Will You Cross the ~~Skies~~ Threshold for Me?

Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs

Prasanna Ravi[1,2], Martianus Frederic Ezerman[3], Shivam Bhasin[1], Anupam Chattopadhyay[1,2] and Sujoy Sinha Roy[4]

[1] Temasek Laboratories, Nanyang Technological University, Singapore
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[4] Institute of Applied Information Processing and Communications, TU Graz, Graz, Austria
prasanna.ravi@ntu.edu.sg fredezerman@ntu.edu.sg sbhasin@ntu.edu.sg
anupam@ntu.edu.sg sujoy.sinharoy@iaik.tugraz.at

Abstract. In this work, we propose generic and novel side-channel assisted chosen-ciphertext attacks on NTRU-based key encapsulation mechanisms (KEMs). These KEMs are IND-CCA secure, that is, they are secure in the chosen-ciphertext model. Our attacks involve the construction of malformed ciphertexts. When decapsulated by the target device, these ciphertexts ensure that a targeted intermediate variable becomes very closely related to the secret key. An attacker, who can obtain information about the secret-dependent variable through side-channels, can subsequently recover the full secret key. We propose several novel CCAs which can be carried through by using side-channel leakage from the decapsulation procedure. The attacks instantiate three different types of oracles, namely a plaintext-checking oracle, a decryption-failure oracle, and a full-decryption oracle, and are applicable to two NTRU-based schemes, which are NTRU and NTRU Prime. The two schemes are candidates in the ongoing NIST standardization process for post-quantum cryptography. We perform experimental validation of the attacks on optimized and unprotected implementations of NTRU-based schemes, taken from the open-source `pqm4` library, using the EM-based side-channel on the 32-bit ARM Cortex-M4 microcontroller. All of our proposed attacks are capable of recovering the full secret key in only a few thousand chosen ciphertext queries on all parameter sets of NTRU and NTRU Prime. Our attacks, therefore, stress on the need for concrete side-channel protection strategies for NTRU-based KEMs.

Keywords: lattice-based cryptography · electromagnetic-based side-channel attack · learning with error · learning with rounding · chosen ciphertext attack · public key encryption · key encapsulation mechanism

1 Introduction

The NIST standardization process for post-quantum cryptography is currently in the third round with seven finalists and eight alternates for public key encryption (PKE), key encapsulation mechanisms (KEMs), and digital signatures (DS) [AASA+20]. For this round, NIST has made it clear that resistance to side-channel attacks (SCAs) and fault injection attacks (FIAs) is an important criteria in the standardization process, especially amongst schemes with tightly matched security and efficiency [AH21].

Three out of the four finalists for PKE/KEMs are schemes from lattice-based cryptography. Lattice-based PKE/KEMs can be broadly classified into two categories. Schemes based

on the *learning with errors* (LWE) [Reg09] and *learning with rounding* (LWR) [BPR12] problems are in the first category. The second category collects schemes that are based on the N^{th} *order truncated polynomial ring unit* (NTRU) problem [HPS98]. The security of IND-CPA secure lattice-based schemes in a static key setting, that is, when the secret key is reused, has been studied for a long time. Several works have proposed efficient *chosen-ciphertext attacks* (CCAs) on both LWE/LWR-based schemes as well as on NTRU-based schemes [DCQ19, QCD19, BDHD+19, BGRR19, Flu16]. These attacks rely on the assumed presence of an oracle that provides some information about the decrypted message.

Depending on the setting, at least three types of oracles can be instantiated when using an IND-CPA secure scheme. These are the *key-mismatch* or *plaintext-checking* (PC) oracle, the *decryption-failure* (DF) oracle, and the *full-decryption* (FD) oracle. The PC oracle typically provides a binary response, either `correct` or `wrong`, about the attacker's guess of the decrypted message (resp., shared secret key) of a PKE (resp., KEM) for a chosen ciphertext. In the presence of a DF oracle, an attacker can infer whether or not a given ciphertext results in a decryption failure. While both the PC and DF oracles only provide binary information, an FD oracle provides information about the complete message for the chosen ciphertexts. Based on the available oracle, an attacker carefully chooses query ciphertexts so that the corresponding oracle's responses reveal the secret key.

All finalists and alternates for PKE/KEMs apply well-known CCA conversions to achieve IND-CCA security against adaptive CCAs. These IND-CCA secure schemes detect invalid/malformed ciphertexts with a very high probability and, upon detection, return failure or a pseudo-random output as a concrete protection against CCAs. The schemes remove the presence of all three aforementioned oracles in an ideal classical *black-box* setting. Any cryptographic algorithm implemented on a real device, however, leaks information about some intermediate values through side-channels. Examples include timing, power consumption, and electromagnetic (EM) emanation.

Following this line of thought, several side-channel assisted CCAs on LWE/LWR-based PKE/KEMs have been proposed. They instantiate different types of oracles to gain information about the decryption output to facilitate secret key recovery in several LWE/LWR-based candidates, including the finalists Kyber [ABD+20b], Saber [DKSRV20], and Frodo [ABD+20a]. A similar analysis is, however, lacking for schemes based on the NTRU problem, including the finalist NTRU [CDH+19] and the alternate NTRU Prime [BBC+20]. Extending such attacks to NTRU-based schemes is nontrivial. The framework and the arithmetic behind NTRU-based schemes are vastly different from those based on the LWE/LWR paradigm. Mounting successful side-channel assisted CCAs on NTRU-based schemes has remained an open challenge.

There are known CCAs on NTRU-based schemes that work in a classical *black-box* setting [JJ00, HGNP+03, DDSV19]. Most existing CCAs have targeted older variants of NTRU. Adapting them to the newer variants requires a significant effort due to differences in the underlying arithmetic and finer technical details. Let us consider the work of Zhang *et al.* in [ZCD21], for example. It presents an attack on the NTRU-HPS variant with a 100% success rate. The attack fails to achieve the same success rate when targeting the NTRU-HRSS variant. NTRU Prime incorporates several optimizations, including the use of rounded ciphertexts. Its arithmetic, over a noncyclotomic field, throws significant challenges to attackers that perform CCAs in either a black-box or a side-channel setting.

Thus, a gap persists in our theoretical understanding on how to mount CCAs over the newer variants of NTRU-based schemes. *Can such attacks be carried out successfully?* If the answer is yes, then another question naturally arises. *Are there significant differences in costs between attacking NTRU-based schemes and LWE/LWR-based schemes?*

To address these critical questions, we propose the first *practical* side-channel assisted CCAs on IND-CCA secure NTRU-based schemes, including NTRU and NTRU Prime KEMs. We attempt to traverse the landscape by demonstrating attacks that instantiate

the three types of oracles on *all parameter sets* of NTRU and NTRU Prime. The key idea is to construct ciphertexts that can instantiate the oracles. The early inspiration comes from the work of Jaulmes and Joux in [JJ00]. They proposed the first CCA that works in a black-box setting on the original NTRU PKE scheme from 1998. Our novel and generic adaptations of their attack achieve full key recovery on unprotected implementations of NTRU and NTRU Prime, with only a few thousand chosen-ciphertext queries to the target device, without the need for offline analysis (or brute forcing).

Assuming the presence of a plaintext-checking oracle for key recovery, we come up with an attack that works with perfect success rate on the NTRU-HRSS variant of NTRU and Streamlined NTRU Prime. We devise novel techniques to surmount the challenges posed by the use of rounded ciphertexts in NTRU Prime and the deployment of arbitrary-weight secrets in the NTRU-HRSS.

We formulate approaches to accomplish three important tasks. The first task is to utilize side-channel leakage from the decapsulation procedure. The next task is to realize practical oracles for plaintext-checking and decryption-failure. The final task is to efficiently recover the keys. Since these oracles only provide binary information, the side-channel analysis relies on simple techniques and can be performed with minimal knowledge about the target implementation. Our claims are *experimentally validated* by exploiting electromagnetic emanation (EM) side-channel on optimized implementations of NTRU and NTRU Prime KEM, taken from the `pqm4` library [KRSS19], running on the 32-bit ARM Cortex-M4 microcontroller.

Please note that all our attacks are demonstrated on unprotected implementations of NTRU-based schemes. Masking in the first or higher order serves as a concrete countermeasure against our attacks. While there are several masking strategies for LWE/LWR-based schemes [BDK+21, BGR+21, OSPG18], we are unaware of a concrete masking scheme for NTRU-based schemes. Thus, our work stresses on the need for concrete masking countermeasures for NTRU-based PKE/KEMs to protect against side-channel assisted CCAs.

Availability of software

For scrutiny and reproducibility, we have made our implementation softwares available at https://github.com/PRASANNA-RAVI/SCA_Assisted_CCA_on_NTRU.

Organization of the Paper

Section 2 provides the necessary background by introducing the required notation and concepts as well as useful known results. Sections 3 and 4 present our proposed PC oracle-based attack. The discussion covers the attack routes on NTRU Prime and on NTRU, respectively. Sections 5 and 6 discuss our DF oracle-based and FD oracle-based SCAs, in that order. Section 7 contains concluding remarks, including a brief discussion on potential countermeasures.

2 Lattice Preliminaries

2.1 Notation

We denote by $\mathbb{Z}/q\mathbb{Z}$ or $\mathbb{Z}_q$, the ring of integers modulo an integer q. The elements are zero-centered in $[-q/2, q/2-1] \cap \mathbb{Z}$ when q is even and in $[-(q-1)/2, (q-1)/2] \cap \mathbb{Z}$ when q is odd. For brevity, the *threshold* is $q/2$ throughout, irrespective of the parity of q. Let $\mathbb{Z}_q[x]/\langle \phi(x) \rangle$ denote the polynomial ring whose reduction polynomial is $\phi(x)$. Its elements are polynomials whose coefficients come from $\mathbb{Z}_q$. We use R_q to denote a polynomial ring. Polynomials in R_q are written in bold lower case letters. The i^{th} coefficient of a polynomial

$\mathbf{a} \in R_q$ is denoted by $\mathbf{a}[i]$. The product of polynomials $\mathbf{a}$ and $\mathbf{b}$ is written as $\mathbf{a} \cdot \mathbf{b}$. A polynomial is *small* if its coefficients are in $\mathbb{Z}_3 := \{-1, 0, 1\}$. A polynomial is *of weight* w if exactly w of its coefficients are nonzero. An element $\mathbf{x} \in R_q$ which is sampled from a distribution $\mathcal{D}$ with standard deviation σ is denoted by $\mathbf{x} \leftarrow \mathcal{D}_\sigma(R_q)$.

An array of bytes of an arbitrary length is denoted by $\mathcal{B}^*$. Byte arrays of length n are written as $\mathcal{B}^n$. The i^{th} bit in an element $x \in \mathbb{Z}_q$ is denoted by x_i. The acquisition of a side-channel trace t corresponding to a particular operation $\mathcal{X}$ on an input p is denoted by $t \Longleftarrow \mathcal{X}(p)$.

2.2 NTRU One-Way Function

In 1998, Hoffstein, Pipher, and Silverman proposed the original NTRU PKE in [HPS98]. Its security relies on a conjectured circular security assumption called the *NTRU assumption* or the *NTRU one-way function* (OWF).

Definition 1 (NTRU OWF)**.** Given $R_{\text{NTRU}} := \mathbb{Z}_q[x]/\langle x^N - 1\rangle$, a small invertible polynomial $\mathbf{p} \leftarrow \mathcal{D}_\sigma(R_{\text{NTRU}})$, and another small polynomial $\mathbf{g} \leftarrow \mathcal{D}_\sigma(R_{\text{NTRU}})$, distinguish structured samples $\mathbf{g} \cdot \mathbf{p}^{-1} \in R_{\text{NTRU}}$ from uniformly random samples in $\mathcal{U}(R_{\text{NTRU}})$.

The problem is reducible to a *shortest vector problem* (SVP) over a special class of lattices known as the *NTRU lattices* [CS97]. The NTRU cryptosystem has survived cryptanalysis for almost 24 years now. This instills confidence in its security claims, despite the lack of provable security guarantees. To distinguish the original NTRU PKE from the finalist NTRU and the alternate NTRU Prime, we call the original scheme NTRU-1998.

2.3 NTRU Prime

NTRU Prime is a suite of two IND-CCA secure KEMs, namely Streamlined NTRU Prime and NTRU LPRime. The former is based on the NTRU paradigm. The latter is based upon the LPR Encrypt paradigm [BBC+20]. We focus on the Streamlined NTRU Prime variant and, henceforth, refer to it as NTRU Prime. At its core is a perfectly correct and deterministic IND-CPA secure PKE. It is defined by the tuple (n, q, w), where n and q are prime numbers and w is a positive integer with the restrictions

$$2n \geq 3w, \quad q \geq 16w + 1, \quad x^n - x - 1 \text{ is irreducible in } \mathbb{Z}_q[x].$$

Unlike NTRU-1998, which operates in a *cyclotomic ring* $(\mathbb{Z}/q\mathbb{Z})[x]/\langle x^n - 1\rangle$, with $n = 2^k$, NTRU Prime operates in the *field* $R_q := \mathbb{Z}_q[x]/\langle x^n - x - 1\rangle$, which is *not cyclotomic*. The choice is motivated by the need to protect against potential attacks, *e.g.*, those discussed in [KEF20], that exploit the cyclotomic structure.

Algorithm 1 describes the NTRU Prime PKE. GenSmall() takes in a seed $\rho \in \mathcal{B}^*$ and samples for small polynomials in R_3. GenShort uses $\rho \in \mathcal{B}^*$ to sample for polynomials of small weight w from the space R_{sh}. Round rounds every coefficient of a given polynomial to its nearest multiple of 3.

The key generation NTRU_PRIME_PKE.KeyGen produces an NTRU instance $\mathbf{h} = \mathbf{g}/(3\mathbf{f}) \in R_q$ with $\mathbf{g} \in R_3$ and $\mathbf{f} \in R_{\text{sh}}$. The secret key is formed by $\mathbf{f}$ and $\mathbf{g}$. The public key is $\mathbf{h} \in R_q$. The encryption NTRU_PRIME_PKE.Encrypt takes as input the *message polynomial* $\mathbf{r} \in R_{\text{sh}}$ and generates the ciphertext $\mathbf{c} = \text{Round}(\mathbf{r} \cdot \mathbf{h}) \in R_q$. Its coefficients are multiples of 3. The decryption NTRU_Prime_PKE.Decrypt takes $\mathbf{c}$ to compute $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c} \in R_q$. The parameters are chosen to ensure that the true, that is the nonreduced, value of every coefficient $\mathbf{a}[i]$ for $i \in [0, p-1]$ always lies in the zero-centered range $(-q/2, q/2]$. A suitable choice for the parameters, leading to the $\mathbf{a}$ in line 3, is key to the correctness of the decryption procedure. The resulting $\mathbf{a}$ is then reduced modulo 3 to yield $\mathbf{e} = \mathbf{g} \cdot \mathbf{r} \in R_3$. The latter, upon multiplication with

Algorithm 1: Streamlined NTRU Prime PKE Core

Procedure NTRU_PRIME_PKE.KeyGen()
  while $\mathbf{g}$ *is not invertible in* R_3 **do**
    $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$, $\mathbf{g} \leftarrow \mathsf{GenSmall}(\rho) \in R$
  end
  $\hat{\mathbf{g}} = 1/\mathbf{g} \in R_3$
  $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$, $\mathbf{f} \leftarrow \mathsf{GenShort}(\rho) \in R_{\text{sh}}$
  $\mathbf{h} = \mathbf{g}/(3\mathbf{f}) \in R_q$
  return $(pk = \mathbf{h}, sk = (\hat{\mathbf{g}}, \mathbf{f}))$

Procedure NTRU_PRIME_PKE.Encrypt$(pk, \mathbf{r} \in R_{\text{sh}})$
  $\mathbf{d} = \mathbf{h} \cdot \mathbf{r} \in R_q$
  $\mathbf{c} = \mathsf{Round}(\mathbf{d}) \in R_q$
  $ct = \mathsf{Encode}(\mathbf{c})$
  return (ct)

Procedure NTRU_PRIME_PKE.Decrypt(ct, sk)
  $\mathbf{c} = \mathsf{Decode}(ct) \in R_q$
  $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c} \in R_q$
  $\mathbf{e} = \mathbf{a} \bmod R_3$
  $\mathbf{b}' = \mathbf{e} \cdot \hat{\mathbf{g}} \in R_3$
  if $\mathsf{Weight}(\mathbf{b}') = w$ **then**
    return $\mathbf{r}' = \mathbf{b}'$
  end
  else
    return $\mathbf{r}' = (1, 1, \ldots, 1, 0, 0, \ldots, 0) \in R_{\text{sh}}$
  end

$\hat{\mathbf{g}} \in R_3$, results in $\mathbf{b}'$. Subsequently, the weight of $\mathbf{b}'$ is checked. If $\mathsf{Weight}(\mathbf{b}') = w$, then $\mathbf{r}' = \mathbf{b}'$ is the valid decryption output. Otherwise, the decryption output is fixed to be $(1, 1, \ldots, 1, 0, 0, \ldots, 0) \in R_3$.

The NTRU Prime PKE core is only IND-CPA secure and, hence, is susceptible to CCAs. The well-known Fujisaki Okamoto (FO) transform [FO99] can convert it into an IND-CCA secure KEM. The transform instantiates NTRU_PRIME_PKE.Encrypt, NTRU_PRIME_PKE.Decrypt, and several instances of hash functions in the IND-CCA secure encapsulation and decapsulation procedures. Algorithm 2 supplies the details. In theory, the FO transform helps check the validity of ciphertexts through a re-encryption procedure after decryption in line 5 of NTRU_Prime_KEM.Decaps. Thus, the attacker only sees, with a very high probability, decapsulation failures for invalid ciphertexts. This provides strong *theoretical* security guarantees against CCAs.

2.4 NTRU

NTRU provides a suite of IND-CCA secure KEMs. Similar to NTRU Prime, NTRU's core contains a perfectly correct and deterministic IND-CPA secure PKE. It is parameterized by pairwise coprime integers n, p, q, sample spaces $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$, and an injection lift operation $\mathsf{Lift} : \mathcal{L}_m \to \mathbb{Z}_x$, $p = 3$ and q is a power of 2. Let $k \in \mathbb{Z}^+$, $\phi_1 = (x - 1)$, and $\phi_n = (x^{n-1} + x^{n-2} + \ldots + 1)$. We note that $\phi_1 \cdot \phi_n = (x^n - 1)$.

NTRU computes over two polynomial rings $S_k := \mathbb{Z}_k[x]/\langle \phi_n \rangle$ and $T_k := \mathbb{Z}_k[x]/\langle \phi_1 \cdot \phi_n \rangle$. It offers parameter sets that fall into two broad categories, namely, NTRU-HPS and NTRU-HRSS. While they share several unified design choices, there are notable differences.

Algorithm 2: The FO transform from IND-CPA into IND-CCA secure KEM

Procedure NTRU_Prime_KEM.Encaps(pk)
  $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
  $\mathbf{r} = \mathsf{GenShort}(\rho) \in R_{\text{sh}}$
  $\mathbf{c} = \mathsf{NTRU\_PRIME\_PKE.Encrypt}(pk, \mathbf{r})$
  $\mathbf{d} = \mathcal{H}(\mathbf{r}, pk)$
  $ct = (\mathbf{c}, \mathbf{d}),\ \ K = \mathcal{G}(1, \mathbf{r}, ct)$
  return ct, K

Procedure KEM.Decaps(sk, pk, ct)
  $ct = (\mathbf{c}, \mathbf{d})$
  $\mathbf{r}' = \mathsf{NTRU\_PRIME\_PKE.Decrypt}(ct, sk)$
  $d' = \mathcal{H}(\mathbf{r}', pk)$
  $c' = \mathsf{NTRU\_PRIME\_PKE.Encrypt}(pk, \mathbf{r}')$
  $ct' = (c', d')$
  if $ct' = ct$ **then**
    return $K = \mathcal{G}(1, \mathbf{r}', ct')$
  end
  else
    return $K = \mathcal{G}(1, \rho', ct')$ `/* ` $\rho' \in \mathcal{B}^{32}$ `is a random secret */`
  end

NTRU-HPS, like NTRU-1998, selects coefficients from *fixed-weight* sample spaces. NTRU-HRSS selects coefficients from an *arbitrary-weight* sample space. We refer the reader to [CDH+19] for the respective details of both variants.

Without loss of generality, we use the NTRU-HPS PKE to describe the procedures of the NTRU PKE core in Algorithm 3. Sample_fg() takes in a seed $\rho \in \mathcal{B}^*$ and samples the secret polynomials $\mathbf{f}, \mathbf{g} \in R_3$. The key generation procedure NTRU_PKE.KeyGen produces an instance $\mathbf{h} = 3\mathbf{g}/\mathbf{f} \in T_q$, with $(\mathbf{f}, \mathbf{g})$ forming the secret key and $\mathbf{h} \in T_q$ forming the public key. We highlight here the change in position of the multiplier 3 in $\mathbf{h}$ compared to its position in NTRU Prime, where $\mathbf{h} = \mathbf{g}/(3\mathbf{f}) \in R_q$.

The encryption NTRU_PKE.Encrypt takes a random $\mathbf{r} \in \mathcal{L}_r$ and a message $\mathbf{m} \in \mathcal{L}_m$ to generate the ciphertext $\mathbf{c}$ as $\mathbf{h} \cdot \mathbf{r} + \mathsf{Lift}(\mathbf{m}) \in T_q$, as shown in line 3. The decryption NTRU_PKE.Decrypt uses $\mathbf{c}$ to compute $\mathbf{a} \in \mathbf{f} \cdot \mathbf{c} \in T_q$ in line 7. Just like in NTRU Prime, the true value of every coefficient of $\mathbf{a}$ is in $\mathbf{Z}_q$. This is the key to the perfect correctness of the NTRU PKE. Subsequently, $\mathbf{a} \in T_q$ is reduced modulo S_3 and multiplied with $\mathbf{f}_p$ to form the message polynomial $\mathbf{m}'$, which is then used to recover the random polynomial $\mathbf{r}'$ in lines 10 and 11. Line 12 says that the decryption procedure returns the polynomial pair $(\mathbf{r}', \mathbf{m}')$ as the decryption output only if $(\mathbf{r}', \mathbf{m}') \in (\mathcal{L}_r \times \mathcal{L}_m)$. Otherwise, it returns the fixed value $(1, 1)$. The decryption procedure also generates a single bit called `fail`, with `fail`$= 0$ denoting success and `fail`$= 1$ denoting failure.

Unlike NTRU Prime KEM and several other LWE/LWR-based KEMs, NTRU KEM achieves IND-CCA security without re-encryption, since the underlying NTRU PKE core achieves the Bernstein-Persichetti rigidity [BP18]. This makes the decapsulation procedure of NTRU among the fastest compared to other lattice-based KEMs. Algorithm 4 gives the encapsulation and decapsulation procedures of NTRU KEM.

2.5 Side-Channel assisted CCAs on LWE/LWR-based schemes

While IND-CCA secure KEMs are theoretically secure against CCAs, their security properties are only valid as long as an attacker is unable to obtain any information about

Algorithm 3: NTRU PKE Core

Procedure NTRU_PKE.KeyGen()
- $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
- $(\mathbf{f}, \mathbf{g}) \leftarrow \mathsf{Sample\_fg}(\rho) \in (\mathcal{L}_f, \mathcal{L}_g)$
- $\mathbf{f}_q = (1/\mathbf{f}) \in S_q$
- $\mathbf{h} = (3 \cdot \mathbf{g} \cdot \mathbf{f}_q) \in T_q$
- $\mathbf{h}_q = (1/\mathbf{h}) \in S_q$
- $\mathbf{f}_p = (1/\mathbf{f}) \in S_3$
- **return** $(pk = (\mathbf{h}, \mathbf{h}_q), sk = (\mathbf{f}, \mathbf{f}_p))$

Procedure NTRU_PKE.Encrypt$(pk, (\mathbf{r}, \mathbf{m}) \in (\mathcal{L}_r \times \mathcal{L}_m))$
- $\bar{\mathbf{m}} = \mathsf{Lift}(\mathbf{m}) \in S_3$
- $\mathbf{c} = \mathbf{h} \cdot \mathbf{r} + \bar{\mathbf{m}} \in T_q$
- $ct = \mathsf{Encode}(\mathbf{c})$
- **return** (ct)

Procedure NTRU_PKE.Decrypt(ct, sk)
- $\mathbf{c} = \mathsf{Decode}(ct) \in T_q$
- **if** $\mathbf{c} \not\equiv 0 \ mod\ (q, \phi_1)$ **then**
  - $\mathtt{fail} = 1$
  - **return** $(0, 0, \mathtt{fail})$
- **end**
- $\mathbf{a} = \mathbf{f} \cdot \mathbf{c} \in T_q$
- $\mathbf{e} = \mathbf{a} \bmod S_3$
- $\mathbf{m}' = \mathbf{e} \cdot \mathbf{f}_p \in S_3$
- $\bar{\mathbf{m}}' = \mathsf{Lift}(\mathbf{m}')$
- $\mathbf{r}' = (\mathbf{c} - \bar{\mathbf{m}}') \cdot \mathbf{h}_q \in S_q$
- **if** $\mathbf{r}', \mathbf{m}' \in (\mathcal{L}_r \times \mathcal{L}_m)$ **then**
  - $\mathtt{fail} = 0$
  - **return** $(\mathbf{r}', \mathbf{m}', \mathtt{fail})$
- **end**
- **else**
  - $\mathtt{fail} = 1$
  - **return** $(0, 0, \mathtt{fail})$
- **end**

the intermediate variables in the decapsulation procedure. Side-channel leakage that reveals sensitive information about any of the variables can lead to serious security flaws. The most severe outcome is a complete recovery of the secret key.

KEMs based on the LWE/LWR problem have been subjected to several side-channel assisted CCAs [DTVV19, RRCB20, GJN20]. Their modus operandi starts with the attacker constructing specially structured ciphertexts. When decrypted/decapsulated, the ciphertexts ensure that a certain intermediate variable, referred to as the *anchor variable*, is very closely related to a targeted portion or, in the best scenario for the attacker, the complete secret key. CCAs on IND-CPA secure LWE/LWR-based schemes have revealed the efficacy of specially constructed ciphertexts to turn the decrypted message into an anchor variable. Once the attacker recovers the value of the anchor variable for the chosen ciphertexts using side-channels, the full secret key can be recovered. Based on the type and amount of side-channel information available, existing attacks on LWE/LWR-based schemes fall into the following three categories.

Algorithm 4: IND-CCA secure NTRU KEM

Procedure NTRU_KEM.Encaps(pk)
- $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
- $(\mathbf{r}, \mathbf{m}) = \mathsf{Sample\_rm}(\rho)$
- $\mathbf{c} = \mathsf{NTRU\_PKE.Encrypt}(pk, \mathbf{r}, \mathbf{m})$
- $k = \mathcal{H}(\mathbf{r}, \mathbf{m})$
- $ct = \mathbf{c}$
- **return** (ct, k)

Procedure NTRU_KEM.Decaps(sk, pk, ct)
- $ct = (\mathbf{c}, \mathbf{d})$
- $(\mathbf{r}', \mathbf{m}', \texttt{fail}) = \mathsf{NTRU\_PKE.Decrypt}(sk, ct)$
- $k_1 = \mathbf{H}(\mathbf{r}', \mathbf{m}')$
- $k_2 = \mathbf{G}(s, \mathbf{c})$
- `/*` $s \in \mathcal{B}^{32}$ `is a random secret */`
- **if** $fail = 0$ **then**
  - **return** k_1
- **end**
- **else**
  - **return** k_2
- **end**

2.5.1 Plaintext-Checking Oracle-Based SCA

The attacker constructs chosen ciphertexts such that the anchor variable only assumes a very small number of possible values known to the attacker. Each possible value exclusively depends on a targeted portion of the secret key. An attacker who can utilize side-channels to retrieve the value of the anchor variable realizes an artificial *plaintext-checking* (PC) oracle. Its responses can then be used to recover the full secret key.

For LWE/LWR-based schemes such as Kyber and Saber, the decrypted messages for chosen ciphertexts can be restricted to two values. These are $m = 0$, on the occurrence of the all-zero bit string m_0, and $m = 1$, on the occurrence of the string m_1 whose entries are all 0 except at the least significant bit, where the entry is 1. Side-channels such as timing and electromagnetic emanation have been shown to be efficiently exploitable to realize a PC oracle in IND-CCA secure schemes. The binary responses, each in the form of $m \in \{0, 1\}$, can recover the full secret key in a few thousand chosen-ciphertext queries to the target decapsulation device [DTVV19, RRCB20].

2.5.2 Decryption-Failure Oracle-Based SCA

The second class of attacks performs key recovery by exploiting side-channels to obtain information about decryption failures for the attacker's chosen ciphertexts. Crafted errors are added to a valid ciphertext to trigger decryption failures. Whether $m = m_{\text{valid}}$ or m_{invalid} depends upon a targeted portion of the secret key. Similar to the PC oracle-based SCA, side-channels can detect decryption failures. This realizes a *decryption-failure* (DF) oracle whose responses can recover the full secret key. Guo, Johansson, and Nilsson in [GJN20] exploited timing side-channel information from nonconstant time ciphertext comparison in Frodo KEM to detect decryption failures. Subsequently, Bhasin *et al.* in [BDH+21] exploited EM side-channel vulnerabilities in several masked ciphertext comparison approaches to realize a DF oracle in Kyber KEM. Both attacks could perform a full key recovery with several thousand chosen ciphertext queries to the target device.

Table 1: Classification of side-channel assisted CCAs on IND-CCA secure LWE/LWR-based schemes according to oracle type. The anchor variable is denoted by anchor and m_x, with or without subscript, is the decrypted message.

Type of Oracle	Oracle Response
plaintext-checking (PC)	anchor $\in \{m_0, m_1\}$
decryption-failure (DF)	anchor $\in \{m_{\text{valid}}, m_{\text{invalid}}\}$
full-decryption (FD)	anchor $= m$

Both PC oracle and DF oracle-based SCA only extract binary information about the anchor variable through side-channels. Thus, these attacks can be carried out with a relatively simple attack setup. They do not pose stringent requirements on the *signal to noise ratio* (SNR) for trace acquisition. The analysis is fairly simple and can be performed with very limited knowledge of the target implementation.

2.5.3 Full-Decryption Oracle-Based SCA

The PC oracle and DF oracle attacks only extract one bit of information about the anchor variable through side-channel traces. They typically require a few thousand queries for full key recovery, especially given the size of secrets used in lattice-based KEMs. This raises a natural question about the possibility of more efficient attacks with a more powerful oracle to gather more than just binary information about the decrypted message. In this direction, Xu *et al.* [XPRO20] showed that an attacker who can obtain a complete knowledge of the decrypted message m for chosen ciphertexts can effectively run the CCA *in parallel mode*, resulting in full key recovery after only a handful of queries. The authors demonstrated full key recovery using only 8 to 16 in LWE/LWR-based KEMs such as Kyber and Saber. They exploited vulnerabilities in the message encoding as treated, for examples, in Amiet *et al.* [ACLZ20] and Sim *et al.* [SKL+20], and in the decoding procedure that leaks the complete message as discussed, for examples, by Ravi *et al.* in [RBRC20] and Ngo *et al.* in [NDGJ21]. Table 1 lists side-channel assisted CCAs on IND-CCA secure LWE/LWR-based schemes by their oracle types.

While the above attacks work on IND-CCA secure LWE/LWR-based KEMs, they do not extend trivially to NTRU-based KEMs. This is because the underlying arithmetic of schemes based on the LWE/LWR paradigm is vastly different compared with schemes in the NTRU paradigm. Mounting similar side-channel attacks in a chosen-ciphertext setting on NTRU-based schemes has been an open problem. Even if nontrivial extension

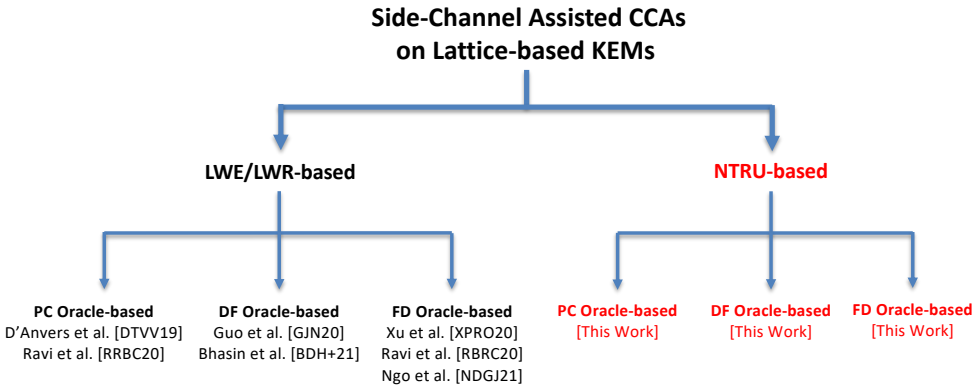


Figure 1: Classification of various side-channel assisted CCAs on lattice-based KEMs.

of such attacks can be carried out, the comparative cost of attacking NTRU-based KEMs in a chosen-ciphertext setting is previously unknown. To address these two questions we exhibit the first side-channel assisted CCAs on NTRU-based schemes. Our attacks are practical, generic, and capable of exploiting all three types of oracles for full key recovery. The attacks apply to all variants of the NTRU-based KEMs in the final round of the NIST PQC process. Figure 1 gives a classification of the various side-channel assisted CCAs attacks on lattice-based KEMs. Ours are highlighted in red.

2.6 CCAs on NTRU-based schemes

Several CCAs have been proposed on different variants of the NTRU PKE cryptosystem. Jaulmes and Joux [JJ00] presented the first CCA on the unpadded version of NTRU-1998. Their attack requires knowledge of the full decryption output and can recover the full secret key with a handful of ciphertexts. They also gave an adaptation of their attack to the padding scheme, which was akin to the optimal asymmetric encryption padding. The adapted variant only needs the DF oracle for key recovery. Similarly, Hoffstein and Silverman used the DF oracle on the unpadded NTRU-1998 in their CCAs [HS99].

Han *et al.* [HHHK03] subsequently came up with very efficient CCAs, based on the FD oracle, on optimized variants of unpadded NTRU-1998. Their attacks utilize chosen ciphertexts that are completely precomputed offline, independent of the previous outputs. While the aforementioned attacks utilize invalid or maliciously crafted ciphertexts, another class of CCAs exploits decryption failures for valid ciphertexts [HGNP+03, GN07]. While these attacks apply to variants of NTRU cryptosystem with nonnegligible decryption failure rate, they are not relevant for NTRU and NTRU Prime.

More recently, Ding *et al.* [DDSV19] formulated a novel CCA on NTRU-1998 using the DF oracle. This attack, with trivial modifications, can be adapted to the NTRU-HPS parameter set, assuming a PC oracle. Zhang *et al.* [ZCD21] showed that the attack fails on NTRU-HRSS, due to the use of secrets with arbitary weight. Although they managed to modify the attack to work on NTRU-HRSS, the improved technique can only recover 93.6% of the keys. Thus, there is no known CCA against NTRU-HRSS that works with a 100% success rate. To the best of our knowledge, there is no CCA on NTRU Prime.

We will soon show that mounting CCAs on NTRU Prime is particularly challenging since the scheme uses rounded ciphertexts as well as conditional checks on the decrypted message. In this work, we improve on the CCA of Jaulmes and Joux and propose generic and novel adaptations to NTRU and NTRU Prime. Ours perform full key recovery with perfect success rate on all parameter sets, assuming the presence of a suitable oracle.

2.7 Test Vector Leakage Assessment

The test vector leakage assessment (TVLA) from [GJJR11] is a popular conformance-based methodology in side-channel analysis. It has been widely used in both academia and industry to evaluate cryptographic implementations. TVLA computes the univariate Welch's t-test over two given sets of side-channel measurements to identify their differentiating features. By testing for a null hypothesis that the mean of the two sets is identical, a `PASS/FAIL` decision is made. TVLA is formulated over measurement sets $\mathcal{T}_r$ and $\mathcal{T}_f$ by

$$\text{TVLA} := \frac{\mu_r - \mu_f}{\sqrt{\frac{\sigma_r^2}{m_r} + \frac{\sigma_f^2}{m_f}}} \ , \tag{1}$$

where μ_r, σ_r, and m_r (resp. μ_f, σ_f, and m_f) are the mean, standard deviation and cardinality of the trace set $\mathcal{T}_r$ (resp. $\mathcal{T}_f$). The null hypothesis is rejected with a confidence of 99.9999% only if the absolute value of the t-test score is > 4.5. A rejected null hypothesis

implies that the two sets of measurements are different. It might leak some side-channel information and, hence, is considered a `FAIL` test. The threshold was later shown to depend on the length of the side-channel trace [DZD+17]. We choose 5 as the threshold based on experimental settings.

While TVLA is mainly used as a metric for side-channel evaluation, it has also been used as a tool for feature selection in multiple cryptanalytic efforts [RJJ+18]. Here we use TVLA as a tool for *feature selection* from side-channel measurements [GLRP06].

3 Plaintext-Checking Oracle-Based SCA

We primarily use NTRU Prime, instead of NTRU, to describe our PC-oracle attack. The former comes with complications that arise due to the use of rounded ciphertexts. Once we have described the attack on NTRU Prime, we adapt it to NTRU. Our attack works in two phases. We construct malicious ciphertexts and, subsequently, utilize side-channel information from the decryption of these malicious ciphertexts to perform key recovery.

1. **Preprocessing Phase:** We search for a ciphertext that, when decrypted, leads to what we refer to as a *single collision* event. We query the decapsulation device with specially crafted ciphertexts and analyze their side-channel leakage to detect the event. Such a ciphertext is called a *base ciphertext*, denoted by $\mathbf{c}_{\text{base}}$. We use it to infer crucial information about the secret polynomials $\mathbf{f}$ and $\mathbf{g}$.

2. **Key Recovery Phase:** We use $\mathbf{c}_{\text{base}}$ to construct new attack ciphertexts. They are built in such a way that, upon decryption, their corresponding internal variable $\mathbf{e}$, in line 4 of NTRU_Prime_PKE.Decrypt procedure in Algorithm 1, can only belong to either one of two exclusive classes, namely $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$, with a single nonzero coefficient. Moreover, the value of $\mathbf{e}$ depends on a targeted portion of the secret key. We exploit side-channel leakage from the operations that manipulate $\mathbf{e}$ to obtain information about its value and devise a practical PC oracle. The oracle's responses, consisting of $\mathbf{e} = 0$ or $\mathbf{e} \neq 0$, obtained for several attack ciphertexts, are used to recover the full secret key.

Figure 2 describes our PC oracle-based SCA on the decryption procedure of Streamlined NTRU Prime KEM. The next two subsections describe the phases in our attack.

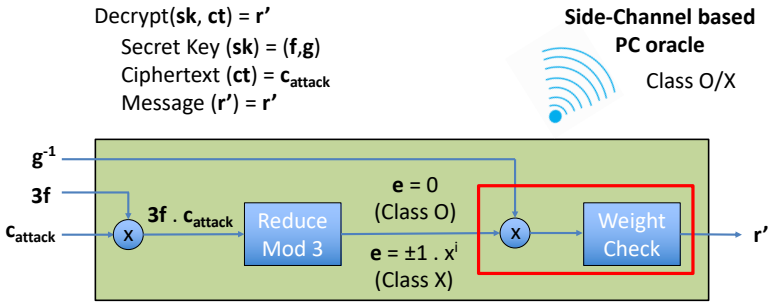


Figure 2: A pictorial illustration of our PC oracle-based SCA on NTRU Prime

3.1 Preprocessing Phase: Retrieving the Base Ciphertext

We start with an intuition for the approach before proposing a concrete methodology. The notation used is from Algorithm 1.

3.1.1 Intuition

We first analyze the effect of decrypting $\mathbf{c} = k + k \cdot \mathbf{h}$, where $k \in \mathbb{Z}^+$, by looking at $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}$ in line 3 of NTRU_Prime_PKE.Decrypt procedure as

$$\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c} = k \cdot 3\mathbf{f} + k \cdot 3\mathbf{f} \cdot \mathbf{h} = k \cdot 3\mathbf{f} + k \cdot 3\mathbf{f} \cdot (\mathbf{g}/3\mathbf{f}) = 3k \cdot \mathbf{f} + k \cdot \mathbf{g}. \tag{2}$$

The coefficients of both $\mathbf{f}$ and $\mathbf{g}$ are in $\{-1, 0, 1\}$. Thus, the largest absolute value of any coefficient $\mathbf{a}[i]$ is obtained when the corresponding $\mathbf{f}[i]$ and $\mathbf{g}[i]$ simultaneously take their absolute maximum values, that is, when $\mathbf{f}[i] = \mathbf{g}[i] = \pm 1$. We call the event when the corresponding coefficients of two or more polynomials attain their maximum absolute value a *collision*. Thus, $\mathbf{a}[i] = 4k$ (resp. $-4k$) when $\mathbf{f}[i] = \mathbf{g}[i] = +1$ (resp. -1). We now choose a suitable positive integer k, with $3 \mid k$, based on the conditions

$$4k > q/2 \text{ and } s \cdot k < q/2 \text{ for } s \in [0, 3]. \tag{3}$$

For the sake of explanation, let $\mathbf{f}$ and $\mathbf{g}$ only collide at the the i^{th} coefficient with the value of $+1$. Hence, $\mathbf{a}$ has the coefficients

$$\mathbf{a}[j] > q/2 \text{ if } j = i \text{ and } \mathbf{a}[j] < q/2 \text{ if } j \neq i. \tag{4}$$

Since $3 \mid k$, it is clear that $3 \mid \mathbf{a}[i]$, for $i \in [0, n-1]$. When $\mathbf{a}$ is reduced modulo q and zero-centered in $(-q/2, q/2]$, all coefficients, except for $\mathbf{a}[i]$, retain their true value and remain a multiple of 3. This is because every time $\mathbf{a}[i]$ crosses the $q/2$ threshold, that is, whenever $\mathbf{a}[i] > q/2$, and upon subsequent reduction modulo q, we subtract the prime q from $\mathbf{a}[i]$. More explicitly,

$$\mathbf{a} \bmod q = \mathbf{a} - q \cdot x^i. \tag{5}$$

Hence, $\mathbf{e} = \mathbf{a} \bmod 3 \ \in R_3$ is nothing but

$$\mathbf{e} = (-q \bmod 3) \cdot x^i. \tag{6}$$

The approach ensures that $\mathbf{a}[i]$ crosses the $q/2$ threshold only during a collision. When there is no collision, $\mathbf{a}[i] < q/2$. Thus, for a choice of k in Equation (3), $\mathbf{e}[i] \neq 0$ signifies a collision at i, while all other coefficients remain zero.

The same scenario applies when the collision value is -1. Subsequently, $\mathbf{a}[i] < -(q/2)$ and, hence, when q is added to $\mathbf{a}[i]$ to zero-center it in the range $[-q/2, q/2]$, the corresponding $\mathbf{e}[i] \neq 0$, implying a collision at i. Henceforth, to avoid repetitions, we focus only on collision with the highest positive value of $+1$. The same analysis holds for the lowest negative value of -1.

In our attack, it would be ideal to have a *single collision* between $\mathbf{f}$ and $\mathbf{g}$, resulting in an $\mathbf{e}$ that has a single nonzero coefficient. For illustration we use one particular parameter set of NTRU Prime. Our choice falls on sntrup761 whose $(n, q, w) = (761, 4591, 286)$. We denote by ρ_{single} the probability of a single collision between $\mathbf{f}$ and $\mathbf{g}$. The probability of a collision at any given coefficient is ρ. Letting ρ_x, for $x \in \{-1, 1\}$, be the probability of a collision between $\mathbf{f}$ and $\mathbf{g}$ with a matching coefficient of either -1 or 1, we define

$$\rho_{\text{match}} := \rho_1 + \rho_{-1}.$$

For $\mathbf{f} \in R_{\text{sh}}$ and $\mathbf{g} \in R_3$, we have $\rho_{\text{match}} := (w/3n)$. Hence, for sntrup761, $\rho_{\text{match}} \approx 0.125$ and $\rho_{\text{single}} = n \cdot \rho_{\text{match}} \cdot (1 - \rho_{\text{match}})^{n-1}$, which is impractically low at $8 \cdot 10^{-43}$. We require better choices for the ciphertexts to limit the number of collisions and, thus, the number of nonzero coefficients in $\mathbf{e}$.

3.1.2 Constructing Ciphertexts for Single Collision

We split the value of $\mathbf{a}$ in Equation (2) into

$$\mathbf{a} = 3k \cdot \mathbf{f} + k \cdot \mathbf{g} = 3k \cdot \mathbf{t}_1 + k \cdot \mathbf{t}_2, \tag{7}$$

where $\mathbf{t}_1 = \mathbf{f}$ and $\mathbf{t}_2 = \mathbf{g}$. To limit the number of collisions between $\mathbf{t}_1$ and $\mathbf{t}_2$ we make a generic choice for $\mathbf{c}$. This choice is

$$\mathbf{c} = k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h} = k_1 \cdot \mathbf{d}_1 + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}, \tag{8}$$

where both $\mathbf{d}_1$ and $\mathbf{d}_2$ are polynomials with, respectively, m and n nonzero coefficients ± 1. The corresponding $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}$ is given by

$$\mathbf{a} = k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} \cdot 3\mathbf{f} = 3k_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} = 3k_1 \cdot \mathbf{t}_1 + k_2 \cdot \mathbf{t}_2, \tag{9}$$

where $\mathbf{t}_1 = \mathbf{d}_1 \cdot \mathbf{f}$ and $\mathbf{t}_2 = \mathbf{d}_2 \cdot \mathbf{g}$. The product of $\mathbf{d}$ with x^i modulo $(x^n - x - 1)$ is

$$\begin{aligned}(\mathbf{d} \cdot x^i) \bmod (x^n - x - 1) = \; & \mathbf{d}_{n-i} + (\mathbf{d}_{n-i} + \mathbf{d}_{n-i-1})\, x + \ldots \\ & + (\mathbf{d}_{n-1} + \mathbf{d}_0)x^i + \mathbf{d}_1 x^{i+1} + \ldots + \mathbf{d}_{n-i-1}x^{n-1},\end{aligned} \tag{10}$$

with all coefficients in $\{-2, -1, 0, 1, 2\}$. We denote the resulting product by $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{d}, i)$ and refer to it informally as the rotation of $\mathbf{d}$ by i degrees. Thus,

$$\begin{aligned}\mathbf{t}_1 = \mathbf{d}_1 \cdot \mathbf{f} &= (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) \cdot \mathbf{f} = \mathbf{f} \cdot x^{i_1} + \mathbf{f} \cdot x^{i_2} + \ldots + \mathbf{f} \cdot x^{i_m} \\ &= \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, i_1) + \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, i_2) + \ldots + \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, i_m)\end{aligned} \tag{11}$$

is the sum of rotations of $\mathbf{f}$ by varying degrees, governed by $\{i_1, i_2, \ldots, i_m\}$. Similarly, $\mathbf{t}_2$ is the sum of rotations of $\mathbf{g}$ by the degrees in $\{j_1, j_2, \ldots, j_n\}$. A collision occurs at index i only if all the corresponding coefficients of $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)$, for $u \in \{i_1, i_2, \ldots, i_m\}$, and $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{g}, v)$, for $v \in \{j_1, j_2, \ldots, j_n\}$, are either $+2$ or -2. In other words, a collision occurs at any given coefficient i, if all the m random rotations of $\mathbf{f}$ and n random rotations of $\mathbf{g}$ simultaneously have a value of $+2$ or -2. We observe that the probability of number of collisions between the rotations of $\mathbf{f}$ and $\mathbf{g}$ quickly degrades as (m, n) increase. It is possible to choose (m, n) for the chosen-ciphertexts such that, either one of m or n is zero. If $m = 0$, then we only consider rotations of $\mathbf{g}$ for the attack analysis, while $n = 0$ means, we only consider rotations of $\mathbf{f}$. However, both m and n cannot simultaneously take a value of 0 for the chosen-ciphertexts, since the resulting chosen-cipherext is zero.

For the choice of $\mathbf{c}$ in Equation (8), the maximum possible value for $\mathbf{a}[i]$ in Equation (9) is $3k_1 \cdot 2m + k_2 \cdot 2n$, which is obtained upon a collision. We therefore choose (k_1, k_2) that satisfy three conditions:

$$3 \mid k_1, \quad 3 \mid k_2, \quad 3k_1 \cdot r + k_2 \cdot s \begin{cases} > q/2, & \text{if } r = 2m, s = 2n, \\ < q/2, & \text{otherwise,} \end{cases} \tag{12}$$

with $0 \le r \le 2m$ and $0 \le s \le 2n$. In other words, we choose (k_1, k_2) such that $\mathbf{a}[i] > q/2$ only when there is a collision at i, while $\mathbf{a}[i] < q/2$, otherwise. Thus, $\mathbf{e}[i] \neq 0$ for a collision at i and $\mathbf{e}[i] = 0$, otherwise.

In summary, we select values for (m, n) and (k_1, k_2) for our chosen ciphertexts in the form of Equation (8). The choice for (m, n) should ensure that a single collision can be obtained with a high probability. Given (m, n), we then choose (k_1, k_2) which satisfies the conditions in Equation (12) such that $\mathbf{e}[i] \neq 0$ indicates a collision at the i^{th} coefficient. Thus, we use the term *true single collision* to refer to the scenario where there is only a single collision at index say i which results in $\mathbf{e}[i] \neq 0$ while all other coefficients of $\mathbf{e}$ are zero. We use the term *multiple collisions* to denote the scenario where there are collisions at more than one index, resulting in two or more coefficients of the corresponding $\mathbf{e}$ to be nonzero. Thus, a true single collision is a favourable case for our attack, while multiple collisions is an unfavourable case.

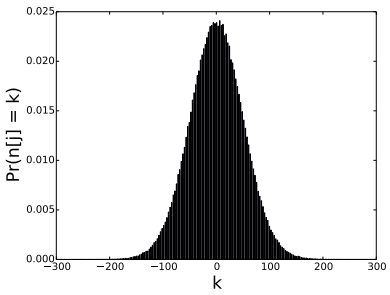


Figure 3: Distribution of the coefficients of the noise $\mathbf{n} := 3\mathbf{f} \cdot \mathbf{m}'$ for sntrup761. It has mean 0 and standard deviation $\sigma \approx 57$.

3.1.3 Additional Challenge: Use of Rounded Ciphertexts

NTRU Prime's encryption procedure generates ciphertexts whose coefficients are rounded to multiples of 3. This rounding is done in line 3 of NTRU_Prime_PKE.Encrypt. The scheme sends only the quotient of each coefficient upon division by 3, reducing the ciphertext size. Thus, every coefficient of the received ciphertext is multiplied by 3 in the decryption. However, our chosen ciphertexts in Equation (8) do not yield exact multiples of 3 and, hence, must be rounded. This introduces a rounding noise $\mathbf{m}' \in R_3$. The actual value of our chosen-ciphertext in the decryption is

$$\begin{aligned}\mathbf{c} &= \mathsf{Round}(k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h}) \\ &= k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h} + \mathbf{m}' \\ &= k_1 \cdot \mathbf{d}_1 + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} + \mathbf{m}' \end{aligned} \tag{13}$$

The corresponding $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}$ becomes

$$\begin{aligned}\mathbf{a} &= k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} \cdot 3\mathbf{f} + \mathbf{m}' \cdot 3\mathbf{f} \\ &= (3k_1 \cdot \mathbf{d}_1 \cdot \mathbf{f}) + (k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g}) + (3\mathbf{f} \cdot \mathbf{m}') = \mathbf{s} + \mathbf{n}, \end{aligned} \tag{14}$$

where $\mathbf{s} := 3k_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g}$ is the *signal component* while $\mathbf{n} := 3\mathbf{f} \cdot \mathbf{m}'$ is the *noise component*. Since $\mathbf{m}' \in R_3$ and $\mathbf{f} \in R_{\text{sh}}$ are small polynomials, the size of the noise is much smaller in comparison to the range q.

Figure 3 shows the distribution of the coefficients $\mathbf{n}[j]$ of sntrup761. It is Gaussian with mean 0 and $\sigma \approx 57$, which is much less than $q = 4591$. The noise polynomial $\mathbf{n} = 3\mathbf{f} \cdot \mathbf{m}'$ is a multiple of 3 and gets rounded to 0 when $\mathbf{a}$ is reduced modulo 3. When $\mathbf{n}$ is added to coefficients of $\mathbf{a}$ near $q/2$, however, the noise is capable of giving rise to a false positive or a false negative collision. For a given choice of (m, n) and (k_1, k_2), the *largest possible coefficient* of $\mathbf{a}$ is $m_1 := 3k_1 \cdot 2m + k_2 \cdot 2n$. The *second largest possible coefficient* is called m_2. As stated in Equation (12), we choose values for (k_1, k_2) such that $m_1 > q/2$ and $m_2 < q/2$. Let $0 \leq r \leq 2m$ and $0 \leq s \leq 2n$. Let

$$\begin{aligned} dm_1 &= \|(3k_1 \cdot 2m + k_2 \cdot 2n) - q/2\| \text{ and} \\ dm_2 &= \left\| \left(\max_{(r,s) \neq (2m, 2n)} (3k_1 \cdot r + k_2 \cdot s) \right) - q/2 \right\| \end{aligned} \tag{15}$$

denote the *distance* between m_1 and m_2, respectively, from $q/2$.

We use the term *false positive* collision to denote the scenario when $\mathbf{e}[i] \neq 0$ even when there is no collision at i (i.e.) $\mathbf{s}[i] < q/2$. For instance, if $\mathbf{s}[i] = m_2$ and the corresponding

Table 2: Concrete values used to build chosen ciphertexts in the two phases of our PC Oracle-based SCA for the stated parameters of NTRU Prime.

Scheme	(m,n)	**Preprocessing Phase**		**Key Recovery Phase**			
		(k_1,k_2)	(dm_1,dm_2)	$(\ell_{11},\ell_{12},\ell_{13})$	(dm_1,dm_2)	$(\ell_{21},\ell_{22},\ell_{23})$	(dm_1,dm_2)
sntrup653	$(0,4)$	$(0,309)$	$(162,147)$	$(0,279,48)$	$(66,69)$	$(0,243,81)$	$(120,123)$
sntrup761	$(0,4)$	$(0,306)$	$(153,153)$	$(0,279,42)$	$(63,63)$	$(0,237,84)$	$(105,132)$
sntrup857	$(0,4)$	$(0,342)$	$(153,189)$	$(0,312,54)$	$(75,75)$	$(0,270,93)$	$(135,135)$
sntrup953	$(0,4)$	$(0,414)$	$(141,273)$	$(0,384,60)$	$(81,99)$	$(0,327,120)$	$(165,162)$
sntrup1013	$(0,4)$	$(0,465)$	$(132,333)$	$(0,435,72)$	$(108,108)$	$(0,375,129)$	$(186,189)$
sntrup1277	$(0,4)$	$(0,510)$	$(141,369)$	$(0,477,78)$	$(111,123)$	$(0,414,138)$	$(201,213)$

coefficient of the noise component $\mathbf{n}[i] > dm_2$, this leads to $\mathbf{a}[i] = \mathbf{s}[i] + \mathbf{n}[i] > q/2$ and thus $\mathbf{e}[i] \neq 0$. Similarly, we use the term *false negative* collision to denote the scenario when $\mathbf{e}[i] = 0$ even when there is a collision at i. This is possible when $\mathbf{s}[i] = m_1$ due to a valid collision, but if the corresponding noise coefficient $\mathbf{n}[i] < -dm_1$, then $\mathbf{s}[i] + \mathbf{n}[i] < q/2$ and $\mathbf{e}[i] = 0$, which suppresses the collision. Both the false positive and false negative scenarios are unfavourable cases, useless for key recovery.

Although the rounding noise $\mathbf{n}$ cannot be removed, the possible occurrence of a false positive or negative collision can be minimized by placing additional constraints in choosing (k_1, k_2). Along with the constraints on (k_1, k_2) in Equation (12), we choose the tuple that *maximizes* dm_1 for m_1 and dm_2 for m_2 to prevent $\mathbf{n}[j]$ from growing large enough to push $\mathbf{a}[j]$ to the other side of $q/2$, which would occasion an error in $\mathbf{e}$. As long as $\mathbf{n}[j]$ does not push $\mathbf{a}[j]$ to the other side of $q/2$, $\mathbf{e}$ remains error-free. In other words, m_1 and m_2 should lie as far as possible on either side of the threshold $q/2$.

Table 2 lists concrete values of (m, n), (k_1, k_2), and the distance tuple (dm_1, dm_2) for different parameter sets of NTRU Prime. These values can be chosen beforehand. Other choices can also be used, albeit with appropriate adjustment in the trace complexity. Table 3 lists the probability of obtaining the ciphertexts corresponding to the different types of collisions, for the chosen parameters. The numbers were empirically obtained through simulations over 1000 attack trials (1000 secret keys) for each parameter set of NTRU Prime. For each secret key, we try with different random chosen ciphertexts until we obtain a true single collision.

We classify the collisions into four types: (1) true single collision (2) multiple collisions (3) false positive collision and (4) false negative collision. The probability to obtain a true single collision, that is, the favourable case, is in the range of 0.032 to 0.064. This indicates the need for between 15 to 32 trials to obtain a true single collision as we go over all parameters. The probability for multiple collisions are roughly an order of magnitude lesser, however nonnegligible. The probabilities for false positive and false collision are much lower than multiple collisions for all parameter sets of NTRU Prime. The base ciphertexts corresponding to the unfavourable cases (multiple, false positive and false negative collisions) do not lead to key recovery. We will explain in Subsection 3.2.3 that instances of unfavourable cases increase the trace complexity without preventing the eventual key recovery.

For all parameter sets of NTRU Prime, we chose $(m, n) = (0, 4)$ since it yields the best trace complexity for key recovery. The tuple provides a nice balance of the favourable and unfavourable cases. Choosing $m = 0$ means that the base ciphertext is built by considering only n rotations of the secret polynomial $\mathbf{g}$. Irrespective of the value of the chosen (m, n), the key recovery phase remains the same and always involves the recovery of the secret polynomial $\mathbf{f}$.

Table 3: The probability of obtaining each type of collisions for the chosen ciphertexts across all parameters of NTRU Prime. The numbers were empirically obtained from 1000 attack trials (1000 secret keys) for each parameter set.

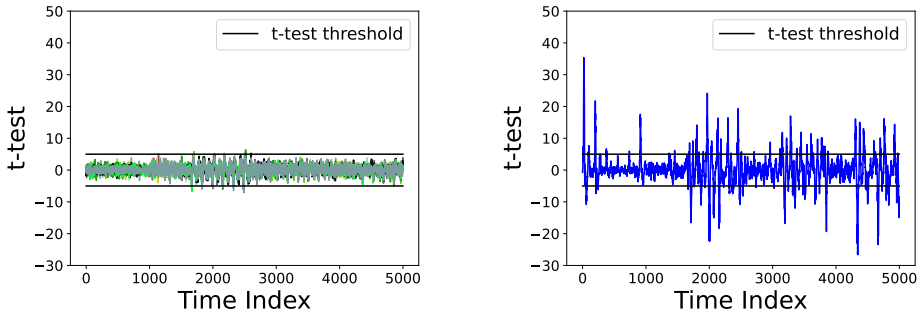
Scheme	Collision Probability			
	True Single	Multiple	False Positive	False Negative
sntrup653	0.0324	0.0016	0.0003	0.0005
sntrup761	0.0372	0.0018	0.0002	0.0015
sntrup857	0.0445	0.0026	0.0003	0.0004
sntrup953	0.0477	0.0032	0.0010	≈ 0
sntrup1013	0.0495	0.0033	0.0019	≈ 0
sntrup1277	0.0642	0.0051	0.0026	≈ 0

3.1.4 Detecting Collision through Side-Channels

Given (m, n) and (k_1, k_2), we randomly select polynomials $\mathbf{d}_1$ and $\mathbf{d}_2$ in Equation (8) until we arrive at a ciphertext $\mathbf{c}$ that has a single nonzero coefficient for $\mathbf{e}$. Since $\mathbf{e}$ is an internal variable, it is impossible to classically obtain information about its value. Hence, we utilize side-channel to identify $\mathbf{e} \neq 0$. This leads to a classification problem with two classes, namely $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$.

For $\mathbf{e} = 0$, line 5 of NTRU_Prime_PKE.Decrypt implies $\mathbf{b}' = \mathbf{e} \cdot \hat{\mathbf{g}} = 0$. Hence, $\mathsf{Weight}(\mathbf{b}') = w_{\mathbf{b}'} = 0$. For $\mathbf{e} \neq 0$ with a single nonzero coefficient, however, $\mathbf{b}' \neq 0$ with uniformly random coefficients in $\{-1, 0, 1\}$ and, hence, $w_{\mathbf{b}'} \neq 0$. Although the exact value depends on the secret polynomial $\mathbf{g}$, the average value of $w_{\mathbf{b}'}$ is 500 for sntrup761. The large weight difference between the two classes should be easily distinguishable through the EM side-channel. The same applies to the other parameter sets.

We ran the optimized implementation of sntrup761 from the open-source `pqm4` library [KRSS] on the STM32F4DISCOVERY board (DUT) housing the STM32F407, ARM Cortex-M4 microcontroller. The implementation, compiled with the options `-O3 -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16`, was clocked at the maximum clock frequency of 168 MHz. EM measurements were observed from the DUT using a near-field probe and processed using a Lecroy HD6104 oscilloscope at a sampling rate of 500MSam/sec. We adopt the Welch's t-test to detect a collision for a chosen ciphertext.



(a) $\mathbf{e} = 0$ (b) $\mathbf{e} \neq 0$

Figure 4: The t-test plots between $\mathcal{T}_{\mathrm{O}}$ and $\mathcal{T}_{\mathrm{X}}$ for sntrup761.

Welch's t-test for Collision Detection: Due to the large weight difference, we fo-

cus on capturing the EM signals from the weight calculation operation in line 6 of NTRU_Prime_PKE.Decrypt. We first obtain T replicated measurements from the decryption of $\mathbf{c} = 0$, which corresponds to $\mathbf{e} = 0$. The trace set is denoted by $\mathcal{T}_{\mathrm{O}}$. To test if a given ciphertext $\mathbf{c}'$ results in a collision, we similarly collect T replicated measurements from the decryption of $\mathbf{c}'$ to form the set $\mathcal{T}_{\mathrm{X}}$. Let $\mathcal{T} = \mathcal{T}_{\mathrm{O}} \cup \mathcal{T}_{\mathrm{X}}$. The Welch's t-test is performed on $\mathcal{T}_{\mathrm{O}}$ and $\mathcal{T}_{\mathrm{X}}$.

- We center each trace $t_i \in \mathcal{T}$ by removing the mean and dividing by its standard deviation to obtain t'_i.
- We compute the Welch's t-test between the *normalized* traces in $\mathcal{T}_{\mathrm{O}}$ and $\mathcal{T}_{\mathrm{X}}$ based on Equation (1). Figure 4(a) and 4(b) depict the t-test plot if $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$ on $T = 10$ replicated measurements. When $\mathbf{e} = 0$ for $\mathbf{c}'$, we do not see any significant peak above and below the ± 5 threshold indicating similarity in both the computations. However for $\mathbf{e} \neq 0$, we can clearly identify several well above the t-test threshold, which clearly denote a collision for $\mathbf{c}'$.

For $\mathbf{c}'$ whose $\mathbf{e} = 0$ in Figure 4(a), there are very few points which border the threshold or marginally exceed it, indicating a relatively minor difference compared with when $\mathbf{c} = 0$. We have examined the internal registers and the control flow to identify any change in behaviour that could result in the t-test values bordering the threshold. We found no discernible change in the state of the device between cases. We hypothesize that computing over an all zero ciphertext $\mathbf{c} = 0$ introduces an *inconsequential bias* in comparison to a nonzero ciphertext $\mathbf{c}'$. This difference is much smaller compared with the difference between $\mathbf{c}'$ with a single collision and $\mathbf{c} = 0$, as can be seen from the very high t-test peaks in Figure 4(b). Thus, the minor difference has a negligible impact on the identification of the base ciphertext and on the success rate of the classification in the key recovery phase, which we will show in Subsection 3.2.2. Instead of using $\mathbf{c} = 0$, we have verified that the usage of chosen-ciphertexts with guaranteed no collision, that is, with high (m, n), also results in a perfect classification with the added benefit of a reduced t-test bias.

It is evident that leakage detection to identify $\mathbf{e} \neq 0$ does not assume any knowledge about the implementation of the decapsulation procedure. In the worst case, the attacker only needs to know the location of the targeted operations within the decryption procedure. Previous works in [ACLZ20, NDGJ21] have shown that visual inspection can identify operations within the decapsulation procedure.

We repeat the test for different choices of $(\mathbf{d}_1, \mathbf{d}_2)$ until we obtain one for which $\mathbf{e} \neq 0$, indicating a possible collision. There is a chance that this collision, instead of being a valid one, is a false positive. Moreover, our technique only realizes a binary oracle that can distinguish between $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$. Thus, we do not know the number of nonzero coefficients in $\mathbf{e}$. If we identify a tuple $(\mathbf{d}_1, \mathbf{d}_2)$ that corresponds to $\mathbf{e} \neq 0$, the corresponding ciphertext is considered to be the base ciphertext $\mathbf{c}_{\mathrm{base}}$, and we simply proceed to the key recovery phase.

Note on False Positive and Multiple Collisions: At this point in the attack, we do not know whether the identified $\mathbf{c}_{\mathrm{base}}$ corresponds to a true single collision. Having $\mathbf{e} \neq 0$ is possible in three different scenarios: (1) true single collision (2) false positive collision or (3) multiple collisions. If $\mathbf{c}_{\mathrm{base}}$ corresponds to a true single collision, then the attack ciphertexts, built using the valid $\mathbf{c}_{\mathrm{base}}$, yield the correct secret key. In the other two scenarios, the attack ciphertexts fail to yield the secret key. When this happens, we simply restart the attack with the search for a new base ciphertext that corresponds to a true single collision.

As shown in Table 3, the parameters to build the ciphertexts are chosen such that a ciphertext that corresponds to true single collision can be identified with a high probability. For analytical purposes, we assume that the $\mathbf{c}_{\mathrm{base}}$ that corresponds to $\mathbf{e} \neq 0$ has a single

nonzero coefficient at index i. We use $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$ to denote the tuple $(\mathbf{d}_1, \mathbf{d}_2)$, with m and n nonzero coefficients, respectively, that corresponds to

$$\begin{aligned}\mathbf{c}_{\text{base}} &= k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h} \\ &= k_1 \cdot \mathbf{d1}_{\text{att}} + k_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h}. \end{aligned} \tag{16}$$

Upon the retrieval of $\mathbf{c}_{\text{base}}$, we proceed to the key recovery phase.

3.2 Key Recovery Phase

Overview: The key recovery phase works by constructing new attack ciphertexts based on $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$. When decrypted, they result in only two possible values $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$ with $\mathbf{e}[i] \neq 0$, where i is the index of the single collision. The value of $\mathbf{e}$ depends on the value of a targeted coefficient of $\mathbf{f}$. This binary information obtained using side-channels over several chosen ciphertexts leads to a complete recovery of $\mathbf{f}$ one coefficient at a time.

3.2.1 Attack Methodology

We build, using $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$, the attack ciphertext

$$\mathbf{c}_{\text{att}} = \ell_1 \cdot \mathbf{d1}_{\text{att}} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h} + \ell_3 \cdot x^u = \mathbf{c}_{\text{base}} + \ell_3 \cdot x^u, \tag{17}$$

where $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$, $u \in [0, n-1]$, and $\mathbf{c}_{\text{base}} = \ell_1 \cdot \mathbf{d1}_{\text{att}} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h}$. Let the error introduced due to rounding be $\mathbf{m}' \in R_3$. Thus, $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{att}}$ is given by

$$\begin{aligned}\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{att}} &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h} \cdot 3\mathbf{f} + \ell_3 \cdot 3\mathbf{f} \cdot x^u + 3\mathbf{f} \cdot \mathbf{m}' \\ &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{g} + 3\ell_3 \cdot \mathbf{f} \cdot x^u + 3\mathbf{f} \cdot \mathbf{m}' \\ &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{g} + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u) + \mathbf{n}, \end{aligned}$$

where $\mathbf{n} = 3\mathbf{f} \cdot \mathbf{m}'$ is the noise term. Please note that this noise term $\mathbf{n}$ is different from the noise term of $\mathbf{c}_{\text{base}}$. For the sake of explanation, we assume that $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$ collide at i with a value of $+2$. Thus, the coefficients of $\mathbf{a}$ can be expressed as

$$\mathbf{a}[j] = \begin{cases} 3\ell_1 \cdot 2m + \ell_2 \cdot 2n + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[j] + \mathbf{n}[j], & \text{if } j = i, \\ 3\ell_1 \cdot r + \ell_2 \cdot s + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[j] + \mathbf{n}[j], & \text{if } j \neq i \text{ and } (r, s) \neq (2m, 2n). \end{cases} \tag{18}$$

In particular, given a constant $\delta := 3\ell_1 \cdot 2m + \ell_2 \cdot 2n + \mathbf{n}[i]$, we can represent the coefficient of $\mathbf{a}$ at the colliding index i as

$$\mathbf{a}[i] = \delta + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]. \tag{19}$$

Thus, $\mathbf{a}[i]$ is linearly dependent on $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$.

Let β_u denote $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$. Based on the rotational property of polynomial multiplication mod $(x^n - x - 1)$ in Equation (10), we know that

$$\beta_u := \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i] = \begin{cases} \mathbf{f}[i-u], & \text{for } 0 \leq u < i, \\ \mathbf{f}[0] + \mathbf{f}[n-1], & \text{if } u = i, \\ \mathbf{f}[n-1+i-u] + \mathbf{f}[n+i-u], & \text{for } i < u < n. \end{cases} \tag{20}$$

By simply changing the rotation index u we can ensure the dependency of $\mathbf{a}[i]$, that is, the colliding index i, with different coefficients of the secret polynomial $\mathbf{f}$. For a given u, the five values in $\{-2, -1, 0, 1, 2\}$ are possible candidates for β_u. Our task is, therefore, to select values for (ℓ_1, ℓ_2, ℓ_3) such that the occurrence of $\mathbf{a}[i] > q/2$ and therefore $\mathbf{e}[i] \neq 0$,

acts as a binary distinguisher capable of identifying every candidate for β_u. To distinguish $\beta_u = +2$, for example, we choose integers ℓ_1, ℓ_2, ℓ_3 multiples of 3, that satisfy the condition

$$3\ell_1 \cdot r + \ell_2 \cdot s + 3\ell_3 \cdot \beta_u \begin{cases} > q/2, & \text{if } r = 2m, s = 2n, \text{ and } \beta_u = 2, \\ < q/2, & \text{otherwise,} \end{cases} \tag{21}$$

with $0 \leq r \leq 2m$ and $0 \leq s \leq 2n$. This ensures that $\mathbf{a}[i] > q/2$ and $\mathbf{e}[i] \neq 0$ at the colliding index i when $\beta_u = +2$, while $\mathbf{a}[i] < q/2$ and $\mathbf{e}[i] = 0$ otherwise. For $j \neq i$, the coefficients are $\mathbf{a}[j] < q/2$ and $\mathbf{e}[j] = 0$, since there is no other collision than at index i. Similarly, we can identify $\beta_u = -2$ by simply changing the sign of ℓ_3, that is, by using $(\ell_1, \ell_2, -\ell_3)$. If, however, $\mathbf{e} = 0$ for both ciphertexts, then $\beta_u \in \{-1, 0, 1\}$. Let $\mathbf{O}$ denote the $\mathbf{e} = 0$ event and $\mathbf{X}$ denote the $\mathbf{e} \neq 0$ event. This binary information serves as a distinguisher for every candidate for β_u. An attacker who can realize a PC oracle to extract the binary information is therefore capable of distinguishing all candidates for β_u.

Effect of Rounding Error: Some rounding error $\mathbf{n}$ is present on $\mathbf{a}$. Adopting a similar strategy to the one in Section 3.1.3, we select a tuple (ℓ_1, ℓ_2, ℓ_3) that minimizes the possibility of a false positive or a false negative in the collision. To distinguish $\beta_u = 2$, the tuple must satisfy Equation (21). At the colliding index when $\beta_u = 2$, the largest possible coefficient of $\mathbf{a}$ is $m_1 := 3\ell_1 \cdot 2m + \ell_2 \cdot 2n + 3\ell_3 \cdot 2 > q/2$. Let us call the second largest coefficient $m_2 < q/2$. A good choice of (ℓ_1, ℓ_2, ℓ_3) maximizes the distances

$$dm_1 = \|(3\ell_1 \cdot 2m + \ell_2 \cdot 2n + 3\ell_3 \cdot 2) - q/2\| \text{ and}$$
$$dm_2 = \left\| \max_{(r,s,t) \neq (2m,2n,2)} (3\ell_1 \cdot r + \ell_2 \cdot s + 3\ell_3 \cdot t) - q/2 \right\|,$$

with $0 \leq r \leq 2m$, $0 \leq s \leq 2n$, and $0 \leq t \leq 2$. In other words, we should give enough leeway to ensure that the possible error $\mathbf{n}[i]$ does not push $\mathbf{a}[i]$ to tho other side of $q/2$. The same must be done for all choices of (ℓ_1, ℓ_2, ℓ_3) that are used to distinguish every other candidate for β_u. Similar to (m, n) and (k_1, k_2) in Subsection 3.2, (ℓ_1, ℓ_2, ℓ_3) can be chosen ahead and fixed.

Table 4 is the decision table for the sntrup761 parameter set. Assuming a collision with a value of $+2$, the table shows unique distinguishability for every candidate for $\beta_u \in \{-2, -1, 0, 1, 2\}$, based on $\mathbf{O}$ or $\mathbf{X}$ for chosen ciphertexts constructed using concrete values for (ℓ_1, ℓ_2, ℓ_3). If the collision value is -2 instead, we swap the responses for $\beta_u = +1$ (resp. $+2$) with those for $\beta_u = -1$ (resp. -2). Every candidate for $\beta_u = \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$ can be *uniquely identified* based on the information about $\mathbf{O}$ or $\mathbf{X}$ from no more than *four chosen ciphertext* queries. Certain candidates, such as $+1$ and $+2$, only require 2 queries to be identified when going from left to right, 0 can be uniquely identified in 3 queries, while -1 and -2 require all 4 queries. We can thus deploy such a greedy approach to identify the value of β_u faster.

Going back to Table 2, we see the concrete values of (ℓ_1, ℓ_2, ℓ_3) and the corresponding distance tuple (dm_1, dm_2) that work on different parameter sets of NTRU Prime. The notation $(\ell_{x1}, \ell_{x2}, \ell_{x3})$ refers to the tuple that we use to distinguish $x \in \{1, 2\}$. We emphasize that there are several other values for (ℓ_1, ℓ_2, ℓ_3) which can also be chosen to construct attack ciphertexts for key recovery.

Since $\mathbf{e}$ is an internal variable, we use side-channel information to distinguish between the classes $\mathbf{O}$ and $\mathbf{X}$. In Subsection 3.1.4, we used the Welch's t-test to identify if $\mathbf{e} \neq 0$ to retrieve $\mathbf{c}_{\text{base}}$. The peaks in the t-test plot exceeding the pass/fail threshold of ± 5 in Figure 4(b) are precisely the features that identify $\mathbf{e} \neq 0$. In the following discussion, we demonstrate techniques to leverage the identified features in the t-test plot to build templates for the two classes $\mathbf{O}$ and $\mathbf{X}$. The templates will then be used to classify a given single trace into either of the two classes.

Table 4: Unique distinguishability of every candidate for $\beta_u \in [-2, 2]$ depending on $\mathbf{e} = 0$ ($\mathbf{O}$) or $\mathbf{e} \neq 0$ ($\mathbf{X}$) for sntrup761. We assume that the collision value is $+2$.

Secret Coeffs.	Either $\mathbf{e} = 0$ or $\mathbf{e} \neq 0$			
	(ℓ_1, ℓ_2, ℓ_3)			
	$(0, 279, 42)$	$(0, 237, 84)$	$(0, 279, -42)$	$(0, 237, -84)$
-2	**O**	**O**	**X**	**X**
-1	**O**	**O**	**X**	**O**
0	**O**	**O**	**O**	**O**
1	**X**	**O**	**O**	**O**
2	**X**	**X**	**O**	**O**

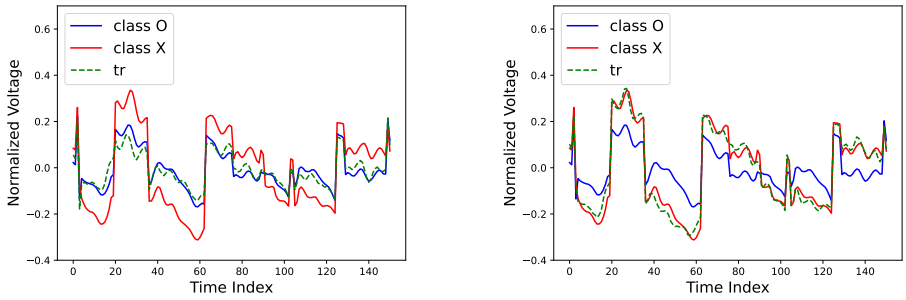
3.2.2 Classification using Reduced Templates

We select features of the t-test plot between $\mathcal{T}_\mathrm{O}$ and $\mathcal{T}_\mathrm{X}$ whose *absolute t-test value* is greater than a certain chosen threshold T_select as our set $\mathcal{P}$ of *points of interest* (PoI). A reduced trace set $\mathcal{T}'_\mathrm{O}$ or $\mathcal{T}'_\mathrm{X}$ is constructed by using points in $\mathcal{P}$. We choose a greater threshold than ± 5 for better distinguishability. For the t-test results in Figure 4, we set ± 7 as the larger threshold. This threshold is a parameter of the attack setup. We subsequently calculate the respective means $m_{\mathrm{O},\mathcal{P}}$ and $m_{\mathrm{X},\mathcal{P}}$ of $\mathcal{T}'_\mathrm{O}$ and $\mathcal{T}'_\mathrm{X}$ to use as the *reduced templates* for each class. We do not utilize a covariance matrix for template construction as we only exploit univariate leakage.

A single trace t for classification is normalized to $t' = t - \bar{t}$ to obtain a reduced trace $t'_\mathcal{P}$. The sum-of-squared difference Γ_* of the trace is computed with each reduced template as

$$\Gamma_\mathrm{O} = (t'_\mathcal{P} - m_{\mathrm{O},\mathcal{P}})^\top \cdot (t'_\mathcal{P} - m_{\mathrm{O},\mathcal{P}}) \text{ and } \Gamma_\mathrm{X} = (t'_\mathcal{P} - m_{\mathrm{X},\mathcal{P}})^\top \cdot (t'_\mathcal{P} - m_{\mathrm{X},\mathcal{P}}). \tag{22}$$

The trace t falls into the class that corresponds to the least sum-of-squared difference. Given a single power/EM trace of the targeted operation, this is sufficient to distinguish between $\mathbf{X}$ or $\mathbf{O}$. Thus, single side-channel traces from the decryption of chosen ciphertexts constructed according to Equation (17) can recover $\beta_u = \mathsf{Rotp}_\mathsf{R}(\mathbf{f}, u)[i]$. Figure 5 visualizes the matching of a section of the reduced trace tr with the reduced templates of the respective classes $\mathbf{O}$ and $\mathbf{X}$. There is a clear distinguishability between the reduced templates of the two classes, leading to a classification with 100% success rate.



(a) $\mathsf{Class}(tr) = \mathbf{O}$ (b) $\mathsf{Class}(tr) = \mathbf{X}$

Figure 5: Matching the reduced template tr of a given attack trace with the respective reduced templates of the two classes $\mathbf{O}$ and $\mathbf{X}$.

3.2.3 Recovering the Full Secret Key

We have, thus far, demonstrated the recovery of a single coefficient $\beta_u = \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$. Simply by changing the rotation index u, we can recover $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$ for all $u \in [0, n-1]$. However, recovering the exact value of the secret polynomial $\mathbf{f}$ requires knowing *both* the colliding index i and the collision value (either $+2$ or -2). Neither of which can be inferred through side-channels by using our technique. Thus, we need to try out all n possible colliding indices $i \in [0, n-1]$ as well as the two possible collision values ± 2. This amounts to $2n$ choices for $\mathbf{f}$. For sntrup761, $2n = 1,522$. For each choice, we compute the secret key $\mathbf{f}'$ and check if $\mathbf{f}' \in R_{\text{sh}}$ and also attempt to decrypt known ciphertexts. We empirically verified, for all parameter sets, that the search space is reduced drastically to only a handful, ≈ 10, of possibilities, up to a certain rotation of $\mathbf{f}$.

On Repeated Attack Iterations for Key Recovery: It is possible that none of the guessed $\mathbf{f}'$ recovered from the key recovery phase is correct. This could be due to two reasons. First, the base ciphertext from the preprocessing phase itself could have come from a multiple collisions or a false positive collision, leading to erroneous oracle responses. Second, even if the base ciphertext corresponds to a true single collision, the rounding noise $\mathbf{n}$ within the attack ciphertexts $\mathbf{c}_{\text{att}}$ could be too large, yielding erroneous oracle's responses. In both cases, we simply reject the current $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$ and initiate a search for a new base ciphertext to repeat the attack until the correct $\mathbf{f}$ is recovered. We reiterate that, if the secret key obtained after the key recovery phase is incorrect, it does not mean that our key recovery has failed. We may need to perform several attack iterations until the correct secret key is recovered. We recall that the correct key can be identified by validating the distribution of coefficients of the recovered secret polynomial or by attempting to decrypt known valid ciphertexts. Figure 6 summarizes the attack flow of our PC oracle-based SCA on NTRU Prime.

On Reducing the Impact of Failed Attack Iterations: Failed attack iterations significantly impact the trace complexity. We can adopt a few optimization approaches to lessen the impact. If the side-channel oracle's responses do not match the expected responses in the decision table, we immediately abort the key recovery phase and restart the preprocessing phase for a fresh base ciphertext. We similarly abort and start afresh if the recovered values of β_u for $u \in [0, n-1]$ appear very skewed and fail to follow the expected distribution.

3.2.4 Experimental Results

We implemented our PC oracle-based SCA on sntrup761. The preprocessing phase to identify $\mathbf{c}_{\text{base}}$ took, on average, 39 attempts. The number of attempts, denoted by A, includes failed attack iterations. Each attempt requires the capture of $N = 10$ traces to carry out the Welch's t-test for leakage detection. Thus, it takes $t_{\text{base}} := A \cdot N \approx 390$ traces to identify $\mathbf{c}_{\text{base}}$. The attack phase requires up to 4 chosen-ciphertext queries to recover one coefficient. The secret $\mathbf{f}$ has $n = 761$ coefficients. The number of traces needed in the attack phase is denoted by t_{attack}. Thus, we require $t_{\text{total}} = t_{\text{base}} + t_{\text{attack}} \approx 3269$ traces for a complete recovery of $\mathbf{f}$. Our attack works with a success rate of about 100% in recovering the secret key. No additional brute force or offline analysis is necessary.

The same attack works on all other parameter sets of NTRU Prime. Table 5 gives the estimated trace complexity. The numbers are estimated with $N = 10$ in the preprocessing phase. It suffices to use 4700 traces for full key recovery across all parameter sets of NTRU Prime, confirming the efficacy of our attack.

Table 5: The trace complexity of our PC oracle-based SCA on different variants of NTRU Prime. We use t_{base} to denote the number of traces used in the preprocessing phase, for $N = 10$ repeated measurements, and t_{total} to denote the number of traces required for full key recovery.

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
sntrup653	420	3005	sntrup953	270	3601
sntrup761	390	3269	sntrup1013	320	4026
sntrup857	420	3731	sntrup1277	240	4688

4 PC Oracle-based SCA on NTRU

We now adapt our PC oracle-based SCA, that we have implemented successfully on NTRU Prime KEM, to NTRU KEM. We keep to the notation in Algorithm 3. Since our attack applies in the same manner to both NTRU-HPS and NTRU-HRSS, we use NTRU-HPS in the attack description, with details on aspects that differ from those in NTRU-HRSS supplied whenever necessary.

4.1 Preprocessing Phase

Let $k_1, k_2 \in \mathbb{Z}^+$. We construct the chosen ciphertext $\mathbf{c}$ as

$$\mathbf{c} = k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h} = k_1 \cdot \mathbf{d}_1 + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}, \quad (23)$$

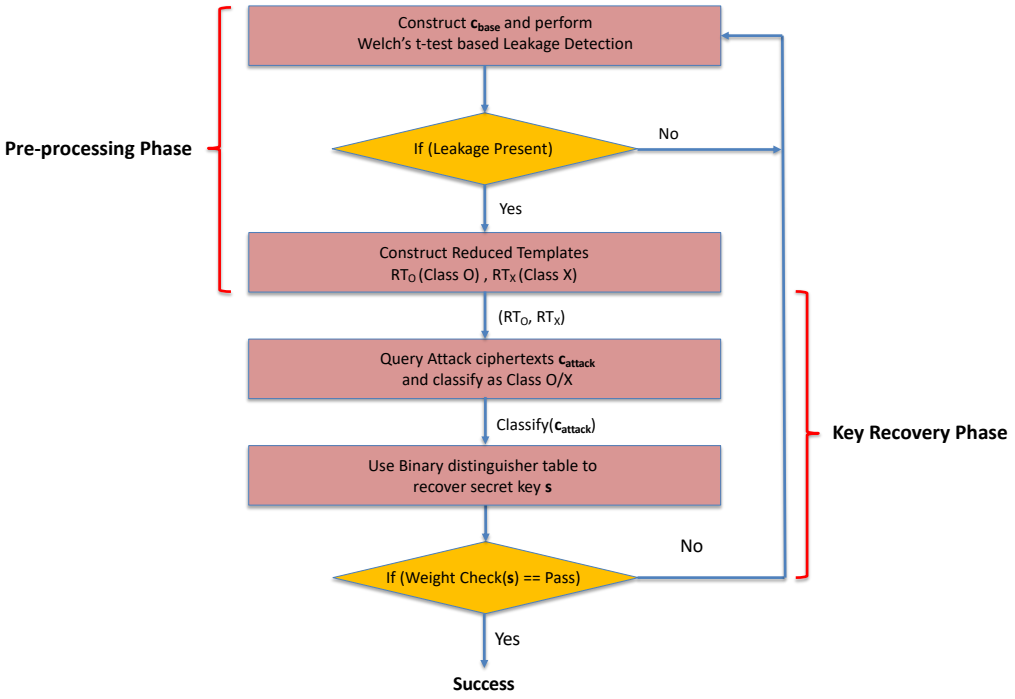


Figure 6: The attack flow diagram of our PC oracle-based SCA on NTRU Prime

where $3 \mid k_i$ for $i \in \{1, 2\}$ and $\mathbf{d}_1$ and $\mathbf{d}_2$ are polynomials with, respectively, m and n nonzero coefficients taking the value of $+1$. The corresponding $\mathbf{a} = \mathbf{f} \cdot \mathbf{c} \in T_q$ in line 7 of NTRU_PKE.Decrypt is

$$\mathbf{a} = k_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} \cdot \mathbf{f} = k_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot 3\mathbf{g} = k_1 \cdot \mathbf{t}_1 + 3k_2 \cdot \mathbf{t}_2, \tag{24}$$

where $\mathbf{t}_1 = \mathbf{d}_1 \cdot \mathbf{f}$ and $\mathbf{t}_2 = \mathbf{d}_2 \cdot \mathbf{g} \in T_q$. The polynomial $\mathbf{t}_1$ (resp. $\mathbf{t}_2$) in the cyclotomic ring $T = \mathbb{Z}[x]/\langle x^n - 1\rangle$ is the sum of exact rotations of the secret polynomial $\mathbf{f}$ by varying degrees, that is, for $u \in \{i_1, i_2, \ldots, i_m\}$ (resp. $v \in \{j_1, j_2, \ldots, j_n\}$). Thus, a collision at index i occurs when all the corresponding coefficients of the rotations of $\mathbf{f}$ and $\mathbf{g}$ have a value of $+1$ or -1.

We choose (m, n) to maximize the probability of a single collision and, then, proceed to choose (k_1, k_2) such that a collision at index i results in $\mathbf{a}[i] > q/2$ while keeping $\mathbf{a}[i] < q/2$ when there is no collision. From Equation (24), we observe that the absolute maximum value of a coefficient of $\mathbf{a}$ upon collision is $\mathbf{a}[i] = k_1 \cdot m + 3k_2 \cdot n$. Thus, we choose (k_1, k_2) such that

$$3 \mid k_1, \quad 3 \mid k_2, \quad k_1 \cdot r + 3k_2 \cdot s \begin{cases} > q/2, & \text{if } r = m \text{ and } s = n, \\ < q/2, & \text{otherwise,} \end{cases} \tag{25}$$

with $0 \leq r \leq m$ and $0 \leq s \leq n$. If there is a collision at i, then the corresponding coefficient of $\mathbf{e} = \mathbf{a} \bmod S_3$ in line 8 of NTRU_PKE.Decrypt is $\mathbf{e}[i] \neq 0$. Otherwise, we have $\mathbf{e}[i] = 0$.

NTRU-HRSS uses $\mathbf{g} = \mathbf{g}' \cdot \phi_1$ with the coefficients of $\mathbf{g}'$ coming from $\{-1, 0, 1\}$. Hence, the coefficients of the secret polynomial $\mathbf{g}$ are elements in $\{-2, -1, 0, 1, 2\}$. The absolute maximum value possible for a coefficient of $\mathbf{a}$ is $\mathbf{a}[i] = k_1 \cdot m + 3k_2 \cdot 2n$ and, hence, Equation (25) is adjusted accordingly.

4.1.1 Additional Challenge: Ciphertext Compression

Similar to the use of rounded ciphertexts in NTRU Prime to reduce ciphertext size, NTRU uses compression to exploit the inherent property of valid ciphertexts. The decryption procedure of NTRU expects valid ciphertexts to be a multiple of ϕ_1 modulo q. In other words, the sum of coefficients of a valid ciphertext is expected to be 0 modulo q as is evident in the conditional check in line 3 of NTRU_PKE.Decrypt. Thus, the scheme proposes to only send the first $n - 1$ coefficients of $\mathbf{c}$, while the last coefficient $\mathbf{c}[n-1]$ is computed within the decryption procedure as

$$\mathbf{c}[n-1] = -\sum_{i=0}^{i=n-2} \mathbf{c}[i] \tag{26}$$

Although the $\mathbf{c}$ in Equation (23) is not a multiple of ϕ_1 modulo q, it can be modified to satisfy the requirement as

$$\mathbf{c} = k_1 \cdot (x^{i_1} - x^{i_2} + x^{i_3} - \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + x^{j_3} + \ldots + x^{j_n}) \cdot \mathbf{h} = k_1 \cdot \mathbf{d}_1 + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}, \tag{27}$$

where m is even and the polynomial $\mathbf{d}_1$ having an equal number of positive and negative nonzero coefficients. It has $m/2$ coefficients $+1$ and $m/2$ coefficients -1. This ensures that its coefficients sum to 0. The same is not required for $\mathbf{d}_2$ since $\mathbf{h}$ is already a multiple of ϕ_1. The $\mathbf{c}$ in Equation (27) is processed without any errors in the decryption procedure. Unlike the chosen ciphertexts for NTRU Prime which inherently contain rounding error, chosen ciphertexts for NTRU do not contain any error. This significantly simplifies our attack on both NTRU-HPS and NTRU-HRSS.

Table 6 lists the concrete values of (m, n) and (k_1, k_2) used in constructing chosen ciphertexts for single collision for the indicated parameter sets of NTRU. Table 7 lists

Table 6: Concrete values for the various parameters used to build chosen ciphertexts for both attack phases of our PC oracle-based SCA for different variants of NTRU.

Scheme	(m, n)	**Preprocessing Phase**	**Key Recovery Phase**	
		(k_1, k_2)	$(\ell_{11}, \ell_{12}, \ell_{13})$	$(\ell_{21}, \ell_{22}, \ell_{23})$
ntruhps2048509	(4, 3)	(147, 51)	(144, 45, 66)	(114, 39, 120)
ntruhps2048677	(4, 3)	(147, 51)	(144, 45, 66)	(114, 39, 120)
ntruhps4096821	(4, 3)	(288, 102)	(273, 93, 141)	(228, 78, 228)
ntruhrss701	(4, 2)	(492, 180)	(483, 162, 243)	(411, 138, 411)

Table 7: The probability of obtaining each type of collision for the chosen ciphertexts across all parameters of NTRU. The numbers were empirically obtained from 1000 attack trials (1000 secret keys) for each parameter set.

Scheme	**Collision Probability**		Scheme	**Collision Probability**	
	True Single	Multiple		True Single	Multiple
ntruhps2048509	0.1533	0.0145	ntruhps4096821	0.3381	0.1148
ntruhps2048677	0.0944	0.0047	ntruhrss701	0.1475	0.0160

the probability of obtaining true single collisions and multiple collisions for the chosen parameters. A single true collision could be obtained between 7-10 trials, while the proabability of occurence of multiple collisions is a factor of $3 - 10$ times lower across all parameters of NTRU.

4.1.2 Detecting Collision through Side-Channels

Given (m, n) and (k_1, k_2), we construct several chosen ciphertexts $\mathbf{c}$ based on Equation (27) until we identify $\mathbf{c}_{\text{base}}$ whose $\mathbf{e} \neq 0$. This identification is done through side-channel leakage in a similar manner to our attack on NTRU Prime. If $\mathbf{e} = 0$, then $\mathbf{m}' = \mathbf{e} \cdot \mathbf{f}_p = 0$ as in line 9 of NTRU_PKE.Decrypt. If $\mathbf{e} = \pm x^i$, however, $\mathbf{m}'$ contains uniformly random coefficients in $\{-1, 0, 1\}$. This large difference in the value of $\mathbf{m}'$ can be easily identified through side-channels to distinguish between the two classes $\mathbf{e} = 0$ (the class $\mathbf{O}$) and $\mathbf{e} \neq 0$ with $\mathbf{e}[i] \neq 0$ (the class $\mathbf{X}$).

We performed experiments on ntruhps2048677. Side-channel measurements were acquired from the same target platform and experimental setup described in Subsection 3.1.4. The Welch's t-test was used to identify leakage corresponding to $\mathbf{e} \neq 0$. Figure 7(a) depicts the t-test plots for several ciphertexts $\mathbf{c}'$ whose $\mathbf{e} = 0$. We see no significant peaks about the threshold, which indicates $\mathbf{e} = 0$. Figure 7(b) corresponds to $\mathbf{e} \neq 0$ for $\mathbf{c}'$, where we can clearly identify several peaks, well beyond the threshold, confirming $\mathbf{e} \neq 0$.

The identified ciphertext is called $\mathbf{c}_{\text{base}}$ and its corresponding tuple $(\mathbf{d}_1, \mathbf{d}_2)$ according to Equation (27) is marked as $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$, to be used in creating the attack ciphertexts for key recovery. Since we can only differentiate between $\mathbf{e} = 0$ and $\mathbf{e} \neq 0$, it is possible that $\mathbf{e}$ contains multiple non-zero coefficients. In such a case, key recovery cannot be performed correctly and thus the search for $\mathbf{c}_{\text{base}}$ has to be repeated until the correct key is recovered.

4.2 Key Recovery Phase

We use $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$ to build the attack ciphertext

$$\mathbf{c}_{\text{att}} = \ell_1 \cdot \mathbf{d1}_{\text{att}} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h} + \ell_3 \cdot (x - 1) \cdot x^u = \mathbf{c}_{\text{base}} + \ell_3 \cdot (x - 1) \cdot x^u, \tag{28}$$

Table 8: Unique distinguishability of every candidate for $\beta_u \in [-2, 2]$ depending on $\mathbf{e} = 0$ (the event $\mathbf{O}$) or $\mathbf{e} \neq 0$ (the event $\mathbf{X}$) for ntruhps2048677

Secret Coeffs.	Either $\mathbf{e} = 0$ or $\mathbf{e} \neq 0$			
	(ℓ_1, ℓ_2, ℓ_3)			
	$(144, 66, 45)$	$(114, 120, 39)$	$(144, 66, -45)$	$(114, 120, -39)$
-2	$\mathbf{O}$	$\mathbf{O}$	$\mathbf{X}$	$\mathbf{X}$
-1	$\mathbf{O}$	$\mathbf{O}$	$\mathbf{X}$	$\mathbf{O}$
0	$\mathbf{O}$	$\mathbf{O}$	$\mathbf{O}$	$\mathbf{O}$
1	$\mathbf{X}$	$\mathbf{O}$	$\mathbf{O}$	$\mathbf{O}$
2	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{O}$	$\mathbf{O}$

where $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$ and $u \in [0, n-1]$. The term $\ell_3 \cdot (x-1) \cdot x^u$ ensures that $\mathbf{c}_{\text{att}}$ is a multiple of ϕ_1 modulo q. Let $\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, j)$ denote the product of $\mathbf{f}$ with x^j in the ring T. If we assume $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$ collide at i with a value of $+1$, then

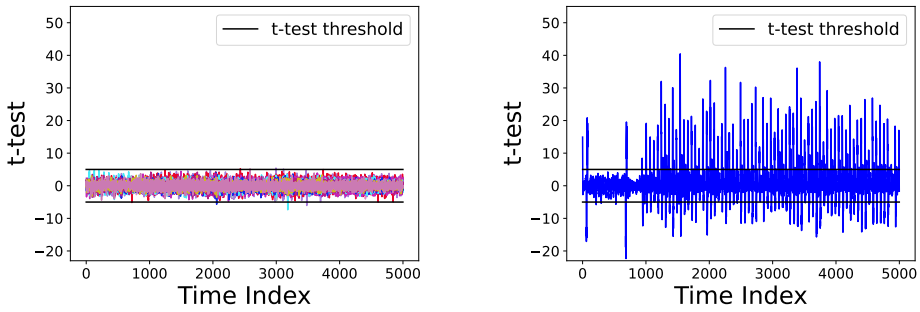
$$\mathbf{a}[j] = \begin{cases} \ell_1 \cdot m + 3\ell_2 \cdot n + \ell_3 \cdot (\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u+1)[j] - \mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u)[j]), & \text{if } j = i \\ \ell_1 \cdot r + 3\ell_2 \cdot s + \ell_3 \cdot (\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u+1)[j] - \mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u)[j]), & \text{if } \begin{cases} (j \neq i) \text{ and} \\ (r, s) \neq (m, n). \end{cases} \end{cases} \tag{29}$$

Letting $\delta := \ell_1 \cdot m + 3\ell_2 \cdot n$ gives us

$$\mathbf{a}[i] = \delta + \ell_3 \cdot (\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u+1)[i] - \mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u)[i]). \tag{30}$$

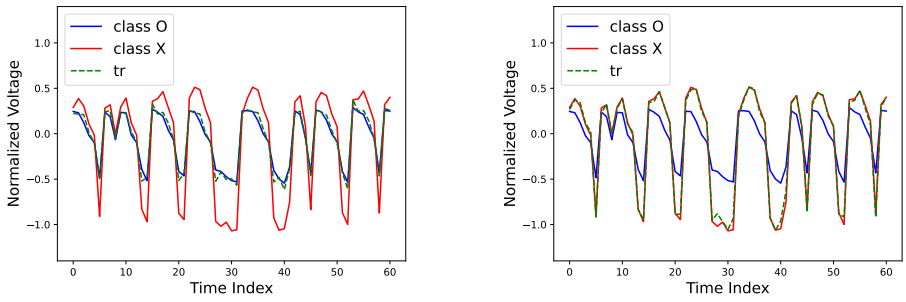
Thus, $\mathbf{a}[i]$ is linearly dependent on $\beta_u = (\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u)[i] - \mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u+1)[i])$, which, in turn, depends on two coefficients of $\mathbf{f}$. For a given $u \in [0, n-1]$, the possible candidates for β_u are $\{-2, -1, 0, 1, 2\}$. We choose (ℓ_1, ℓ_2, ℓ_3) such that $\mathbf{a}[i] > q/2$. Hence, $\mathbf{e}[i] \neq 0$ (Class $\mathbf{X}$) acts as a binary distinguisher for β_u. We choose (ℓ_1, ℓ_2, ℓ_3) for NTRU in the same manner as for NTRU Prime in Subsection 3.2.1 without the additional constraints placed to deal with the rounding error.

Table 8 is the decision table for ntruhps2048677. It demonstrates unique distinguishability for every candidate for $\beta_u \in \{-2, -1, 0, 1, 2\}$ based on $\mathbf{O}$ or $\mathbf{X}$. Every candidate for $\beta_u = (\mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u)[i] - \mathsf{Rotp}_{\mathsf{T}}(\mathbf{f}, u+1)[i])$ can be *uniquely identified* in *no more than four chosen ciphertext queries*. Table 6 gives the concrete (ℓ_1, ℓ_2, ℓ_3) values for different parameter sets of NTRU. We write $(\ell_{x1}, \ell_{x2}, \ell_{x3})$ to denote the tuple used to distinguish $x \in \{1, 2\}$.



(a) $\mathbf{e} = 0$ (b) $\mathbf{e} \neq 0$

Figure 7: The t-test plots for chosen ciphertexts for ntruhps2048677.



(a) Class$(tr) = \mathbf{O}$ (b) Class$(tr) = \mathbf{X}$

Figure 8: Matching the reduced template tr of a given attack trace with the reduced templates of the two classes **O** and **X**.

4.2.1 Classification using Reduced Templates

We have seen in Subsection 3.2.2 how side-channel leakage from the decryption of $\mathbf{c}_{\text{att}}$ classifies a given ciphertext as either **O** of **X** to realize a PC oracle. We use the distinguishing features of the t-test plot in Figure 7 to construct reduced templates for both classes **O** and **X**. We then use the templates for classification based on a simple least squared (LSQ) difference test. Figure 8 visualizes the matching of a section of the attack trace with the reduced templates of the respective classes **O** and **X**. Here, again, we observe clear distinguishability between the two classes. We experimentally obtained 100% success rate in classification, demonstrating that the realized PC oracle is highly accurate.

4.2.2 Recovering the Full Secret Key

The realized PC oracle can uniquely recover the value of β_u in up to *four* traces in obtaining information about two coefficients of $\mathbf{f}$. The same can be repeated for indices $u \in [0, n-1]$ to build a well-defined linear system. The system can be trivially solved to recover all n coefficients of the $\mathbf{f}$. We recover $\mathbf{f}' = \mathbf{f} \cdot x^i$, which is the secret up to a rotation of i indices. The attacker does not know the collision index i. The multiplication of $\mathbf{f}$ by x^i in the ring T_q, however, does not change the coefficients of $\mathbf{f}$. Since the decryption involves multiplication and division by $\mathbf{f}$, the rotated secret $\mathbf{f}'$ can already be used to decrypt any message encrypted with the secret polynomial $\mathbf{f}$.

As before, the secret key might not be recovered correctly if there are multiple colliding indices in $\mathbf{c}_{\text{base}}$. We simply repeat the attack until the complete key is recovered correctly.

4.2.3 Experimental Results

On ntruhps2048677, retrieving $\mathbf{c}_{\text{base}}$ required, on average, only 10 attempts. For $N = 10$ replicated measurements, the t_{base} is ≈ 100 traces. The attack phase requires no more than 4 chosen-ciphertext queries to recover one coefficient. There are $n = 677$ coefficients. Altogether, including failed attack iterations, the complete secret polynomial $\mathbf{f}$ can be recovered in $t_{\text{total}} \approx 2364$ traces. Our attack works with a success rate very close to 100%. Neither brute force nor offline analysis is necessary.

We successfully verified our attack using a simulated PC oracle on the remaining parameter sets of NTRU. Table 9 presents the estimated trace complexity. The numbers are estimated with $N = 10$ in the preprocessing phase. We find that 2900 traces is enough

Table 9: Trace Complexity of our proposed PC oracle-based SCA on the indicated variants. We use t_{base} to denote the number of traces needed to retrieve the base ciphertext and t_{total} to denote the number of traces required for full key recovery.

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
ntruhps2048509	70	1791	ntruhps4096821	30	2911
ntruhps2048677	100	2364	ntruhrss701	70	2447

for full key recovery across all parameter sets of NTRU. This demonstrates that our attack is effective.

The attack complexity of NTRU is less than that of NTRU Prime by a factor of ≈ 1.5 for comparable secret polynomials. This can be attributed mainly to the absence of rounding noise in NTRU. The attack analysis is simpler. It allows for more relaxed choices of attack parameters, reducing the number of attempts to identify the base ciphertext as well as the number of failed attack iterations.

4.2.4 Comparison with PC Oracle-based SCA on LWE/LWR-based schemes

There are a few subtle but critical differences between our PC oracle-based SCA on NTRU-based schemes and similar attacks on LWE/LWR-based schemes. The main difference lies in the anchor variable whose value is controlled carefully through the chosen ciphertexts for key recovery. On NTRU and NTRU Prime, the internal variable $\mathbf{e}$ within the decryption procedure serves as the anchor variable. The underlying arithmetic of LWE/LWR-based schemes allows for a direct control over the output of the decryption procedure. There, the decrypted message m of the chosen ciphertexts serves as the anchor variable for key recovery.

Another differing aspect lies on the ability to control the value of the anchor variable. While our attacks can restrict $\mathbf{e}$ to two classes $\mathbf{O}$ and $\mathbf{X}$, the value of $\mathbf{e}$ in class $\mathbf{X}$ cannot be controlled. Our preprocessing phase involves a search for a base ciphertext whose $\mathbf{e} = \pm 1 \cdot x^i$. The attacker can neither control nor know the colliding index i, since it depends on the secret key. In LWE/LWR-based schemes, the two classes $m = 0$ (the class $\mathbf{O}$) and $m = 1$ (the class $\mathbf{X}$) are fixed irrespective of the secret key. It is possible to build attack ciphertexts to exactly restrict m to either 0 or 1. Since the decrypted message m is the anchor variable, an attacker can also easily construct ciphertexts for $m = 0$ and $m = 1$ to build side-channel templates. Thus, the search for a base ciphertext is not necessary, which heavily simplifies the PC oracle-based SCA on LWE/LWR-based schemes.

Though the attack seems to be more involved in NTRU-based schemes, we do not observe a significant difference in the attacker's cost, that is, the trace complexity, to perform full key recovery. For comparison, we use the experimental results reported in the work of Ravi *et al.* in [RRCB20] on PC oracle-based SCA on LWE/LWR-based schemes, using the same target platform and attack setup. Their attack on Kyber512 required ≈ 7700 traces for full key recovery. The dimension is $n = 512$ with coefficients in $\{-2, -1, 0, 1, 2\}$. The count corresponds to three attack iterations to improve the success rate through majority voting. A single attack iteration, therefore, takes ≈ 2560 traces. Thus, the trace complexity of our proposed attack is comparable to the reported attack on LWE/LWR-based schemes.

4.3 Limitations of the PC Oracle-Based SCA

Our PC oracle-based SCA can perform full key recovery on all parameter sets of NTRU and NTRU Prime KEMs. We observe, however, that side-channel leakage from only a few operations within the decryption procedure can be used to obtain information about the

anchor variable $\mathbf{e}$ for key recovery. The attacker has a narrow scope to obtain side-channel leakage to instantiate a PC oracle for key recovery.

This is particularly true for NTRU Prime and we refer to NTRU_Prime_PKE.Decrypt in Algorithm 1. The attack ciphertexts result in $\mathbf{e} = 0$ or $\mathbf{e} = \pm 1 \cdot x^i$. If $\mathbf{e} = 0$, then $\mathbf{b}' = 0$ by line 5. If $\mathbf{e} = \pm 1 \cdot x^i$, then $\mathbf{b}'$ has uniformly random coefficients in $\{-1, 0, 1\}$ and its exact value depends on the secret polynomial $\mathbf{g}$. In both cases, the weight of $\mathbf{b}'$ is not equal to w, which is a requirement to be satisfied by the decrypted message. Thus, by line 10, the decryption procedure only returns a fixed value of $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$ for all attack ciphertexts.

The effect of the anchor variable $\mathbf{e}$ for the attack ciphertexts does not propagate beyond the decryption procedure. The PC oracle attack can only be carried out using side-channel information from operations that manipulate $\mathbf{e}$ and other dependent variables *within* the decryption procedure. This restricts an attacker from utilizing side-channel information from operations performed *after decryption*. These operations take place within the re-encryption procedure from line 5 of KEM.Decaps in Algorithm 2.

In the next section, we improve upon the PC oracle-based SCA by proposing a novel DF oracle-based SCA. The improved attack widens the scope of the attacker to obtain side-channel leakage from several other operations, which aids in key recovery, within the decapsulation procedure.

5 Decryption-Failure Oracle-Based SCA

We start by providing some intuition for the decryption-failure (DF) oracle attack. We demonstrate our attack on NTRU Prime only since the same approach extends trivially to NTRU. The main idea is to *carefully perturb valid ciphertexts, followed by observing the effect of perturbation on the decrypted message*. The perturbations are similar to those used in the PC oracle-based attack.

Let $\mathbf{c}_{\text{valid}}$ be a valid ciphertext whose anchor variable $\mathbf{e}$ is denoted by $\mathbf{e}_{\text{valid}}$. Let $\mathbf{c}'$ be an element in a set of specially crafted ciphertexts. These are similar to those used for the PC oracle-based attack. Upon decryption of $\mathbf{c}'$, the corresponding $\mathbf{e}'$ can only have two possible values, namely $\mathbf{e}' = 0$ and $\mathbf{e}' = \pm 1 \cdot x^i$. We add the perturbation ciphertext $\mathbf{c}'$ to the valid ciphertext $\mathbf{c}_{\text{valid}}$ to obtain a *perturbed ciphertext* $\mathbf{c}_{\text{pert}}$. Perturbing $\mathbf{c}_{\text{valid}}$ in this manner, in turn, perturbs $\mathbf{e}_{\text{valid}}$ so that the corresponding $\mathbf{e}_{\text{pert}}$ for $\mathbf{c}_{\text{pert}}$ admits two possible values, namely $\mathbf{e}_{\text{pert}} = \mathbf{e}_{\text{valid}}$ (the class $\mathbf{O}$) or $\mathbf{e}_{\text{valid}}$ with a single coefficient error at i, that is, $\mathbf{e}_{\text{pert}} = \mathbf{e}_{\text{valid}} \pm 1 \cdot x^i$, denoted as $\mathbf{e}_{\text{invalid}}$ (the class $\mathbf{X}$).

Decryption never fails for the class of valid ciphertexts. The decryption procedure returns $\mathbf{r}'_{\text{valid}}$. For the second class of ciphertexts, however, there is a single coefficient error in the anchor variable $\mathbf{e}$, with $\mathbf{e}_{\text{invalid}} = \mathbf{e}_{\text{valid}} \pm 1 \cdot x^i$. This triggers a decryption

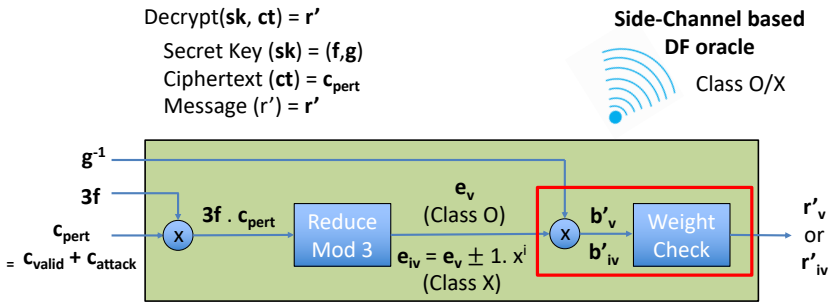


Figure 9: A pictorial illustration of our DF oracle-based SCA on NTRU Prime. The subscript `v` denotes `valid` whereas the subscript `iv` denotes `invalid`.

failure and, hence, $\mathbf{r}'_{\text{invalid}} = (1, 1, \ldots, 1, 0, 0, \ldots, 0)$ is returned as the decrypted message. Here, the perturbed ciphertext $\mathbf{c}_{\text{pert}}$ restricts the decrypted message $\mathbf{r}'$ to two possibilities, namely $\mathbf{r}'_{\text{valid}}$ and $\mathbf{r}'_{\text{invalid}}$. There, the decrypted message always takes the form of $\mathbf{r}'_{\text{invalid}}$. The success or failure of decryption for the perturbed ciphertexts depends upon a targeted portion of the secret key. Thus, an attacker who can obtain information about the decryption outcomes through a decryption-failure (DF) oracle can fully recover the secret key. At this point, we have ensured that the effect of the anchor variable $\mathbf{e}$ propagates to the decrypted message $\mathbf{r}'$. This was not the case with the PC oracle-based attack where the decrypted message always takes the form of $\mathbf{r}'_{\text{invalid}}$. Figure 9 illustrates the attack targeting leakage from the decryption procedure. We can also rely on leakage from the re-encryption procedure to instantiate the DF oracle.

A decryption failure can be identified through side-channel leakage from two sets of operations. The first one consists of operations that manipulate the anchor variable $\mathbf{e}$. The second one includes operations that manipulate the decrypted message $\mathbf{r}'$ in the re-encryption procedure. Thus, an attacker enjoys a wider scope to obtain side-channel information from several operations in the decapsulation procedure, including the re-encryption operation, toward a key recovery.

Remark 1. We observe that the DF oracle-based attack works with information about the decrypted message $\mathbf{r}'$. This can be used to perform key recovery over the IND-CPA secure NTRU Prime PKE, even without the requirement of side-channels. Thus, our proposed DF oracle-based attack on NTRU Prime is also *the first theoretical chosen-ciphertext attack against the IND-CPA secure NTRU Prime PKE.*

Similar to the PC oracle-based SCA, our DF oracle-based attack also works in two phases, namely the preprocessing phase and the key recovery phase.

5.1 Preprocessing Phase

As in line 3 of NTRU_PRIME_PKE.Encrypt in Algorithm 1, we construct a valid ciphertext $\mathbf{c}_{\text{valid}} = \mathsf{Round}(\mathbf{h} \cdot \mathbf{r})$. Its corresponding $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{valid}}$ is

$$\mathbf{a}_{\text{valid}} = \mathbf{g} \cdot \mathbf{r} + 3\mathbf{f} \cdot \mathbf{m}, \tag{31}$$

where $\mathbf{m}$ is the rounding error. We then construct perturbations using the methodology that was used to build the ciphertexts to obtain a single collision for NTRU Prime in Subsection 3.1.2. Such a perturbation $\mathbf{c}'$ is given by

$$\mathbf{c}' = k_1 \cdot (x^{i_1} + x^{i_2} + \ldots + x^{i_m}) + k_2 \cdot (x^{j_1} + x^{j_2} + \ldots + x^{j_n}) \cdot \mathbf{h} = k_1 \cdot \mathbf{d}_1 + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}, \tag{32}$$

with $3 \mid k_1$, $3 \mid k_2$, and $\mathbf{d}_1$ and $\mathbf{d}_2$ having, respectively, m and n nonzero coefficients $+1$. The corresponding $\mathbf{a}' = 3\mathbf{f} \cdot \mathbf{c}'$ is

$$\mathbf{a}' = k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} + 3\mathbf{f} \cdot \mathbf{m}''. \tag{33}$$

We use $\mathbf{c}'$ to perturb $\mathbf{c}_{\text{valid}}$ as

$$\mathbf{c}_{\text{pert}} = \mathsf{Round}(\mathbf{h}\cdot\mathbf{r}+\mathbf{c}') = \mathsf{Round}(\mathbf{h}\cdot\mathbf{r}+k_1\cdot\mathbf{d}_1+k_2\cdot\mathbf{d}_2\cdot\mathbf{h}) = \mathbf{h}\cdot\mathbf{r}+k_1\cdot\mathbf{d}_1+k_2\cdot\mathbf{d}_2\cdot\mathbf{h}+\mathbf{m}', \tag{34}$$

where $\mathbf{m}'$ is the rounding error. Upon decrypting $\mathbf{c}'$, we express $\mathbf{a}_{\text{pert}} = 3\mathbf{f} \cdot \mathbf{c}_{\text{pert}}$ as

$$\mathbf{a}_{\text{pert}} = \mathbf{g} \cdot \mathbf{r} + k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} + 3\mathbf{f} \cdot \mathbf{m}', \tag{35}$$

Thus, $\mathbf{a}_{\text{pert}} \approx \mathbf{a}_{\text{valid}} + \mathbf{a}'$. Let $\mathbf{s} := k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g}$ be the *signal component* of $\mathbf{a}_{\text{pert}}$. The noise $\mathbf{n}$ comprises of the rounding noise $3\mathbf{f} \cdot \mathbf{m}'$ acting together with $\mathbf{g} \cdot \mathbf{r}$, written as $\mathbf{gr}$, from $\mathbf{a}_{\text{valid}}$. For simplicity, we denote variables $\mathbf{a}_{\text{pert}}$ and $\mathbf{e}_{\text{pert}}$ corresponding to the perturbed ciphertext $\mathbf{c}_{\text{pert}}$ by $\mathbf{a}$ and $\mathbf{e}$, respectively.

To induce a decryption failure, that is, to perturb a single coefficient of $\mathbf{e} = \mathbf{a} \bmod 3$, we need a single coefficient of $\mathbf{a}$, say $\mathbf{a}[i]$, to be greater than $q/2$. This is achieved by choosing (m, n) for the polynomials $(\mathbf{d}_1, \mathbf{d}_2)$ in Equation (32) to maximize the probability of a single collision. If there is a collision at i, then $\mathbf{s}[i]$ should be large enough to push $\mathbf{a}[i]$ beyond the $q/2$ threshold. The coefficient $\mathbf{s}[i]$ at the colliding index is $m_1 := 3k_1 \cdot 2m + k_2 \cdot 2n$. Let m_2 denote the next largest possible value. We thus choose (k_1, k_2) such that $m_1 > q/2$ and $m_2 < q/2$.

The noise component $\mathbf{n} = \mathbf{g} \cdot \mathbf{r} + 3\mathbf{f} \cdot \mathbf{m}'$ in $\mathbf{a}$, however, contributes to crossovers near the $q/2$ threshold for coefficients of $\mathbf{a}$, resulting in false positives and false negatives in decryption failures. For sntrup761, the distribution of $\mathbf{n}'$ is Gaussian with mean 0 and a slightly larger standard deviation of $\sigma \approx 53$ than $\sigma = 50$ for $\mathbf{n}$ in the PC oracle-based attack. Though the increase is insignificant, we will soon see in Section 5.3 that the noise term $\mathbf{gr}$ in $\mathbf{n}$ is also present as a constant bias in the attack ciphertexts used for key recovery, along with the inherent rounding error. This additional bias poses challenges and, thus, requires a slightly different approach to construct chosen ciphertexts.

5.1.1 Additional Challenge: Dealing with Bias

To negate the effect of $\mathbf{gr}$, we slightly modify the constraints in choosing (k_1, k_2) so as to obtain a single collision at the index where the corresponding coefficient $\mathbf{gr}[i]$ of $\mathbf{gr}$ has a high absolute value. We choose (k_1, k_2) such that

$$3 \mid k_1, \quad 3 \mid k_2, \quad (q/2 - \epsilon_1) < m_1 < (q/2 - \epsilon_2), \quad m_2 < q/2, \tag{36}$$

with $\epsilon_1, \epsilon_2 > 0$. This way, even if there is a collision at i, we keep $\mathbf{s}[i] = m_1 < q/2$ in the range $[(q/2 - \epsilon_1), (q/2 - \epsilon_2)]$. Such a constraint for (k_1, k_2) gives us several advantages.

The main advantage is that $\mathbf{a}[i] > q/2$ only when two conditions, namely a collision at i and $\mathbf{n}[i] > \epsilon_2$, hold simultaneously. In allowing the noise coefficient to have a large value, we increase the chances of $\mathbf{gr}[i]$ to also have a large value at the colliding index. Instead of simply identifying a collision at any index, we increase the chances of achieving collision at an index where $\mathbf{gr}[i]$ has a large value. Thus, even if $\mathbf{a}'[i] = m_1$ and is close to $q/2$, it gets pushed further away from $q/2$ by $\mathbf{gr}[i]$. This has a positive influence on the key recovery as it decreases the chance of a false negative for decryption failure in the key recovery phase. With m_1 chosen to be $< q/2$ by $\epsilon_2 < dm_1 < \epsilon_1$, there is a leeway to increase dm_2. This reduces the chances of false positives at the other indices $j \neq i$ where no collisions occur. In short, the modified constraints for choosing (k_1, k_2) reduce the chance of false positives and false negatives.

The tuples (m, n) and (k_1, k_2) are chosen to identify ciphertexts with a single collision. Table 10 presents concrete values used in our attack on the specified parameter sets of NTRU Prime. Table 11 lists the probability of obtaining ciphertexts corresponding to different types of collisions, for the chosen parameters. The numbers were empirically

Table 10: Concrete values used to construct chosen ciphertexts for both phases of our DF oracle-based SCA on NTRU Prime.

Scheme	(m, n)	**Preprocessing Phase**		**Key Recovery Phase**			
		(k_1, k_2)	(dm_1, dm_2)	$(\ell_{11}, \ell_{12}, \ell_{13})$	(dm_1, dm_2)	$(\ell_{21}, \ell_{22}, \ell_{23})$	(dm_1, dm_2)
sntrup653	$(0, 4)$	$(0, 285)$	$(30, 315)$	$(0, 279, 48)$	$(66, 69)$	$(0, 243, 81)$	$(120, 123)$
sntrup761	$(0, 4)$	$(0, 282)$	$(39, 321)$	$(0, 279, 42)$	$(63, 63)$	$(0, 237, 84)$	$(105, 132)$
sntrup857	$(0, 4)$	$(0, 318)$	$(39, 357)$	$(0, 312, 54)$	$(75, 75)$	$(0, 270, 93)$	$(135, 135)$
sntrup953	$(0, 4)$	$(0, 393)$	$(27, 420)$	$(0, 384, 60)$	$(81, 99)$	$(0, 327, 120)$	$(165, 162)$
sntrup1013	$(0, 4)$	$(0, 444)$	$(36, 480)$	$(0, 435, 72)$	$(108, 108)$	$(0, 375, 129)$	$(186, 189)$
sntrup1277	$(0, 4)$	$(0, 489)$	$(27, 516)$	$(0, 477, 78)$	$(111, 123)$	$(0, 414, 138)$	$(201, 213)$

Table 11: The probability of obtaining each type of collisions for the chosen ciphertexts on NTRU Prime. The numbers were obtained from 1000 attack trials (1000 secret keys) for each parameter set.

Scheme	Collision Probability			
	True Single	Multiple	False Positive	False Negative
sntrup653	0.0099	0.0008	≈ 0	0.0228
sntrup761	0.0095	0.0009	≈ 0	0.0287
sntrup857	0.0111	0.0010	≈ 0	0.0315
sntrup953	0.0177	0.0018	≈ 0	0.0305
sntrup1013	0.0151	0.0017	≈ 0	0.0342
sntrup1277	0.0213	0.0036	≈ 0	0.0393

obtained through simulations from 1000 attack trials (1000 secret keys) for each parameter set. We observe that a true single collision can be obtained anywhere between 47 to 111 trials across all parameters of NTRU Prime, while the probability of multiple collision is generally an order of magnitude lower across all parameters. Since the parameters were chosen to particularly minimize false positive collisions to deal with the constant bias, we do not observe any false positive collisions in our experiments. However, this leads to a significant increase in false negative in collisions (suppression of true single collisions), which are more likely to occur than true single collisions across all parameters of NTRU Prime. This leads to a slight increase in the number of traces required in the preprocessing phase, as we show in the experimental results in Table 12.

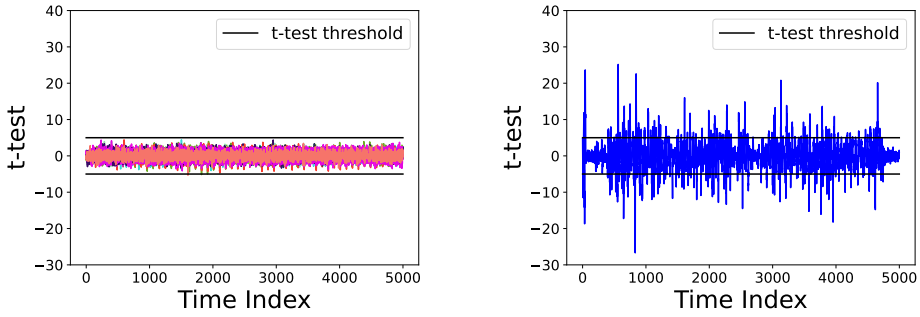
Given (m, n) and (k_1, k_2), we randomly select $\mathbf{d}_1$ and $\mathbf{d}_2$ to construct perturbations $\mathbf{c}'$ based on Equation (32). The aim is to identify a perturbation which, when added to a valid ciphertext $\mathbf{c}_{\text{valid}}$, induces a single coefficient error in the corresponding variable $\mathbf{e}$ by producing $\mathbf{e}_{\text{invalid}} = \mathbf{e}_{\text{valid}} \pm x^i$. This results in a decryption failure by yielding $\mathbf{r}'_{\text{invalid}}$.

5.2 Detecting Decryption Failure through Side-Channels

Decryption failures can be identified by obtaining information, through side-channels, about either the anchor variable $\mathbf{e}'$ within the decryption procedure or the decrypted message $\mathbf{r}'$ used in the re-encryption procedure. We can, therefore, utilize side-channel leakage from two sources. The first source consists of operations that manipulate $\mathbf{e}'$ within the decryption procedure in lines 5 to 6 of NTRU_PRIME_PKE.Decrypt in Algorithm 1. Operations within the re-encryption procedure in line 5 of NTRU_Prime_KEM.Decaps in Algorithm 2 form the second source.

When experimenting on sntrup761, we keep the same target platform and experimental setup used to perform the PC oracle-based attack to take measurements. In particular, we obtained side-channel leakage from the encoding of the decrypted message $\mathbf{r}'$ just after the decryption procedure. Other operations within the re-encryption can also be deployed to infer information on $\mathbf{r}'$.

We used the Welch's t-test described in Subsection 3.1.4 to identify leakage that differentiates $\mathbf{r}'_{\text{invalid}}$ (decryption failure) from $\mathbf{r}'_{\text{valid}}$ (decryption success). Figure 10(a) depicts the t-test plot for several perturbed ciphertexts $\mathbf{c}_{\text{pert}}$ whose decryption does not trigger any error. One sees no significant peaks beyond the threshold, which indicates $\mathbf{r}' = \mathbf{r}'_{\text{valid}}$. Figure 10(b), however, exhibits the t-test plot when the decryption fails for the perturbed ciphertext. One can clearly identify several peaks, well above the threshold, indicating $\mathbf{r}' = \mathbf{r}'_{\text{invalid}}$. The ciphertext which successfully induces a decryption failure is called $\mathbf{c}_{\text{base}}$. Its corresponding polynomials $\mathbf{d}_1$ and $\mathbf{d}_2$ are labelled $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$, with m and n terms, respectively.



(a) $\mathbf{r}' = \mathbf{r}'_{\text{valid}}$ (b) $\mathbf{r}' = \mathbf{r}'_{\text{invalid}}$

Figure 10: The t-test plots used to identify decryption failure for sntrup761.

5.3 Key Recovery Phase

We now use the polynomials $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$ of $\mathbf{c}_{\text{base}}$ to build new perturbed attack ciphertexts. Side-channel leakage from their decapsulation is used to identify decryption failures, which subsequently leads to full recovery of the secret polynomial $\mathbf{f}$.

5.3.1 Attack Methodology

Our approach here very closely resembles the one used in the PC oracle-based attack on NTRU Prime in Section 3.2. We first build the perturbation ciphertext $\mathbf{c}'$, using $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$ of $\mathbf{c}_{\text{base}}$, as

$$\mathbf{c}' = \ell_1 \cdot \mathbf{d1}_{\text{att}} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h} + \ell_3 \cdot x^u, \tag{37}$$

with $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$ and $u \in [0, n-1]$. We add $\mathbf{c}'$ to the term $\mathbf{h} \cdot \mathbf{r}$ of $\mathbf{c}_{\text{valid}}$ and generate the invalid perturbed ciphertext $\mathbf{c}_{\text{pert}}$ as in Equation (34). The corresponding $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{pert}}$ is given by

$$\begin{aligned} \mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{pert}} &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{h} \cdot 3\mathbf{f} + \ell_3 \cdot 3\mathbf{f} \cdot x^u + \mathbf{gr} + 3\mathbf{f} \cdot \mathbf{m}' \\ &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{g} + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u) + \mathbf{gr} + 3\mathbf{f} \cdot \mathbf{m}' \\ &= 3\ell_1 \cdot \mathbf{d1}_{\text{att}} \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d2}_{\text{att}} \cdot \mathbf{g} + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u) + \mathbf{n}, \end{aligned} \tag{38}$$

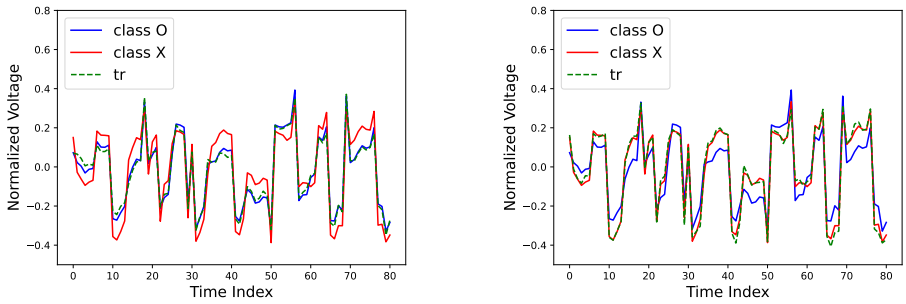
where $\mathbf{n} = 3\mathbf{f} \cdot \mathbf{m}' + \mathbf{gr}$. If $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$ collide at i, then we can write

$$\mathbf{a}[j] = \begin{cases} 3\ell_1 \cdot 2m + \ell_2 \cdot 2n + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[j] + \mathbf{n}[j], & \text{if } j = i \\ 3\ell_1 \cdot r + \ell_2 \cdot s + 3\ell_3 \cdot \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[j] + \mathbf{n}[j], & \text{if } j \neq i \text{ and } (r, s) \neq (2m, 2n). \end{cases} \tag{39}$$

As in the PC oracle-based attack, we choose (ℓ_1, ℓ_2, ℓ_3) to ensure that the following conditions are met. First, $\mathbf{a}[j] < (q/2)$, for $j \neq i$, and, thus, $\mathbf{e}[j] = \mathbf{e}_{\text{valid}}[j]$. Second, the coefficients of $\mathbf{a}$ near the threshold $q/2$ are as far as possible from the threshold to avoid accidental crossovers due to $\mathbf{n}$. Third, the occurrence of $\mathbf{a}[i] > q/2$ and, therefore, $\mathbf{e}[i] \neq \mathbf{e}_{\text{valid}}[i]$, depends on a single coefficient $\beta_u \in \{-2, -1, 0, 1, 2\}$ of the rotated secret polynomial $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$. Thus, $\mathbf{e} = \mathbf{e}_{\text{valid}}$ (the class $\mathbf{O}$) or $\mathbf{e} = \mathbf{e}_{\text{invalid}}$ (the class $\mathbf{X}$) can act as a binary distinguisher for every candidate of β_u. These constraints used to select (ℓ_1, ℓ_2, ℓ_3) are the same as that used for the PC oracle-based SCA for NTRU Prime. We arrive at the values used for the PC oracle-based SCA as listed in Table 10. Our decision table for unique distinguishability here is, therefore, Table 4 in Subsection 3.2.1.

5.3.2 Classification using Reduced Templates

We utilize the differentiating features in the t-test plot in Figure 10(b) to build the reduced templates for classes $\mathbf{O}$ and $\mathbf{X}$. Subsequently, they can be used to classify any given trace corresponding to the decapsulation of an attack ciphertext into either class. This was treated earlier in Subsection 3.2.2. Figure 11 visualizes the matching of a small section of an attack trace tr with the reduced templates of the respective classes $\mathbf{O}$ and $\mathbf{X}$, showing a clear distinguishability. This enables us to correctly classify a given single trace with a 100% success rate.



(a) $\mathsf{Class}(tr) = \mathbf{O}$ (b) $\mathsf{Class}(tr) = \mathbf{X}$

Figure 11: Matching the reduced template tr of a given attack trace with the reduced templates of classes $\mathbf{O}$ and $\mathbf{X}$.

5.3.3 Recovering the Full Secret Key

So far, we have demonstrated the recovery of a single coefficient β_u of the rotated secret polynomial $\mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)$. By changing the rotation index u, we can recover $\mathsf{Rotp}(\mathbf{f}, u)[i]$ for all $u \in [0, n-1]$. In line with the PC oracle attack, recovering the exact secret polynomial $\mathbf{f}$ requires knowing the colliding index i and the value, either $+2$ or -2, of the collision. By trying out all possible choices for $i \in [0, n-1]$ and the colliding values $+2$ and -2, we check, for each choice, if $\mathbf{f}' \in R_{\mathrm{sh}}$ and attempt to decrypt known ciphertexts. We empirically verified that the search space is drastically reduced to ≈ 10, up to a certain rotation of $\mathbf{f}$. It is also possible that the secret is not recovered correctly, due to a bad choice of $\mathbf{c}_{\mathrm{base}}$ or large rounding noise in the attack ciphertexts. When this happens, we simply rerun the attack until the correct key is recovered, similar to the PC oracle-based attack.

5.3.4 Experimental Results

We ran our attack on sntrup761. The preprocessing phase to retrieve $\mathbf{c}_{\mathrm{base}}$ requires an average of ≈ 165 attempts. For $N = 10$, the number of required t_{base} goes to ≈ 1650. The number is almost 4.2 times higher than the one in the PC oracle-based attack, which only requires ≈ 39 attempts. The increase is partly due to the additional constraints to deal with the constant bias $\mathbf{gr}$. The subsequent attack phase can recover a single coefficient in up to 4 ciphertexts and, thus, the complete attack requires $t_{\mathrm{total}} = t_{\mathrm{base}} + t_{\mathrm{attack}} \approx 4566$ traces to completely recover $\mathbf{f}$. Our attack works with a success rate of very close to 100% with no additional brute force or offline analysis to perform.

We successfully verified our attack methodology using a simulated DF oracle on all parameter sets of NTRU Prime. Table 12 gives the attack's estimated trace complexity.

Table 12: The trace complexity of our DF oracle-based SCA on NTRU Prime. We use t_{base} to denote the number of traces required in the preprocessing phase (for $N = 10$ repeated measurements) and t_{total} to denote the number of traces required for full key recovery.

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
sntrup653	1630	4182	sntrup953	760	4436
sntrup761	1650	4566	sntrup1013	740	4603
sntrup857	1200	4631	sntrup1277	410	5287

The numbers are estimated with $N = 10$ in the preprocessing phase. Roughly, between 4100 to 5300 traces are enough for full key recovery across all listed parameter sets with a 100% success rate. The numbers are 1.2 to 1.4 times the numbers for the PC oracle-based attack. This increase comes mainly from the longer preprocessing phase for the DF oracle-based attack.

5.3.5 Comparison with DF Oracle-based SCA on LWE/LWR-based schemes

Known DF oracle-based SCA on LWE/LWR-based schemes [GJN20, BDH+21] modified the coefficients of the ciphertext to perturb the corresponding bits in the decrypted message m that served as the anchor variable. Whether or not the perturbations resulted in a decryption failure is linearly dependent on the secret key. This information, if obtainable by a DF oracle, led to full key recovery. For LWE/LWR-based schemes, the location of the perturbed bit in the decrypted message can be precisely controlled.

Although the underlying arithmetic is too different for a direct comparison, we can identify a few comparative aspects. Our approach does not allow us to control the location of the perturbed bit of the decrypted message. The more important subtle difference lies in the type of error used for perturbation. We use carefully constructed perturbations which, in fact, are the chosen ciphertexts used to carry out the PC oracle-based attacks. In contrast, the attacks on LWE/LWR-based schemes use simpler errors which perturb targeted collection of single coefficients of the ciphertext polynomial.

For a quantitative comparison of the attacker's effort, we utilize experimental results from the work of Bhasin *et al.* [BDH+21]. The work demonstrated a practical side-channel attack on a side-channel resistant implementation of Kyber KEM. Their attack exploited side-channel vulnerabilities in the ciphertext comparison operation to instantiate a DF oracle. Their attack on Kyber512 took about 2^{17} decapsulation queries and an additional offline analysis, with a computational complexity of 2^{65}, for full key recovery. Similarly, Guo *et al.* [GJN20] proposed a timing side-channel attack targeting the ciphertext comparison operation to instantiate a DF oracle-based attack on Frodo KEM. Their attack required $\approx 2^{30}$ decapsulation queries for full key recovery in $\mathsf{Frodo-KEM-1344-AES}$. While the number of measurements includes replicated queries for better signal-to-noise ratio, the number of decapsulation queries without replications is still very high at ≈ 118000. Our attack on NTRU-based schemes requires much less number of traces, in the range of 4500 to 7000, for full key recovery with a 100% success rate across all parameter sets of NTRU Prime.

6 Full-Decryption Oracle-Based SCA

We have thus shown that PC and DF oracles can be realized using side-channels through careful crafting of chosen ciphertexts for NTRU Prime and NTRU KEMs to perform full key recovery. The PC and DF oracles, however, only provide 1 bit of information about

the secret-dependent anchor variable, requiring a few thousand chosen ciphertext queries to the target device. A more powerful side-channel adversary, who can extract more that just 1 bit of information about the anchor variable, can perform more efficient attacks.

In this respect, Sim *et al.* [SKL+20] demonstrated single-trace message recovery attacks over several IND-CCA secure NIST PQC KEMs. In particular, their attack targeted routines which manipulate sensitive variables, such as the decrypted messages, one coefficient or one bit at a time. Targeting NTRU, they showed that the polynomial lift operation computed on the decrypted message $\mathbf{m}'$ in line 10 of NTRU_PKE.Decrypt is susceptible to side-channel attacks. They exhibited successful single-trace message recovery with close to a 100% success rate.

Though they have not demonstrate a message recovery for NTRU Prime, we speculate that the weight check operation on the variable $\mathbf{b}'$ in line 6 of NTRU_Prime_PKE.Decrypt could also be susceptible to similar single-trace attacks, because it involves manipulation of single coefficients of $\mathbf{b}'$. The feasibility of performing a single trace recovery of $\mathbf{b}'$ is outside the scope of our present work.

The aforementioned side-channel vulnerabilities can potentially be exploited to recover the complete decrypted message $\mathbf{m}'$ in NTRU or the variable $\mathbf{b}'$ in NTRU Prime in a single trace. We show that such vulnerabilities can also be used to instantiate a full-decryption (FD) oracle in a CCA setting to mount very efficient key recovery attacks. We first describe our attack on NTRU Prime KEM before going to NTRU KEM.

6.1 Attack Methodology on NTRU Prime KEM

The attack methodology directly follows from our PC oracle-based SCA on NTRU Prime KEM in Section 3. We conduct the preprocessing phase to retrieve $\mathbf{c}_{\text{base}}$ whose $\mathbf{e} = \pm x^i$. Using the side-channel based FD oracle, we assume complete recovery of $\mathbf{b}'$ for $\mathbf{c}_{\text{base}}$ in a single trace. From line 5 of NTRU_Prime_PKE.Decrypt, we have

$$\mathbf{b}' = \mathbf{e} \cdot \hat{\mathbf{g}} \in R_3, \tag{40}$$

where $\hat{\mathbf{g}}$ is the inverse of the secret polynomial $\mathbf{g}$ in R_3. Since $\mathbf{e} = \pm x^i$, $\mathbf{g}$, we can recover $\mathbf{g} = \mathbf{e} \cdot \hat{\mathbf{b}'} \in R_3$, where $\hat{\mathbf{b}'}$ is the inverse of $\mathbf{b}' \in R_3$. The attacker does not know i, but can simply try out all possible choices for $i \in [0, n-1]$ to recover the secret polynomials $\mathbf{f}$ and $\mathbf{g}$, up to a rotation.

6.2 Attack Methodology on NTRU KEM

The attack follows the preprocessing phase of the PC oracle-based SCA of NTRU KEM from Subsection 4.1 to retrieve the base ciphertext $\mathbf{c}_{\text{base}}$ whose $\mathbf{e} = \pm x^i$. Using the side-channel based FD oracle, we assume a complete recovery of $\mathbf{m}'$ for $\mathbf{c}_{\text{base}}$ in a single trace. From line 9 of NTRU_PKE.Decrypt procedure, we know that

$$\mathbf{m}' = \mathbf{e} \cdot \mathbf{f}_p \in S_3, \tag{41}$$

where $\mathbf{f}_p$ is the inverse of $\mathbf{f}$ in S_3. Since $\mathbf{e} = \pm x^i$, we can compute $\mathbf{f}_p = \hat{\mathbf{b}'} \cdot \hat{\mathbf{e}} \in S_3$, where $\hat{\mathbf{e}}$ is the inverse of $\mathbf{e} \in R_3$. An attacker can try out all possible values of i to fully retrieve $\mathbf{f}_p$ before calculating the secret key polynomials $\mathbf{f}$ and $\mathbf{g}$.

Unlike in the PC oracle or DF oracle-based attacks, the attacker can perform full key recovery by using the base ciphertext $\mathbf{c}_{\text{base}}$ only for both NTRU and NTRU Prime KEMs. This completely eliminates the need for the key recovery phase. Thus, the trace requirement of the FD oracle-based SCA primarily comes from the preprocessing phase of the attack. We refer to the column corresponding to t_{base} in Tables 5 and 9 for the estimated trace complexity of the FD oracle-based SCA on the different parameter sets of NTRU Prime and NTRU KEMs, respectively.

7 Concluding Remarks

We have thus demonstrated the first practical side-channel assisted CCAs on NTRU and NTRU Prime, which are final round candidates in the onging NIST PQC standardization process. Our attacks involve a careful construction of malformed ciphertexts which, when decrypted, can instantiate three different types of oracles through side-channel leakage from the decapsulation procedure. The resulting responses can then be used to perform full key recovery. The oracles are plaintext-checking, decryption-failure, and full-decryption oracles. We validate our attacks experimentally on optimized implementations of NTRU-based schemes, using the EM-based side-channel on the 32-bit ARM Cortex-M4 microcontroller. All of our attacks are capable of recovering the full secret key in only a few thousand chosen ciphertext queries to the target device on all parameter sets of NTRU and NTRU Prime. The attacks stress on the need for concrete masking strategies for NTRU-based KEMs to protect against side-channel assisted CCAs.

On protection against proposed SCA-assisted CCA: The primary attack methodology behind all our proposed attacks relies on fixing targeted intermediate variables to known values. The subsequent utilization of side-channel leakage identifies the values to perform key recovery. Thus, a complete randomization of the internal computation through masking can serve as a concrete countermeasure. We address the countermeasures for NTRU Prime and NTRU separately.

In NTRU Prime, the PC oracle-based attack only exploits leakage from the decryption procedure. Thus, masking only the decryption procedure in decapsulation protects against the attack. The same applies for the FD oracle-based attack since it primarily relies upon leakage from the decryption procedure. The DF oracle-based attack, however, is capable of exploiting leakage from the re-encryption procedure for key recovery. Thus, the entire decapsulation procedure needs to be masked for a concrete protection to thwart key recovery.

In NTRU, the decapsulation procedure does not perform any re-encryption of the decrypted message. Thus, the decryption procedure remains the only source of side-channel leakage to instantiate the oracles for key recovery. All three attacks target NTRU by exploiting leakage from the decryption procedure. We believe that masking the decryption procedure within decapsulation is sufficient to thwart our attacks. However, the other unmasked operations within the decapsulation procedure could also offer an opportunity for the attacker to instantiate oracles for key recovery. We leave a concrete analysis of this possible attack route for some future work.

There are several works, see, *e.g.*, [LSCH10, WZW13, HCY20, SMS19], on protecting NTRU-based primitives against side-channel attacks. Thus far, existing attacks as well as countermeasures only target the polynomial multiplier involving the secret key in the decryption procedure in lines 3 and 5 in NTRU_Prime_PKE.Decrypt of NTRU Prime in Algorithm 1 and in lines 7 and 9 of NTRU_PKE.Decrypt of NTRU in Algorithm 3. Our attacks have shown that other operations within decryption and decapsulation can also be targeted for key recovery. Moreover, schemes such as Streamlined NTRU Prime include nonlinear operations, which are nontrivial to mask. An example is the weight check in line 6 of NTRU_Prime_PKE.Decrypt of NTRU Prime. To the best of our knowledge, a concrete and complete masking scheme for NTRU-based PKE/KEMs is yet to be devised. Developing efficient and concrete masking strategies for NTRU-based PKE/KEMs, therefore, warrants an urgent attention from our community.

On masking countermeasures for lattice-based schemes: Masking countermeasures, in general, are known to be costly in performance. This is especially true in schemes based on the LWE/LWR problem, such as Kyber and Saber. First-order secure masked implementations of these schemes are slower by 2.5 to 4 times compared to unmasked

implementations on the ARM Cortex-M4 microcontroller [BDK+21, BGR+21]. Designing secure masking schemes is tricky, as subtle flaws, if present, can be exploited. This was shown by Bhasin *et al.* [BDH+21] on Kyber KEM. In addition, higher order attacks are possible on masked implementations, as shown by Ngo *et al.* [NDGJ21] on Saber KEM. Although the aforementioned attacks target masked implementations, the analysis used in performing key recovery by exploiting the observed leakage, is the same as the one for key recovery on unprotected implementations of the targeted schemes. We stress that our proposed approaches to build chosen-ciphertexts as well as the subsequent key recovery could also be extended to target protected/masked implementations of NTRU-based schemes.

References

[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+20a] Erdem Alkim, Joppe W. Bos, Leo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM learning with errors key encapsulation: Algorithm specifications and supporting documentation (September 30, 2020). *Submission to the NIST post-quantum project*, 2020.

[ABD+20b] 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 3.0): Algorithm specifications and supporting documentation (October 1, 2020). *Submission to the NIST post-quantum project*, 2020.

[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.

[AH21] Daniel Apon and James Howe. Attacks on NIST PQC 3rd Round Candidates, 2021. Invited talk at Real World Crypto 2021, https://iacr.org/submit/files/slides/2021/rwc/rwc2021/22/slides.pdf.

[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.

[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. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):334–359, Jul. 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.

[BDK+21] Michiel Van Beirendonck, Jan-Pieter D'anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of saber. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(2):1–26, 2021.

[BGR+21] Joppe W Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First-and higher-order implementations. *IACR Cryptol. ePrint Arch.*, 2021:483, 2021.

[BGRR19] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. Assessment of the key-reuse resilience of NewHope. In *Cryptographers' Track at the RSA Conference*, pages 272–292. Springer, 2019.

[BP18] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. *IACR Cryptol. ePrint Arch.*, 2018:526, 2018.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, pages 719–737. Springer, 2012.

[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.

[CS97] Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 52–61. Springer, 1997.

[DCQ19] Jintai Ding, Chi Cheng, and Yue Qin. A simple key reuse attack on LWE and Ring-LWE encryption schemes as key encapsulation mechanisms (KEMs). *IACR Cryptol. ePrint Arch.*, 2019:271, 2019.

[DDSV19] J Ding, J Deaton, K Schmidt, and Zhang Vishakha. Z.: A simple and practical key reuse attack on NTRU cryptosystem. *IACR Cryptol. ePrint Arch.*, 2019:1022, 2019.

[DKSRV20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. SABER: Mod-LWR based KEM (Round 3 Submission). *Submission to the NIST post-quantum project*, 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. ACM, 2019.

[DZD+17] Aidong Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In *International Conference on Smart Card Research and Advanced Applications*, pages 105–122. Springer, 2017.

[Flu16] Scott R. Fluhrer. Cryptanalysis of Ring-LWE based key exchange with key share reuse. *IACR Cryptol. ePrint Arch.*, 2016:085, 2016.

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

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. In *NIST Non-Invasive Attack Testing Workshop*, volume 7, pages 115–136, 2011.

[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.

[GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 15–29. Springer, 2006.

[GN07] Nicolas Gama and Phong Q. Nguyen. New chosen-ciphertext attacks on NTRU. In *International Workshop on Public Key Cryptography*, pages 89–106. Springer, 2007.

[HCY20] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on NTRU Prime. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):123–151, 2020.

[HGNP+03] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In *Annual International Cryptology Conference*, pages 226–246. Springer, 2003.

[HHHK03] Daewan Han, Jin Hong, Jae Woo Han, and Daesung Kwon. Key recovery attacks on NTRU without ciphertext validation routine. In *Australasian Conference on Information Security and Privacy*, pages 274–284. Springer, 2003.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. NTRU: A ring-based public key cryptosystem. *Algorithmic number theory*, pages 267–288, 1998.

[HS99] Jeffrey Hoffstein and Joseph H Silverman. Reaction attacks against the NTRU public key cryptosystem. Technical Report 15, NTRU Cryptosystems, 1999.

[JJ00] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In *Annual International Cryptology Conference*, pages 20–35. Springer, 2000.

[KEF20] Paul Kirchner, Thomas Espitau, and Pierre-Alain Fouque. Fast reduction of algebraic lattices over cyclotomic fields. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 155–185. Springer, 2020.

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

[KRSS19] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4. In *Second PQC Standardization Conference: University of California, Santa Barbara and co-located with Crypto 2019*, pages 1–22, 2019.

[LSCH10] Mun-Kyu Lee, Jeong Eun Song, Dooho Choi, and Dong-Guk Han. Countermeasures against power analysis attacks for the NTRU public key cryptosystem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer sciences*, 93(1):153–163, 2010.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked IND-CCA secure Saber KEM. *IACR Cryptol. ePrint Arch.*, 2021:079, 2021.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical cca2-secure and masked ring-lwe implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 142–174, 2018.

[QCD19] Yue Qin, Chi Cheng, and Jintai Ding. A complete and optimized key mismatch attack on NIST candidate NewHope. *IACR Cryptol. ePrint Arch.*, 2019:435, 2019.

[RBRC20] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery and key recovery attacks. *IACR Cryptol. ePrint Arch.*, 2020:1559, 2020.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[RJJ+18] Prasanna Ravi, Bernhard Jungk, Dirmanto Jap, Zakaria Najm, and Shivam Bhasin. Feature selection methods for non-profiled side-channel attacks on ECC. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5, 2018.

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

[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. *IEEE Access*, 8:183175–183191, 2020.

[SMS19] Thomas Schamberger, Oliver Mischke, and Johanna Sepulveda. Practical evaluation of masking for NTRUEncrypt on ARM Cortex-M4. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2019.

[WZW13] An Wang, Xuexin Zheng, and Zongyue Wang. Power analysis attacks and countermeasures on NTRU-based wireless body area networks. *KSII Transactions on Internet and Information Systems (TIIS)*, 7(5):1094–1107, 2013.

[XPRO20] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David Oswald. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. *IACR Cryptol. ePrint Arch.*, 2020:912, 2020.

[ZCD21] Xiaohan Zhang, Chi Cheng, and Ruoyu Ding. Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. In Debin Gao, Qi Li, Xiaohong Guan, and Xiaofeng Liao, editors, *Information and Communications Security*, pages 283–300. Springer, 2021.