UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

Reveal the invisible secret

Xu, Zhuang; Pemberton, Owen; Oswald, David; Zheng, Zhiming

DOI:
10.1007/978-3-031-25319-5_12

License:
Other (please specify with Rights Statement)

Document Version
Peer reviewed version

Citation for published version (Harvard):
Xu, Z, Pemberton, O, Oswald, D & Zheng, Z 2023, Reveal the invisible secret: chosen-ciphertext side-channel attacks on NTRU. in *International Conference on Smart Card Research and Advanced Applications: CARDIS 2022: Smart Card Research and Advanced Applications.* Lecture Notes in Computer Science, vol. 13820, Springer, pp. 227–247, 21st Smart Card Research and Advanced Application Conference, Birmingham, United Kingdom, 7/11/22. https://doi.org/10.1007/978-3-031-25319-5_12

Link to publication on Research at Birmingham portal

Publisher Rights Statement:
This version of the contribution has been accepted for publication, after peer review (when applicable) but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-031-25319-5_12. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

General rights
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 12. Jan. 2024

Reveal the Invisible Secret: Chosen-Ciphertext Side-Channel Attacks on NTRU

Zhuang Xu[1], Owen Pemberton[2], David Oswald[2], and Zhiming Zheng[3,4,5]

[1] School of Mathematical Sciences and Shenyuan Honors College, Beihang University, Beijing, China, xu_zhuang@buaa.edu.cn
[2] School of Computer Science, University of Birmingham, Birmingham, U.K. o.m.pemberton@pgr.bham.ac.uk, d.f.oswald@bham.ac.uk
[3] Institute of Artificial Intelligence, LMIB, NLSDE and Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, China
[4] Peng Cheng Laboratory, Shenzhen, China
[5] Institute of Medical Artificial Intelligence, Binzhou Medical University, Yantai, China, zzheng@pku.edu.cn

Abstract. NTRU is a well-known lattice-based cryptosystem that has been selected as one of the four key encapsulation mechanism finalists in Round 3 of NIST's post-quantum cryptography standardization. This paper presents two succinct and efficient chosen-ciphertext side-channel attacks on the latest variants of NTRU, i.e., NTRU-HPS and NTRU-HRSS as in Round 3 submissions. Both methods utilize the leakage from the polynomial modular reduction to recover the long-term secret key. For the first attack, although the side-channel leakage does not directly reveal the secret polynomial $\mathbf{f}$, we recover differences between adjacent coefficients using appropriately chosen ciphertexts, and finally reconstruct $\mathbf{f}$ through linear algebra. The second attack is based on the inherent relation between the secret key and the public key in NTRU-HPS: we first reveal the "invisible" secret polynomial $\mathbf{g}$ with chosen ciphertexts and then use $\mathbf{g}$ and the public polynomial $\mathbf{h}$ to compute $\mathbf{f}$. In theory, these attacks only need 4 and 2 ciphertexts, respectively. We then practically apply those attacks on all reference implementations of four instances in the `PQClean` library and show that the accuracy of secret-key recovery can reach 100% with only few traces (4 to 24 and 2 to 6, respectively). We also observe similar leakage in optimized implementations in the `pqm4` library and propose an according analysis scheme.

Keywords: Lattice-based cryptography · NTRU · Chosen-ciphertext attack · Side-channel analysis.

1 Introduction

When will a practical quantum computer that can break current Public-Key Cryptography (PKC) based on RSA or elliptic curve cryptography be available? Quantum computing experts anticipate that it might take 10 to 15 years. But one thing is for sure: we need cryptographic algorithms that are secure even in

the context of quantum computing to replace current PKC schemes. This kind of cryptography is also known as Post-Quantum Cryptography (PQC).

There are several proposals for PQC schemes relying on lattice-based, code-based, hash-based, isogeny-based and multivariate polynomial-based problems. To select sound successors, the National Institute of Standards and Technology (NIST) held a public competition-like PQC standardization project in 2016 [2]. After two rounds of competition, four Key Encapsulation Mechanisms (KEMs) and three signature schemes have been chosen as finalists in Round 3 [1]. Out of the four KEMs, three are lattice-based schemes: CRYSTALS–Kyber [7], Saber [9] and NTRU [8]. The security of the first two are based on the hardness of Learning with Errors (LWE) problem variants, i.e., the Module-LWE problem and the Module Learning with Rounding problem; while that of NTRU is based on the Shortest Vector Problem, a different assumption. As the one with the longest history among them, NTRU has stood the test of time, i.e., NTRU-based schemes had been widely studied in theoretical and implementation security. Therefore, it is a very competitive finalist. Recently, NIST published the report on the third round. Although currently NTRU is not considered for future standardization as per the evaluation and selection results, NIST stated that NTRU may replace Kyber as a standard if the patent issue on Kyber is not resolved by the end of 2022 [3]. So, it is still significant to study the latest variants of NTRU in depth.

Theoretical security, performance and characteristics (in terms of algorithm and implementation) are core evaluation criteria in the first two rounds [1]. The implementation security of finalists is also a key criterion in Round 3. A deeper understanding of the threats to the implementation can help designers create more effective countermeasures to mitigate such attacks. In this paper, we focus on Side-Channel Analysis (SCA), a classic threat exposing the relation between secret intermediate values and side-channel leakage when the cryptographic algorithm runs on a specific hardware/software platform. More specifically, we focus on Simple Power Analysis (SPA) [18] using chosen ciphertexts for the NTRU KEM. To our knowledge, this is the first attempt to apply this kind of analysis on NTRU instances in Round 3.

1.1 Related Work

SCA on NTRU. There are many side-channel attacks [4,6,19,29] targeting the early version of NTRU. Most attacks exploit the leakage from the polynomial multiplication which involves the secret polynomial and a controllable/known polynomial. But these attacks cannot be directly applied to the latest NTRU instances that use Toom-Cook and Number Theoretic Transform (NTT) techniques to optimize the multiplication process in the `PQClean` and `pqm4` [16] library respectively, instead of "multiply and accumulate" approach in early NTRU implementations. Recently, Mujdei *et al.* [20] exploited the leakage from polynomial multiplication through Toom-Cook and NTT strategies and mounted successful correlation power analysis.

In addition to multiplication, other operations have also been targeted to mount SCA. In [17], Karabulut *et al.* showed the vulnerability of the sorting

algorithm in the key generation step of NTRU. Askeland *et al.* [5] performed a partial key recovery by utilizing the leakage from the unpacking operation and the $\mathbb{Z}_3$ to $\mathbb{Z}_q$ mapping in a single-trace setting. They then applied the Block Korkin-Zolotarev (BKZ) lattice reduction algorithm to find the remainder of the secret key. Ueno *et al.* [26] presented a deep-learning-based SCA on the Fujisaki-Okamoto transformation and its variants and demonstrated that the scheme can be applied to most candidates in Round 3 (including NTRU). In [22], Ravi *et al.* proposed several chosen-ciphertext collision attacks on the NTRU family of KEMs through side-channel-based oracles. Their method is inspired by a Chosen-Ciphertext Attack (CCA) [15] on early versions of NTRU.

There are also attacks which target the session key instead of the long-term secret key. For example, Sim *et al.* [24] utilized the leakage from message encoding in encapsulation to recover the secret shared message which can be used to calculate the shared session key.

Chosen-Ciphertext SPA on Other Lattice-Based Schemes. As lattice-based KEMs share a common characteristic—the coefficients of the secret polynomial are sampled from a small value range (e.g., $\{-1, 0, 1\}$ in NTRU)—they are vulnerable under chosen-ciphertext-assisted SPA which can take advantage of this feature to recover the secret key using a small number of traces. This kind of attack becomes more practical when an attacker cannot collect enough traces to mount Differential Power Analysis (DPA) [18] or template attacks.

Park *et al.* [21] proposed the first chosen-ciphertext SPA targeting Ring-LWE encryption on an 8-bit microcontroller. In [13], several power analysis techniques were applied to NTRU Prime by Huang *et al.* One of them is a chosen-input SPA targeting the polynomial multiplication. With the assistance of chosen ciphertexts, an adversary can recover the key according to special patterns in the trace. Xu *et al.* [27] devised an efficient SPA on Montgomery reduction in the reference implementation of Kyber. With four traces from different crafted ciphertexts, they achieved a high accuracy of secret-key recovery.

1.2 Contributions

In this paper, we explore the feasibility of chosen-ciphertext SCA on all NTRU instances in Round 3. We also determine a lower bound of the required number of traces, exploiting a unique property in three NTRU-HPS instances.

Concretely, we extend the chosen-ciphertext SPA method from [27] to Round-3 NTRU. Interestingly, the relation between the public key and the secret key in NTRU-HPS simplifies this kind of attack. Our main contributions are:

1) We identify a leaky function, the $\mathcal{R}_q$ to S_3 polynomial modular reduction in decryption in the decapsulation phase, which leaks the Hamming Weight (HW) of a secret-dependent intermediate value.
2) We mount chosen-ciphertext Electro-Magnetic (EM) SCA on the reference implementations of all instances (i.e., `ntruhps2048509`, `ntruhps2048677`, `ntruhps4096821` and `ntruhrss701`) in `PQClean`, and recover the core secret polynomial $\mathbf{f}$ successfully with 4 ciphertexts supported by 4 to 24 traces.

3) We propose a more efficient attack utilizing an inherent property of NTRU-HPS—for that, we first recover the "invisible" secret polynomial $\mathbf{g}$ with 2 ciphertexts (2 to 6 traces) and then compute $\mathbf{f}$ from $\mathbf{h}$ and $\mathbf{g}$.
4) We discuss the extension of our analysis to implementations from the optimized `pqm4` library.

Compared to several previous SCA schemes on NTRU [20,22,26], our schemes need less queries from the victim device. A stronger leakage point (compared to [5]) is exploited, so no additional lattice reduction is required.

1.3 Organization

The remainder of this paper is structured as follows: in Sect. 2, we introduce the necessary notations, background, and adversary model. In Sect. 3, we present our two types of chosen-ciphertext SPA on reference implementations of NTRU. Next, we propose an analysis strategy on optimized implementations in Sect. 4. Finally, we conclude in Sect. 5.

2 Preliminaries

2.1 Notation

Let q be a positive integer. Centered modular reduction is represented as $r' = r \mod q$ where $r' \in [-q/2, q/2-1]$ when q is even or $r' \in [-(q-1)/2, (q-1)/2]$ when q is odd. The non-negative modular reduction $r' = r \mod^+ q$ means $r' \in [0, q-1]$. Here, $\mathbb{Z}_q$ is the ring of integers modulo q with centered modular reduction. A polynomial is represented by a bold-font lowercase letter, e.g., $\mathbf{a}$, its i-th coefficient is represented as a_i, and then $\mathbf{a} = \sum_i a_i x^i$. Polynomial multiplication is denoted using the $\cdot$ operator, whereas for scalar multiplication, the operator is omitted. We define $\mathbf{\Phi}_1 = x-1$, $\mathbf{\Phi}_n = (x^n-1)/(x-1)$, for some integer n. The quotient rings of polynomials are $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n-1) = \mathbb{Z}[x]/(q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$, $S_q := \mathbb{Z}_q[x]/(\mathbf{\Phi}_n)$. The polynomial multiplication of $\mathbf{a}$ and $\mathbf{b}$ in the above rings is represented by $\mathbf{a} \cdot \mathbf{b} \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$ and $\mathbf{a} \cdot \mathbf{b} \bmod (q, \mathbf{\Phi}_n)$, respectively. $\mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n = 0$ (or $\mathbf{\Phi}_n = 0$) is applied to terms which have a degree greater than or equal to n (or $n-1$). We refer to the side-channel measurement (i.e., trace) as $p(t)$, where t is discretized time in sample points.

2.2 NTRU in NIST PQC Round 3

NTRU is one of the most-studied lattice-based schemes. Several variants have been proposed since it was first proposed by Hoffstein in 1996 [11]. As our targets are the latest variants that have advanced to Round 3 in the NIST PQC standardization, we refer the reader to [23] for more information about the comparisons among different variants.

The submission of NTRU KEM in Round 3 is a merge of NTRU-HPS [12] and NTRU-HRSS [14]. Here, we briefly describe the core algorithms; for a full description, please refer to [8].

Algorithm 1 NTRU.PKE.KeyGen $(seed)$ [8]

1: $(\mathbf{f}, \mathbf{g}) = \text{Sample\_fg}(seed)$
2: $\mathbf{f}_p = (1/\mathbf{f}) \bmod (p, \mathbf{\Phi}_n)$ /* $p = 3$ */
3: $\mathbf{f}_q = (1/\mathbf{f}) \bmod (q, \mathbf{\Phi}_n)$
4: $\mathbf{h} = (3\mathbf{g} \cdot \mathbf{f}_q) \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$/*HPS*/ or $(3\mathbf{g} \cdot \mathbf{\Phi}_1 \cdot \mathbf{f}_q) \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$/*HRSS*/
5: $\mathbf{h}_q = (1/\mathbf{h}) \bmod (q, \mathbf{\Phi}_n)$
6: $sk = \text{Pack\_S3}(\mathbf{f}) \parallel \text{Pack\_S3}(\mathbf{f}_p) \parallel \text{Pack\_Sq}(\mathbf{h}_q)$
7: $pk = \text{Pack\_Rq0}(\mathbf{h})$
8: **return** (sk, pk)

Algorithm 2 NTRU.PKE.Enc $(pk, packed\_rm)$ [8]

1: $(\mathbf{r}, \mathbf{m}) = \text{Unpack\_S3}(packed\_rm)$
2: $\mathbf{m}' = \text{Lift}(\mathbf{m})$
3: $\mathbf{h} = \text{Unpack\_Rq0}(pk)$
4: $\mathbf{c} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}') \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$
5: **return** $ct = \text{Pack\_Rq0}(\mathbf{c})$

Algorithm 3 NTRU.PKE.Dec (sk, ct) [8]

1: $\mathbf{c} = \text{Unpack\_Rq0}(ct)$
2: $(\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q) = \text{Unpack}(sk)$
3: $\mathbf{v} = (\mathbf{c} \cdot \mathbf{f}) \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$ /* Target step in this paper */
4: $\mathbf{m} = (\mathbf{v} \cdot \mathbf{f}_p) \bmod (p, \mathbf{\Phi}_n)$ /* $p = 3$ */
5: $\mathbf{m}' = \text{Lift}(\mathbf{m})$
6: $\mathbf{r} = ((\mathbf{c} - \mathbf{m}') \cdot \mathbf{h}_q) \bmod (q, \mathbf{\Phi}_n)$
7: $packed\_rm = \text{Pack\_S3}(\mathbf{r}) \parallel \text{Pack\_S3}(\mathbf{m})$
8: **if** $(\mathbf{r}, \mathbf{m}) \in \mathcal{L}_r \times \mathcal{L}_m$, $fail = 0$
9: **else** $fail = 1$
10: **return** $(packed\_rm, fail)$

The NTRU KEM with Indistinguishability under Chosen-Ciphertext Attack (IND-CCA) is built by a generic transformation from a deterministic NTRU Public-Key Encryption (PKE) scheme. The deterministic PKE consists of Algorithms 1 to 3. All elements and corresponding computations are in three rings: $\mathcal{R}_q$, S_q and S_p, where q varies for different instances but $p = 3$ for all schemes. The specific parameters are shown in Table 1. $\mathbf{f}$, $\mathbf{g}$, $\mathbf{r}$ and $\mathbf{m}$ are polynomials in S_3, i.e., ternary polynomials of degree at most $n - 2$. The specific sample spaces are different: in NTRU-HPS, $\mathbf{g}$ and $\mathbf{m}$ have exactly $(q/16 - 1)$ coefficients equal

Table 1. Parameters for different NTRU instances

NTRU	NTRU-HPS			NTRU-HRSS
	`ntruhps2048509`	`ntruhps2048677`	`ntruhps4096821`	`ntruhrss701`
n	509	677	821	701
q	2048	2048	4096	8192

Algorithm 4 NTRU.KEM.KeyGen $(seed)$ [8]

1: $(sk, pk) = \text{NTRU.PKE.KeyGen}(seed)$
2: $s \leftarrow_{\$} \{0, 1\}^{256}$
3: **return** $((sk, s), pk)$

Algorithm 5 NTRU.KEM.Encaps (pk) [8]

1: $coins \leftarrow_{\$} \{0, 1\}^{256}$
2: $(\mathbf{r}, \mathbf{m}) = \text{Sample\_rm}(coins)$
3: $packed\_rm = \text{Pack\_S3}(\mathbf{r}) \parallel \text{Pack\_S3}(\mathbf{m})$
4: $ct = \text{NTRU.PKE.Enc}(pk, packed\_rm)$
5: $K = H_1(packed\_rm)$ /* Shared Key */
6: **return** (ct, K)

Algorithm 6 NTRU.KEM.Decaps $((sk, s), ct)$ [8]

1: $(packed\_rm, fail) = \text{NTRU.PKE.Dec}(sk, ct)$
2: $K_1 = H_1(packed\_rm), K_2 = H_2(s, ct)$
3: **if** $fail = 0$ **return** K_1 /* Shared Key */
4: **else return** K_2 /* Random Key */

to 1 and $(q/16-1)$ coefficients equal to -1; in NTRU-HRSS, $\mathbf{f}$ and $\mathbf{g}$ satisfy the non-negative correlation property [14]. As the polynomial $\mathbf{g}$ only appears in the key generation step and is not a component of the secret key sk, we call it the "invisible" secret polynomial in this paper. The transformations between polynomial and bytes are implemented by the Pack and Unpack functions. $\text{Lift}(\mathbf{m}) = \mathbf{m}$ for NTRU-HPS, but equal to $\mathbf{\Phi}_1 \cdot (\mathbf{m}/\mathbf{\Phi}_1 \bmod (3, \mathbf{\Phi}_n))$ in NTRU-HRSS.

The main steps of the IND-CCA NTRU KEM are shown in Algorithms 4 to 6. A successful key exchange will let two communication parties share a same session key, otherwise a random key will be generated. s and $coins$ are strings of uniform random bits. H_1 and H_2 are hash functions.

2.3 Threat Model

Target. The adversary's target is to obtain the secret key sk. Although it is recommended that applications do not reuse the secret key, in many scenarios (e.g., resource-constrained embedded devices), sk will be reused or stored locally. Once the long-term secret key is leaked, the attacker can intercept KEM session involving the victim and other party and then recover the shared session key. Moreover, the attacker can impersonate the victim. The secret key of NTRU consists of three parts: $\mathbf{f}$, $\mathbf{f}_p$ and $\mathbf{h}_q$, where $\mathbf{h}_q$ can be computed using the public key. If one can recover $\mathbf{f}$, the other components can be computed accordingly. Therefore, the ternary polynomial $\mathbf{f}$ in the decryption phase during decapsulation becomes the core target of our analysis.

Attacker Capabilities. We assume a standard side-channel attacker who can initiate a KEM process with the victim, and further:

1) Encapsulate maliciously chosen ciphertexts and send them to the victim;
2) Has physical access to EM traces collected from the victim device while it executes the decapsulation algorithm.

Experimental Setup. For all experiments, we compile target implementations using ARM GCC version `10.3-2021.10` with the default (maximum) optimization `-O3` and run on an STM32F407G discovery development board. The STM32 chip runs at a clock frequency of 168 MHz. We made minimal modifications to the source code to perform our experiments: we raise a General Purpose I/O (GPIO) pin to trigger our oscilloscope, output the randomly generated secrets $\mathbf{f}$ and $\mathbf{g}$ over serial to confirm our results, and add functions which can change the ciphertext as required in encryption step. A Langer RF-U 5-2 H-field probe is placed at a fixed position over microcontroller as shown in Fig. 1. The probe is connected to a Langer PA 306 preamplifier to amplify the signal by 30 dB. We use a PicoScope 6404C digital oscilloscope to sample traces at 1.25 GHz, which are then downsampled to 625 MHz before analysis. In some cases, we average across multiple traces of the same ciphertext to reduce the impact of measurement noise, as detailed in our attack discussion.

All experiments for the reference implementation and ARM-optimized implementation of NTRU use the `PQClean` library commits `964469d` and `pqm4` library commits `0b50e72`, respectively. We provide our data sets and code under the following link: https://mega.nz/folder/DFBTlb4A#oMeh9Z8DgNSxUHyzgYYjCA

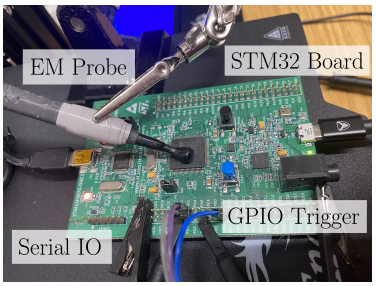


Fig. 1. Hardware setup

3 SPA on NTRU Reference Implementation

In this section, we discuss an SPA on implementations of NTRU in the `PQClean` library. We start with a preliminary idea and then propose two types of practical chosen-ciphertext attacks, exploiting the leakage from polynomial modular reduction. The first attack is applicable to all the four NTRU instances. The

second attack utilizes an inherent property of NTRU-HPS, so it can only be applied to three NTRU-HPS instances.

3.1 A Preliminary Idea

Our first intuition was to recover $\mathbf{f}$ from the leakage of the output in line 3 in Algorithm 3 following the method from [27]—when the ciphertext is generated in the encryption of the encapsulation phase, if we choose the polynomial $\mathbf{c}$ so that it consists only of a nonzero constant term c_0, then the coefficient of $\mathbf{c} \cdot \mathbf{f}$ only has three possible values: $\{-c_0, 0, c_0\}$. If we can find a difference in power consumption between these values during computation, then we can recover the coefficients of $\mathbf{f}$. However, there is an implicit restriction:

$$\mathbf{c} \equiv 0 \pmod{(q, \mathbf{\Phi}_1)}, \tag{1}$$

which means $\mathbf{c}(1) = \sum_{i=0}^{n-1} c_i \equiv 0 \pmod{q}$. We thus cannot simply set $\mathbf{c} = c_0$.

3.2 Attack 1: Recovering Differences Between Adjacent Coefficients

In order to overcome the above restriction, we can choose the polynomial $\mathbf{c}$ with the following `Type-I` structure:

$$\texttt{Type-I} : \mathbf{c} = c_0 + (q - c_0)x,\ c_0 \in \{1, 2, \ldots, q-1\}. \tag{2}$$

In the decryption step, the coefficient of $\mathbf{v}$ (before mod$q$) is:

$$v_i = c_0(f_i - f_{(i-1) \bmod^+ n}) \bmod^+ q \in \{-2c_0, -c_0, 0, c_0, 2c_0\} \bmod^+ q. \tag{3}$$

As Eq. (3) shows, there are at most five possible values for each coefficient. If side-channel leakage can be used to distinguish these values, then we can recover all differences between adjacent coefficients, i.e., $\delta_i = f_i - f_{(i-1) \bmod^+ n}$. The equations are written in matrix form as follows:

$$\begin{pmatrix} \delta_{n-1} \\ \delta_{n-2} \\ \vdots \\ \delta_1 \\ \delta_0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \\ -1 & & & & 1 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_1 \\ f_0 \end{pmatrix}. \tag{4}$$

Because $\mathbf{f} \in S_3$, $f_{n-1} = 0$, we have:

$$\begin{pmatrix} \delta_{n-2} \\ \vdots \\ \delta_1 \\ \delta_0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ & & & 1 \end{pmatrix} \begin{pmatrix} f_{n-2} \\ \vdots \\ f_1 \\ f_0 \end{pmatrix}, \tag{5}$$

$$\begin{pmatrix} f_{n-2} \\ \vdots \\ f_1 \\ f_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ & \ddots & \ddots & \vdots \\ & & 1 & 1 \\ & & & 1 \end{pmatrix} \begin{pmatrix} \delta_{n-2} \\ \vdots \\ \delta_1 \\ \delta_0 \end{pmatrix}. \tag{6}$$

We can then recover each coefficient by computing $f_i = \sum_{j=0}^{i} \delta_j$.

Target after the First Polynomial Multiplication. A question remains of whether we can find leakage suitable as a classifier. In the reference implementation from the `PQClean` library, we find a function[1] immediately after the multiplication between $\mathbf{c}$ and $\mathbf{f}$ that has significant leakage related to v_i. Our target is the iteration over coefficients (lines 4–8) in Listing 1.1.

```
1 void PQCLEAN_NTRUHPS2048509_CLEAN_poly_Rq_to_S3(poly *r, const poly *a) {
2     int i; uint16_t flag;
3     //1. Translation: non-negative integer ==> representative in [-q/2,q/2)
4     for (i = 0; i < NTRU_N; i++) {
5         r->coeffs[i] = MODQ(a->coeffs[i]);
6         flag = r->coeffs[i] >> (NTRU_LOGQ - 1);
7         r->coeffs[i] += flag << (1 - (NTRU_LOGQ & 1));
8     }
9     //2. Reduce mod (3, Phi)
10    PQCLEAN_NTRUHPS2048509_CLEAN_poly_mod_3_Phi_n(r);
11 }
```

Listing 1.1. Polynomial reduction from $\mathcal{R}_q$ to S_3 in reference C implementation

After the call to modular reduction `MODQ()` in line 5, each coefficient is in $\{0, 1, \ldots, q-1\}$. The code in lines 6–7 then implements the centered modular reduction (i.e., mod$q$): if the output of line 5 is less than $q/2$, then the consequent two computations do not change its value, or else $(-q) \bmod^{+} 3$ (i.e., 1 for $q = 2048, 8192$; 2 for $q = 4096$) will be added. As the coefficient of a polynomial is non-negative in the implementation setting, $-q$ is taken modulo 3 in advance to avoid the output becoming a negative number and also make sure that the result after modulo 3 is correct. From experimental observation, one can find a periodic pattern in the trace where the target code runs. We choose the local downward peak in each small interval as Point of Interest (PoI) (in total n PoI corresponding to n intermediate values), as it is related to the HW of the output in line 7 (see Fig. 2). From the result in Fig. 2, we conclude that:

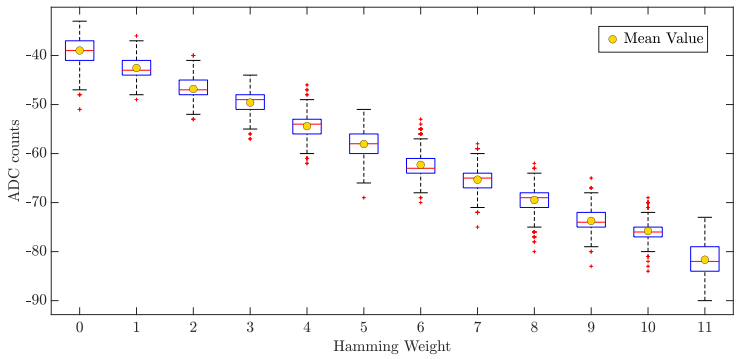


Fig. 2. p(PoI) vs. Hamming Weight of intermediate value in `PQClean` (1000 PoI for each HW value, red line indicates median, red "+" indicates outliers).

[1] https://github.com/PQClean/PQClean/blob/964469d/crypto_kem/ntruhps2048509/clean/owcpa.c#L139

$$|p\left(\mathrm{PoI}_i\right)| \propto h(v_i) = \begin{cases} \mathrm{HW}\left(v_i\right), & \text{if } v_i < q/2; \\ \mathrm{HW}\left(v_i + (-q) \bmod^{+} 3\right), & \text{otherwise.} \end{cases} \tag{7}$$

This means that a significant difference in the HWs of intermediate values can be observed in the corresponding sub-traces. Next, we choose a special c_0 to distinguish different possible values of δ via the HW leakage.

Precomputation. For our chosen ciphertext $\mathbf{c} = c_0 + (q - c_0)x$, the corresponding v_i is in $\{(-2c_0) \bmod^{+} q,\ (-c_0) \bmod^{+} q,\ 0,\ c_0 \bmod^{+} q,\ 2c_0 \bmod^{+} q\}$. By going through all the possible values of c_0, we establish a data set (h vs. (c_0, δ)) by evaluating the function $h\left(\right)$ for the above five values (denoted by h_δ when c_0 is fixed). A partial data set is shown below. From Fig. 3, we know that when $c_0 = 1$, we can distinguish -2 from $\{-1, 0, 1, 2\}$, because the corresponding h is substantially larger.

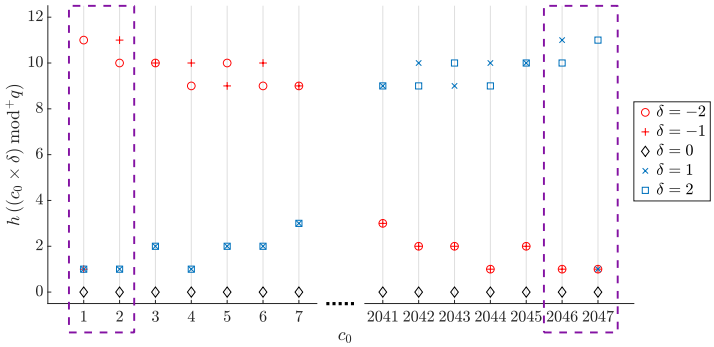


Fig. 3. Selected $h\left((c_0 \times \delta) \bmod^{+} q\right)$ vs. c_0 (dotted box indicates our chosen c_0).

As mentioned in [27], the task to find appropriate c_0 can be considered as a constrained clustering problem [25]. To some degree, more clusters reduces the distance between different clusters in our scenario. In order to achieve better separation in real traces with measurement noise, we only use c_0 which can divide five possible values of δ into two clusters according to high and low h_δ values (represented by "H" and "L"). This ensures that points within any one cluster are similar and the difference between clusters is big. Specifically, for $i, j \in \{-2, -1, 0, 1, 2\}$, we define the similarity within any one cluster B as $S_{\mathrm{B}} = \underset{i,j \in \mathrm{B}}{max}(|h_i - h_j|)$ and the distance between two clusters H and L as $D_{\text{H-L}} = \underset{i \in \mathrm{H}, j \in \mathrm{L}}{min}(|h_i - h_j|)$. We first run a hierarchical clustering on the data set for different c_0 individually to separate all the five possible values into two clusters. We then refine the process of selecting the best c_0: for the same partition, i.e., each cluster has exactly the same elements, we choose the c_0 which has the highest value of $D_{\text{H-L}} - 0.5(S_{\mathrm{H}} + S_{\mathrm{L}})$.

Actual Analysis. In our actual analysis, we only use the best c_0 for good separability. For example, in the analysis of `ntruhps2048509`, we choose four `Type-I` ciphertexts where $c_0 = 1, 2, 2046, 2047$. Following the notation of "partition with HW feature tags" in [27], we can acquire four different partitions:

$$\begin{aligned}
&\{\mathrm{H}:\{-2\},\ \mathrm{L}:\{-1,\ 0,\ 1,\ 2\}\},\ \text{when } c_0 = 1;\\
&\{\mathrm{H}:\{-2,\ -1\},\ \mathrm{L}:\{0,\ 1,\ 2\}\},\ \text{when } c_0 = 2;\\
&\{\mathrm{H}:\{2,\ 1\},\ \mathrm{L}:\{-2,\ -1,\ 0\}\},\ \text{when } c_0 = 2046;\\
&\{\mathrm{H}:\{2\},\ \mathrm{L}:\{-2,\ -1,\ 0,\ 1\}\},\ \text{when } c_0 = 2047.
\end{aligned} \tag{8}$$

Based on these partitions, we can build a decision tree (see Fig. 4). Next, we send the chosen ciphertexts to the victim and collect traces while it runs the decapsulation. After locating the section of each trace related to our target function, we find the n PoI and use k-means clustering to separate them into two clusters (H and L) according to the amplitude. With the knowledge of the clustering results under four ciphertexts and the decision tree, we can recover δ. This process is visually illustrated in Fig. 5.

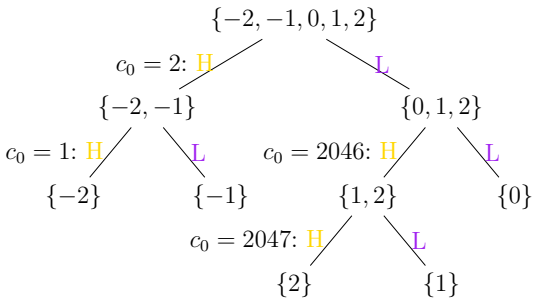
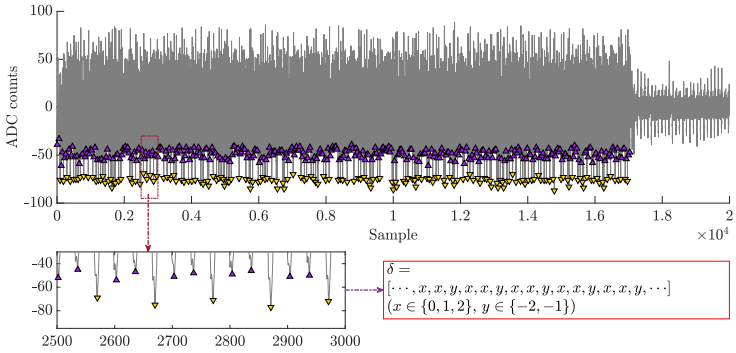


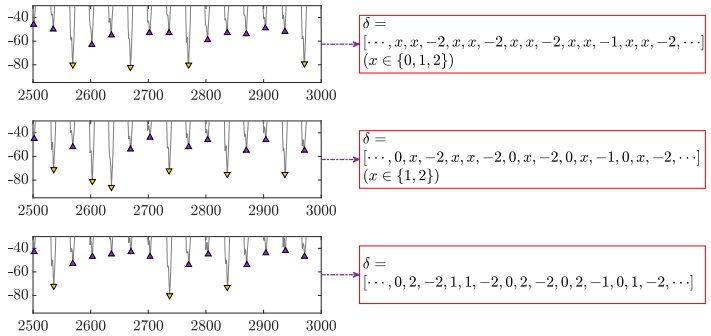
Fig. 4. Decision tree for recovery of δ_i.

However when analyzing the traces, we found it hard to correctly classify the first PoI (corresponding to δ_0). Consequently, instead of focusing on the recovery of the top $(n-1)$ δ_i, we turned to aim at the recovery of all δ_i except δ_0, i.e., $\delta_1, \delta_2, \ldots, \delta_{n-1}$. From Eq. (4), δ_0 can be calculated using the remaining δ_i, i.e., $\delta_0 = -\sum_{i=1}^{n-1} \delta_i$. We can then recover the secret polynomial $\mathbf{f}$ from Eq. (6). As all instances of NTRU call this reduction function, our analysis scheme is universal. The chosen ciphertexts for the other three instances are provided in our data sets (see URL in Sect. 2.3).

Experimental Results. We apply the analysis strategy to all the four NTRU instances. We examine the success rate of recovery of δ for 16 randomly generated key pairs. For `ntruhps2048509`, in a single-trace setting (i.e., one trace for each chosen ciphertext), we can recover all 508 δ_i for 13 out of 16 tests. This means



(a) Single trace under $c_0 = 2$



(b) Partial traces under $c_0 = 1$, 2046, 2047

Fig. 5. Traces used to recover δ_i; yellow (or purple) triangle indicates the corresponding δ_i is clustered to "H" (or "L").

we can reveal the secret polynomial $\mathbf{f}$ with just 4 traces. In the remaining 3 cases, the success rate is $507/508 \approx 99.80\%$, and can be improved by averaging corresponding PoI of multiple traces from the same challenge. The results for the remaining instances are given in Table 2.

In one test for `ntruhps4096821` (marked with $\perp$ in Table 2), noise prevented recovery of both δ_0 and δ_1 even after averaging 10 repeated traces. Since more than one δ_i was incorrect we could not use Eq. (4) to reconstruct $\mathbf{f}$. To solve this issue, we altered ciphertext `Type-I` to rotate the PoI as shown below:

$$\texttt{Type-IB} : \mathbf{c} = (c_0 + (q - c_0)\, x) \cdot x^j,\ c_0 \in \{1, 2, \ldots, q-1\},\ j \in \{0, 10\}. \quad (9)$$

For each chosen c_0, we send a `Type-IB` ciphertext ($j = 0$) and a `Type-IB` ciphertext with shifted PoI ($j = 10$). We then replace $p\,(\mathrm{PoI}_0)$ and $p\,(\mathrm{PoI}_1)$ of ($j = 0$) with $p\,(\mathrm{PoI}_{10})$ and $p\,(\mathrm{PoI}_{11})$ of ($j = 10$), and perform clustering to recover δ. Although 4 more ciphertexts were used, the success rate can reach 100% without averaging, i.e., $4 + 4 = 8$ traces can recover δ, thus infer a correct $\mathbf{f}$.

Table 2. Keys with 100% success rate and lower success rate for all NTRU instances, the number after percentage indicates the needed number of traces in averaging for each ciphertext in order to improve the SR to 100%

Instance	$\#\delta_i$	#SR=100%	#SR<100%	SR in <100% cases
`ntruhps2048509`	508	13	3	99.80% (2), 99.80% (4), 99.80% (5)
`ntruhps2048677`	676	16	0	-
`ntruhps4096821`	820	13	3	99.88% (4), 99.88% (6), 99.88% ($\perp$)
`ntruhrss701`	700	15	1	99.86% (2)

In summary, our method is effective on the reference implementations of all the four NTRU instances. In the best situation (also the most common case), the secret key can be recovered with only 4 traces, while the worst case (one test in `ntruhps4096821`) requires at least 6 traces per ciphertext, meaning $4 \times 6 = 24$ traces in total. As the target function runs coefficient-wise, we can handle all n PoI together, and thus the complexity will not increase as n increases.

3.3 Attack 2: Recovery of the "Invisible" Secret Polynomial

In the previous subsection, we have shown that as few as four chosen ciphertexts could recover the secret key. The question arises of whether four is the lower bound for the chosen-ciphertext SCA. In this section, we propose a more efficient attack using an inherent property of NTRU that succeeds with fewer ciphertexts.

Analysis of NTRU-HPS. Firstly, we target the NTRU-HPS variant. We consider the polynomial $\mathbf{c}$ with the following structure:

$$\texttt{Type-II} : \mathbf{c} = r_0\mathbf{h},\ r_0 \in \{1, 2, \ldots, q-1\}. \tag{10}$$

`Type-II`-like ciphertext have been discussed in several CCA-based mismatch analysis schemes [10, 28]: the idea of these techniques is to send a ciphertext $\mathbf{c} = \mathbf{r} \cdot \mathbf{h}$ with carefully chosen coefficients of $\mathbf{r}$ and utilize a decryption-failure oracle to recover the polynomial $\mathbf{g}$. Unfortunately this oracle is unavailable in CCA-secure NTRU KEM, as the chosen polynomial $\mathbf{r}$ in those schemes cannot pass the validation test (line 8 in Algorithm 3) in the decryption phase. Without depending on a decryption-failure oracle, we only leave the constant term of $\mathbf{r}$ and reveal the coefficients of the "invisible" polynomial $\mathbf{g}$ via the leakage of polynomial modular reduction. Inspired by the formula derivation in [10,28], we make the following proposition:

Proposition 1. *For NTRU-HPS, given the chosen ciphertext* $\mathbf{c} = r_0\mathbf{h}$*, each coefficient of the intermediate polynomial* $\mathbf{v}$ *only has three possible values:* $\{-3r_0, 0, 3r_0\}$.

Proof.

$$\begin{aligned} \mathbf{v} &= \mathbf{c} \cdot \mathbf{f} && \bmod(q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \\ &= r_0\mathbf{h} \cdot \mathbf{f} && \bmod(q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \\ &= 3r_0\mathbf{g} \cdot \mathbf{f}_q \cdot \mathbf{f} && \bmod(q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \end{aligned} \tag{11}$$

Since $\mathbf{f}_q \cdot \mathbf{f} \equiv 1 \pmod{(q, \mathbf{\Phi}_n)}$, $\mathbf{f}_q \cdot \mathbf{f} = 1 + \mathbf{k} \cdot \mathbf{\Phi}_n \bmod q$, where $\mathbf{k} \in \mathbb{Z}_q[x]$ and $\deg \mathbf{k} = \deg \mathbf{f}_q + \deg \mathbf{f} - (n-1)$. As key generation in NTRU-HPS forces $\mathbf{g} \equiv 0 \pmod{(q, \mathbf{\Phi}_1)}$, we can write it as $\mathbf{g} = \mathbf{g}^* \cdot \mathbf{\Phi}_1 \bmod q$, where $\mathbf{g}^* \in \mathbb{Z}_q[x]$ and $\deg \mathbf{g}^* = \deg \mathbf{g} - 1$. Then we have:

$$\begin{aligned} \mathbf{v} &= 3r_0\mathbf{g} \cdot (1 + \mathbf{k} \cdot \mathbf{\Phi}_n) && \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \\ &= 3r_0\mathbf{g} + 3r_0\mathbf{g} \cdot \mathbf{k} \cdot \mathbf{\Phi}_n && \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \\ &= 3r_0\mathbf{g} + 3r_0\mathbf{k} \cdot \mathbf{g}^* \cdot \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n && \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n) \\ &= 3r_0\mathbf{g} && \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n). \end{aligned} \tag{12}$$

□

From Proposition 1, we know v_i only has three possible values: $\{(-3r_0) \bmod^+ q, 0, 3r_0 \bmod^+ q\}$. Utilizing the same HW leakage and strategy of choosing appropriate ciphertexts as in Sect. 3.2 (but on a different data set (h vs. (r_0, g_i)), we can recover $\mathbf{g}$ with only 2 chosen ciphertexts.

Next, we take the analysis of `ntruhps2048509` as an example to explain the process of recovery of secret polynomials $\mathbf{g}$ and $\mathbf{f}$. The chosen r_0 and corresponding partitions are shown below:

$$\begin{aligned} &\{\mathrm{H} : \{1\}, \mathrm{L} : \{-1, 0\}\}, \text{ when } r_0 = 682; \\ &\{\mathrm{H} : \{-1\}, \mathrm{L} : \{0, 1\}\}, \text{ when } r_0 = 1366. \end{aligned} \tag{13}$$

With these two partitions, we can build a decision tree (see Fig. 6) to reveal each coefficient of $\mathbf{g}$.

The actual analysis process with the PoI in the traces is shown in Fig. 7. As $\mathbf{g}$ has a fixed type that $(q/16-1)$ coefficients are equal to 1 and another $(q/16-1)$ coefficients equal to -1, we can use a sorting algorithm instead of clustering to find the highest $(q/16-1)$ $|p(\mathrm{PoI}_i)|$, then the corresponding g_i will be assigned to 1 when $r_0 = 682$ (or -1 when $r_0 = 1366$).

Proposition 2. *For NTRU-HPS, with the knowledge of* $\mathbf{g}$ *and the relation* $3\mathbf{g} = \mathbf{f} \cdot \mathbf{h} \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$*, one can recover* $\mathbf{f}$ *by computing* $3\mathbf{g} \cdot \mathbf{h}_q \bmod (q, \mathbf{\Phi}_n)$.

Proof. From the above relation, we know that for some $\mathbf{t} \in \mathbb{Z}_q[x]$, $\deg \mathbf{t} = \deg \mathbf{f} + \deg \mathbf{h} - n$, the following equation holds:

$$3\mathbf{g} - \mathbf{f} \cdot \mathbf{h} \equiv \mathbf{t} \cdot \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n \pmod{q}. \tag{14}$$

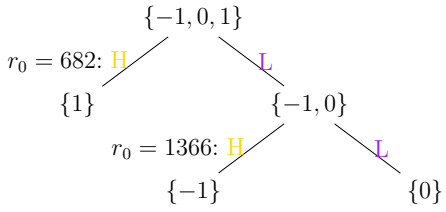
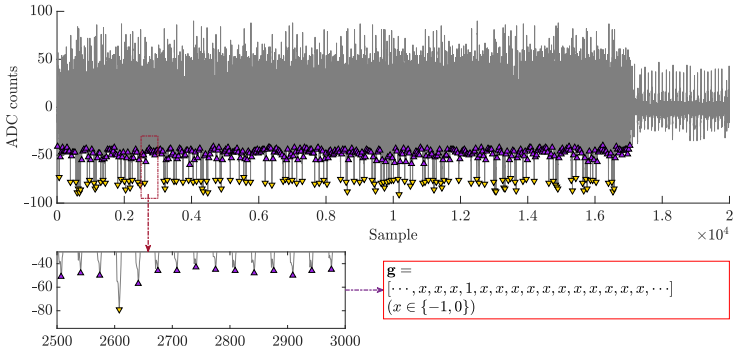
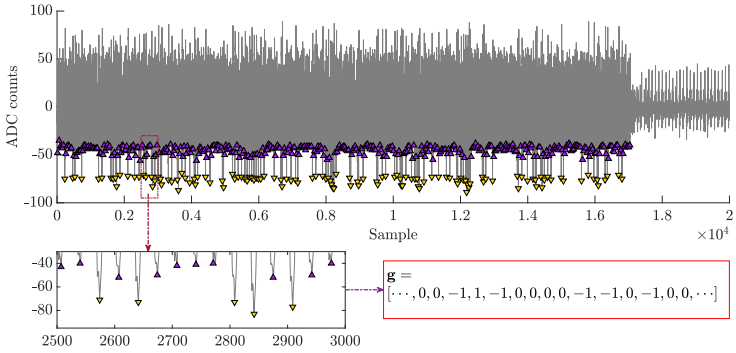


Fig. 6. Decision tree for recovery of $\mathbf{g}$.



(a) Single trace under $r_0 = 682$



(b) Single trace under $r_0 = 1366$

Fig. 7. Traces used to recover the secret ternary polynomial $\mathbf{g}$; yellow (or purple) triangle indicates the corresponding g_i is clustered to "H" (or "L").

So after modular computation, we have:

$$\begin{aligned} 3\mathbf{g} - \mathbf{f} \cdot \mathbf{h} &\equiv 0 \qquad (\mathrm{mod}\ (q,\ \mathbf{\Phi}_n)), \\ i.e.,\ \mathbf{f} \cdot \mathbf{h} &\equiv 3\mathbf{g} \qquad (\mathrm{mod}\ (q,\ \mathbf{\Phi}_n)). \end{aligned} \tag{15}$$

Furthermore, $\mathbf{h} \cdot \mathbf{h}_q \equiv 1\ (\mathrm{mod}\ (q,\ \mathbf{\Phi}_n))$. Finally, we get the following formula:

$$\mathbf{f} \equiv 3\mathbf{g} \cdot \mathbf{h}_q \quad (\mathrm{mod}\ (q,\ \mathbf{\Phi}_n)). \tag{16}$$

□

As $\mathbf{h}$ is a public polynomial, it is easy to compute its inverse. Based on the formula in Proposition 2, we can recover the secret polynomial $\mathbf{f}$.

Analysis on NTRU-HRSS. The above strategy cannot be directly applied to NTRU-HRSS, because the term $\mathbf{\Phi}_1$ disturbs the recovery of $\mathbf{g}$. More specifically,

if we choose $\mathbf{c} = r_0\mathbf{h}$, then $\mathbf{v} = 3r_0\mathbf{g} \cdot \mathbf{\Phi}_1 \bmod (q, \mathbf{\Phi}_1 \cdot \mathbf{\Phi}_n)$. As the coefficient of $\mathbf{g} \cdot \mathbf{\Phi}_1$ is in $\{-2, -1, 0, 1, 2\}$, the complexity of recovery of $\mathbf{g}$ is the same as that of $\mathbf{f}$. Therefore, we do not provide a similar, more efficient strategy for NTRU-HRSS here.

Experimental Results. We apply the second analysis strategy to three NTRU-HPS instances. The chosen ciphertexts and corresponding partitions for the other two instances are provided in our data sets (please refer to the URL in Sect. 2.3). We again examine the success rate of recovery of $\mathbf{g}$ on 16 randomly generated key pairs. For `ntruhps2048509`, in a single-trace setting, we can recover all g_i correctly in all 16 cases. This means we can reveal the secret polynomial $\mathbf{f}$ with just 2 traces. The results for the remaining instances are shown in Table 3. In the worst case (with respect to the number of traces used), i.e., in one test in `ntruhps2048677`, we note that we could improve the success rate from 99.70% to 100% by averaging the corresponding PoI of three traces from the same challenge, i.e., $2 \times 3 = 6$ traces are needed.

Table 3. Keys with 100% success rate and lower success rate for all NTRU-HPS instances (cause $g_{n-1} = 0$, $\#g_i = n - 1$), the number after percentage indicates the needed number of traces in averaging for each ciphertext in order to improve the SR to 100%

Instance	$\#g_i$	#SR=100%	#SR<100%	SR in <100% cases
`ntruhps2048509`	508	16	0	-
`ntruhps2048677`	676	15	1	99.70% (3)
`ntruhps4096821`	820	14	2	99.51% (2), 99.76% (2)

4 Applicability to `pqm4`

We also checked if the same or similar vulnerability exists in `pqm4` [16] implementations which have been optimized for ARM processors. In this library, the function that transfers an element in $\mathcal{R}_q$ to S_3[2] is slightly different from the one in the `PQClean` library (see Listing 1.2).

```
void poly_Rq_to_S3(poly *r, const poly *a) {
    int i;
    //1. Translation: integer in [0,q) ==> representative in [3q-q/2,3q+q/2)
    for (i = 0; i < NTRU_N; i++) {
        r->coeffs[i] = ((a->coeffs[i] >> (NTRU_LOGQ - 1)) ^ 3) << NTRU_LOGQ;
        r->coeffs[i] += a->coeffs[i];
    }
    //2. Reduce mod (3, Phi)
    ...
}
```

Listing 1.2. Polynomial reduction from $\mathcal{R}_q$ to S_3 in `pqm4` implementation

[2] https://github.com/mupq/pqm4/blob/0b50e72/crypto_kem/ntruhps2048509/m4f/owcpa.c#L150

We found that the local upward peak (i.e., PoI) in the EM trace is related to the HW of the output in line 6 in Listing 1.2. From Fig. 8, we assume the following HW leakage model:

$$|p\left(\mathrm{PoI}_i\right)| \propto h'(v_i) = \begin{cases} \mathrm{HW}\left(3q + v_i\right), \text{ if } v_i < q/2; \\ \mathrm{HW}\left(2q + v_i\right), \text{ otherwise.} \end{cases} \tag{17}$$

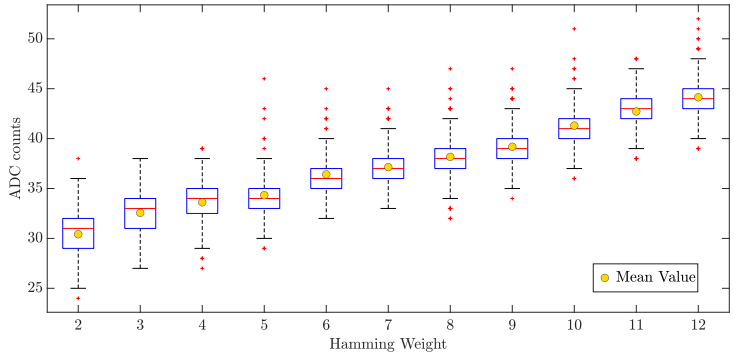


Fig. 8. $p\left(\mathrm{PoI}\right)$ vs. Hamming Weight of intermediate value in `pqm4` (1000 PoI for each HW value, red line indicates median, "+" indicates outlier).

4.1 Direct Recovery of f

We first tried to apply the recovery of δ in Sect. 3.2 to `pqm4`. The same ciphertext (i.e., $\mathbf{c} = c_0 + (q - c_0)x$) was chosen. However, the two different implementations of polynomial reduction cause different effects on our analysis. In the reference implementation in `PQClean`, based on our leakage model, we could find c_0 to distinguish between positive numbers $\{1, 2\}$, negative numbers $\{-1, -2\}$ and zero. Furthermore, some c_0 can be used to distinguish 2 from 1, and -1 from -2. In the `pqm4` implementation, it is possible to find c_0 to distinguish positive numbers, negative numbers, and zero through the leakage model. Yet, for arbitrary $c_0 \in \{1, 2, \ldots, q-1\}$, we have:

$$\begin{aligned} \left|h'\left(\left(2c_0\right) \bmod^{+} q\right) - h'\left(\left(c_0\right) \bmod^{+} q\right)\right| &\leq 1, \\ \left|h'\left(\left(-2c_0\right) \bmod^{+} q\right) - h'\left(\left(-c_0\right) \bmod^{+} q\right)\right| &\leq 1. \end{aligned} \tag{18}$$

It will be impractical to distinguish 2 from 1, and -2 from -1 in actual analysis.

However, the ability to distinguish between positive, negative, and zero is sufficient to recover $\mathbf{f}$. According to Eq. (4) and $f_{n-1} = 0$, we can find the following pair of equations:

$$\begin{cases} f_0 = \delta_0 \\ f_{n-2} = -\delta_{n-1}. \end{cases} \tag{19}$$

There are just three possible values for δ_0 and δ_{n-1}. It is feasible to find two c_0 that form two partitions like Eq. (13). By observing the clustering result of the PoI in the first sub-trace and the last sub-trace, we can reveal f_0 and f_{n-2} by two chosen ciphertexts with the following structure:

$$\widetilde{\texttt{Type-I}} : \mathbf{c}^{(i)} = c_0 + (q - c_0)x^i, \tag{20}$$

where $c_0 \in \{1, 2, \ldots, q-1\}$ and $i \in \{1, 2, \ldots, (n-1)/2\}$.

As mentioned, ciphertexts like $\mathbf{c}^{(1)}$ can help recover f_0 and f_{n-2}. In a similar way, ciphertexts like $\mathbf{c}^{(2)}$ can help recover f_1 and f_{n-3}. Using ciphertexts with the structures $\mathbf{c}^{(1)}$, $\mathbf{c}^{(2)}$, $\ldots$, $\mathbf{c}^{((n-1)/2)}$, we can recover all coefficients of $\mathbf{f}$. The number of ciphertexts needed is $2 \times (n-1)/2 = n - 1$. Notably, this analysis scheme applies to both NTRU-HPS and NTRU-HRSS.

4.2 Recovery via g

As the leakage can still be used to distinguish between zero, positive, and negative numbers, an approach similar to the strategy in Sect. 3.3 can be applied to `pqm4` implementation of NTRU-HPS. Only two ciphertexts with the structure of Eq. (10) are needed to launch the attack. However, note that the number of required traces may be much larger than 2, because the amplitude of variation of p(PoI) against HW of intermediate value is not as significant as that in `PQClean` implementations. We leave an experimental evaluation of this for future work.

5 Conclusion

In this paper, we proposed two chosen-ciphertext SCA schemes on the latest versions of NTRU. Targeting the HW leakage from the polynomial modular reduction—which is an essential component in NTRU—we first used four `Type-I` ciphertexts to exploit the differences between contiguous coefficients (i.e., δ) of the core secret polynomial $\mathbf{f}$. Combining the inherent property of NTRU-HPS, we put forward a more efficient second strategy—first revealing the "invisible" secret polynomial $\mathbf{g}$ via 2 `Type-II` ciphertexts, then recovering $\mathbf{f}$ from Eq. (16). In practical experiments, we found the required number of traces to be low, from two to 24. We also pinpointed similar leakage in the optimized `pqm4` implementations. As the above two analysis strategies are succinct and efficient, an effective countermeasure is required to mitigate the side-channel leakage when one uses NTRU with a long-term secret key. We plan to investigate appropriate countermeasures like masking and shuffling in a future work.

Acknowledgements. This work is partially supported by the National Key Research and Development Program of China (2020YFB1005700) and by the Engineering and Physical Sciences Research Council (EPSRC) under grants EP/R012598/1 and EP/V000454/1. We thank the anonymous reviewers for the valuable comments and Sitong Zong for her helpful proofreading advice.

References

1. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., et al.: Status report on the second round of the NIST post-quantum cryptography standardization process. Rep. NISTIR 8309, US Department of Commerce, NIST (July 2020). https://doi.org/10.6028/NIST.IR.8309
2. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., et al.: Status report on the first round of the NIST post-quantum cryptography standardization process. Rep. NISTIR 8240, US Department of Commerce, NIST (January 2019). https://doi.org/10.6028/NIST.IR.8240
3. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., et al.: Status report on the third round of the NIST post-quantum cryptography standardization process. Tech. Rep. NISTIR 8413, US Department of Commerce, NIST (July 2022). https://doi.org/10.6028/NIST.IR.8413
4. An, S., Kim, S., Jin, S., Kim, H., Kim, H.: Single trace side channel analysis on NTRU implementation. Applied Sciences **8**(11) (2018). https://doi.org/10.3390/app8112014
5. Askeland, A., Rønjom, S.: A side-channel assisted attack on NTRU. Cryptology ePrint Archive, Paper 2021/790 (2021), https://eprint.iacr.org/2021/790
6. Atici, A.C., Batina, L., Gierlichs, B., Verbauwhede, I.: Power analysis on NTRU implementations for RFIDs: First results. In: RFIDSec 2008 (2008)
7. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., et al.: CRYSTALS–Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
8. Chen, C., Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., et al.: NTRU: Algorithm specifications and supporting documentation. Tech. rep., NIST (2020), https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions
9. D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology – AFRICACRYPT 2018. pp. 282–305. Springer International Publishing, Cham (2018)
10. Ding, J., Deaton, J., Schmidt, K., Vishakha, Zhang, Z.: A simple and efficient key reuse attack on NTRU cryptosystem. Cryptology ePrint Archive, Paper 2019/1022 (2019), https://eprint.iacr.org/2019/1022
11. Hoffstein, J.: NTRU: A new high speed public key cryptosystem. presented at the rump session of Crypto 96 (1996)
12. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) Algorithmic Number Theory. LNCS, vol. 1423, pp. 267–288. Springer (1998). https://doi.org/10.1007/BFb0054868
13. Huang, W.L., Chen, J.P., Yang, B.Y.: Power analysis on NTRU Prime. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 123–151 (November 2019). https://doi.org/10.13154/tches.v2020.i1.123-151
14. Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P.: High-speed key encapsulation from NTRU. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. LNCS, vol. 10529, pp. 232–252. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_12
15. Jaulmes, É., Joux, A.: A chosen-ciphertext attack against NTRU. In: Bellare, M. (ed.) Advances in Cryptology – CRYPTO 2000. LNCS, vol. 1880, pp. 20–35. Springer (2000). https://doi.org/10.1007/3-540-44598-6_2

16. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4
17. Karabulut, E., Alkim, E., Aysu, A.: Single-trace side-channel attacks on ω-small polynomial sampling: With applications to NTRU, NTRU Prime, and CRYSTALS–Dilithium. In: 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 35–45. IEEE (2021). https://doi.org/10.1109/HOST49136.2021.9702284
18. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO' 99. LNCS, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25
19. Lee, M., Song, J.E., Choi, D., Han, D.: Countermeasures against power analysis attacks for the NTRU public key cryptosystem. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E93.A**(1), 153–163 (2010). https://doi.org/10.1587/transfun.E93.A.153
20. Mujdei, C., Beckers, A., Mera, J.M.B., Karmakar, A., Wouters, L., Verbauwhede, I.: Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication. Cryptology ePrint Archive, Paper 2022/474 (2022), https://eprint.iacr.org/2022/474
21. Park, A., Han, D.G.: Chosen ciphertext simple power analysis on software 8-bit implementation of Ring-LWE encryption. In: 2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST). pp. 1–6 (2016). https://doi.org/10.1109/AsianHOST.2016.7835555
22. Ravi, P., Ezerman, M.F., Bhasin, S., Chattopadhyay, A., Roy, S.S.: Will you cross the threshold for me? generic side-channel assisted chosen-ciphertext attacks on NTRU-based KEMs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 722–761 (2022). https://doi.org/10.46586/tches.v2022.i1.722-761
23. Schanck, J.M.: A comparison of NTRU variants. Cryptology ePrint Archive, Paper 2018/1174 (2018), https://eprint.iacr.org/2018/1174
24. Sim, B., Kwon, J., Lee, J., Kim, I., Lee, T., Han, J., et al.: Single-trace attacks on message encoding in lattice-based KEMs. IEEE Access **8**, 183175–183191 (2020). https://doi.org/10.1109/ACCESS.2020.3029521
25. Tizpaz-Niari, S., Cerný, P., Trivedi, A.: Data-driven debugging for functional side channels. In: 27th Annual Network and Distributed System Security (NDSS) Symposium, San Diego, California, USA, February 23-26, 2020. The Internet Society (2020). https://doi.org/10.14722/ndss.2020.24269
26. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 296–322 (November 2021). https://doi.org/10.46586/tches.v2022.i1.296-322
27. Xu, Z., Pemberton, O.M., Sinha Roy, S., Oswald, D., Yao, W., Zheng, Z.: Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. IEEE Transactions on Computers **71**(9), 2163–2176 (2022). https://doi.org/10.1109/TC.2021.3122997
28. Zhang, X., Cheng, C., Ding, R.: Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. In: Gao, D., Li, Q., Guan, X., Liao, X. (eds.) Information and Communications Security. LNCS, vol. 12919, pp. 283–300. Springer (2021). https://doi.org/10.1007/978-3-030-88052-1_17
29. Zheng, X., Wang, A., Wei, W.: First-order collision attack on protected NTRU cryptosystem. Microprocessors and Microsystems **37**(6), 601–609 (2013). https://doi.org/10.1016/j.micpro.2013.04.008