore, concretely demonstrates the power of PC oracle attacks on Kyber KEM, thereby stressing the need for concrete countermeasures such as masking for Kyber and other lattice-based KEMs.

Keywords: lattice-based cryptography · Side-Channel Analysis · Kyber · Plaintext-Checking Oracle · Chosen-Ciphertext Attack · Key Encapsulation Mechanism

1 Introduction

NIST very recently announced results for the third round of the Post-Quantum Cryptography (PQC) standardization process [AAC+22], in which CRYSTALS-Kyber [ABD+21] was selected as the sole candidate for standardization of Key Encapsulation Mechanisms (KEMs). The security of Kyber is based on the well known Module Learning With Errors (MLWE) problem, and served as one of the most promising candidates for KEMs in the

Licensed under Creative Commons License CC-BY 4.0.
Received: 2022-10-15 Accepted: 2022-12-15 Published: 2023-03-06

NIST PQC process, owing to the confidence in its theoretical security guarantees, while also offering one of the best implementation performance compared to other PQC based KEMs [AASA+20]. Thus, one can expect Kyber KEM to be implemented and designed on a wide-variety of computational platforms and applications, in the coming years.

In this respect, security of Kyber against physical attacks such as Side-Channel Attacks (SCA) naturally arises as an immediate concern, particularly for applications involving embedded devices. NIST had also particularly encouraged more research on analysing the security of PQC schemes against SCA [AASA+20]. The cryptographic community has shown significant interest towards research on development of new attacks on several lattice-based schemes including Kyber KEM [RBRC21, RRCB20, BDH+21], as well as development of efficient side-channel protection techniques [HKL+22, BGR+21]. While there exists a wide variety of SCA particularly on Kyber KEM, we observe that they can be broadly classified into two main categories.

The first category of attacks only require a single trace or very few traces to perform key recovery or message recovery [NDGJ21, RBRC21, ACLZ20]. However, these attacks typically target very precise leakages from targeted operations within the scheme. They typically require a fairly sophisticated setup, as well as a detailed knowledge of the target device. Moreover, exploitation of such leakages on different implementations and targets, is not very straightforward, and at the very least requires significant adaptations.

The second category of attacks are more generic, and exploit inherent vulnerabilities in the algorithm for key recovery, while remaining relatively somewhat agnostic to the target/implementation. These generic attacks typically work by querying the target device with chosen-ciphertexts, and subsequently utilizing leakage from the decapsulation of chosen-ciphertexts as an oracle, to recover information about the secret key [RRCB20, UXT+22, SCZ+22]. However, such side-channel assisted chosen-ciphertext attacks typically suffer from a disadvantage of requiring a few thousand queries for key recovery. In particular, the attacks realizing such a Plaintext-Checking (PC) oracle typically exploit 1-bit of information about the secret key (i.e.) binary PC oracle, thereby requiring a few thousand queries/traces for full key recovery. Thus, we observe a clear trade-off for both the categories of attacks, based on the ease of mounting the attack versus number of traces for key recovery.

In this work, we attempt to *bridge this gap* through our proposal of *parallel* PC oracle attacks for LWE-based KEMs, with main focus on Kyber KEM. While all our attacks are demonstrated on Kyber KEM, we believe our attack can be adapted to similar LWE/LWR-based KEMs such as Saber [DKR+20], FrodoKEM [ABD+20] etc. The main contributions of our work are as follows:

1. We propose generic and novel adaptations of side-channel assisted binary PC oracle-based attacks, referred to as P-way parallel PC oracle attacks, which have the ability to simultaneously recover an arbitrary P number of bits of information about the secret key per query, while state-of-the-art PC oracle attacks are only capable of extracting one bit per query.

2. We identify that existing approaches to construct chosen-ciphertext queries for optimal key recovery in the case of binary PC oracle attacks, are not always optimal for the proposed PC oracle attacks in the parallel setting. We therefore propose improved constructions of Binary Decision Trees (BDTs) to identify lower bounds for the number of queries for the proposed attacks.

3. We also adapt the binary side-channel classifiers used for the binary PC oracle attack, to develop multi-class classifiers for an arbitrary 2^P number of classes to realize a practical P-way parallel PC oracle. We practically validated that our multi-class classifier is able to uniquely classify between 1024 classes (for $P = 10$) with 100%

success rate. While higher values of P are possible in theory, it is hard to estimate a bound on the highest value of P that is possible to achieve in practice.

4. We experimentally validated our attacks on the fastest implementation of Kyber KEM in the *pqm4* library [KRSS19], a well known benchmarking and testing framework for PQC schemes on the ARM Cortex-M4 microcontroller. We practically validated improvements in the range of 2.89× and 7.65×, compared to state-of-the-art binary PC oracle attacks. However, we note that significant improvements are possible as shown later in the paper. Such improvements provide concrete inputs to a designer for determining safe key refresh rates when ephemeral setting for key-exchange is not possible.

5. We also conduct a comprehensive analysis of the capabilities of our attack in different attack scenarios, based on (1) the presence/absence of a clone device and (2) partial key recovery for attackers with different capabilites for offline computation. We observe that our attack brings about arbitrarily high improvements in the number of traces, especially when the attacker can construct a very high number of templates on the clone device[1]

6. We also practically validated the applicability of our attacks to implementations protected with low-cost countermeasures such as shuffling. Our attack yields the lowest number of traces compared to existing state-of-the-art side-channel assisted chosen-ciphertext attacks targeting the shuffled implementation of Kyber KEM.

2 Preliminaries

2.1 Notation

We denote the ring of integers modulo $q \in \mathbb{Z}^+$ as $\mathbb{Z}_q$. Elements in $\mathbb{Z}_q$ are denoted using lower case letters (i.e.) $a \in \mathbb{Z}_q$, and the i^{th} bit of $a \in \mathbb{Z}_q$ is denoted as a_i. We use R_q to denote the polynomial ring $\mathbb{Z}_q[x]/(x^n + 1)$ and polynomials in R_q are denoted using bold lower case letters (i.e.) $\mathbf{a} \in R_q$. The i^{th} coefficient of $\mathbf{a} \in R_q$ is denoted as $\mathbf{a}[i]$. A vector of polynomials in R_q (i.e.) R_q^k with $k \in \mathbb{Z}^+$ is denoted using bold lower case letters, while a matrix of polynomials in $R_q^{k\times\ell}$ with $(k, \ell) \in \mathbb{Z}^+$ are denoted using bold upper case letters. The i^{th} polynomial of $\mathbf{a} \in R_q^k$ is denoted as $\mathbf{a}_i$. Matrices and vectors of polynomials in R_q are together referred to as *modules*. The product of two polynomials $\mathbf{a}$ and $\mathbf{b}$ in R_q is denoted as $\mathbf{c} = \mathbf{a} \cdot \mathbf{b} \in R_q$, while pointwise multiplication using $\circ$ (i.e.) $\mathbf{c} = \mathbf{a} \circ \mathbf{b} \in R_q$. Byte arrays of length n are denoted as $\mathcal{B}^n$. The transpose of a matrix $\mathbf{A}$ is denoted as $\mathbf{A}^T$. The NTT representation of $\mathbf{a} \in R_q$ is denoted as $\hat{\mathbf{a}} \in R_q$.

2.2 Kyber KEM

Kyber is a chosen-ciphertext secure KEM (IND-CCA), built upon the hardness of the Module-LWE (MLWE) problem. It offers three parameter sets, hereby listed in increasing levels of security - (1) `Kyber-512` (NIST Security Level 1), (2) `Kyber-768` (Level 3) and (3) `Kyber-1024` (Level 5) with $k = 2, 3$ and 4 respectively. The CCA secure Kyber KEM is built upon a simpler chosen-plaintext secure PKE (IND-CPA), denoted as CPA.Kyber PKE. Refer to Algorithm 1 for a simplified description of the key-generation (CPA.KeyGen), encryption (CPA.Encrypt) and decryption procedures (CPA.Decrypt) of CPA.Kyber PKE. Sample_U is used to denote the operation sampling coefficients from a uniform distribution, Sample_B to denote sampling from a centered binomial distribution (CBD) and the function

[1] $\approx 25\times$ improvement for an arbitrarily strong attacker capable of constructing 2^{32} templates on the clone device

Expand, to denote expanding a small seed into a uniformly random matrix in $R_q^{k\times k}$. The function $\mathsf{Compress}(\mathbf{u}, d)$ lossily compresses $\mathbf{u} \in \mathbb{Z}_q$ into $v \in \mathbb{Z}_{2^d}$ with $q > 2^d$, while $\mathsf{Decompress}(\mathbf{v}, d)$ extrapolates $\mathbf{v} \in \mathbb{Z}_{2^d}$ into $u' \in \mathbb{Z}_q$.

Algorithm 1 CPA Secure Kyber PKE (Simplified)

procedure CPA.KeyGen
- $(seed_A, seed_B) \in \mathcal{B}^* \leftarrow \mathsf{Sample}_U()$ ▷ Generate uniform seeds $seed_A, seed_B$
- $\hat{\mathbf{A}} \in R_q^{k\times k} \leftarrow \mathsf{Expand}(seed_A)$ ▷ Expand $seed_A$ into $\hat{\mathbf{A}}$
- $\mathbf{s}, \mathbf{e} \in (R_q^k \times R_q) \leftarrow \mathsf{Sample}_B(seed_B, coins)$ ▷ Sample $\mathbf{s}, \mathbf{e}$
- $\hat{\mathbf{s}} \in R_q^k \leftarrow \mathsf{NTT}(\mathbf{s})$; $\hat{\mathbf{e}} \in R_q^k \leftarrow \mathsf{NTT}(\mathbf{e})$ ▷ $\mathsf{NTT}(\mathbf{s})$, $\mathsf{NTT}(\mathbf{e})$
- $\hat{\mathbf{t}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$ ▷ $\mathbf{t} = \mathbf{A}\cdot\mathbf{s} + \mathbf{e}$ in NTT domain
- Return $(pk = (seed_A, \hat{\mathbf{t}}), sk = (\hat{\mathbf{s}}))$

end procedure

---

procedure CPA.Encrypt$(pk, m \in \{0,1\}^{256}, seed_R \in \{0,1\}^{256})$
- $\hat{\mathbf{A}} \in R_q^{k\times k} \leftarrow \mathsf{Expand}(seed_A)$
- $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \in (R_q^k \times R_q^k \times R_q) \leftarrow \mathsf{Sample}_B(seed_R)$ ▷ Sample $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2}$
- $\hat{\mathbf{r}} \in R_q^k \leftarrow \mathsf{NTT}(\mathbf{r})$ ▷ $\mathsf{NTT}(\mathbf{r})$
- $\mathbf{u} \in R_q^k \leftarrow \mathsf{INTT}(\mathbf{A}^T \circ \hat{\mathbf{r}}) + \mathbf{e_1}$ ▷ $\mathbf{u} = \mathbf{A}^T \cdot \mathbf{r} + \mathbf{e_1}$
- $\mathbf{v}' \in R_q \leftarrow \mathsf{INTT}(\hat{t}^T \circ \hat{\mathbf{r}}) + \mathbf{e_2}$ ▷ $\mathbf{v}' = \mathbf{t}^T \cdot \mathbf{r} + \mathbf{e_2}$
- $\mathbf{v} \in R_q \leftarrow \mathbf{v} + \mathsf{Decompress}(m, 1)$ ▷ $\mathbf{v} = \mathbf{v} + \mathsf{Encode}(m)$
- Return $ct = \mathsf{Compress}(\mathbf{u}, d_1), \mathsf{Compress}(v, d_2)$

end procedure

---

procedure CPA.Decrypt(sk, ct)
- $(\mathbf{u}, \mathbf{v}) \leftarrow \mathsf{Decompress}(ct, d_1, d_2)$
- $\hat{\mathbf{u}} = \mathsf{NTT}(\mathbf{u})$
- $\mathbf{m} = \mathbf{v} - \mathsf{INTT}(\hat{\mathbf{u}} \circ \mathbf{s})$ ▷ $\mathbf{m} = \mathbf{v} - \mathbf{u}\cdot\mathbf{s}$
- $m \in R_q \leftarrow \mathsf{Compress}(\mathbf{m}, 1)$ ▷ Decoding $\mathbf{m} \in R_q$ into $m \in \mathcal{B}^{32}$
- Return m

end procedure

2.2.1 IND-CCA Security

The IND-CPA secure PKE is transformed into an IND-CCA secure KEM using a post-quantum variant of well-known Fujisaki-Okamoto transformation [FO99]. It involves the use of two hash functions $(\mathcal{H}, \mathcal{G})$ and a key-derivation function KDF, forming a wrapper denoted as encapsulation (CCA.Encaps) and decapsulation (CCA.Decaps) procedures of CCA.Kyber KEM (Refer Alg.2).

Within this framework, the encryption procedure is deterministic and depends solely on the message m for a given public key pk. This is done by ensuring that the seed input r' to the encryption procedure is derived by hashing m with pk (Line 4-5 in CCA.Encaps). In the decapsulation procedure (CCA.Decaps), the decrypted message m is *re-encrypted* (Line 12-13) to generate the ciphertext ct'. The received ciphertext ct is compared with ct', and the valid shared key is generated (Line 17) only if $ct = ct'$, denoting a valid ciphertext, else a pseudo-random key is generated (invalid ciphertext). This enables to detect invalid/malicious ciphertexts, thereby offering concrete protection against chosen-ciphertext attacks. We refer the reader to [ABD+21] for more details on CCA secure Kyber KEM.

Algorithm 2 CCA secure Kyber KEM

procedure CCA.ENCAPS(pk)
    $m \leftarrow \{0,1\}^{256}$
    $m' = \mathcal{H}(m)$
    $(\bar{K}', r') = \mathcal{G}(m' \| \mathcal{H}(pk))$ ▷ Generate $\bar{K}'$ and r' using m and pk
    $ct = \mathsf{CPA.Encrypt}(pk, m', r')$
    $K = \mathsf{KDF}(\bar{K}' \| \mathcal{H}(ct))$
    Return (ct, K)
end procedure

procedure CCA.DECAPS(sk, ct)
    $(pk, \mathcal{H}(pk), z) \leftarrow \mathsf{UnpackSK}(sk)$
    $m = \mathsf{CPA.Decrypt}(sk, ct)$
    $(\bar{K}, r) = \mathcal{G}(m, \mathcal{H}(pk))$
    $ct' = \mathsf{CPA.Encrypt}(pk, m, r)$ ▷ Re_Encrypt(m, pk)
    if $(ct' == ct)$ **then**
        Return $K = \mathsf{KDF}(\bar{K} \| \mathcal{H}(ct'))$ ▷ Ciphertext Comparison Success
    else
        Return $K = \mathsf{KDF}(z \| \mathcal{H}(ct'))$ ▷ Ciphertext Comparison Failure
    end if
end procedure

2.3 Prior Side-Channel Attacks and Motivation

LWE/LWR-based KEMs including Kyber KEM have been subjected to a wide variety of side-channel attacks whose primary target is the long-term secret key $\mathbf{s}$ used in the decapulation procedure. Recovery of a single secret key $\mathbf{s}$ leads to compromise of all the corresponding session keys K, that were derived using $\mathbf{s}$.

In this respect, we can broadly classify existing attacks on Kyber KEM into two categories: (1) Target Operation Independent attacks (TO_Indep) and (2) Target Operation Dependent (TO_Dep) attacks. TO_Dep attacks are those that exploit side-channel leakage from a specific targeted operation within the decapsulation procedure [XPR+21, ACLZ20, PPM17]. On the other hand, TO_Indep attacks are not limited to any single operation, but can collectively exploit leakage from several operations within the decapsulation procedure [RRCB20, DTVV19, GJN20]. Moreover, TO_Indep attacks are generic and to a certain degree, agnostic to the target implementation, while TO_Dep attacks are more specific to the target device/implementation.

2.3.1 Target Operation Independent Attacks (TO_Indep)

Several works have shown that an adversary with the ability to query the decapsulation device with chosen-ciphertexts can amplify side-channel leakage related to the secret key [RRCB20, DTVV19, GJN20]. The modus operandi of such attacks is as follows: The attacker submits malformed ciphertexts ct to the decapsulation device, whose corresponding decrypted message m has a close relation to the secet key $\mathbf{s}$. An attacker who can exploit side-channel leakage to recover information about m, in effect instantiates an *oracle* which leads to recovery of $\mathbf{s}$. These attacks can be further classified into two categories:

Plaintext-Checking (PC) Oracle-based SCA: D'Anvers *et al.* [DTVV19] demonstrated that chosen-ciphertexts can be constructed to restrict the decrypted message to a only two possible values (i.e.) $m = 0$ or $m = 1$, with the value of m depending on targeted single coefficients of $\mathbf{s}$. They utilized the timing side-channel from variable time error correcting codes, to recover m (i.e.) $\mathsf{Time}(\mathsf{ECC\_Decode}(0))! = \mathsf{Time}(\mathsf{ECC\_Decode}(1))$, thereby instantiating a Plaintext-Checking (PC) oracle. Ravi *et al.* [RRCB20] subsequently generalized the attack to multiple LWE/LWR-based KEMs including Kyber KEM through the EM side-channel. They showed that a single bit change between $m = 0$ and $m = 1$ results in vastly different computations in the re-encryption procedure, which can be easily distinguished in a single trace. Every chosen-ciphertext query only enables recovery of a single bit information about the secret $\mathbf{s}$, thus full key recovery is only possible in a few thousand queries ($\approx 2k - 3k$).

Decryption-Failure (DF) Oracle-based SCA: These attacks work by submitting perturbed ciphertexts $ct' = ct_{\mathsf{valid}} + \epsilon$ to the target, which induce a *decryption failure* depending on the value of the secret key $\mathbf{s}$. An attacker who can detect a decryption failure (i.e.) $m = m_{\mathsf{valid}}$ or $m = m_{\mathsf{invalid}}$ can construct *linear hints* about the secret key, enabling full key recovery in a few thousand queries ($5k - 6k$).

Guo *et al.* [GJN20] exploited variable time implementations of the ciphertext comparison operation in the decapsulation procedure (Line 14 in CCA.Decaps of Alg.2) to instantiate a DF oracle, while subsequent works [BDH+21, DHP+22] demonstrated exploitation of the EM side-channel for key recovery. While these attacks specifically targeted the ciphertext comparison operation for leakage, these attack can also exploit leakage from the entire re-encryption procedure of an unprotected implementation, to easily distinguish between $\mathsf{Re-Encrypt}(m_{\mathsf{valid}}, pk)$ and $\mathsf{Re-Encrypt}(m_{\mathsf{invalid}}, pk)$.

In essence, all the aforementioned attacks recover upto a single bit information about the secret key sk, thereby instantiating a *binary* oracle, which requires a few thousand queries for key recovery. These attacks are particularly attractive for their inherent *simplicity* in performing key recovery, where the attacker requires very minimal information about the underlying target.

2.3.2 Target Operation Dependent Attacks (TO_Dep)

These attacks target leakage from specific leaky operations in the decapsulation procedure for key recovery. We discuss two major types of attacks.

Targeting Message Encoding/Decoding Operation: Several attacks have targeted the message encoding (Line 15 of CPA.Encrypt in Alg.1) and message decoding (Line 22 of CPA.Decrypt) operations, enabling recovery of the entire message m in a single trace [ACLZ20, SKL+20, RBRC21, NDJ21]. These attacks mainly exploit very fine leakages from manipulations of single bits of the sensitive message m, enabling full message recovery. Xu *et al.* [XPR+21] showed that the aforementioned leakage can be exploited to instantiate a *Full Decryption* (FD) oracle in a chosen-ciphertext setting. This enabled full key recovery in only 6 queries for Kyber512. Adaptations of the same attack has also been demonstrated on masked and shuffled implementations of Kyber KEM [NDJ21, NDGJ21].

While these attacks consume very few queries for full key recovery, they suffer from inherent disadvantages. Firstly, they exploit very fine leakages from single bit manipulations of the message. Thus, leakage is limited to single clock cycles or very few samples for each message bit, thereby naturally being sensitive to acquisition noise (SNR), as shown in [RBRC21]. Moreover, it is not clear if similar leakages can be exploited on complex devices with features such as heavy parallelism, deep pipelining and inherent jitter, especially given the sensitivity of these attacks to noise.

Targeting NTT Operation: Several attacks have targeted the Number Theoretic Transform (NTT), used for polynomial multiplication in Kyber KEM [PPM17, PP19]. These attacks enable full key recovery in a single trace or very few traces, exploiting advanced algebraic side-channel techniques. However, they also suffer from the same disadvantages of exploiting fine leakages from multiple targeted instructions, while also requiring a sophisticated setup or a powerful side-channel adversary for key recovery. While improved attacks to counter noise have been demonstrated [HHP+21], the attacks still are relatively hard to implement. Moreover, the threat of the same attack on more sophisticated platforms is not clear.

2.3.3 Trade-off In TO_Indep and TO_Dep Attacks

We observe a very clear trade-off between the ease of attack and the number of traces/queries to perform full key recovery. While TO_Dep attacks only require a handful of traces for key recovery, these attacks rely on very delicate leakages from targeted operations/instructions for key recovery. However, TO_Indep attacks fall on the other side of the spectrum, with respect to ease of key recovery. The attacks can exploit leakage from practically the entire re-encryption procedure, but require traces/queries ranging in the few thousands for key recovery. The number of queries is particularly important since refreshing the key pair is a common strategy to reduce exposure of the key to possible classical/side-channel attacks [RBRC21]. Thus, a natural question that arises is "*whether it is possible to construct attacks that can obtain the best of both worlds?*" (i.e.) attacks that can exploit leakage independent of the target operation, as well as enable efficient key recovery in very few queries.

In this work, we answer this question positively by proposing generic and novel *parallel* Plaintext-Checking (PC) oracle based attacks, bringing about significant improvement in number of queries in key recovery, compared to the binary PC oracle attacks [RRCB20]. We lay main focus on improving the efficiency of simple and generic side-channel attacks on unprotected implementations. Thus, masking is naturally out of scope of this paper. Our work is motivated from the view point of a designer, contemplating the decision to use heavy countermeasures such as masking for SCA protection of Kyber [HKL+22, BGR+21]. In this respect, our work attempts to answer the question: "*what is the simplest and most efficient attack on an unprotected implementation, with a basic SCA setup and very limited knowledge about the target?*.

The ephemeral setting for KEMs used for key-exchange is often recommended, where the public-private key pair (pk, sk) is refreshed for every new key-exchange. However, it is not always practical due to huge performance overhead of frequent key generation. Thus, the static key setting with regular refreshment of the key pair is a more preferable setting, where the public-private key pair (pk, sk) is refreshed once every X number of key-exchanges, where X is chosen by the designer. Here, the static secret key sk is more exposed to the attacker due to its use in multiple key-exchanges, compared to the ephemeral setting. If an attacker is able to recover the secret key in Y number of key-exchanges where $Y < X$, then the remaining $Z = X - Y$ number of key-exchanges using the same secret key sk with other legitimate devices are compromised. This is because all corresponding session keys can be recovered with the knowledge of sk. Lower the value of Y, higher is the number of session keys that can be compromised by the attacker for a given secret key sk. In this context, our proposed attacks which improve upon the number of queries for key recovery provide concrete inputs to a designer for choosing an appropriate key refresh rate. As we show later in Section 5, the impact of our attacks depend upon different scenarios such as availability of clone device, ability to perform offline computations after partial key-recovery etc.

In this work, we primarily focus on unprotected implementations, but our proposed attacks also perform better than existing attacks on *lightly protected* implementations using countermeasures such as shuffling [RBRC21]. Refer to Fig.1 for an illustration of a qualitative comparison of reported SCA applicable to Kyber, with respect to target dependency and number of traces.

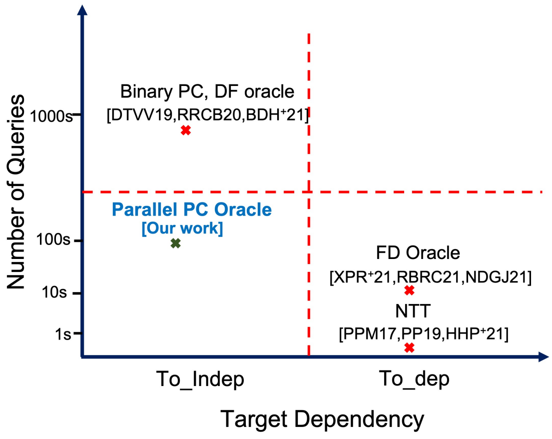


Figure 1: Qualitative comparison of reported SCA applicable to Kyber, with respect to target dependency and number of traces. Due to lack of space, we do not list all the attacks in the different categories

3 Improved PC Oracle-based CCA

3.1 Attacker Model

We assume that the attacker has physical access to the target device implementing the decapsulation procedure of Kyber KEM. The attacker has the ability to query the target device with chosen-ciphertexts ct of his/her choice. Moreover, prior knowledge of the secret key of the DUT or detailed knowledge about the underlying implementation such as the source code or compiled executable is not required. The attacker also does not require the ability to profile the side-channel leakage of the Device Under Test (DUT) with known keys. In the following, we explain the binary PC oracle attack of Ravi *et al.* [RRCB20] on Kyber KEM, which serve as the basis of our improved attacks.

3.2 Binary PC Oracle-based CCA

The attack works by recovering the secret key one coefficient at a time. We therefore demonstrate recovery of a single coefficient $\mathbf{s}[0]$, while other coefficients can be recovered in a similar manner. For simplicity, we also assume that all the components are only polynomials in R_q, however the same technique can be extended to higher dimensions (i.e.) R_q^k.

3.2.1 Construction of Chosen-Ciphertexts

The attacker constructs chosen-ciphertexts $ct = (\mathbf{u}, \mathbf{v}) \in (R_q \times R_q)$ as $\mathbf{u} = k_u \cdot x^0$ and $\mathbf{v} = k_v \cdot x^0$ where $(k_u, k_v) \in \mathbb{Z}_q$. The corresponding decrypted message m is given as:

$$m_i = \begin{cases} \mathsf{Decode}(k_v - k_u \cdot \mathbf{s}[0]) & \text{for} \quad i = 0 \\ \mathsf{Decode}(-k_u \cdot \mathbf{s}[i]) & \text{for} \quad i \in \{1, n-1\} \end{cases} \tag{1}$$

for $i \in [0, n-1]$. Thus, every bit m_i of the decrypted message is only dependent on the corresponding secret coefficient $\mathbf{s}[i]$. Now, the attacker can choose values for (k_u, k_v) such that:

$$m_i = \begin{cases} \mathcal{F}(\mathbf{s}[0]), & \text{if } i = 0 \\ 0, & \text{for } 1 \le i \le n-1 \end{cases} \tag{2}$$

where m can only take two possible values (i.e.) $m = 0$ and $m = 1$ (all bits except LSB have a value of 0). Moreover, $m = 0/1$ for a given ct, solely depends upon the value of $\mathbf{s}[0]$. Here, $\mathcal{F}$ represents the relation between the secret coefficient $\mathbf{s}[0]$ and m_i such that multiple values of the tuple (k_u, k_v) can uniquely identify the value of $\mathbf{s}[0]$ based on the corresponding value of m_i. In other words, the attacker needs to identify the appropriate values for the tuple (k_u, k_v) such that the corresponding values for the message bit $m_i = 0/1$ serves as a binary distinguisher for $\mathbf{s}[0]$ based on $m = 0/1$. Thus, the value of $m_i = 0/1$ serves as a *binary plaintext checking (PC) oracle* for the attacker to obtain information about the secret coefficient $\mathbf{s}[0]$.

An attacker who can instantiate such a binary PC oracle through *side-channels* can uniquely recover the secret coefficient $\mathbf{s}[0]$. Similarly, other coefficients can be recovered by exploiting the rotational property of polynomial multiplication in the ring R_q. Multiplying $\mathbf{r} \in R_q$ with x^p rotates $\mathbf{r}$ by p positions in an anti-cyclic fashion [RRCB20]. Thus, for the chosen-ciphertext $\mathbf{u} = k_u \cdot x^p$ and $\mathbf{v} = k_v \cdot x^0$ with $p \in \mathbb{Z}^+$, the first message bit m_0 is given as:

$$m_0 = \begin{cases} k_v - k_u \cdot \mathbf{s}[0], & \text{if } p = 0 \\ k_v - k_u \cdot (-\mathbf{s}[n-p]), & \text{for } 1 \le p \le n-1 \end{cases} \tag{3}$$

while the other bits are fixed to a value of 0. Thus, changing the parameter p ensures that m_0 depends upon different secret coefficients of $\mathbf{s}$, which can also be recovered in the same manner as $\mathbf{s}[0]$.

3.2.2 Instantiating Binary PC Oracle through SCA

A close observation of the decapsulation procedure (CCA.Decaps in Alg.2) reveals that the decrypted message m is hashed with the public-key ($\mathcal{G}$ in Line 12), and its result r along with the message m is fed into the re-encryption procedure (Line 13). For brevity, we denote these operations together as Re_Encrypt(m, pk). This re-encryption procedure is deterministic and solely depends upon m for a given public key pk. Thus, a single bit difference between $m = 0$ and $m = 1$ induces very different computations throughout the re-encryption procedure. This amounts to a few thousand leaky Points of Interest (PoI) which can be used to easily distinguish between $m = 0$ and $m = 1$, thereby instantiating a binary PC oracle.

3.3 Optimizing the Number of Queries for Key-Recovery

Ravi *et al.* [RRCB20] targeted the Round 2 specification of Kyber512 with secret coefficients in $[-2, 2]$. They utilized a *non-adaptive* approach to query the decapsulation device with chosen-ciphertexts, thereby using 5 ciphertexts to recover a single coefficient in $[-2, 2]$, which is clearly suboptimal. Thus, subsequent works [BDHD+19, HDV20] proposed improved adaptive attacks to reduce the number of queries for key recovery. The most recent work of Qin *et al.* [QCZ+21] proposed a systematic approach to find the lower bounds for the number of queries. They propose to build a Binary Decision Tree (BDT), which can be traversed based on the oracle's responses $m = 0/1$ to efficiently recover the correct coefficient.

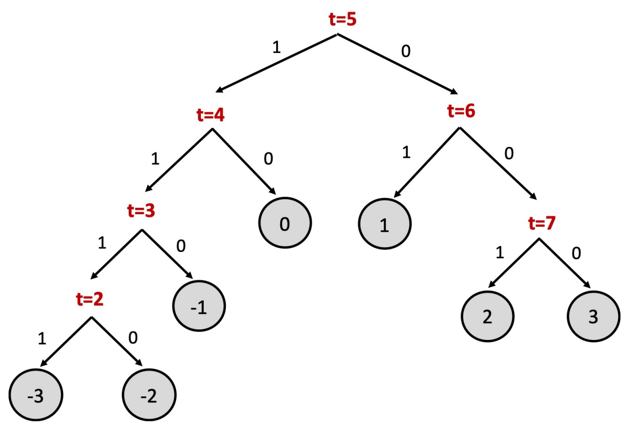


Figure 2: Optimal BDT for Kyber512 to minimise $\mathcal{Q}_{bin}$ for binary PC oracle-based CCA

3.3.1 Construction of an Optimal BDT

The core idea to construct an optimal BDT is based on the observation that the secret coefficients of Kyber follow a *non-uniform* CBD distribution. Thus, the optimal minimum for queries can be attained by constructing a non-uniform distinguisher with the following strategy: *higher the frequency of a secret coefficient candidate, lower should be the number of queries for unique distinguishability.* Thus, the number of queries to uniquely recover a candidate is inversely proportional to the probability of its occurrence.

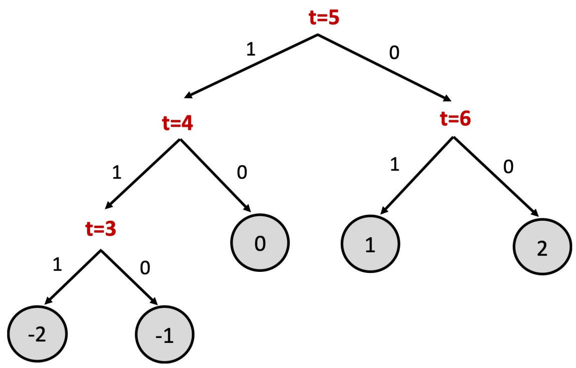


Figure 3: Optimal BDT for Kyber768, Kyber1024 to minimise $\mathcal{Q}_{bin}$ for binary PC oracle-based CCA

Let q_x denote the number of queries to uniquely distinguish $\mathbf{s}[i] = x$, and $\mathsf{Pr}(x)$ denote the probability that a secret coefficient $\mathbf{s}[i] = x$. Then, the objective is to build a BDT that yields a minimum for $\mathcal{Q}_{bin}$ which is given as:

$$\mathcal{Q}_{bin} = \sum_{i=-\eta}^{i=\eta} q_x \cdot \mathsf{Pr}(x) \tag{4}$$

We adopted the technique of Qin *et al.* [QCZ+21] to construct BDTs for all parameter sets of Kyber. For our chosen-ciphertexts, we choose $(k_u, k_v) = (208, 208 \cdot t)$ where $t \in \mathbb{Z}^+$.

Refer to Figure.2 for the corresponding optimal BDT for Kyber512 with $\mathcal{Q}_{bin} = 2.5625$, distinguishing every candidate in $[-3, 3]$ in not more than 4 queries. A node, edge and leaf of the BDT denotes a chosen-ciphertext query, oracle response and a recovered secret coefficient respectively. The optimal BDT for Kyber768 and Kyber1024 (with coefficients in $[-2, 2]$) is shown in Figure.3, with $\mathcal{Q}_{bin} = 2.3125$. Thus, the average number of queries for full key recovery is given as:

$$\mathcal{Q}_{attack} = 2^8 \cdot k \cdot \mathcal{Q}_{bin} \tag{5}$$

Thus, $\mathcal{Q}_{attack}$ amounts to 1312, 1776 and 2368 for Kyber512, Kyber768 and Kyber1024 respectively. We refer to the constructed trees as $\mathsf{BDT}_{\mathsf{min\_ent}}$, since the minimum average number of queries is very similar to computation of a certain Shannon entropy.

3.3.2 Critical Observations on the Binary PC Oracle-based CCA

We make two critical observations on the binary PC oracle CCA.

1. Observation-1: The decrypted message m for the chosen-ciphertexts only contains a single secret dependent message bit (i.e.) $m_0 = 0/1$, while all other bits $m_i = 0$ $\forall\, i = \{1, n-1\}$.
2. Observation-2: Leakage from the entire re-encryption procedure has only been exploited to recover a single bit of m, and therefore the secret key $\mathbf{s}$, especially when there are a few thousand leakage points [RRCB20].

This motivates us to investigate if it is possible to recover multiple bits of information about the secret key, exploiting leakage from the re-encryption procedure in a generic manner. In the following, we propose novel extensions of the binary PC oracle attack, which parallelize secret key recovery in a generic and configurable manner. This yields significant improvements in the possible lower bounds achievable for key recovery.

3.4 Parallel PC Oracle-based CCA

The core idea of our attack lies in constructing ciphertexts, such that multiple targeted bits of the message m (i.e.) m_i for $i \in \{0, P-1\}$ ($P \in \mathbb{Z}^+$) depend upon the P corresponding coefficients of the secret key. To achieve the same, we choose $\mathbf{u} = 208 \cdot x^0$ (i.e.) $k_u = 208$ and $\mathbf{v} = 208 \cdot t \cdot (\sum_{i=0}^{i=(P-1)} x^i)$ where $t \in \mathbb{Z}^+$ (i.e.) $k_v = 208 \cdot t$. We explain our choice for the exact value for $(\mathbf{u}, \mathbf{v})$ later in this section. Thus, the decrypted message m is given as:

$$m_i = \begin{cases} \mathsf{Decode}(208 \cdot t - 208 \cdot \mathbf{s}[i]), & \text{if } i \in [0, P-1] \\ \mathsf{Decode}(-208 \cdot \mathbf{s}[i]), & \text{for } i \in [P, n-1] \end{cases} \tag{6}$$

For the same values of t used to build the optimal BDT (Fig.2-3), the decrypted message m is given as:

$$m_i = \begin{cases} \mathcal{F}(\mathbf{s}[i]), & \text{if } i \in [0, P-1] \\ 0, & \text{for } i \in [P, n-1] \end{cases} \tag{7}$$

Thus, the first P bits of m are now dependent on the corresponding coefficients of $\mathbf{s}$, while all the other bits are fixed to 0. This technique is subtly different from attacks exploiting the *Full-Decryption* (FD) oracle [XPR+21,RBRC21], where all the bits of m are dependent on the corresponding coefficients of $\mathbf{s}$. However, our technique allows us to control the number and position of the secret dependent message bits. As seen later in Sec.4, this nuanced difference in approach allows to exploit leakage from the re-encryption procedure, in a parallel as well as generic manner.

An adversary able to recover the correct value of the message (i.e.) $m \in [0, 2^P - 1]$ can simultaneously recover a configurable P bits of information about $\mathbf{s}$. While the binary PC oracle attack traverses one BDT in a single query (Fig.2-3), our attack allows us to simultaneously traverse P distinct BDTs (BDT_i for $i \in [0, P-1]$) in a single query, where each BDT_i is traversed using the corresponding message bit m_i. This simultaneous traversal of P BDTs for P message bits m_i for $i \in [0, P-1]$ is made possible due to the choice of our chosen-ciphertexts. In particular, the value of the ciphertext component $\mathbf{u}$ is fixed to $\mathbf{u} = 208 \cdot x^0$, while the coefficients of the ciphertext component $\mathbf{v}$ (i.e.) the value of t for $\mathbf{v}[i] = 208 \cdot t$ with $i \in [0, P-1]$, can be decided based on the traversed node of the corresponding BDT (i.e.) BDT_i and the corresponding message bit m_i. We exhaustively searched for a single value for the non-zero coefficient of $\mathbf{u}$ (i.e.) k_u such that, simply changing $\mathbf{v}$ can yield different oracle responses to uniquely identify all possible values for the secret coefficients. In this manner, the traversal of all the P BDTs for the message bits m_i for $i \in [0, P-1]$ can be made completely independent of one-another.

Thus, a generic number of P secret coefficients can be simultaneously recovered in not more than 4 queries for Kyber512 using $\mathsf{BDT}_{\mathsf{min\_ent}}$ (Figure.2). Similarly, P secret coefficients can be simultaneously recovered in not more than 3 queries for Kyber768, Kyber1024 using $\mathsf{BDT}_{\mathsf{min\_ent}}$ (Figure.3). Similar to the binary attack, the rotational property of polynomial multiplication can be used to recover different P coefficients at a time, thereby recovering the full key. We refer to P as the parallelization factor and our attack as the P-way parallel PC oracle attack.

3.5 Optimal Key Recovery for P-way Parallel Attack

While the approach of Qin *et al.* [QCZ+21] yields the lower bound for queries for the binary attack, it is not clear if it is optimal in the P-way parallel attack. Our intuition is based on the following observation. If we utilize $\mathsf{BDT}_{\mathsf{min\_ent}}$ (Figure.2) for a 2-way parallel attack on Kyber512, then recovering two coefficients with a value of $(0, 0)$ simultaneously, only requires two queries. However, pairs with values of $(-2, 0)$ or $(-3, 0)$, can only be recovered in 4 queries, since -2 and -3 occur at a higher depth in the BDT. Thus, the number of queries to recover a given set of P random coefficients depends upon that coefficient in the set, with the maximum depth in the BDT. Moreover, increasing the parallelization factor P, only increases the probability of observing coefficients with a higher depth (e.g.) $-2, -3$, leading to increase in average number of queries to recover a given set of P coefficients.

3.5.1 More efficient BDTs for the P-way Parallel Attack

This leads us to hypothesize if BDTs with a lower depth yield lower average number of queries for the P-way parallel attack, in particular for higher values of P. We refer to Eqn.6 to construct our chosen-ciphertexts. After careful consideration of all possible values of t and corresponding m_i, we construct a BDT with a depth of 3 (Figure.4), which is the lowest achievable depth for unique distinguishability of coefficients in $[-3, 3]$. We also verified that it is not possible to build a BDT, with lower entropy and a depth of 3. We refer to this alternate tree as $\mathsf{BDT}_{\mathsf{min\_depth}}$.

We now derive an expression to compute the average number of queries to recover a set of P random coefficients for a generic BDT. We introduce some notation to explain our analysis. Refer to Figure.5 for the corresponding illustration of the same. We use *subtree* to denote a tree with smaller depth starting from the root. We use the notation st_d to denote a subtree with depth d where the depth of the root node is 0. So, for any BDT with maximum depth d, there are $d+1$ such subtrees (i.e.) $\{st_0, st_1, \ldots, st_d\}$. We denote the set of subtrees that contain at least a single leaf (recovered coefficient) as $\mathcal{V}$. The set of leaves in a given sub-tree st_i is denoted as $\mathcal{L}_{st_i}$ and specifically the leaves in the last layer of the subtree are denoted as $\mathcal{M}_{st_i}$. The average number of queries required to recover all P coefficients using the tree BDT denoted as $\mathcal{Q}_{set}$ is therefore given as:

$$\mathcal{Q}_{set} = \sum_{\forall st_i \in \mathcal{V}} i \cdot \mathsf{R}_i \tag{8}$$

where R_i denotes the probability that a set of random P coefficients belong to $\mathcal{L}_{st_i}$, with at least one coefficient in the set $\mathcal{M}_{st_i}$. With knowledge about the apriori distribution of the secret coefficients, it is possible to compute $\mathcal{Q}_{set}$ for any given BDT.

Note that it might be possible to obtain a more efficient tree using all possible P-tuples and not splitting into 2 branches after each node, but instead splitting into 2^P branches. While this approach might slightly further reduce the number of queries needed during the attack, it requires a much higher number of templates (one for each node). As such, trying to reach the optimal BDT-mindepth in the $P > 1$ scenario is not always desirable.

Refer to Figure.6 for the plot of $\mathcal{Q}_{set}$ for different values of P for Kyber512, for both $\mathsf{BDT}_{\mathsf{min\_ent}}$ (Figure.2) and $\mathsf{BDT}_{\mathsf{min\_depth}}$ (Figure.4). We can clearly see that $\mathcal{Q}_{set}$ for our proposed $\mathsf{BDT}_{\mathsf{min\_depth}}$ graph is lower than that of $\mathsf{BDT}_{\mathsf{min\_ent}}$ for $P \geq 3$, thereby confirming our hypothesis. Thus, our proposed BDT (i.e.) $\mathsf{BDT}_{\mathsf{min\_depth}}$ clearly yields fewer number of queries for Kyber512, compared to $\mathsf{BDT}_{\mathsf{min\_ent}}$ for $P \geq 3$. For the case of Kyber768 and Kyber1024, the secret coefficients lie in the span of $[-2, 2]$. Moroever, the same BDT used for the binary PC oracle attack (i.e.) $\mathsf{BDT}_{\mathsf{min\_ent}}$ in Fig.3 can be used for both Kyber768 and Kyber1024. The BDT has the minimum possible achievable depth of

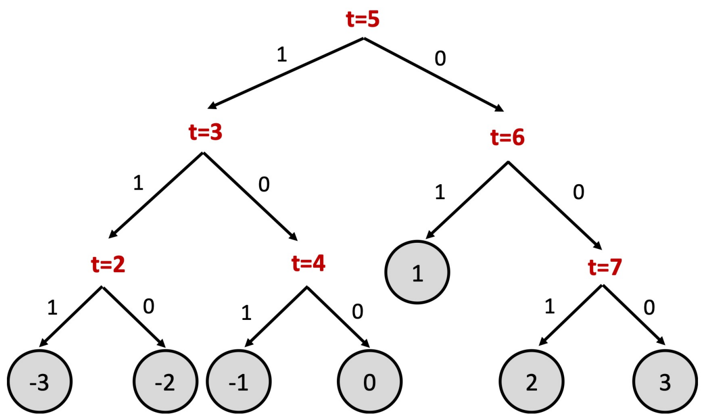


Figure 4: Optimal BDT ($\mathsf{BDT}_{\mathsf{min\_depth}}$) for P-way parallel oracle attack on Kyber512

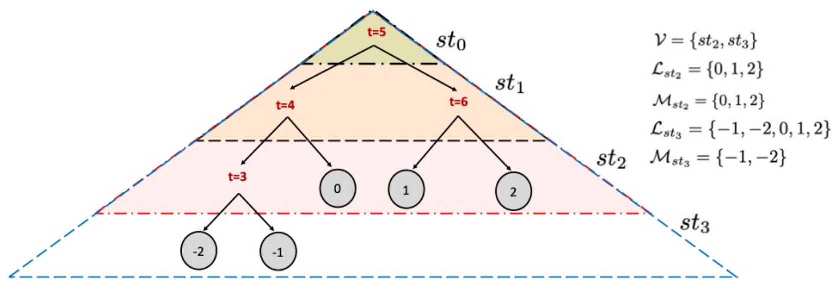


Figure 5: Illustration for calculation of $\mathcal{Q}_{set}$ (i.e.) average number of queries to recover P coefficients for a given BDT

3. Since the BDT already has the lowest entropy as well as minimum depth, $\mathsf{BDT}_{\mathsf{min\_ent}}$ yields the lowest number of queries for the P-way parallel attack, for both Kyber768 and Kyber1024 (i.e.) $\mathsf{BDT}_{\mathsf{min\_ent}} = \mathsf{BDT}_{\mathsf{min\_depth}}$. This therefore yields the same value for $\mathcal{Q}_{set}$, for both Kyber768 and Kyber1024 as shown in Fig.6.

Putting it all together, if an adversary has access to a perfect P-way PC oracle, then the average number of queries for full key recovery, denoted as $\mathcal{Q}_{attack}$ is given as:

$$\mathcal{Q}_{attack} = \lceil \frac{2^8}{P} \rceil \cdot k \cdot \mathcal{Q}_{set} \tag{9}$$

Refer to Figure.7 for $\mathcal{Q}_{attack}$ versus the parallelization factor P, for all parameter sets of Kyber. Our experimental simulations assuming a perfect P-way parallel PC oracle yielded a 100% success rate in recovering the secret key for a generic value of P. In the following, we demonstrate that an attacker can realize a very efficient and practical P-way parallel PC oracle through exploitation of side-channel leakage from the re-encryption procedure.

4 Realizing a Side-Channel based P-way Parallel PC Oracle

4.1 Experimental Setup

Our Device Under Test (DUT) is the STM32F407VG microcontroller, mounted on the STM32F4DISCOVERY evaluation board. We target the fastest implementation of Kyber KEM (m4speed version), taken from the public *pqm4* library [KRSS19], a benchmarking and testing framework for PQC schemes on the 32-bit ARM Cortex-M4 microcontroller. The target is clocked at 24 MHz. We utilize the Electromagnetic Emanation (EM) side-channel for our experiments. We obtain EM leakage using a near-field EM probe mounted on top of the chip, with the measurements collected on a Lecroy HD6104 oscilloscope, using a sampling rate of 250 MSam/sec, amplified 30dB with a pre-amplifier.

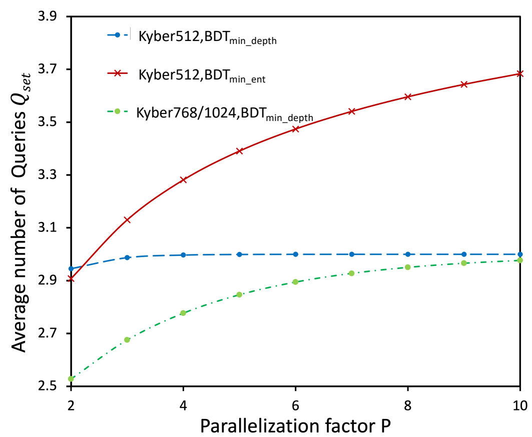


Figure 6: Average number of queries to recover P coefficients (i.e.) $\mathcal{Q}_{set}$ versus the parallelization factor P, for all parameter sets of Kyber

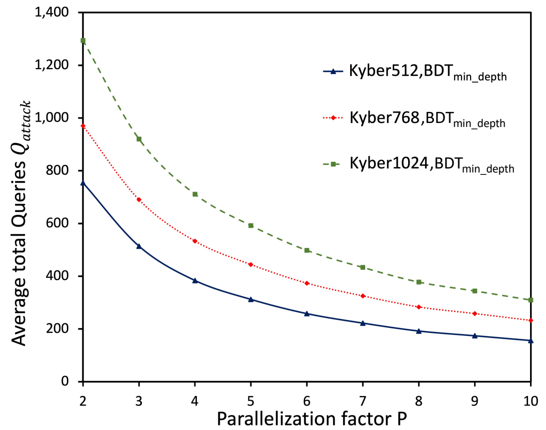


Figure 7: Average number of queries $\mathcal{Q}_{attack}$ for full key recovery, versus parallelization factor P for all parameter sets of Kyber

4.2 Side-Channel Methodology

Our task is to build a side-channel classifier for $m \in [0, 2^P - 1]$, using leakage from the re-encryption procedure (i.e.) $\mathsf{Re\_Encrypt}(m, pk)$. While prior works [RRCB20, QCZ+21] utilized the same leakage to distinguish between $m = 0$ and $m = 1$, we demonstrate that it is possible to classify an arbitrary number of values for m with a very high accuracy.

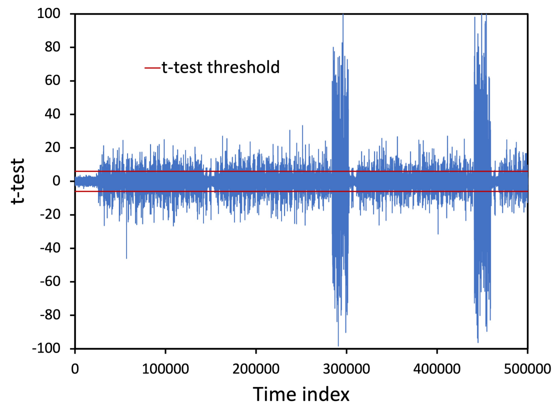


Figure 8: Welch's t-test plot computed for Tr_0 and Tr_1 for Kyber768

4.2.1 Building a Multi-Class Side-Channel Classifier

Our approach for multi-class classification, builds upon the binary classification approach using the well-known Welch's t-test [RRCB20]. We will briefly explain the binary classification method, and subsequently explain our generic extensions to arbitrary 2^P number of classes.

The binary classification is done in two phases: (1) Pre-Processing Phase and (2) Classification Phase. The pre-processing phase involves construction of side-channel templates for each class 0 and 1. The subsequent classification phase uses the templates to classify a given side-channel trace into one of the 2 classes. At no point during any of the two phases, does the attacker require to operate the target device with known secret keys.

Pre-Processing Phase: The adversary obtains T repeated measurements corresponding to $\mathsf{Re\_Encrypt}(m, pk)$ for both $m = 0$ and $m = 1$. This is done by repeatedly querying the decapsulation device with valid ciphertexts for $m = 0$ and $m = 1$ (T times each). We denote the trace set for $m = i$ as Tr_i, and the complete trace set as $\mathsf{Tr} = \cup_{i=0}^{i=1}\mathsf{Tr}_i$. The number of repeated measurements T is a parameter of the experimental setup.

- Every trace tr_j in Tr is normalized by removing the mean and dividing by its standard deviation to obtain t'_j.
- The Welch's t-test is computed between Tr_0 and Tr_1 to detect univariate leakage, based on Equation (10).

$$\mathsf{t\text{-}value} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{T} + \frac{\sigma_2^2}{T}}} \tag{10}$$

  where μ_i and σ_i are the mean and standard deviation of trace set Tr_i.

  Refer to Figure.8 for the t-test plot between Tr_0 and Tr_1 for $\mathsf{Kyber768}$ ($T = 20$ traces). The plot shows several peaks about the t-test threshold of ± 5, clearly indicating significant difference between the two computations. In particular, there are two distinct peaks (over multiple samples) with very high t-test values. Upon inspection, we identified it to be the sampling of polynomials of $\mathbf{r}$ from the CBD distribution (Line 11 of $\mathsf{CPA.Encrypt}$ in Alg.1). The attacker however does not require this information to perform the attack.
- Those features/points whose *absolute t-test value* is greater than a chosen threshold $\mathsf{Th}_{\mathsf{PoI}}$ are selected as the Points of Interest (PoI) set, denoted as $\mathcal{P}$. We do not have other criteria, apart from the t-test value to select the PoI. The threshold value $\mathsf{Th}_{\mathsf{PoI}}$ is also a parameter of the experimental setup and is empirically determined.
- The set $\mathcal{P}$ is used to derive a reduced trace set for each class, which we denote as Tr'_i for $i \in \{0, 1\}$, and the mean of the reduced trace set Tr'_i is the reduced template $m_{(i,\mathcal{P})}$ for class i with $i \in \{0, 1\}$.

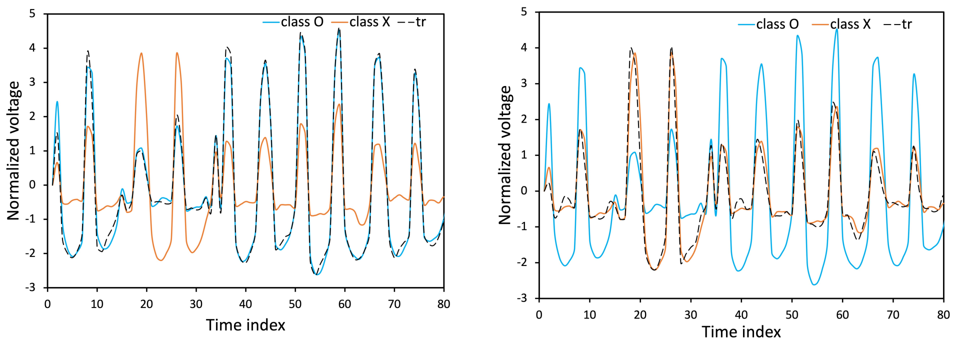
Thus, the reduced templates $m_{(i,\mathcal{P})}$ for $i \in \{0, 1\}$ are the output of the pre-processing phase. Since the target operation $\mathsf{Re\_Encrypt}(m, pk)$ depends upon both the message m and the public key pk, the pre-processing phase is not *one-time*, and therefore has to be carried out for every new public key.

Classification Phase: The reduced templates obtained from the pre-processing phase are now used to classify a given trace tr for a chosen-ciphertext, into either $m = 0/1$. The trace tr is first normalized, and the reduced trace $t'_{\mathcal{P}}$ is obtained. Then, the sum-of-squared difference Γ_* is computed with the reduced template of each class $m_{(i,\mathcal{P})}$ for $i \in \{0, 1\}$ as follows:

$$\Gamma_0 = (t'_{\mathcal{P}} - m_{0,\mathcal{P}})^\top \cdot (t'_{\mathcal{P}} - m_{0,\mathcal{P}}) \text{ and } \Gamma_1 = (t'_{\mathcal{P}} - m_{1,\mathcal{P}})^\top \cdot (t'_{\mathcal{P}} - m_{1,\mathcal{P}}). \tag{11}$$

The trace tr belongs to the class with the least sum-of-squared difference (i.e.) $\mathsf{Class}(tr) = 0$ if $\Gamma_0 < \Gamma_1$, else $\mathsf{Class}(tr) = 1$. Thus, a single side-channel trace can be used to distinguish between the two classes $m = 0$ and $m = 1$, thereby instantiating a binary PC oracle.

Refer to Figure.9 which shows clear distinguishability of a sample trace tr into either of the two classes. The figures only show a short segment of the reduced trace for better visual distinguishability, while the reduced templates used for our experiments span a few hundreds to few thousand points. We were able to obtain a 100% success rate for the binary classification $m = 0/1$, as also shown in Figure.9.



(a) $\mathsf{Class}(tr) = 0$ (b) $\mathsf{Class}(tr) = 1$

Figure 9: Matching the reduced attack trace tr' with the reduced templates of the two classes $m = 0$ and $m = 1$

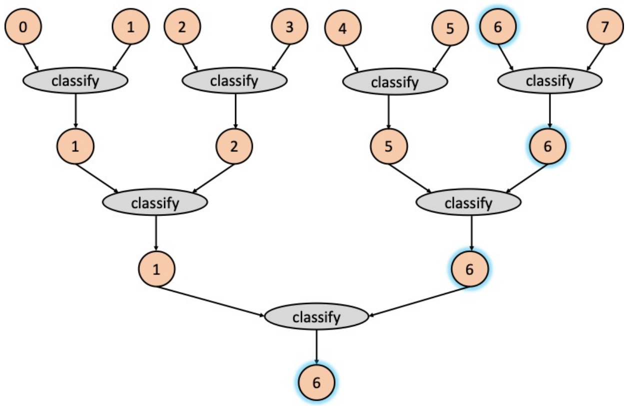


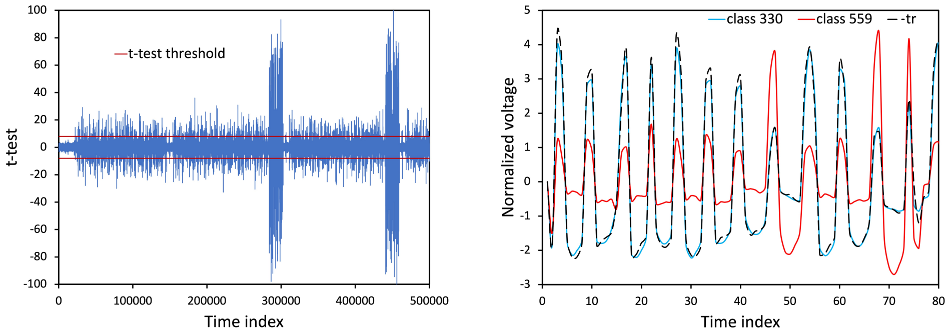
Figure 10: Illustration to classify the attack trace tr among 8 classes, m = [0,7]

4.2.2 Towards Multi-Class Classification

Our approach towards multi-class classification is based on the observation that it is possible to classify any two random values of m in the same manner, as $m = 0/1$. This is rendered possible due to the diffusion property of hash functions used in the re-encryption procedure. For an illustration, refer to Figure.11 for the t-test based binary classification between $m = 330$ and $m = 559$, as illustration. It is well known that unique identification of a particular candidate within a group is possible, if there exists a pairwise classifier for every possible pair of candidates [KU02]. For a P-way parallel PC oracle attack, there are 2^P possible classes for m. Thus, if we are able to classify between any two pairs of m with $m \in [0, 2^P - 1]$, it is also possible to uniquely identify the value of m. This applies for any generic value of P. Thus, the P-way parallel PC oracle can be realized in two phases in the following manner.

Pre-Processing Phase: The adverary collects T repeated measurements corresponding to $\mathsf{Re\_Encrypt}(m, pk)$ for all 2^P values of $m \in [0, 2^P - 1]$. The complete trace set for all classes is denoted as $\mathsf{Tr} = \cup_{i=0}^{i=2^P-1} \mathsf{Tr}_i$.

Classification Phase: Given an attack trace tr, the adversary uses pairwise binary classification similar to a knock-out tournament with 2^P players. The correct class (resp. winner) is selected after $(2^P - 1)$ pairwise classifications (resp. matches). A match in this context is nothing but binary classification of a given trace tr between two classes $m = x$ and $m = y$, which is denoted as $\mathsf{Classify}(x, y)$. Refer to Figure.10 for an illustration for 8 classes (i.e.) $m = [0, 7]$, where the attack trace tr corresponds to $m = 6$.



(a) t-test (b) $\mathsf{Class}(tr) = 330$

Figure 11: t-test plot and matching a given reduced attack trace tr' correponding to class 330, against reduced templates for classes 330 and 559

This approach only requires $h - 1$ pairwise binary classifications for h classes, and is optimal in terms of the number of pairwise classifications necessary for unique distinguishability. There is another costlier approach, of doing pairwise classification of all possible pairs of classes and adopting a majority voting approach to select the correct class. However, this yields a much higher h^2 binary classifications for h classes. Thus, we adopt the former and more efficient approach for classification. The aforementioned technique yields the correct candidate as long as the correct candidate for m is correctly classified, when paired with any other value of $m \in [0, 2^P - 1]$. This is similar to the case of having a player, who is capable of winning against any other player in the tournament, and therefore emerges as the winner.

4.2.3 Experimental Validation

We validated our proposed attack on the speed optimized implementation of Kyber768 from the *pqm4* library (Refer Sec.4.1 for the experimental setup). We were able to achieve full key recovery with 100% success rate for a parallelization factor of $P = 10$ (1024 classes). We utilize $T = 5$ traces to build templates for each class, which amounts to 5520 traces. While this is the maximum value of P used for our experiments, it is possible to increase P to any arbitrary value. In this respect, we also verified the success of binary classification of several pairs of messages with $P = 12$ (i.e.) which amounts to 4096 classes. We were able to classify all the collected pairs correctly with 100% accuracy.

The achieved success rate significantly depends upon the the experimental setup, and particularly the SNR of the collected traces. For success rates below 100% due to the effect of random noise, it is possible to utilize majority voting based on multiple traces or utilize error correcting codes to encode the oracle responses as shown in [NDGJ21, SCZ+22] to enhance the success rate. Nevertheless, our experiments provide sufficient evidence that there is enough information in the leakage available from the re-encryption procedure, that allows to recover a generic P bits of information about the secret key, while prior works [RRCB20] underutilized leakage from the re-encryption procedure to only recover a single bit. This is also due to the fact that there are several hundred PoIs for classification, to distinguish any pair of values for m.

We also believe that an upper limit for P for perfect classification if exists, is challenging to determine through experiments. It depends upon a variety of factors such as the target device and target implemenation, Signal to Noise Ratio (SNR) etc. Thus, it can only be determined empirically by attempting key recovery for different values of P, which we leave for future work. However, we show later in Section 5, that arbitrarily increasing P also exponentially increases the number of traces for the pre-processing phase, thereby decreasing the relevance of the attack. Moreover, we can see that our proposed attack is generic and clearly agnostic to the target implementation, and requires almost no information about the design of the target.

5 Evaluating Total Cost for Key Recovery

From an attacker's perspective, the number of queries is the primary cost of the attack, as he/she looks for key recovery with minimum possible interaction with the target device. The cost of key recovery, therefore includes the number of queries for the pre-processing phase denoted as $\mathcal{Q}_{template}$, as well as for the classification phase, denoted as $\mathcal{Q}_{attack}$. The pre-processing phase requires T queries for each of the 2^P classes, which amounts to $2^P \cdot T$ queries. The number of queries in the classification phase is nothing but the total number of chosen-ciphertext queries for key recovery (Refer Eqn.8 and Fig.7 in Sec.3.4). Thus, the total number of queries for key recovery denoted as $\mathcal{Q}_{total}$ is given as:

$$\mathcal{Q}_{total} = \mathcal{Q}_{template} + \mathcal{Q}_{attack} \quad (12)$$

$$= 2^P \cdot T + \lceil \frac{2^8}{P} \rceil \cdot k \cdot \mathcal{Q}_{set} \quad (13)$$

5.1 Analysis for Partial Key Recovery

In a bid to reduce the number of traces, an attacker can also resort to recovering $m < (k \cdot n)$ coefficients of Kyber, and recovering the remaining coefficients using suitable lattice-based solvers in an offline manner. In this respect, we consider three possible cases for an attacker with differing capabilities to perform offline computations:

1. Full_Recovery - Full Key Recovery with 0 remaining offline computations.
2. Partial_Recovery_2^{32} - Partial Key Recovery with 2^{32} remaining offline computations.
3. Partial_Recovery_2^{64} - Partial Key Recovery with 2^{64} remaining offline computations.

We utilized the leaky LWE estimator developed by Dachman-Soled *et al.* [DSDGR20] to estimate the number of coefficients to be recovered, to reduce the security strength of Kyber to 2^{32} and 2^{64} respectively. The tool allows us to include exact or approximate hints and estimate the remaining cost to recover the secret. Refer to Table.1 for the exact number of coefficients to be recovered for the aforementioned attacker scenarios.

Table 1: Number of coefficients to be recovered, for scenarios considering attackers with different offline computational capabilities

	Full_Recovery	Partial_Recovery_2^{32}	Partial_Recovery_2^{64}
Kyber512	512	354	184
Kyber768	768	667	463
Kyber1024	1024	1010	782

5.2 On the Presence of Clone Device

A close observation of Eqn.13 to calculate the total number of queries $\mathcal{Q}_{total}$ reveals that the cost of pre-processing phase to generate templates cannot be ignored, especially given that the pre-processing phase is required to be done for every new public key. In this respect, we identify two possible scenarios, with respect to whether or not the adversary has access to a clone device.

5.2.1 With Clone Device

In this scenario, the adversary has access to a clone device. Thus, he/she can generate templates for Re_Encrypt(m, pk) from the clone device, since the computations only depend upon known values to the attacker (i.e.) m and pk. Thus, the pre-processing phase is completely taken offline. By offline, we mean that templates can be captured on the clone device, and the attacker only requires to capture traces from the target device for the key recovery phase. In this case, the number of queries to the target device, denoted as $\mathcal{Q}_{target}$ is nothing but:

$$\mathcal{Q}_{target} = \mathcal{Q}_{attack} \tag{14}$$

$$= \lceil \frac{2^8}{P} \rceil \cdot k \cdot \mathcal{Q}_{set} \tag{15}$$

Here, $\mathcal{Q}_{template} = 0$ since the pre-processing phase is carried out on the clone device.Thus, $\mathcal{Q}_{target}$ simply scales inversely with the parallelization factor P. Thus, the lower bound for $\mathcal{Q}_{target}$ is only limited by the parallelization factor P achievable on the given target device. We however recall that finding the exact limit on P is very hard to achieve in practice.

Refer to Figure.13(a) for the plot of number of queries to the target versus P for Kyber768, considering both full key recovery and partial key recovery. We can clearly see that the number of queries scales inversely with increase in P. For the experimentally verified case of $P = 10$, the attacker requires ≈ 232 queries for full key recovery, which improves over the state-of-the-art binary PC oracle attack [QCZ+21] by a factor of $\approx 7.6\times$.

(Arbitarily) Best Case Scenario: We also consider an arbitarily strong attacker capable of building 2^{32} templates on the clone device. In this case, full key recovery is possible in just 72 queries, which is an improvement by a factor of ≈ 24.6 compared to the binary PC oracle attack. Naturally, we also observe better improvements for the case of partial key recovery as seen in Table.2.

5.2.2 Without Clone Device

In this scenario, the adversary does not have access to a clone device. Recall that our attack is possible even without knowledge of the key in pre-processing phase. Thus, both the pre-processing as well as classification phase has to be carried out directly on the target

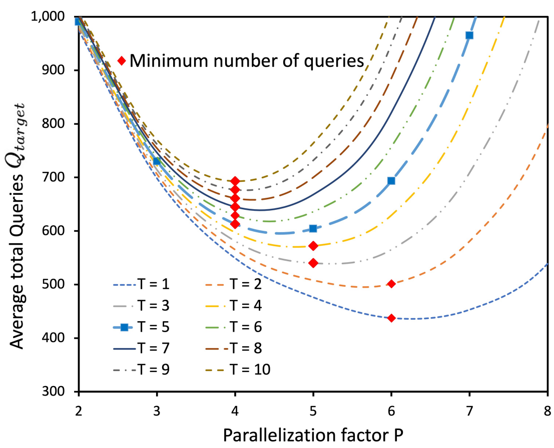


Figure 12: Total number of queries required for full key recovery for Kyber768 in the Scenario_Without_Clone versus the parallelization factor P, for different values of T, where T is the number of traces per template

device. Here $\mathcal{Q}_{target}$ is nothing but:

$$\mathcal{Q}_{target} = \mathcal{Q}_{template} + \mathcal{Q}_{attack} \quad (16)$$

$$= 2^P \cdot T + \lceil \frac{2^8}{P} \rceil \cdot k \cdot \mathcal{Q}_{set} \quad (17)$$

We can observe that $\mathcal{Q}_{template}$ scales exponentially with P (i.e.) 2^P and also increases linearly with T (number of traces per template), while $\mathcal{Q}_{attack}$ scales inversely with P. Thus, there exists a fine trade-off between the cost of pre-processing and classification phase. It is not possible to arbitrarily increase P to improve $\mathcal{Q}_{target}$, as the cost of the pre-processing phase outweighs the cost of the classification phase for higher values of P, unlike when attacker has access to clone device. Refer to Figure.12 for the plot of number of queries versus the parallelization factor P, for different values of T. As expected, we observe a certain minima for number of queries for each value of $T \in [1, 10]$.

We experimentally verified that full key recovery is possible with $T = 5$. For $T = 5$, the parallelization factor $P = 4$ yields the lowest number of queries (i.e.) 613. This is an improvement by a factor of $\approx 2.89\times$ compared to the binary PC oracle attack. However, a lower value for T could be achieved with a better experimental setup with low acquisition noise. For the best possible scenario of $T = 1$ (single trace per template), the parallelization factor $P = 6$ yields the lowest number of queries (i.e.) 437, an improvement factor of $\approx 4\times$ compared to the binary PC oracle attack. Refer to Figure.13(b) for the number of queries versus P, for $T = 5$, for both full key recovery and partial key recovery.

5.3 Extensions to Lightly Protected Implementations

Given the heavy performance penalty of masking countermeasures for LWE/LWR-based KEMs with upto 3.1× in runtime as shown in [BGR+21], there is significant interest in low-cost countermeasures to offer protection against known side-channel attacks. One such approach is the shuffling countermeasure, which was proposed to protect the message encoding procedure against single trace message recovery attacks [ACLZ20, SKL+20]. Moreover, leakage from the message encoding procedure was also shown to be exploitable

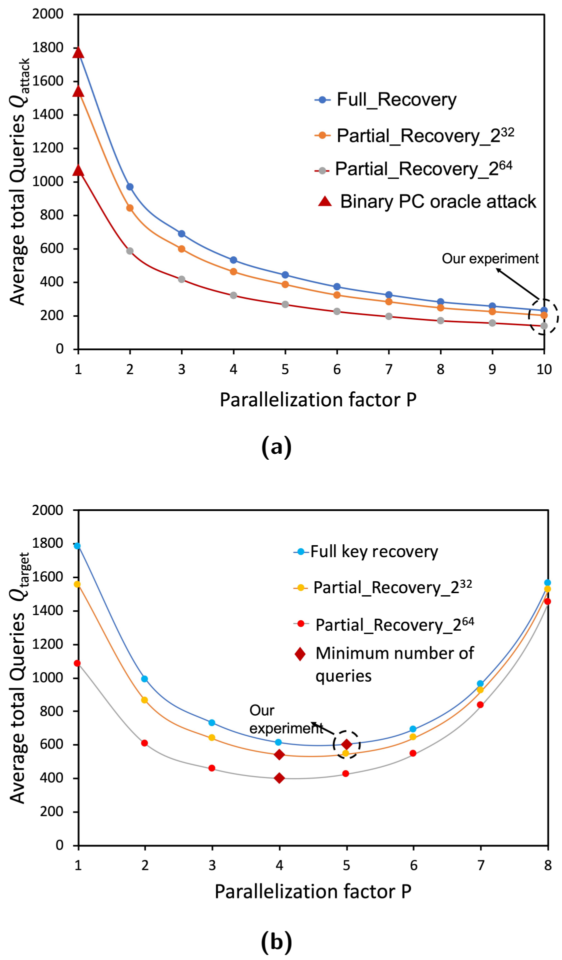


Figure 13: Estimates for the total number of queries to the target versus the parallelization factor P for Kyber768 (a) With clone device and (b) Without clone device, and also considering partial key recovery and full key recovery

for key recovery through the Full-Decryption (FD) oracle attack in [XPR+21]. These attacks are capable of recovering 256 bits of information from a single trace, thereby capable of message recovery in potentially less than 10 traces on unprotected implementations of Kyber KEM, especially in presence of sufficiently high SNR.

In this respect, shuffling was proposed as a concrete countermeasure against attacks targeting the message encoding procedure. Shuffling ensures that the attacker can still recover all the 256 bits of the message, but not its correct order, thereby offering protection against message recovery and also removing the presence of the FD oracle. However, Ravi *et al.* [RBRC21] showed an attack on shuffling countermeasure in a chosen-ciphertext setting and recover 1 targeted message bit per query. While shuffling does not offer concrete protection, it at least prevents single trace message recovery, and reduces the attacker's capability to only recover a single bit per query, which is equivalent to a binary PC oracle attack. This is the best known attack on such a lightly protected shuffled implementation of Kyber KEM. Thus, a simple shuffling countermeasure on the message encoding operation is able to increase the attacker's effort from recovering the secret key in < 10 traces to a few thousand traces.

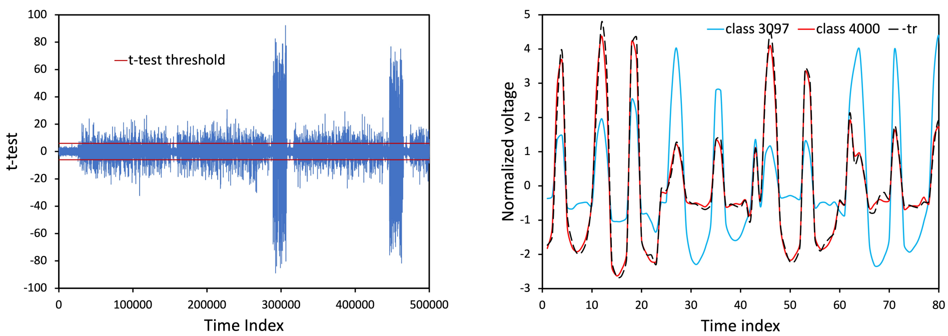
Table 2: Tabulation of the total number of queries for key recovery for Kyber768, considering attack scenarios with respect to clone device, as well as the attacker's offline computational capability.

	Parallelization Factor P					
	With Clone				Without Clone	
	1	10	12	32	4 (T=5)	6 (T =1)
Full_Recovery	1776	232	197	72	613	437
Partial_Recovery_2^{32}	1545	202	170	63	544	388
Partial_Recovery_2^{64}	1073	140	120	45	402	290

However, we observe that our proposed parallel PC oracle attack is independent of leakage from the shuffled message encoding operation. In fact, for our experiments on the unprotected implementation, we utilized leakage only until sampling of the ephemeral secret module **r** during the re-encryption procedure (Line 11 of CPA.Encrypt in Alg.1). Thus, we do not utilize leakage from the message encoding operation for our practical experiments. Thus, we hypothesize that our attack can defeat the lightly protected implementation with the shuffled message encoding operation, in the same manner as that of the unprotected implementation.

5.3.1 Experimental Results

We mounted our parallel PC oracle attack on the decapsulation procedure of Kyber KEM, with a shuffled message encoding procedure. Confirming our hypothesis, we were able to successfully recover the secret key in the same manner as the unprotected implementation. The number of traces for key recovery, remains the same as that of our attack on the unprotected implementation. We also experimentally validated the capability to exhaustively recover all 2^P possible values of the decrypted message for a parallelization factor of $P = 10$, and we were able to correctly distinguish all classes only using single trace, thereby concretely demonstrating the ability to distinguish between 2^P possible values of the decrypted message. Figure.14 shows the t-test based classification between $m = 3097$ and $m = 4000$, as illustration for our attack on the shuffling countermeasure. Thus, our parallel PC oracle attack also serves an effective TO_Indep attack on the lightly protected implementation with the shuffling countermeasure.



(a) t-test (b) Class(tr) = 4000

Figure 14: t-test plot and matching a given reduced attack trace tr correponding to class 4000, against reduced templates for classes 3997 and 4000 with shuffling countermeasure

5.4 Differences from [TUX+22]

In a recent and independently developed work by Tanaka *et al.* [TUX+22], authors also propose to recover multiple bits in parallel from a PC oracle attack. While the objectives of [TUX+22] are aligned with our work, we few subtle differences.

Number of Oracle Queries: We observe that the BDTs for optimal binary PC oracle attack are not always optimal for the parallel PC oracle attack. We demonstrate that BDTs with minimum depth (i.e.) BDT_{min_depth} are optimal for a high parallelization factor P, compared to BDTs with minimum entropy (i.e.) BDT_{min_ent} (Refer Section 3.5.1). However, [TUX+22] use the same BDTs that were used for the binary PC oracle attack, to also perform their attack in the parallel setting. Refer Tab.3(a) of [TUX+22] for the BDT used to attack Kyber512 in the parallel setting and Tab.9(a) of [XIU+21] for the BDT used to attack Kyber512 in the binary setting. Thus, our approach to construct optimal BDTs for the parallel PC oracle attack yields lower number of queries for Kyber, even in the presence of a perfect parallel PC oracle.

Side-Channel based Oracle: The authors of [TUX+22] utilized a multi-class classification neural network (NN) to realize a parallel PC oracle, while we utilize a simple t-test based classifier. We observe that the attacker needs to carry out the pre-processing phase to create templates for every new public key, *it is important to minimize the sum of traces for the pre-processing phase and key recovery phase.* This is especially applicable in the scenario where the attacker does not have access to a clone device. Thus, we chose t-test based classifier with an objective to reduce the no. of traces during pre-processing phase, typically $T < 10$ for each class. On the other hand, NN-based classifiers are typically known to require a very high number of traces for training. [TUX+22] utilize 1000 traces for training and 500 traces for validation for each of the 2^P classes. While NN-based classifiers are suitable for training on the clone device (offline), they are sub-optimal when there is no access to a clone device.

Moreover, we perform experiments on a full implementation of Kyber to realize the parallel PC oracle, while [TUX+22] perform experiments on implementations of SHAKE,SHA3 and AES. Thus, we are able to exploit leakage from several operations within the re-encryption procedure which enable us to yield a 100 % success rate over single traces to realize a practical parallel PC oracle.

5.5 Applicability to other PQC schemes

While we present our parallel PC oracle attack for Kyber KEM, we also believe that our attack can be adapted in a straightforward manner to other lattice-based schemes such as Saber and Frodo. This is because our technique to construct chosen-ciphertexts not only applies to Kyber, but to the broader framework of the LPR encryption scheme [LPR10], which forms the core of several lattice-based KEMs such as Kyber, Saber [DKR+20], NewHope [AAB+], Round5 [BBF+], LAC [LLJ+] and Frodo [ABD+20]. We recall that the binary PC oracle attacks proposed in several prior works [RRCB20, UXT+22] have been shown to be adaptable to several LWE/LWR-based schemes including Kyber and Saber, albeit with appropriate changes in the actual value of the chosen-ciphertexts and the number of queries for key recovery. We also experimentally validated our attack on Saber through simulations of a perfect parallel PC oracle. We would like to briefly sketch the idea to perform parallel PC oracle attack on Saber.

Similar to the chosen-ciphertexts for Kyber, we choose $\mathbf{u} = k_u \cdot x^0$ and $\mathbf{v} = k_v \cdot$

$(\sum_{i=0}^{i=(P-1)} x^i)$ where $t \in \mathbb{Z}^+$. Thus, the decrypted message m is given as:

$$m_i = \begin{cases} \mathsf{Decode}(k_v - k_u \cdot \mathbf{s}[i]), & \text{if } i \in [0, P-1] \\ \mathsf{Decode}(-k_u \cdot \mathbf{s}[i]), & \text{for } i \in [P, n-1] \end{cases} \quad (18)$$

For the recommended parameters of Saber, the secret coefficients are in a slightly larger range of $[-4, 4]$ compared to Kyber768 with coefficients in $[-2, 2]$. Please refer to Fig.15 for the BDT with minimum depth we were able to achieve for the recommended parameter sets of Saber.

For $(k_u = 57)$ and different values for $k_v \in [1, 7]$, we were able to uniquely distinguish candidates for the secret coefficient in the range $[-2, 4]$ in not more than 4 queries. However, we were not able to distinguish between candidates -3 and -4 using $k_u = 57$. Thus, we have to utilize an additional query of $(k_u, k_v) = (54, 1)$ to distinguish between -3 and -4. However, in case of Kyber, we were able to distinguish all candidates between $[-2, 2]$ using the same value of k_u, in not more than 3 queries (Refer Fig.3). We estimate that, for parallelization factor $P = 10$, we would require approximately 390 queries in the key recovery phase for full key recovery, compared to 232 for Kyber768 (Refer Tab.2). Thus, we can see that our technique can be easily adapted to Saber, albeit with differences in the number of traces for key recovery. Similarly, we believe our attack can also be adapted to other LWE/LWR-based KEMs such as NewHope, Round5, LAC and Frodo which are based on the LPR encryption scheme. However, adapting our attack to other lattice-based schemes based on the NTRU paradigm such as NTRU [CDH+19], NTRU Prime [BBC+20] is not trivial, and thus consider them as potential future work.

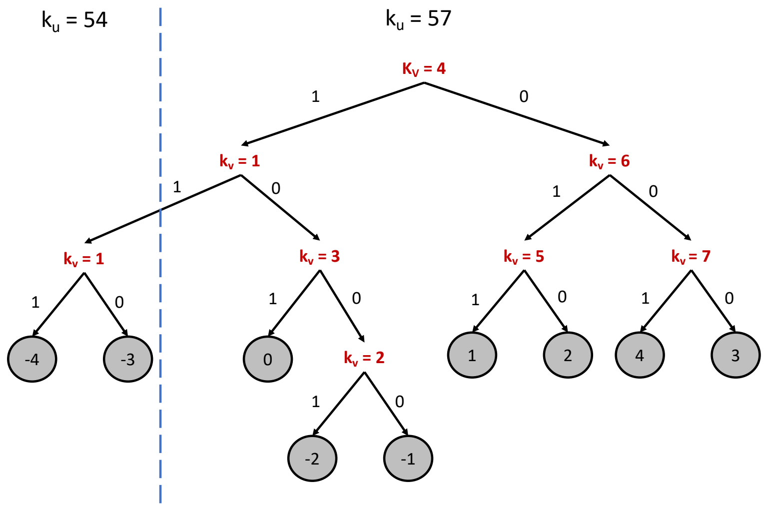


Figure 15: Optimal BDT for Parallel PC oracle based CCA on Saber (recommended parameters)

6 Conclusion

In this work, we propose novel *parallel PC oracle* based side-channel attacks, which are capable of recovering an arbitrary P number of bits of information about the secret key in a single trace. We experimentally validated our attacks on the fastest implementation of unprotected Kyber KEM in the *pqm4* library. Our experiments yielded improvements in the range of 2.89× and 7.65× in the number of queries, compared to state-of-the-art binary PC oracle attacks, while arbitrarily high improvements are possible given the generic nature of the attack. We also show that our proposed attacks are able to achieve the lowest number of queries for key recovery, even over implementations protected with low-cost countermeasures such as shuffling. Masking serves as a concrete countermeasure against our proposed attacks, and therefore we believe our work stresses the strong need to implement masking countermeasures for lattice-based schemes, particularly for embedded applications. The future directions of our presented work are finding theoretical optimal attacks in the context of parallel PC oracle and applicability in the hardware targets.

Acknowledgment

Jan-Pieter D'Anvers is funded by FWO (Research Foundation − Flanders) as junior post-doctoral fellow (contract number 133185 / 1238822N LV).

References

[AAB+] Erdem Alkim, Roberto Avanzi, Joppe W. Bos, Leo Ducas, Antonio de la Piedra, Thomas Poppelmann, Peter Schwabe, and Douglas Stebila. NewHope (Version 1.1): Algorithm Specifications And Supporting Documentation (April 10, 2020). *Submission to the NIST post-quantum project.*

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the third round of the NIST post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2022.

[AASA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[ABD+20] Erdem Alkim, Joppe W. Bos, Leo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. Frodo : Algorithm Specifications And Supporting Documentation (June 4, 2021). *Submission to the NIST post-quantum project*, 2020.

[ABD+21] Roberto Avanzi, Joppe W. Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber (version 3.02): Algorithm specifications and supporting documentation (Auguest 4, 2021). 2021.

[ACLZ20] Dorian Amiet, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. Defeating NewHope with a single trace. In *International Conference on Post-Quantum Cryptography*, pages 189–205. Springer, 2020.

[BBC+20] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, Chitchanok Chuengsatiansup, Tanja Lange, Adrian Marotzke, Bo-Yuan Peng, Nicola Tuveri, Christine van Vredendaal, and Bo-Yin Yang. NTRU Prime: Round 3 (October 7, 2020). *Submission to the NIST post-quantum project*, 2020.

[BBF+] Hayo Baan, Sauvik Bhattacharya, Scott Fluhrer, Oscar Garcia-Morchon, Thijs Laarhoven, Rachel Player, Ronald Rietman, Markku-Juhani O. Saarinen, , Ludo Tolhuizen, José Luis Torre-Arce, and Zhenfei Zhang. Round5 : Algorithm Specifications And Supporting Documentation (10th April, 2020). *Submission to the NIST post-quantum project.*

[BDH+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel van Beirendonck. Attacking and defending masked polynomial comparison for lattice-based cryptography. 2021(3):334–359, 2021.

[BDHD+19] Ciprian Băetu, F Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse attacks on post-quantum cryptosystems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 747–776. Springer, 2019.

[BGR+21] Joppe W Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First-and higher-order implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 173–214, 2021.

[CDH+19] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU: Algorithm specifications and supporting documentation (March 20, 2019). *Submission to the NIST post-quantum project*, 2019.

[DHP+22] Jan-Pieter D'Anvers, Daniel Heinz, Peter Pessl, Michiel Van Beirendonck, and Ingrid Verbauwhede. Higher-order masked ciphertext comparison for lattice-based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):115–139, Feb. 2022.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. Saber. *NIST Round 3 Submissions*, 2020.

[DSDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. Lwe with side information: attacks and concrete security estimation. In *Annual International Cryptology Conference*, pages 329–358. Springer, 2020.

[DTVV19] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. Timing attacks on error correcting codes in post-quantum schemes. In *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*, pages 2–9, 2019.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual international cryptology conference*, pages 537–554. Springer, 1999.

[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 *Annual International Cryptology Conference*, pages 359–386. Springer, 2020.

[HDV20] Loïs Huguenin-Dumittan and Serge Vaudenay. Classical misuse attacks on nist round 2 pqc. In *International Conference on Applied Cryptography and Network Security*, pages 208–227. Springer, 2020.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked cca2 secure kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 88–113, 2021.

[HKL+22] Daniel Heinz, Matthias J Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Daan Sprenkels. First-order masked kyber on arm cortex-m4. *Cryptology ePrint Archive*, 2022.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4, 2019. https://github.com/mupq/pqm4.

[KU02] Boonserm Kijsirikul and Nitiwut Ussivakul. Multiclass support vector machines using adaptive directed acyclic graph. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 980–985. IEEE, 2002.

[LLJ+] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC: Practical Ring-LWE Based Public-Key Encryption with Byte-Level Modulus (19th Dec, 2019).

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked ind-cca secure saber kem implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 676–707, 2021.

[NDJ21] Kalle Ngo, Elena Dubrova, and Thomas Johansson. Breaking masked and shuffled cca secure saber kem by power analysis. In *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, pages 51–61, 2021.

[PP19] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In *International Conference on Cryptology and Information Security in Latin America*, pages 130–149. Springer, 2019.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 513–533. Springer, 2017.

[QCZ+21] Yue Qin, Chi Cheng, Xiaohan Zhang, Yanbin Pan, Lei Hu, and Jintai Ding. A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 92–121. Springer, 2021.

[RBRC21] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On exploiting message leakage in (few) nist pqc candidates for practical message recovery attacks. *IEEE Transactions on Information Forensics and Security*, 2021.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based pke and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[SCZ+22] 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. *Cryptology ePrint Archive*, 2022.

[SKL+20] Bo-Yeon Sim, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. Single-trace attacks on message encoding in lattice-based KEMs. 8:183175–183191, 2020.

[TUX+22] Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum kems. Cryptology ePrint Archive, Paper 2022/940, 2022. https://eprint.iacr.org/2022/940.

[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 Transactions on Cryptographic Hardware and Embedded Systems*, pages 296–322, 2022.

[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 *International Conference on the Theory and Application of Cryptology and Information Security*, pages 33–61. Springer, 2021.

[XPR+21] Zhuang Xu, Owen Michael Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Transactions on Computers*, 2021.