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 for NTRU-based Key Encapsulation Mechanisms (KEM) secure in the *chosen ciphertext model* (IND-CCA security). Our attacks involve construction of malformed ciphertexts which, when decapsulated by the target device, ensure that a targeted intermediate variable has a very close relation with the secret key. Subsequently, an attacker who can obtain information about the secret-dependent variable through side-channels, can recover the full secret key. We propose several novel CCAs which can be carried through instantiating three different types of oracles, namely plaintext-checking oracle, decryption-failure oracle, and full-decryption oracle, using side-channel leakage from the decapsulation procedure. Our proposed attacks are applicable to two NTRU-based schemes: NTRU and NTRU Prime. The two schemes are candidates in the ongoing NIST standardization process for post-quantum cryptography. We perform experimental validation of our proposed attacks on optimized 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 our proposed 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. Our attacks therefore stress on the need for concrete 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 and final round with seven finalist candidates and eight alternate candidates for Public Key Encryption (PKE), Key Encapsulation Mechanisms (KEM), 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) will also be considered as important criteria in the standardization process, especially amongst schemes with tightly matched security and efficiency [AH21]. The cost of implementing protections against such attacks will also be looked into closely. In [AASA+20, Section 2.2.3] NIST states that

NIST hopes to collect more information about the costs of implementing these algorithms in a way that provides resistance to side-channel attacks.

Three out of the four finalist candidates 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 which 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 NTRU-based schemes [DCQ19, QCD19, BDHD+19, BGRR19, Flu16]. These attacks mainly work by assuming the presence of an oracle that provides some information about the decrypted message.

There are at least three types of oracles that can be instantiated, depending upon the setting, 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 Plaintext-Checking 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 a binary information, a full decryption oracle provides information about the complete message for chosen ciphertexts. Based on the available oracle, an attacker carefully chooses query ciphertexts in such a way that the corresponding oracle's responses reveal the secret key.

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

Following this line of thought, several side-channel assisted CCAs on LWE/LWR-based PKE/KEMs have been proposed. They utilize side-channel information to instantiate different types of oracles to gain information about the decryption output. This information facilitates secret key recovery in several LWE/LWR-based NIST candidates. These include the finalists, such as Kyber [ABD+20b], Saber [DKSRV20], and Frodo [ABD+20a]. A similar analysis is, however, lacking for schemes based on the NTRU problem, including the main finalist NTRU [CDH+19] and alternate finalist NTRU Prime [BBC+20]. Extending such attacks to NTRU-based schemes is not trivial since the framework and arithmetic, underlying NTRU-based schemes is vastly different from those based on the LWE/LWR paradigm. Thus, mounting side-channel assisted CCAs on NTRU-based schemes remains an open problem.

This remains the case, even when there are known CCAs on IND-CPA secure 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 the same attacks to the newer variants is not trivial, due to differences in the underlying arithmetic and finer technical details. As an example, we refer to the work of Zhang *et al.* [ZCQD21]. They demonstrated a successful attack on the NTRU-HPS variant of the NIST finalist NTRU KEM with 100% success rate, but failed to achieve the same success rate when targeting the NTRU-HRSS variant of NTRU KEM. Similarly, NTRU Prime also incorporates several optimizations such as the use of rounded ciphertexts. Its arithmetic, over a non-cyclotomic field, poses significant challenges in performing CCAs, both in a black box setting as well as in a side-channel setting.

Thus, there exists a sufficient gap in theoretical understanding in terms of how to mount CCAs over the newer variants of NTRU-based schemes. These aspects make it very interesting to develop side-channel assisted CCAs on NTRU-based schemes. Another pertinent question that arises is, *even if such attacks appear to be possible, is there a significant difference in terms of the cost of side-channel CCAs on NTRU-based schemes, compared to attacks on LWE/LWR-based schemes.*

To address these critical questions, we propose here the first side-channel assisted CCAs on IND-CCA secure NTRU-based schemes. Our attacks are applicable to the IND-CCA secure NTRU and NTRU Prime KEMs, the NTRU-based candidates for PKE/KEMs in the NIST Post Quantum Cryptography (PQC) standardization process. We attempt to traverse the landscape of side-channel assisted CCAs, by demonstrating practical side-channel attacks instantiating three different types of oracles. These are the PC oracle, the DF oracle, and the FD oracle on all parameter sets of NTRU and NTRU Prime. Underlying the attacks is the key idea of building suitably chosen ciphertexts that are capable of instantiating the three types of oracles. The idea for the type of ciphertexts to be built is inspired by the work of Jaulmes and Joux [JJ00]. They proposed the first CCA that works in a black box setting on the original IND-CPA secure NTRU PKE scheme of Hoffstein *et al* [HPS98]. We in this work propose novel and generic adaptations of their attack to mount successful side-channel assisted CCAs on NTRU and NTRU Prime. Remarkably, all our proposed attacks only require a few thousand chosen-ciphertext queries to the target device for full key recovery with a 100% success rate and no offline analysis for key recovery. Our analysis is also backed by successful experimental validation on optimized implementations of NTRU and NTRU Prime KEM taken from the open-source *pqm4* library [KRSS19], on the 32-bit ARM Cortex-M4 microcontroller using the Electromagnetic Emanation (EM) side-channel.

Contributions:

The main contributions of our work can be summarized as follows.

1. We demonstrate the first practical side-channel assisted chosen-ciphertext attacks on NTRU-based schemes. The attacks target two NTRU-based schemes, NTRU and NTRU Prime, which are final round candidates in the onging NIST PQC standardization process. It is worth noting that such attacks until now, have only been demonstrated on LWE/LWR-based schemes. Our work is the first to investigate such attacks on NTRU-based schemes.

2. We traverse the landscape of side-channel assisted CCAs on NTRU-based schemes by demonstrating practical attacks that instantiate three different types of oracles through side-channels on IND-CCA secure NTRU and NTRU Prime KEMs for full key recovery. These are the plaintext-checking oracle, decryption-failure oracle, and full-decryption oracle.

3. The core idea in the construction of chosen ciphertexts to instantiate the three oracles is inspired by the work of Jaulmes and Joux [JJ00]. They proposed the first CCA on the original NTRU scheme of Hoffstein *et al* [HPS98]. We propose generic and novel adaptations of their attack to develop successful CCAs. They are capable of utilizing all three oracles for successful key recovery in NTRU and NTRU Prime KEMs.

4. Remarkably, we also propose the first attack that works with a 100% success rate on the NTRU-HRSS variant of NTRU and Streamlined NTRU Prime, assuming the presence of a plaintext-checking oracle for key recovery. We exhibit novel techniques

to subvert the challenges posed by optimizations such as use of rounded ciphertexts in NTRU Prime and use of arbitrary-weight secrets in the NTRU-HRSS variant of NTRU, to perform successful key recovery.

5. We also demonstrate simple techniques to utilize side-channel leakage from the decapsulation procedure, to realize a practical plaintext-checking oracle and decryption-failure oracle, to combine it with the capabilities of our proposed novel CCAs for efficient key recovery attacks. Since these oracles only provide binary information, the side-channel analysis relies on simple techniques and can be performed with very minimal knowledge about the target implementation.

6. We perform experimental validation of our attacks on optimized implementations of NTRU-based schemes taken from the open-source *pqm4* library [KRSS19], using the EM-based side-channel on the 32-bit ARM Cortex-M4 microcontroller. All our proposed 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.

Availability of software

All softwares utilized for this work is placed into the public domain. They are available at `https://github.com/SCACCAONNTRU/SCACCAONNTRU`.

Organization of the Paper

This paper is organized as follows. 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 discusses potential countermeasures against our proposed attacks. Section 8 concludes our paper.

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, zero-centered in the range $[-q/2, q/2-1] \cap \mathbb{Z}$ if q is even, or $[-(q-1)/2, (q-1)/2] \cap \mathbb{Z}$ if q is odd. For brevity, we denote the threshold as $q/2$ throughout the paper, irrespective of q being even or odd. Let $\mathbb{Z}_q[x]/(\phi(x))$ denote the polynomial ring whose reduction polynomial is $\phi(x)$. The ring 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 multiplication of two polynomials $\mathbf{a}$ and $\mathbf{b}$ is denoted as $\mathbf{c} = \mathbf{a} \cdot \mathbf{b}$. 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

Hoffstein, Pipher, and Silverman in 1998 [HPS98] proposed the N^{th} *order Truncated Polynomial Ring Unit* (NTRU) public key encryption scheme. Its security relies on a conjectured circular security assumption, better known as the *NTRU assumption* or the *NTRU One-Way Function*, involving the factorization of polynomials in R_q [HPS98].

Definition 1 (NTRU OWF)**.** Given $R_{\text{NTRU}} := \mathbb{Z}_q[x]/(x^N - 1)$, 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 was shown to be reducible to a *shortest vector problem* (SVP) over a special class of lattices known as the *NTRU lattices* [CS97]. It is worth noting that the NTRU cryptosystem has survived cryptanalysis for almost 24 years now. This instills a lot of confidence in its security claims, despite the lack of provable security guarantees. Two candidate PKE/KEMs in the NIST PQC standardization process, namely a main finalist NTRU [CDH+19] and an alternate finalist NTRU Prime [BBC+20], are based on the paradigm of the NTRU cryptosystem. For clarity, we refer to the original NTRU PKE proposed in [HPS98] as NTRU-1998 whereas the finalists NTRU and NTRU Prime are referred to by their respective names throughout this paper.

2.3 NTRU Prime

NTRU Prime is a suite of two IND-CCA secure KEMs: 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, it contains a perfectly correct and deterministic IND-CPA secure PKE. It is defined by three parameters (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 the NTRU-1998 PKE which operates in a cyclotomic ring $(\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$ with $n = 2^k$, NTRU Prime operates in the field $R_q := \mathbb{Z}_q[x]/(x^n - x - 1)$, which is not cyclotomic. The choice is motivated by the need to protect against potential attacks that could exploit the cyclotomic structure in lattice-based schemes [KEF20].

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

The key generation procedure $\mathsf{NTRU\_PRIME\_PKE.KeyGen}$ 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 procedure $\mathsf{NTRU\_PRIME\_PKE.Encrypt}$ takes as input the message polynomial $\mathbf{r} \in R_{\text{sh}}$ and generates a product-form NTRU instance $\mathbf{c} = \mathsf{Round}(\mathbf{r} \cdot \mathbf{h}) \in R_q$ as the ciphertext whose coefficients are multiples of 3. The decryption procedure $\mathsf{NTRU\_Prime\_PKE.Decrypt}$ takes the ciphertext $\mathbf{c}$ to first compute $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c} \in R_q$. The parameters are chosen to ensure that the true, that is the non-reduced, 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 $\hat{\mathbf{g}} \in R_3$, 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 $(1, 1, \ldots, 1, 0, 0, \ldots, 0) \in R_3$.

Algorithm 1: Streamlined NTRU Prime PKE Core

```
1  Procedure NTRU_PRIME_PKE.KeyGen()
2      while $\mathbf{g}$ is invertible in $R_3$ do
3          $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$,  $\mathbf{g} \leftarrow \mathsf{GenSmall}(\rho) \in R$
4      end
5      $\hat{\mathbf{g}} = 1/\mathbf{g} \in R_3$
6      $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$,  $\mathbf{f} \leftarrow \mathsf{GenShort}(\rho) \in R_{\text{sh}}$
7      $\mathbf{h} = \mathbf{g}/(3\mathbf{f}) \in R_q$
8      return $(pk = \mathbf{h}, sk = (\hat{\mathbf{g}}, \mathbf{f}))$
9  ____________________________________________
1  Procedure NTRU_PRIME_PKE.Encrypt($pk, \mathbf{r} \in R_{\text{sh}}$)
2      $\mathbf{d} = \mathbf{h} \cdot \mathbf{r} \in R_q$
3      $\mathbf{c} = \mathsf{Round}(\mathbf{d}) \in R_q$
4      $ct = \mathsf{Encode}(\mathbf{c})$
5      return $(ct)$
6  ____________________________________________
1  Procedure NTRU_PRIME_PKE.Decrypt($ct, sk$)
2      $\mathbf{c} = \mathsf{Decode}(ct) \in R_q$
3      $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c} \in R_q$
4      $\mathbf{e} = \mathbf{a} \bmod R_3$
5      $\mathbf{b}' = \mathbf{e} \cdot \hat{\mathbf{g}} \in R_3$
6      if $\mathsf{Weight}(\mathbf{b}') = w$ then
7          return $\mathbf{r}' = \mathbf{b}'$
8      end
9      else
10         return $\mathbf{r}' = (1, 1, \ldots, 1, 0, 0, \ldots, 0) \in R_{\text{sh}}$
11     end
```

2.3.1 IND-CCA Secure NTRU Prime KEM

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 detail. 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 performs computations over two polynomial rings $S_k := \mathbb{Z}_k[x]/(\phi_n)$ and $T_k := \mathbb{Z}_k[x]/(\phi_1 \phi_n)$. 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. NTRU-HPS selects coefficients from fixed-weight sample spaces, similar to the

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

```
1  Procedure NTRU_Prime_KEM.Encaps($pk$)
2      $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
3      $\mathbf{r}$ = GenShort($\rho$) $\in R_{\text{sh}}$
4      $\mathbf{c}$ = NTRU_PRIME_PKE.Encrypt($pk, \mathbf{r}$)
5      $\mathbf{d} = \mathcal{H}(\mathbf{r}, pk)$
6      $ct = (\mathbf{c}, \mathbf{d}),\ \ K = \mathcal{G}(1, \mathbf{r}, ct)$
7      return $ct, K$
8  ____________________________________________________________
1  Procedure KEM.Decaps($sk, pk, ct$)
2      $ct = (\mathbf{c}, \mathbf{d})$
3      $\mathbf{r}'$ = NTRU_PRIME_PKE.Decrypt($sk, c$)
4      $d' = \mathcal{H}(\mathbf{r}', pk)$
5      $c'$ = NTRU_PRIME_PKE.Encrypt($pk, \mathbf{r}'$)
6      $ct' = (c', d')$
7      if $ct' = ct$ then
8          return $K = \mathcal{G}(1, \mathbf{r}', ct')$
9      end
10     else
11         return $K = \mathcal{G}(1, \rho', ct')$ /* $\rho' \in \mathcal{B}^{32}$ is a random secret */
12     end
```

NTRU-1998 PKE. 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. The procedure Sample_fg() takes in a seed $\rho \in \mathcal{B}^*$ and samples the secret polynomials $\mathbf{f}, \mathbf{g} \in R_p$ where $p = 3$. 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 procedure NTRU_PKE.Encrypt takes a random $\mathbf{r} \in \mathcal{L}_r$ and a message $\mathbf{m} \in \mathcal{L}_m$ as input 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 procedure NTRU_PKE.Decrypt uses the ciphertext $\mathbf{c}$ 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 compute 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 denoted as $fail$ which denotes success or failure of decryption, where $fail = 0$ denotes success, and failure otherwise.

2.4.1 IND-CCA Secure NTRU KEM

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. They instantiate NTRU_PKE.Encrypt and NTRU_PKE.Decrypt, respectively, along with several instances of hash functions.

Algorithm 3: NTRU PKE Core

```
Procedure NTRU_PKE.KeyGen()
    $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
    $(\mathbf{f}, \mathbf{g}) \leftarrow \mathsf{Sample\_fg}(\rho) \in (\mathcal{L}_f, \mathcal{L}_g)$
    $\mathbf{f}_q = (1/\mathbf{f}) \in (S_q)$
    $\mathbf{h} = (3 \cdot \mathbf{g} \cdot \mathbf{f}_q) \in (T_q)$
    $\mathbf{h}_q = (1 \cdot \mathbf{h}) \in (S_q)$
    $\mathbf{f}_p = (1/\mathbf{f}) \in (S_3)$
    return $(pk = (\mathbf{h}, \mathbf{h}_q), sk = (\mathbf{f}, \mathbf{f}_p))$
------------------------------------------------------------
Procedure NTRU_PKE.Encrypt($pk, \mathbf{r}, \mathbf{m} \in (\mathcal{L}_r \times \mathcal{L}_m)$)
    $\bar{\mathbf{m}} = \mathsf{Lift}(\mathbf{m}) \in S_3$
    $\mathbf{c} = \mathbf{h} \cdot \mathbf{r} + \mathbf{m}' \in T_q$
    $ct = \mathsf{Encode}(\mathbf{c})$
    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
        $fail = 1$
        return $(0, 0, fail)$
    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
        $fail = 0$
        return $(\mathbf{r}', \mathbf{m}', fail)$
    end
    else
        $fail = 1$
        return $(0, 0, fail)$
    end
```

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 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, bears a very close relation with 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

Algorithm 4: IND-CCA secure NTRU KEM

```
1  Procedure NTRU_KEM.Encaps($pk$)
2      $\rho \leftarrow \mathcal{U}(\mathcal{B}^*)$
3      $(\mathbf{r}, \mathbf{m}) = \mathsf{Sample\_rm}(\rho)$
4      $\mathbf{c} = \mathsf{NTRU\_PKE.Encrypt}(pk, \mathbf{r}, \mathbf{m})$
5      $k = \mathcal{H}(\mathbf{r}, \mathbf{m})$
6      $ct = \mathbf{c}$
7      return $ct, K$
8  ____________________________________________________________
1  Procedure NTRU_KEM.Decaps($sk, pk, ct$)
2      $ct = (\mathbf{c}, \mathbf{d})$
3      $(\mathbf{r}', \mathbf{m}', fail) = \mathsf{NTRU\_PKE.Decrypt}(sk, ct)$
4      $k_1 = \mathbf{H}(\mathbf{r}', \mathbf{m}')$
5      $k_2 = \mathbf{G}(s, \mathbf{c})$
       /* $s \in \mathcal{B}^{32}$ is a random secret */
6      if $fail = 0$ then
7          return $k_1$
8      end
9      else
10         return $k_2$
11     end
```

ciphertexts using side-channels, the full secret key can be recovered. Based on the type and amount of side-channel information available, we categorize the existing attacks on LWE/LWR-based schemes into the following three categories.

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 message 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 exploited to realize a PC oracle in IND-CCA secure schemes whose binary responses $m \in \{0, 1\}$ 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 perform 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 non-constant 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 were capable of performing full key recovery with several thousand chosen ciphertext queries to the target device.

As can be seen, 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 and does not pose stringent requirements on the Signal to Noise Ratio (SNR) for trace acquisition. Moreover, the analysis is also fairly simple and can be performed with very limited knowledge of the target implementation.

2.5.3 Full-Decryption Oracle-Based SCA

While the PC oracle and DF oracle attacks only extract binary information (1-bit) about the anchor variable through side-channel traces, they typically require a few thousand chosen-ciphertext queries to the target device 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 in only a handful of traces/queries. They showed how to perform full key recovery using only 8 to 16 chosen-ciphertext queries in LWE/LWR-basd KEMs such as Kyber and Saber. Their demonstration 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 leak the complete message as discussed in, for examples, Ravi *et al.* [RBRC20] and Ngo *et al.* [NDGJ21]. Table 1 lists side-channel assisted CCAs on IND-CCA secure LWE/LWR-based schemes based on 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 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. We demonstrate that our proposed attacks are practical, generic, and are capable of exploiting all three different

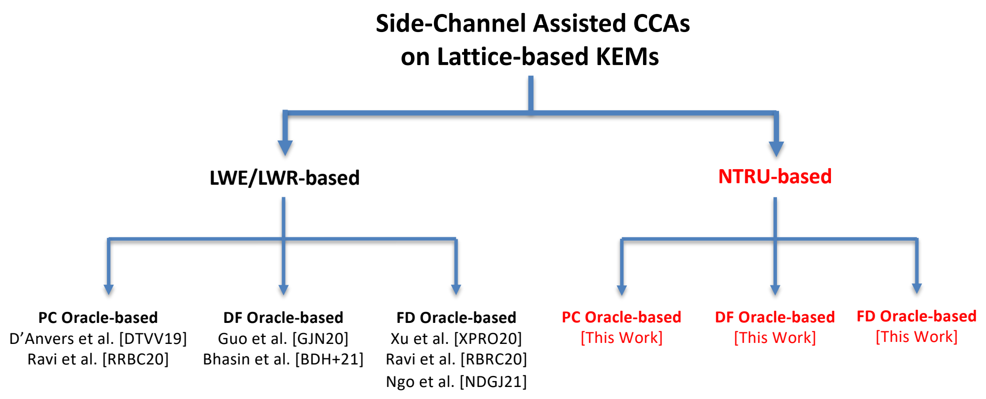


Figure 1: Classification of various side-channel assisted CCAs on lattice-based KEMs.

Table 1: Classification of side-channel assisted CCAs on IND-CCA secure LWE/LWR-based schemes. It is based on the nature of the oracle used for key recovery, where anchor denotes the anchor variable and m_x, with or without subscript, denotes the decrypted message.

Type of Oracle	Oracle Response
Plaintext-Checking (PC)	$\mathsf{anchor} \in \{m_0, m_1\}$
Decryption-Failure (DF)	$\mathsf{anchor} \in \{m_{\text{valid}}, m_{\text{invalid}}\}$
Full-Decryption (FD)	$\mathsf{anchor} = m$

types of oracles for full key recovery. Our 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

Given the existence of the NTRU cryptosystem for almost 24 years now, 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 PKE. Their attack requires knowledge of the full decryption output (i.e.) FD oracle and can recover the full secret key with a handful of ciphertexts. They also present an adaptation of their attack to the OAEP-like padding scheme, which works only with the knowledge of decryption failures (i.e.) DF oracle for key recovery. Hoffstein and Silverman also presented CCAs using the DF oracle on the unpadded NTRU-1998 PKE [HS99].

Han *et al.* [HHHK03] subsequently presented very efficient CCAs based on the FD oracle, on optimized variants of unpadded NTRU-1998 PKE and their proposed attacks utilize chosen ciphertexts that are completely pre-computed offline, independent of the previous outputs. While the aforementioned attacks utilize invalid/maliciously crafted ciphertexts, another class of CCAs exploit decryption failures for valid ciphertexts [HGNP+03, GN07]. While these attacks apply to variants of NTRU cryptosystem with non-negligible decryption failure rate, they are not relevant to the more recent variants, and in particular, the NIST PQC candidates NTRU and NTRU Prime, as they are based on perfectly correct PKE.

More recently, Ding *et al.* [DDSV19] presented a novel CCA on the NTRU-1998 PKE using the DF oracle. While this attack with trivial modifications can be adapted to the NTRU-HPS parameter set of NTRU assuming a PC oracle, Zhang et al. [ZCQD21] showed that it does not work on the NTRU-HRSS variant of NTRU, due to the use of secrets with arbitary weight. They adapt the attack of Ding *et al.* [ZCQD21] to the NTRU-HRSS scheme, but the improved technique can only recover 93.6% of the keys. Thus, a CCA against the NTRU-HRSS scheme that works with a 100% success rate is not known. Moreover, to the best of our knowledge, we are also not aware of a CCA on NTRU Prime. As we later show in the paper, mounting CCAs on NTRU Prime is especially challenging, given its use of rounded ciphertexts and conditional checks on the decrypted message.

In this work, we improve upon the CCA proposed by Jaulmes and Joux [JJ00] and propose generic and novel adapatations to the NIST PQC candidates NTRU and NTRU Prime. Remarkably, our proposed attacks can perform key recovery with 100% success rate on all parameter sets of NTRU and NTRU Prime, assuming the presence of a suitable oracle. To the best of our knowledge, we therefore present the first CCA on IND-CPA secure PKE of NTRU-HRSS and NTRU Prime, that works with a 100% success rate. We also extend the same attacks to the side-channel setting, to propose the first side-channel assisted CCAs on IND-CCA secure NTRU and NTRU Prime KEM.

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. The TVLA formulation over measurement sets $\mathcal{T}_r$ and $\mathcal{T}_f$ is given 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 [GJJR11]. A rejected null hypothesis implies that the two trace/data sets are different and 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 the threshold as 5 based on our 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. **Pre-Processing 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 the base ciphertext 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, $\mathbf{e} = 0$ or $\mathbf{e} \neq 0$ 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 ($\mathbf{e} = 0$ 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.

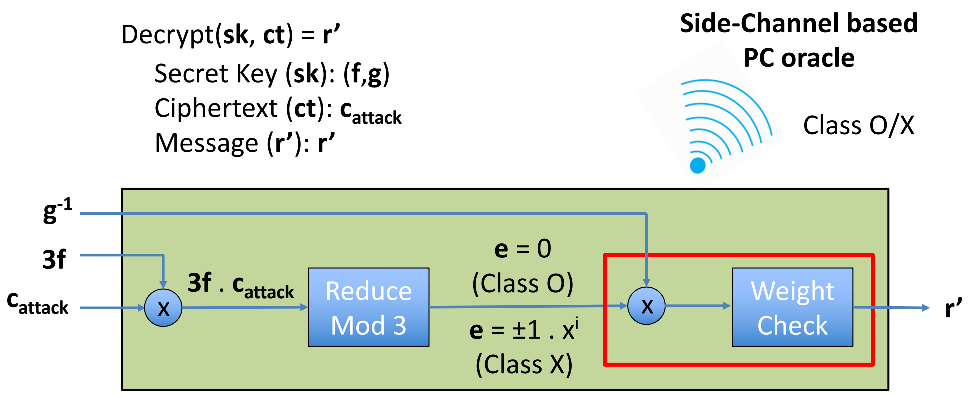


Figure 2: Pictorial Illustration of our PC oracle-based SCA on NTRU Prime

3.1 Pre-Processing Phase: Retrieving the Base Ciphertext $\mathbf{c}_{\mathbf{base}}$

Our construction of chosen ciphertexts is inspired by the attack of Jaulmes and Joux on NTRU-1998 in [JJ00]. We start with an intuition for the approach before proposing a concrete methodology. The notation used is from Algorithm 1 of NTRU Prime.

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,

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

Subsequently, $\mathbf{e} = \mathbf{a} \bmod 3 \in R_3$ 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 $\mathbf{g}$ for sntrup761. We denote by ρ the probability of a collision at any given coefficient and by

$$\rho_{\text{match}} := \rho_1 + \rho_{-1}, \text{ where } \rho_x, \text{ for } x \in \{-1, 1\}$$

the probability of a collision between $\mathbf{f}$ and $\mathbf{g}$ with a matching coefficient of either -1 or 1. For $\mathbf{f} \in R_{\text{sh}}$ and $\mathbf{g} \in R_3$, we get $\rho_{\text{match}} := (w/3n)$ and, hence, $\rho_{\text{match}} \approx 0.125$ for sntrup761. The probability of a single collision between $\mathbf{f}$ and $\mathbf{g}$ is

$$\rho_{\text{single}} = n \cdot \rho_{\text{match}} \cdot (1 - \rho_{\text{match}})^{n-1}.$$

This value 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 a polynomial $\mathbf{d}$ 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 $\mathsf{Rotp_R}(\mathbf{d}, i)$ and refer to it informally as the rotation of $\mathbf{d}$ by i degrees. Thus,

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

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 $\mathsf{Rotp_R}(\mathbf{f}, u)$, for $u \in \{i_1, i_2, \ldots, i_m\}$, and $\mathsf{Rotp_R}(\mathbf{g}, v)$, for $v \in \{j_1, j_2, \ldots, j_n\}$, are either $+2$ or -2. We observe that the probability of collisions quickly degrades as (m, n) increase.

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 $0 \leq r \leq 2m$ and $0 \leq s \leq 2n$. 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.

Summarizing the above discussion, 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) ensures that a single collision takes place 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. The concrete values for (m, n) and (k_1, k_2) can be fixed for a given parameter set of NTRU Prime.

3.1.3 Additional Challenge: Use of Rounded Ciphertexts

The encryption procedure of NTRU Prime generates ciphertexts whose coefficients are rounded to the exact multiples of 3 (line 3 of NTRU_Prime_PKE.Encrypt procedure). Thus, the scheme proposes to send only the quotient of each coefficient upon division by 3, thereby reducing ciphertext size. Thus, every coefficient of the received ciphertext is multiplied by 3 by the decryption procedure. However, our chosen ciphertexts (Equation (8)) are not exact multiples of 3 and thus need to be rounded, which introduces a rounding noise denoted as $\mathbf{m}' \in R_3$. Thus, the actual value of our chosen-ciphertext used in decryption is given by

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

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

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

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. But, $\mathbf{m} \in R_3$ and $\mathbf{f} \in R_{\text{sh}}$ are small polynomials, making the size of noise much smaller in comparison to the range q.

For the parameter set sntrup761, Figure 3 shows the distribution of the coefficients $\mathbf{n}[j]$ for $j \in [0, n-1]$ of $\mathbf{n}$. 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. However, when $\mathbf{n}$ is added to coefficients of $\mathbf{a}$ near $q/2$, 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 value of a coefficient of $\mathbf{a}$ is denoted by $m_1 := (3k_1 \cdot 2m + k_2 \cdot 2n)$. The next largest value is denoted by 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 dm_1 (resp. dm_2) denote the distance between m_1 (resp. m_2) from $q/2$, where

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

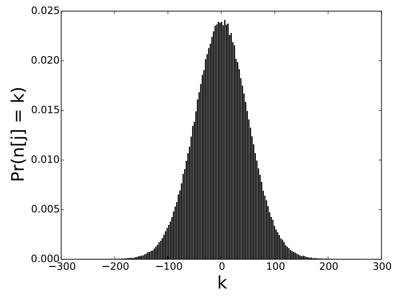


Figure 3: Distribution of the coefficients of the noise $\mathbf{n} := 3\mathbf{f} \cdot \mathbf{m}'$ with mean 0 and $\sigma \approx 57$ for sntrup761.

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

Scheme	(m, n)	**Pre-Processing 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)

A false positive collision occurs when $\mathbf{s}[i] = m_2$ and the corresponding $\mathbf{n}[i] > dm_2$, ensuring that $\mathbf{s}[i] + \mathbf{n}[i] > q/2$ and $\mathbf{e}[i] \neq 0$. Similarly, a false negative can occur when $\mathbf{s}[i] = m_1$ due to a valid collision. If, however, $\mathbf{n}[i] < -dm_1$, then $\mathbf{s}[i] + \mathbf{n}[i] < q/2$ and $\mathbf{e}[i] = 0$, which suppresses the collision.

Though the rounding noise $\mathbf{n}$ cannot be removed, the possibility of a false positive or negative collision can, however, be minimized by placing additional constraints in choosing the tuple (k_1, k_2). Along with constraints for (k_1, k_2) in Equation (12), we choose the tuple that *maximizes* the distance dm_1 (resp. dm_2) for m_1 (resp. m_2) to prevent the noise coefficient $\mathbf{n}[j]$ from growing large enough to push $\mathbf{a}[j]$ to the other side of $q/2$, which is when an error occurs in the value of $\mathbf{e}$. As long as the error $\mathbf{n}[j]$ does not push $\mathbf{a}[j]$ to the other side of $q/2$, there will be no error in $\mathbf{e}$. In other words, m_1 and m_2 should lie as far as possible on either side of the threshold $q/2$. This additional constraint in the choice of (k_1, k_2) is simply to maximize the distance tuple (dm_1, dm_2).

For sntrup761 of NTRU Prime, we empirically choose $(m, n) = (0, 4)$ and $(k_1, k_2) = (0, 306)$. We stress that other values can also be chosen to construct ciphertexts corresponding to single collision. Table 2 lists the concrete values of (m, n), (k_1, k_2) and the corresponding distance tuple (dm_1, dm_2) for different parameter sets of NTRU Prime. These values can be chosen beforehand for any given parameter set.

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 not possible 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 $\mathbf{e} \neq 0$.

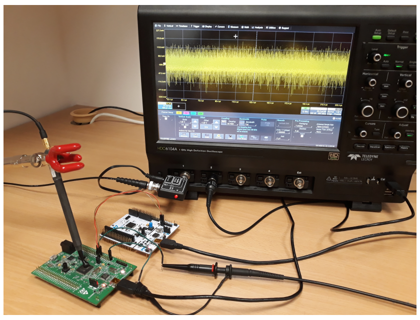


Figure 4: Experimental setup used for EM trace acquisition.

For $\mathbf{e} = 0$, Line 5 of the NTRU_Prime_PKE.Decrypt procedure implies $\mathbf{b}' = \mathbf{e} \cdot \hat{\mathbf{g}} = 0$ and, 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 other parameter sets of NTRU Prime.

In our experiments, 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 following 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. Figure 4 shows our experimental setup to perform EM trace acquisition. We adopt the Welch's t-test to detect a collision for a chosen ciphertext.

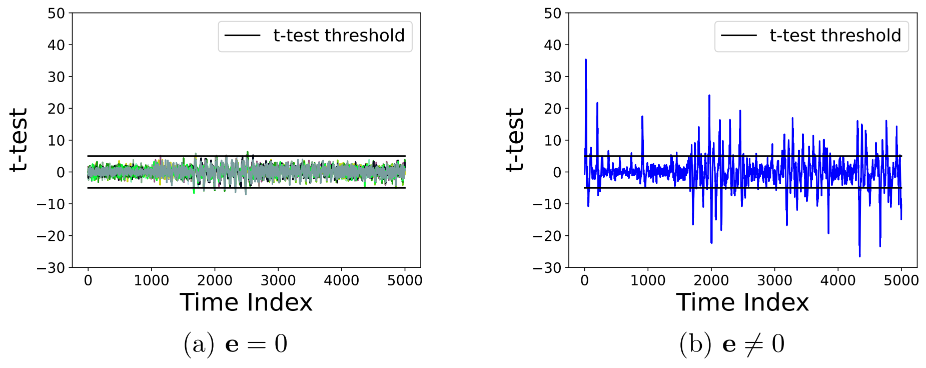


Figure 5: The t-test plots between $\mathcal{T}_{\mathrm{O}}$ and $\mathcal{T}_{\mathrm{X}}$ for `sntrup761`.

Welch's t-test for Collision Detection: Due to the large difference in weights, we focus on capturing EM signals from the weight calculation operation in Line 6 of the decryption procedure 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 obtain T replicated measurements from the decryption of $\mathbf{c}'$, which is denoted by $\mathcal{T}_{\mathrm{X}}$. Let $\mathcal{T} = \mathcal{T}_{\mathrm{O}} \cup \mathcal{T}_{\mathrm{X}}$. We now perform the Welch's t-test between $\mathcal{T}_{\mathrm{O}}$ 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). If there are several peaks well above the t-test threshold of ± 5, then $\mathbf{e} \neq 0$ for $\mathbf{c}'$. Otherwise, $\mathbf{e} = 0$. Figure 5(a) depicts the t-test plot if $\mathbf{e} = 0$ for $\mathbf{c}'$ on $T = 10$ replicated measurements. As can be clearly seen, we do not observe any significant peaks about the above the t-test threshold of ± 5. It is possible that there are a few points bordering the threshold or marginally exceeding it. We performed an examination of the internal registers and the control flow to identify any change in behaviour that could result in t-test values bordering the threshold. But, we were unable to identify any discernible change in the state of the device. Thus, those points with bordering t-test values can be safely ignored. Figure 5(b) corresponds to $\mathbf{e} \neq 0$. We can identify several peaks, well above the threshold, which clearly indicates $\mathbf{e} \neq 0$.

As we can see, leakage detection for identification of $\mathbf{e} \neq 0$ does not assume any knowledge about the implementation of the decapsulation procedure. In the worst case, the attacker only requires to know the location of the targeted operations within the decryption procedure, but as shown in previous works [ACLZ20,NDGJ21], it is also possible to identify operations within the decapsulation procedure, through visual inspection.

We repeat this 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 non-zero coefficients in $\mathbf{e}$. If we identify a tuple $(\mathbf{d}_1, \mathbf{d}_2)$ that corresponds to $\mathbf{e} \neq 0$, we simply proceed to the key recovery phase of the attack. For a faulty base ciphertext with a single false collision or multiple collisions, key recovery cannot be performed correctly. Thus, we simply need to repeat the attack until the correct key is recovered.

We denote the ciphertext corresponding to $\mathbf{e} \neq 0$ as $\mathbf{c}_\mathrm{base}$. For analytical purpose, we assume that the ciphertext $\mathbf{c}_\mathrm{base}$ that corresponds to $\mathbf{e} \neq 0$ has a single non-zero coefficient at index i. We us $(\mathbf{d1}_\mathrm{att}, \mathbf{d2}_\mathrm{att})$ to denote the tuple $(\mathbf{d}_1, \mathbf{d}_2)$, with m and n nonzero coefficients, respectively, that corresponds to $\mathbf{c}_\mathrm{base}$ as . The ciphertext $\mathbf{c}_\mathrm{base}$ is, therefore,

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

Upon retrieval of $\mathbf{c}_\mathrm{base}$, we proceed to the second phase of the attack, which is the key recovery phase.

3.2 Key Recovery Phase

Attack Overview: The key recovery phase works by constructing new attack ciphertexts using ($\mathbf{d1}_\mathrm{att}$,$\mathbf{d2}_\mathrm{att}$), which when decrypted result in only two possible values for $\mathbf{e}$: (1) $\mathbf{e} = 0$ and (2) $\mathbf{e} \neq 0$ 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 ($\mathbf{d1}_\mathrm{att}$,$\mathbf{d2}_\mathrm{att}$), the ciphertext

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

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

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

where $\mathbf{n}$ is the noise term $3\mathbf{f} \cdot \mathbf{m}'$. Please note that this noise term $\mathbf{n}$ is different from the noise term of the base ciphertext $\mathbf{c}_{\text{base}}$. 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

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

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

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

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

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

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

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 thus constitutes a distinguisher for every candidate for β_u. An attacker who can realize a PC oracle to extract this binary information about $\mathbf{e}$, can therefore distinguish 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 (ℓ_1, ℓ_2, ℓ_3) that minimize the possibility of a false positive or a false negative in the collision. For distinguishing $\beta_u = 2$, 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 the second largest value be $m_2 < q/2$ and the distance between m_1 (resp. m_2) and $q/2$ be dm_1 (resp. dm_2). The values for

Table 3: Unique distinguishability of every candidate for $\beta_u \in [-2, 2]$ depending on $\mathbf{e} = 0$ (**O**) or $\mathbf{e} \neq 0$ (**X**) for sntrup761. We assume 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**

(ℓ_1, ℓ_2, ℓ_3) should be chosen so as to maximize the distance dm_1 and dm_2, where

$$dm_1 = \|(3\ell_1 \cdot 2m + \ell_2 \cdot 2n + 3\ell_3 \cdot 2) - q/2\| \text{ and}$$

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

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 candidate for β_u. Similar to the tuple (m, n) and (k_1, k_2) in Subsection 3.2, the tuple (ℓ_1, ℓ_2, ℓ_3) can be chosen ahead and fixed for a given parameter set of NTRU Prime.

Table 3 is the decision table for the sntrup761 parameter set. It shows unique distinguishability for every candidate for $\beta_u \in \{-2, -1, 0, 1, 2\}$, based on **O** or **X** for chosen ciphertexts constructed using concrete values for the (ℓ_1, ℓ_2, ℓ_3) assuming a collision with a value of $+2$. The responses for $\beta_u = +1$ (resp. $+2$) can be swapped with $\beta_u = -1$ (resp. -2) if the collision value is -2. Every candidate for $\beta_u = \mathsf{Rotp_R}(\mathbf{f}, u)[i]$ can be *uniquely identified* based on the information about **O** or **X** from only upto *four chosen ciphertext* queries. We note that certain candidates such as $+1$ and $+2$ only require 2 queries to be identified (going from left to right), 0 can be uniquely identified in 3 queries, while -1 and -2 require all 4 queries. Thus, we can adopt such a greedy approach to identify the value of β_u in a more optimized manner.

Table 2 supplies the concrete values of the tuple (ℓ_1, ℓ_2, ℓ_3) and the corresponding distance tuple (dm_1, dm_2), chosen for our attack on different parameter sets of NTRU Prime. We use the notation $(\ell_{x1}, \ell_{x2}, \ell_{x3})$ to denote the tuple used to distinguish $x \in \{1, 2\}$. While these are specific parameters we used for our attack, we would like to emphasize that there are several other values for (ℓ_1, ℓ_2, ℓ_3) which can 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 **O** and **X**. As seen in Subsection 3.1.4, we used the Welch's t-test to identify if $\mathbf{e} \neq 0$ to retrieve the base ciphertext $\mathbf{c}_{\text{base}}$. The peaks in the t-test plot above the pass/fail threshold of ± 5 in Figure 5(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 **O** and **X**. The templates will then be used to classify a given single trace into either of the two classes.

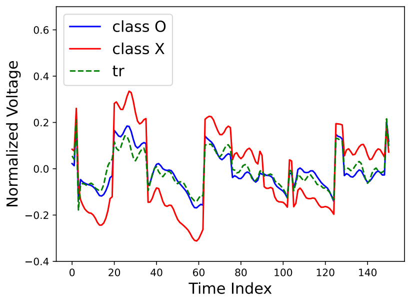
3.2.2 Classification using Reduced Templates

We select features of the t-test plot between $\mathcal{T}_\mathsf{O}$ ($\mathbf{e} = 0$) and $\mathcal{T}_\mathsf{X}$ ($\mathbf{e} \neq 0$) whose *absolute t-test value* is greater than a certain chosen threshold Th_{sel} as our set $\mathcal{P}$ of *Points of Interest* (PoI). A reduced trace set $\mathcal{T}'_\mathsf{O}$ or $\mathcal{T}'_\mathsf{X}$ 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 5, 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.

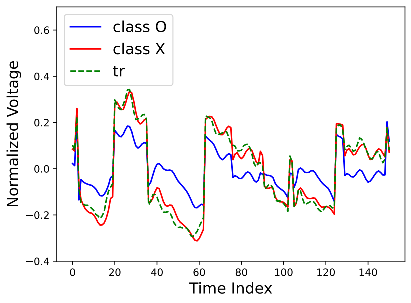
A single trace t for classification is normalized such that $t' = t - \bar{t}$ to obtain a reduced trace $t'_{\mathcal{P}}$. The sum-of-squared difference Γ_* of the trace is computed with each reduced template

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

The trace t falls into the class that corresponds to the least sum-of-squared difference. A single power/EM trace of the targeted operation 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 6 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 6: Matching the reduced template tr of a given attack trace with the reduced template of the two classes $\mathbf{O}$ and $\mathbf{X}$.

3.2.3 Recovering the Full Secret Key

We have thus demonstrated recovery of a single coefficient $\beta_u = \mathsf{Rotp}_{\mathsf{R}}(\mathbf{f}, u)[i]$. By simply 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 knowledge about (1) the colliding index i and (2) the collision value (either $+2$ or -2), both of which cannot be inferred through side-channels 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 $\mathbf{f}' \in R_{\mathrm{sh}}$ 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 of possibilities (≈ 10), upto a certain rotation of $\mathbf{f}$.

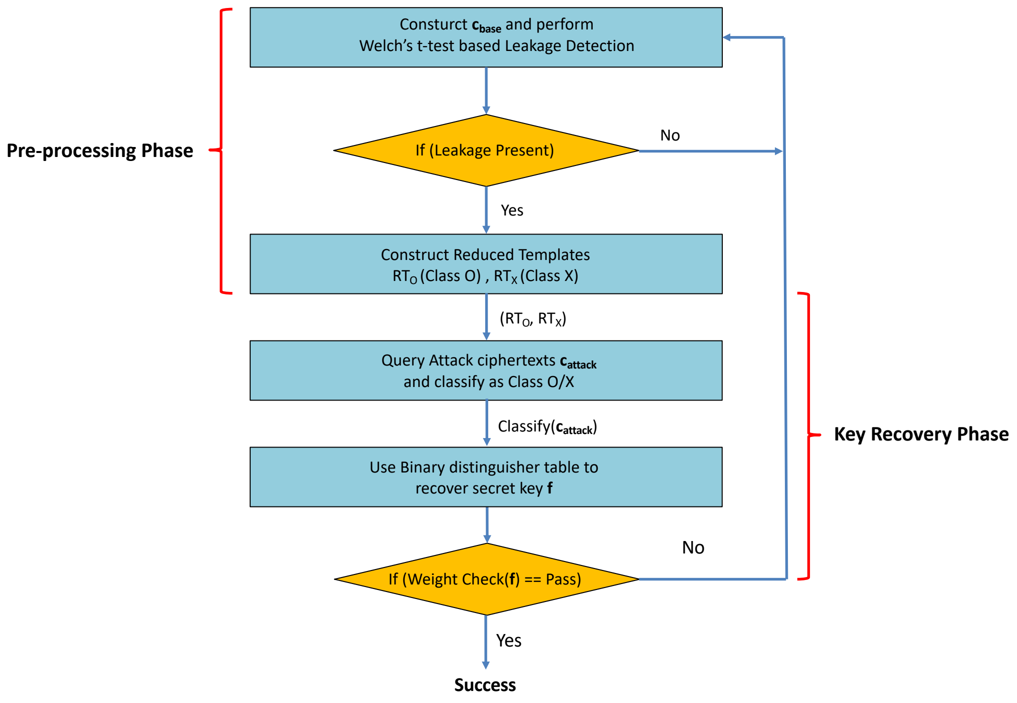
It is possible that none of the guessed $\mathbf{f}'$ turns out to be correct. This could be due to two reasons. Firstly, the rounding noise $\mathbf{n}$ within the attack ciphertexts $\mathbf{c}_{\mathrm{att}}$ could be large enough to induce errors in $\mathbf{e}$, which inturn results in erroneous oracle's responses. Secondly, the chosen base ciphertext $\mathbf{c}_{\mathrm{base}}$ has multiple collisions, which again results in erroneous oracle's responses. In these cases, we simply reject the current $(\mathbf{d1}_{\mathrm{att}}, \mathbf{d2}_{\mathrm{att}})$ and initiate a search for a new pair before repeating the attack until the correct $\mathbf{f}$ is recovered.

We observe that failed iterations of the attack significantly impact the attack's cost, with respect to the number of traces for key recovery. In this respect, we can adopt a few optimization approaches to reduce the impact of failed iterations, particularly in the key recovery phase. If the side-channel oracle's responses do not match the expected responses in the decision table, then the key recovery phase can be immediately aborted, to restart the pre-processing phase for a fresh base ciphertext. Similarly, if the recovered values of β_u for $u \in [0, n-1]$ appear to be very skewed and do not follow the expected distribution, here again the key recovery phase can be immediately aborted, to restart the attack. We summarize the attack flow of our PC oracle-based SCA on NTRU Prime in Figure 7.

3.2.4 Experimental Results

We implemented our proposed PC oracle-based SCA on the optimized implementation of `sntrup761` from the `pqm4` library [KRSS]. The pre-processing phase to identify $\mathbf{c}_{\text{base}}$, took on average, 39 attempts. The number of attempts denoted as A also 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 $A \cdot N \approx 390$ traces to identify $\mathbf{c}_{\text{base}}$, which is denoted as t_{base}. The subsequent attack phase requires up to 4 chosen-ciphertext queries, (i.e.) up to 4 traces, to recover one coefficient. The secret polynomial $\mathbf{f}$ contains $n = 761$ coefficients and we denote the traces required in the attack phase as t_{attack}. Thus, we require $t_{\text{total}} = t_{\text{base}} + t_{\text{attack}} \approx 3269$ traces for complete recovery of $\mathbf{f}$. Our attack works with a success rate of about 100% with no remaining brute force or offline analysis.

We also successfully verified our attack methodology using a simulated PC oracle on other parameter sets of NTRU Prime. Table 4 gives the estimated trace complexity of our attack for different parameter sets of NTRU Prime where the numbers are estimated with $N = 10$ for the pre-processing phase. We can see that 4700 traces is enough for full key



Consturct $\mathbf{c}_{\text{base}}$ and perform Welch's t-test based Leakage Detection
If (Leakage Present)
No
Yes
Construct Reduced Templates RT_O (Class O) , RT_X (Class X)
Pre-processing Phase
(RT_O, RT_X)
Query Attack ciphertexts $\mathbf{c}_{\text{attack}}$ and classify as Class O/X
Classify($\mathbf{c}_{\text{attack}}$)
Use Binary distinguisher table to recover secret key $\mathbf{f}$
Key Recovery Phase
If (Weight Check($\mathbf{f}$) == Pass)
No
Yes
Success

Figure 7: Attack Flow Diagram of our proposed PC Oracle-based SCA on NTRU Prime

Table 4: Trace Complexity of our proposed PC oracle-based SCA on different variants of NTRU Prime. We use t_{base} to denote the number of traces 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}
sntrup653	420	3005	sntrup953	270	3601
sntrup761	390	3269	sntrup1013	320	4026
sntrup857	420	3731	sntrup1277	240	4688

recovery across all parameter sets of NTRU Prime, thereby demonstrating the effectiveness of our attack.

4 PC Oracle-based SCA on NTRU

In this section, we adapt our PC oracle-based SCA on NTRU Prime KEM to NTRU KEM. The notation used is from the IND-CPA secure NTRU PKE described in Algorithm 3. Since our attack applies in the same manner to both NTRU-HPS and NTRU-HRSS, we primarily use NTRU-HPS for description of the attack, but also provide details on those aspects that differ for NTRU-HRSS, wherever necessary.

4.1 Pre-Processing Phase

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

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

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 procedure is given by

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

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 $T = \mathbb{Z}[x]/(x^n - 1)$ 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

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

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

While the above analysis applies for NTRU-HPS, 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 $\{-2, -1, 0, 1, 2\}$. Thus, the absolute maximum value possible for a coefficient of $\mathbf{a}$ is $\mathbf{a}[i] = k_1 \cdot m + 3k_2 \cdot 2n$ and thus Equation (25) can be adapted accordingly.

Table 5: Concrete values for the various parameters used to build chosen ciphertexts for both the Pre-Processing and Key Recovery phase of our PC oracle-based SCA for different variants of NTRU

Scheme	(m, n)	**Pre-Processing 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, 4)	(288, 102)	(273, 93, 141)	(228, 78, 228)
ntruhrss701	(4, 2)	(492, 180)	(483, 162, 243)	(411, 138, 411)

4.1.1 Additional Challenge: Ciphertext Compression

Similar to the use of rounded ciphertexts in NTRU Prime to reduce ciphertext size, NTRU also adopts compression exploiting 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, which can also be seen from the conditional check in line 3 of NTRU_PKE.Decrypt procedure. 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

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

However $\mathbf{c}$ constructed according to Equation 23 is inherently not a multiple of ϕ_1 modulo q. But, it can be adapted to satisfy the requirement in the following manner. Thus, we slightly modify $\mathbf{c}$ 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 $2 \mid m$ (i.e.) m is even and polynomial $\mathbf{d}_1$ has equal number of positive and negative non-zero coefficients (i.e.) $\mathbf{d}_1$ has $m/2$ coefficients with a value of $+1$ and $m/2$ coefficients with a value of -1. This ensures that the sum of coefficients of $\mathbf{c}$ is 0. This is not required for $\mathbf{d}_2$ since $\mathbf{h}$ is already a multiple of ϕ_1, thus the product $\mathbf{d}_2 \cdot \mathbf{h}$ in $\mathbf{c}$ is a multiple of ϕ_1. Thus, ciphertext $\mathbf{c}$ according to 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, which significantly simplifies our attack on NTRU. This applies for both NTRU-HPS as well as NTRU-HRSS variants of NTRU.

Table 5 lists the concrete values of the tuples (m, n) and (k_1, k_2) used to construct chosen ciphertexts for single collision, for different parameter sets of NTRU.

4.1.2 Detecting Collision through Side-Channels

Given chosen tuples (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 is identified through side-channel leakage, in a similar manner to our attack on NTRU Prime. If $\mathbf{e} = 0$, then $\mathbf{m}' = \mathbf{e} \cdot \mathbf{f}_p = 0$ (Line 9 of NTRU_PKE.Decrypt procedure). However, if $\mathbf{e} = \pm x^i$, $\mathbf{m}'$ contains uniformly random coefficients in $\{-1, 0, 1\}$. This large difference in the value of $\mathbf{m}'$ can be easily identified through side-channels, thereby distinguishing between the two classes: (1) $\mathbf{e} = 0$ (Class $\mathbf{O}$) and (2) $\mathbf{e} \neq 0$ with $\mathbf{e}[i] \neq 0$ (Class $\mathbf{X}$).

We performed practical experiments on the ntruhps2048677 parameter set of NTRU. Side-channel measurements were acquired from the same target platform and experimental setup described in Section 3.1.4 and the Welch's t-test based approach was used to identify leakage corresponding to $\mathbf{e} \neq 0$. Figure 8(a) depicts the t-test plots for several ciphertexts $\mathbf{c}'$ whose $\mathbf{e} = 0$. As can be seen, there are no significant peaks about the threshold, which indicates $\mathbf{e} = 0$. However, Figure 8(b) corresponds to $\mathbf{e} \neq 0$ for $\mathbf{c}'$, where we can clearly identify several peaks, well above the threshold, thereby indicating $\mathbf{e} \neq 0$.

The identified ciphertext is denoted as $\mathbf{c}_{\text{base}}$ and its corresponding polynomial tuple $(\mathbf{d}_1, \mathbf{d}_2)$ according to Equation 27 is denoted as $(\mathbf{d1}_{\text{att}}, \mathbf{d2}_{\text{att}})$, which is subsequently used to create 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.

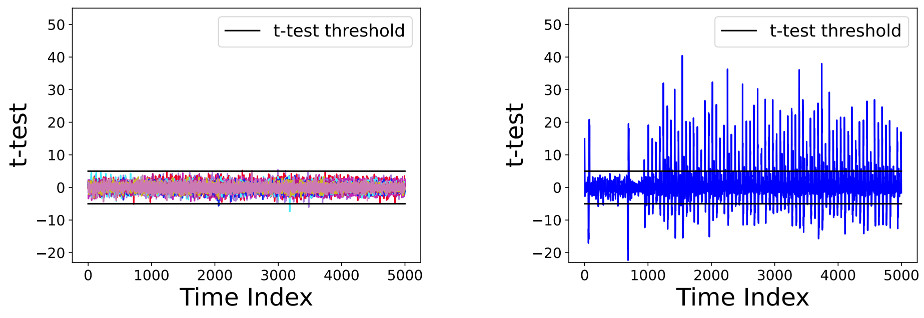


Figure 8: The t-test plots for chosen ciphertexts whose (a) $\mathbf{e} = 0$ and (b) $\mathbf{e} \neq 0$ for ntruhps2048677.

4.2 Key Recovery Phase

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

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

where $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$, $u \in [0, n-1]$. The term $\ell_3 \cdot (x-1) \cdot x^u$ is used to ensure that $\mathbf{c}_{\text{att}}$ is a multiple of ϕ_1 modulo q. Let $\mathsf{Rotp_T}(\mathbf{f}, j)$ denote the product of $\mathbf{f}$ with x^i 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

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

If we denote $3\ell_1 \cdot 2m + 3\ell_2 \cdot 2n$ as a constant δ, then we can represent the coefficient of the colliding index $\mathbf{a}[i]$ as

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

Thus, $\mathbf{a}[i]$ is linearly dependent on $\beta_u = (\mathsf{Rotp_T}(\mathbf{f}, u)[i] - \mathsf{Rotp_T}(\mathbf{f}, u+1)[i])$, which in turn depends upon 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 the occurence of $\mathbf{a}[i] > q/2$ and therefore $\mathbf{e}[i] \neq 0$ (Class $\mathbf{X}$) acts as a binary distinguisher for β_u. Our methodology for choosing (ℓ_1, ℓ_2, ℓ_3) for NTRU resembles our technique used for NTRU Prime described in

Table 6: 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	**O**	**O**	**X**	**X**
-1	**O**	**O**	**X**	**O**
0	**O**	**O**	**O**	**O**
1	**X**	**O**	**O**	**O**
2	**X**	**X**	**O**	**O**

Section 3.2.1, barring the additional constraints placed to deal with the rounding error, as chosen ciphertexts of NTRU are devoid of rounding error.

Table 6 is the decision table for the ntruhps2048677 parameter set, which 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 $\beta_u = (\mathsf{Rotp_T}(\mathbf{f}, u)[i] - \mathsf{Rotp_T}(\mathbf{f}, u + 1)[i])$ can be *uniquely identified* in *no more than four chosen ciphertext queries*. Table 5 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 $x \in \{1, 2\}$.

4.2.1 Classification using Reduced Templates

As shown in Section 3.2.2, side-channel leakage from the decryption of the attack ciphertext $\mathbf{c}_{\mathrm{att}}$ can be used to classify a given ciphertext as either $\mathbf{O}/\mathbf{X}$, thereby realizing a PC oracle. We use the distinguishing features of the t-test plot in Figure 8 to construct reduced templates for both classes $\mathbf{O}$ and $\mathbf{X}$. We then use the templates for classification using a simple LSQ test. Figure 9 visualizes the matching of a section of the attack trace with the reduced templates of the respective classes $\mathbf{O}$ and $\mathbf{X}$. Here again, we are able to observe clear distinguishability between the two classes and we experimentally obtained 100% success rate in classification, thereby demonstrating high accuracy of the realized PC oracle.

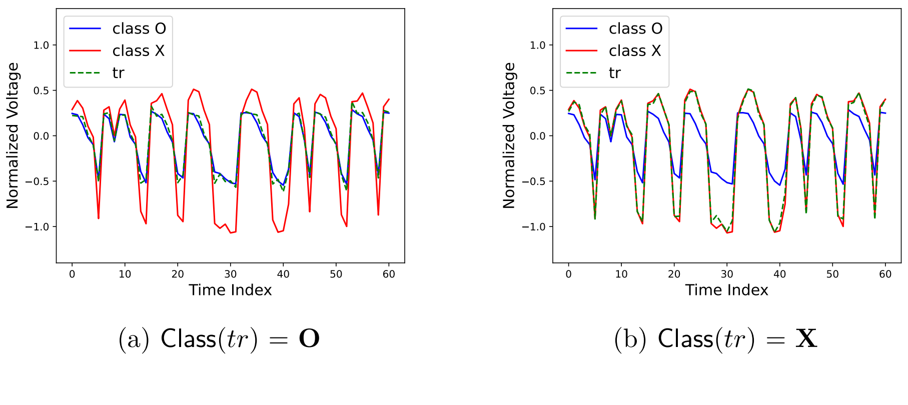


Figure 9: Matching the reduced template tr of a given attack trace with the reduced template of the two classes $\mathbf{O}$ and $\mathbf{X}$.

Table 7: 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}
ntruhps2058509	70	1791	ntruhps4096821	30	2911
ntruhps2058677	100	2364	ntruhrss701	70	2447

4.2.2 Recovering the Full Secret Key

Thus, we can use the realized PC oracle to uniquely recover the value of β_u in upto *four* traces, thereby 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, which can be trivially solved to recover all n coefficients of the $\mathbf{f}$. We can only recover the secret upto a rotation of i indices (i.e.) $\mathbf{f}' = \mathbf{f} \cdot x^i$. The attacker does not know the collision index i, however the multiplication of $\mathbf{f}$ by x^i in the ring T_q does not change the coefficients of $\mathbf{f}$. Moreover, since the decryption involves multiplication and division by $\mathbf{f}$, the rotated secret $\mathbf{f}'$ can also be used to decrypt any message encrypted with the secret polynomial $\mathbf{f}$.

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

4.2.3 Experimental Results

We implemented our attack on the optimized implementation of ntruhps2048677 from the pqm4 library [KRSS]. The pre-processing first step to retrieve $\mathbf{c}_{\text{base}}$, required on average, only 10 attempts. For $N = 10$ replicated measurements, the trace complexity t_{base} of the pre-processing phase is ≈ 100 traces. The subsequent attack phase requires upto 4 chosen-ciphertext queries (4 traces) 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 of about 100%, with no remaining brute force or offline analysis.

We also successfully verified our attack using a simulated PC oracle on the remaining parameter sets of NTRU. Table 7 presents the estimated trace complexity of our attack for different parameter sets of NTRU where the numbers are estimated with $N = 10$ for the pre-processing phase. We can see that 2900 traces is enough for full key recovery across all parameter sets of NTRU Prime, thereby demonstrating the effectiveness of our attack.

As can be seen, the attack complexity of NTRU is lesser than NTRU Prime by a factor of ≈ 1.5 for almost similar dimensions of the secret polynomial $\mathbf{f}$. This can be mainly attributed to absence of rounding noise in NTRU, which simplifies the attack analysis and allows for more relaxed choices of the attack parameters. This reduces the number of attempts to identify the base ciphertext in the pre-processing phase, as well as reduces the number of failed iterations of the attack.

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

We identify a few subtle but critical differences, when comparing our proposed PC oracle-based SCA on NTRU-based schemes compared to 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. As seen from our attacks on NTRU and NTRU Prime, the internal variable $\mathbf{e}$ within the decryption procedure, serves as the anchor variable. However, the underlying arithmetic of LWE/LWR-based schemes is such that, it is possible to exercise direct control over the output of the decryption procedure (i.e.) the decrypted message m through chosen ciphertexts, which serves as the anchor variable for key recovery.

Another differing aspect is the ability to control the value of the anchor variable. While our proposed attacks can restrict $\mathbf{e}$ to two classes $\mathbf{e} = 0$ (Class $\mathbf{O}$) and $\mathbf{e} = \pm 1 \cdot x^i$ (Class $\mathbf{X}$), the value of $\mathbf{e}$ in class $\mathbf{X}$ cannot be controlled. Thus, the pre-processing phase of our attack 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 upon the secret key.

However, for LWE/LWR-based schemes, the two classes are fixed (i.e.) $m = 0$ (Class $\mathbf{O}$) and $m = 1$ (Class $\mathbf{X}$), 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 build 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 for NTRU-based schemes, we do not observe a significant difference in the attacker's cost (trace complexity) to perform full key recovery. For comparison, we utilize experimental results from the work of Ravi *et al.* [RRCB20] who demonstrated PC oracle-based SCA on LWE/LWR-based schemes, using the same target platform and attack setup. Their attack on the Kyber512 parameter set of Kyber required about 7700 traces for full key recovery (dimension $n = 512$ with coefficients in $\{-2, -1, 0, 1, 2\}$). But, this count corresponds to three attack iterations to improve success rate through majority voting. A single attack iteration takes about 2560 traces and thus the trace complexity of our proposed attack is comparable to the attack LWE/LWR-based schemes.

4.3 Limitations of the PC Oracle-Based SCA

Our proposed PC oracle-based SCA can perform full key recovery on all parameter sets of NTRU KEM and NTRU Prime KEM. However, we observe 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. Thus, 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 KEM and we refer to the decryption procedure of NTRU Prime (i.e.) NTRU_Prime_PKE.Decrypt procedure in Algorithm 1. The attack ciphertexts result in $\mathbf{e} = 0/\mathbf{e} = \pm 1 \cdot x^i$. 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 upon the secret polynomial $\mathbf{g}$. However, 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 the attack ciphertexts.

The effect of the anchor variable $\mathbf{e}$ for the attack ciphertexts, does not propagate beyond the decryption procedure. Thus, 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 following 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 within the decapsulation procedure, which aids in key recovery.

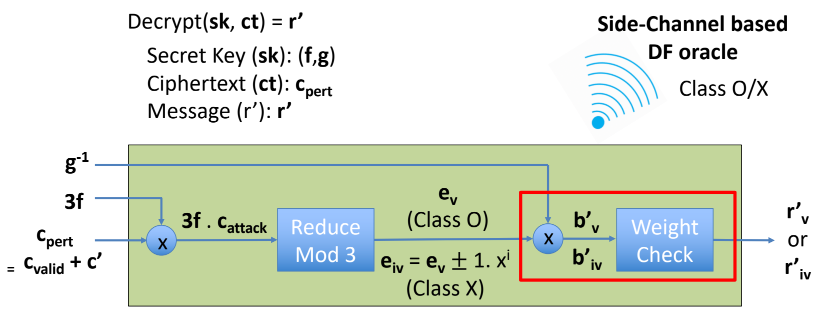


Figure 10: Pictorial Illustration of our DF oracle-based SCA on NTRU Prime. The subscript `v` denotes `valid` whereas the subscript `iv` denotes `invalid`.

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 for the PC oracle-based attack.

Let $\mathbf{c}_{\text{valid}}$ be a valid ciphertext whose anchor variable $\mathbf{e}$ is denoted as $\mathbf{e}_{\text{valid}}$. 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 simply 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 (i.e.) $\mathbf{e}_{\text{pert}} = \mathbf{e}_{\text{valid}} \pm 1 \cdot x^i$, 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 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. In effect, we have ensured that the effect of the anchor variable $\mathbf{e}$ propagates to the decrypted message $\mathbf{r}'$, while 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 10 illustrates the attack targeting leakage from the decryption procedure. But, 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 pre-processing phase and the key recovery phase.

5.1 Pre-Processing Phase

As in Line 3 of NTRU_PRIME_PKE.Encrypt procedure 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 Section 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 now 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 $(k_1 \cdot \mathbf{d}_1 \cdot 3\mathbf{f} + k_2 \cdot \mathbf{d}_2 \cdot \mathbf{g})$ be the signal component $\mathbf{s}$ 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 corresponding to the perturbed ciphertext $\mathbf{c}_{\text{pert}}$ (i.e.) $\mathbf{a}_{\text{pert}}$ and $\mathbf{e}_{\text{pert}}$ as $\mathbf{a}$ 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 we require a slightly different approach to construct chosen ciphertexts.

5.1.1 Additional Challenge: Dealing with the Bias gr

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 $\mathbf{gr}$ whose absolute value is high (both positive and negative 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 over key recovery as it decreases the chance of a false negative for decryption failure in the key recovery phase.

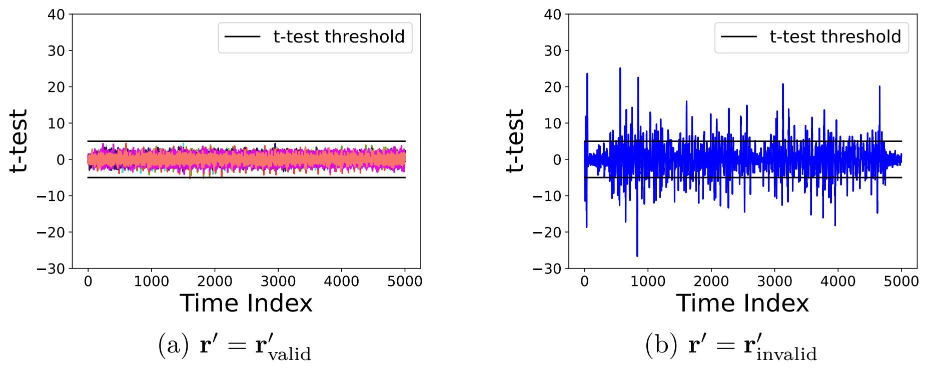
With m_1 chosen to be $< q/2$ by $\epsilon_2 < dm_1 < \epsilon_1$, there is a leeway to increase dm_2. This reduces the chances of false positives at the other indices $j \neq i$ where no collisions occur. Thus, our modified constraints for choosing (k_1, k_2) according to Equation (36) offer several advantages in reducing the false positives as well as false negatives for decryption failures, which aids key recovery.

We choose (m, n) and (k_1, k_2) based on the aforementioned constraints to identify ciphertexts with a single collision. Table 8 presents the concrete values chosen for our attack on all parameter sets of specified NTRU Prime. 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 $\mathbf{e}$ (i.e.) $\mathbf{e}_{\text{invalid}} = \mathbf{e}_{\text{valid}} \pm x^i$. 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 either by obtaining information about the anchor variable $\mathbf{e}'$ within the decryption procedure or the decrypted message $\mathbf{r}'$ used in the re-encryption procedure, through side-channels. 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.

We performed practical experiments on sntrup761. Measurements were acquired from the same target platform and experimental setup used to perform the PC oracle-based attack. 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 $\mathbf{r}'$.



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

Figure 11: The t-test plots used to identify decryption failure for sntrup761.

We used the Welch's t-test described in Section 3.1.4 to identify leakage that differentiates $\mathbf{r}'_{\text{invalid}}$ (decryption failure) from $\mathbf{r}'_{\text{valid}}$ (decryption success). Figure 11(a) depicts the t-test plot for several perturbed ciphertexts $\mathbf{c}_{\text{pert}}$ whose decryption does not fail (i.e.) no error. One sees no significant peaks about the threshold, which indicates $\mathbf{r}' = \mathbf{r}'_{\text{valid}}$. Figure 11(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 denoted as $\mathbf{c}_{\text{base}}$ and its corresponding polynomials $\mathbf{d}_1$ and $\mathbf{d}_2$ are denoted by $\mathbf{d1}_{\text{att}}$ and $\mathbf{d2}_{\text{att}}$, with m and n terms, respectively.

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 to build new perturbation ciphertexts for key recovery very closely resembles the one used for 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 perturbed/invalid ciphertext $\mathbf{c}_{\text{pert}}$ in the same way as done 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 the coefficients of $\mathbf{a}$ as

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

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,

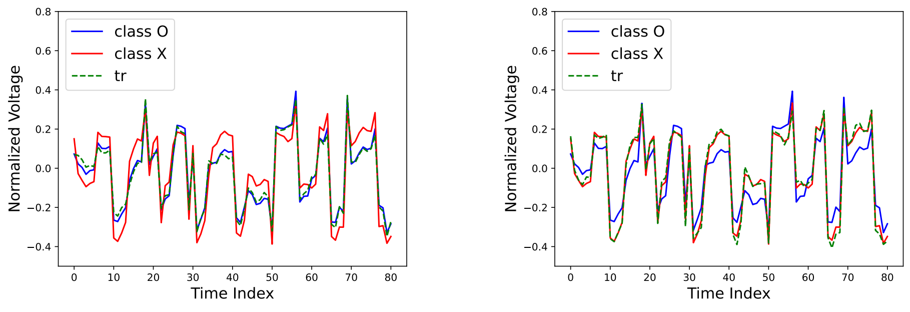
Table 8: Concrete values for the various parameters used to build chosen ciphertexts for both the Pre-Processing and Key Recovery phase of our DF Oracle-based SCA for different variants of NTRU Prime

Scheme	(m, n)	**Pre-Processing 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)

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, $\mathbf{e} = \mathbf{e}_{\text{valid}}$ (Class $\mathbf{O}$) or $\mathbf{e} = \mathbf{e}_{\text{invalid}}$ (Class $\mathbf{X}$) can act as a binary distinguisher for every candidate of $\beta_u \in [-2, 2]$. 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. Thus, we arrive at the same values which were used for the PC oracle-based SCA, as can be seen in Table 8. Thus, the decision table for unique distinguishability also stays the same (cf. Table 3 in Section 3.2.1).

5.3.2 Classification using Reduced Templates

We utilize the differentiating features in the t-test plot shown in Figure 11(b) to build reduced templates for both the 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 of the classes. This was treated earlier in Section 3.2.2. Figure 12 visualizes the matching of a small section of an attack trace tr with the reduced templates of the respective classes $\mathbf{O}$ and $\mathbf{X}$. There is a clear distinguishability between the reduced templates of the two classes. This enables us to correctly classify each given single trace with a 100% success rate.



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

Figure 12: Matching the reduced template tr of a given attack trace with the reduced template of the two 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 $\mathsf{Rotp}_\mathsf{R}(\mathbf{f}, u)$. Similarly, by changing the rotation index u, we can recover $\mathsf{Rotp}(\mathbf{f}, u)[i]$ for all $u \in [0, n-1]$. Just in line with the PC oracle attack, recovering the exact secret polynomial $\mathbf{f}$ requires knowing the colliding index i and the value ($+2$ or -2) of the collision. By simply trying out all possible choices for $i \in [0, n-1]$ and the colliding values $+2$ and -2, we check, for each choice, if $\mathbf{f}' \in R_{\text{sh}}$ 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 the base ciphertext $\mathbf{c}_{\text{base}}$. In this case, we simply retry the attack until the correct key is recovered.

Table 9: Trace Complexity of our proposed DF oracle-based SCA on different variants of NTRU Prime. We use t_{base} to denote the number of traces 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}
sntrup653	163	4182	sntrup953	76	4436
sntrup761	165	4566	sntrup1013	74	4603
sntrup857	120	4631	sntrup1277	41	5287

5.3.4 Experimental Results

We ran our attack on the optimized implementation of sntrup761 taken from the pqm4 library [KRSS] on the ARM Cortex-M4 microcontroller. The pre-processing phase to retrieve $\mathbf{c}_{\text{base}}$ requires an average of ≈ 165 attempts. For $N = 10$, the number of required total attempts goes to ≈ 1650 traces (t_{base}). The number is almost $2.85\times$ higher compared with the PC oracle-based attack. The latter only requires ≈ 58 attempts. The increase is partly due to the additional constraints to deal with the constant bias $\mathbf{gr}$ (Section 5.1.1). The subsequent attack phase can recover a single coefficient in up to 4 ciphertexts and, thus, the complete attack requires $t_{\text{total}} = t_{\text{base}} + t_{\text{attack}} \approx 4566$ traces for complete recovery of $\mathbf{f}$. Our attack works with a success rate of about 100% with no additional brute force or offline analysis to perform.

We also successfully verified our attack methodology using a simulated DF oracle on all the parameter sets of NTRU Prime. Table 9 gives the attack's estimated trace complexity for different schemes. The numbers are estimated with $N = 10$ for the pre-processing phase. We can see that 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 roughly 1.2 to 1.4 times the numbers for the PC oracle-based attack. This increase can be attributed mainly to the longer pre-processing 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 single 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 result in a decryption failure is linearly dependent on the secret key. This information, if could be obtained by a DF oracle, led to full key recovery. For LWE/LWR-based schemes it is notable that the location of the perturbed bit in the decrypted message can be precisely controlled.

Although the underlying arithmetic is vastly different for a direct comparison, we identify a few subtle differences when compared to our proposed DF oracle-based SCA on the NTRU-based schemes. Our approach does not allow us to control the location of the perturbed bit of the decrypted message. The more important but subtle difference lies in the type of error used for perturbation. In our attack, 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 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]. It 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 about 2^{30} decapsulation queries for full key recovery in $\mathsf{Frodo}-\mathsf{KEM}-\mathsf{1344}-\mathsf{AES}$ parameter set. 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 proposed attack on NTRU-based schemes requirer much less number of traces, in the range of 4500 to 7000 traces, 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 choice of the chosen ciphertexts for NTRU Prime and NTRU KEM to perform full key recovery. However, the PC and DF oracles only provide binary information (1-bit) about the secret-dependent anchor variable, and therefore require a few thousand chosen ciphertext queries to the target device. However, a more powerful side-channel adversary who can extract more that just a single-bit information about the anchor variable, can potentially 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 those routines which manipulate sensitive variables such as the decrypted message, 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 the decryption procedure (Line 10 of NTRU_PKE.Decrypt procedure) is susceptible to side-channel attacks. They demonstrated successful single-trace message recovery with close to a 100% success rate.

Though they do not demonstrate message recovery for NTRU Prime, we speculate that the weight check operation on the variable $\mathbf{b}'$ (Line 6 of NTRU_Prime_PKE.Decrypt procedure) could also be susceptible to similar single-trace attacks, especially because it involves manipulation of single coefficients of $\mathbf{b}'$. The feasibility of performing single trace recovery of $\mathbf{b}'$ remains out of scope of this work.

However, the aforementioned side-channel vulnerabilities can potentially be exploited to recover the complete decrypted message $\mathbf{m}'$ in case of NTRU, or the variable $\mathbf{b}'$ in case of 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, and subsequently on NTRU KEM.

6.1 Attack Methodology: NTRU Prime KEM

The attack methodology directly follows from our PC oracle-based SCA on NTRU Prime KEM (cf. Section 3). We conduct the pre-processing phase to retrieve the base ciphertext $\mathbf{c}_{\text{base}}$ 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 procedure, we know that

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

where $\hat{\mathbf{g}}$ is the inverse of the secret polynomial $\mathbf{g}$ in R_3. Since $\mathbf{e} = \pm x^i$, $\mathbf{g}$ can be directly recovered as $\mathbf{g} = \mathbf{e} \cdot \hat{\mathbf{b}'} \in R_3$ 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 $i \in [0, n-1]$ and recover the secret polynomials $\mathbf{f}$ and $\mathbf{g}$ upto a rotation.

6.2 Attack Methodology: NTRU KEM

The attack follows the pre-processing 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 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 $\mathbf{e} = \pm x^i$, $\mathbf{f}_p$ can be simply computed as $\mathbf{f}_p = \hat{\mathbf{b}}' \cdot \hat{\mathbf{e}} \in S_3$ 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 $\mathbf{f}_p$ and thereby calculate the secret key polynomials $\mathbf{f}$ and $\mathbf{g}$.

Unlike the PC oracle or DF oracle-based attacks, the attacker can perform full key recovery only using the base ciphertext $\mathbf{c}_{\text{base}}$ for both NTRU and NTRU Prime KEM, which completely eliminates the need for key recovery phase. Thus, the trace requirement of the FD oracle-based SCA primarily comes from the pre-processing phase of the attack. Please refer to the column corresponding to t_{base} in Tables 4 and 7 for the estimated trace complexity of the FD oracle-based SCA on different parameter sets of NTRU Prime KEM and NTRU KEM respectively.

7 Countermeasures

Our proposed side-channel assisted CCAs rely on fixing targeted intermediate variables to known values and, subsequently, utilizing side-channel leakage to identify its value to perform key recovery. Thus, a complete randomization of the internal computation through masking, can serve as a concrete countermeasure against the attacks. Let us briefly address the countermeasures for the NTRU Prime KEM and the NTRU KEM separately.

In the case of 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 PC oracle-based SCA. 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 the case of 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 therefore 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.

Masking countermeasures, in general, are known to be costly in terms of performance. 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 procedure of NTRU Prime in Algorithm 1 and in Lines 7 and 9 of NTRU_PKE.Decrypt procedure of NTRU in Algorithm 3.

Our attacks have shown that other operations within the decryption and decapsulation procedure 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 in the NTRU_Prime_PKE.Decrypt procedure 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.

8 Conclusion

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 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 oracle, decryption-failure oracle, and full-decryption oracle. We perform experimental validation of our proposed attacks on optimized implementations of NTRU-based schemes, 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 to the target device on all parameter sets of NTRU and NTRU Prime. Our attacks stress on the need for concrete masking strategies for NTRU-based KEMs to protect against side-channel assisted CCAs.

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.

[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. *Advances in Cryptology–EUROCRYPT 2012*, pages 719–737, 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] A 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.

[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.

[ZCQD21] Xiaohan Zhang, Chi Cheng, Yue Qin, and Ruoyu Ding. Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. *IACR Cryptol. ePrint Arch.*, 2021:168, 2021.