Security Assessment of NTRU Against Non-Profiled SCA

Luk Bettale, Julien Eynard, Simon Montoya, Guénaël Renault, Rémi Strullu

Security Assessment of NTRU Against Non-Profiled SCA

Luk Bettale[1], Julien Eynard[2[0000−0002−1118−1383]], Simon Montoya[1,3], Guénaël Renault[2,3[0000−0002−7050−9975]], Rémi Strullu[2]

[1]Crypto and Security Lab IDEMIA Courbevoie, France.
Email: firstname.lastname@idemia.com
[2]ANSSI Paris, France.
Email: firstname.lastname@ssi.gouv.fr
[3]LIX, INRIA, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, France.
Email: firstname.lastname@lix.polytechnique.fr

Abstract. NTRU was first introduced by J. Hoffstein, J. Pipher and J.H Silverman in 1998. Its security, efficiency and compactness properties have been carefully studied for more than two decades. A key encapsulation mechanism (KEM) version was even submitted to the NIST standardization competition and made it to the final round. Even though it has not been chosen to be a new standard, NTRU remains a relevant, practical and trustful post-quantum cryptographic primitive.
In this paper, we investigate the side-channel resistance of the NTRU `Decrypt` procedure. In contrast with previous works about side-channel analysis on NTRU, we consider a weak attacker model and we focus on an implementation that incorporates some side-channel countermeasures. The attacker is assumed to be unable to mount powerful attacks by using templates or by forging malicious ciphertexts for instance. In this context, we show how a non-profiled side-channel analysis can be done against a core operation of NTRU decryption. Despite the considered countermeasures and the weak attacker model, our experiments show that the secret key can be fully retrieved with a few tens of traces.

Keywords: Non-profiled SCA · NTRU · Post-Quantum Cryptography

1 Introduction

The emergence of a quantum computer with the capacity to run Shor's algorithm [29] is a threat to classic cryptography. This threat has led the scientific community to investigate further the field of cryptographic primitives which are resistant against such computers: post-quantum cryptography. A quantum computer could break widely used asymmetric cryptosystems such as RSA or elliptic curve cryptography. In order to keep communications secure in the future, some national agencies have started to study proposals for new algorithms (*e.g.* [8, 2]) and have conducted standardization processes for quantum safe algorithms (*e.g.* [25, 9]). One of them was started in 2016 by the National Institute of Standards

and Technology (NIST) in the form of a competition [25]. It was seemingly the most attended and followed standardization process by the post-quantum cryptography community until the results were announced on July 5th, 2022.

Among the candidates of the final round, two key encapsulation mechanisms, named NTRU [10] (finalist) and NTRU Prime [4] (alternate finalist), are based on the same mathematical problem that was first introduced in the definition of the original NTRU cryptosystem [20]. In particular, these versions of NTRU are improvements of the original cryptosystem. Even though they were not selected to be part of the next standards, their efficiency, compactness, security make them relevant to be considered for quantum-resistant cryptographic applications. Moreover, the interest of NTRU-based cryptosystems is confirmed by real life experiments. In 2019, Google and Cloudflare jointly initiated an experiment, named CECPQ2 [16, 11] that integrates the NTRU finalist key-exchange algorithm into TLS 1.3. Afterwards, Bernstein *et al.* optimized the NTRU Prime key-exchange algorithm in [5] and used it in TLS 1.3 in order to speed-up CECPQ2 protocol.

This kind of cryptographic primitive can be considered for embedded implementations within devices that are limited in terms of CPU frequency, RAM quantity. These kinds of device are naturally threatened by side-channels attacks. In order to protect the implementations against those kind of attacks, the NIST asked the competitors to provide constant-time implementations. However, this countermeasure only protects against timing attacks [22]. Therefore, this protection is not enough against other side-channel attacks (see *e.g.* [7, 23]). The overhead required for additional security depends on the subroutines and parameters used within a cryptosystem. The present work provides new results about the security of NTRU like implementations against side-channel attacks.

Previous works & motivations

The previous works on the side-channel analysis of NTRU-based schemes were done in different contexts.

In [21], the authors apply several power analysis attacks on various implementations of NTRU Prime. The side-channel analysis focus on the polynomial multiplication during the decryption. These attacks are applied to the NTRU Prime reference implementation, an optimized implementation using SIMD instructions and some implementations using classic countermeasures such as masking. In all these cases, the attacker retrieves the whole secret key. To do so, a powerful attacker model is assumed as he is able to profile the targeted embedded device by using a fully controllable similar open device. In [30], the authors present a single-trace side-channel attack against several lattice-based KEMs. This attack aims to retrieve the ephemeral session key exchanged during the encapsulation routine on not secured implementations. Whereas the NTRU implementation is not directly attacked, the authors give a methodology to apply the attack to it. In [28], the authors combine malicious forged ciphertexts and side-channel analysis in order to retrieve the secret key in the NTRU decapsulation routine. The NTRU cryptosystem is IND-CCA secure which means that it is protected against

chosen ciphertext attacks. However, this protection is ensured by a verification done after the whole computation. Therefore, an attacker using side-channel analysis can learn information about these computations and can deduce information about the secret key. Here again, the targeted implementation is not secured against side-channel attacks. Moreover, the attacker can choose the input ciphertext. Even when the secret is masked, side-channel analysis combined with forged ciphertexts are still devastating (see attack [18] for such an attack on Kyber, another finalist to the NIST PQC competition). In [3], the authors mount a side-channel attack against the NTRU round 3 implementation, which is not claimed to be secured against SCA. The targeted operation is the reduction modulo 3 applied to secret polynomials. By using information leakage during this modular reduction, the attacker can retrieve approximately 75% of the secret polynomial. Eventually, the whole key is found by using some lattice reduction techniques. To conclude their work, the authors suggest a more secure modular reduction algorithm that significantly reduces the side-channel leakage.

All these works have paved the way for good practice about secure implementations. Even if they are applied to different contexts, these attacks mainly focus on non-secure implementations and/or suppose that the attacker is powerful enough to profile the target power consumption/electromagnetic radiations. The purpose of the present work is to go further with the side-channel analysis of NTRU by studying the relevance of some classic side-channel countermeasures (tested on the NTRU reference implementation) against weak attackers.

Being able to assess the security provided by lightweight side-channel countermeasures is crucial for constrained devices. Indeed, implementing secure post-quantum algorithms on constrained devices such as smart cards is already a particularly arduous challenge [17]. Since the development of countermeasures can be resource consuming in many ways (development time, extra energy or entropy consumption within the device, etc), the designer needs to fine-tune the set of side-channel countermeasures according to the considered attacker model. The present work aims to help with this by challenging the efficiency of some classic lightweight side-channel protections against a basic, weak attacker model. To do so, we make use of some classic non-profiled side-channel analysis techniques (clustering) that have already been shown useful to attack embedded implementations of classic asymmetric cryptography (*e.g.* see [19]).

Our contributions.

Our work explores how a weak attacker model can still be threatening for embedded implementations that incorporate low-cost SCA countermeasures. Finding a good balance between performance and effectiveness of SCA countermeasures is crucial for developers. For instance, the rotation-shuffle technique considered in this work is a typical example of a dedicated countermeasure that relies on the intrinsic algebraic structures of NTRU in order to limit the development time, performance and entropy costs.

Attacker model and Device Under Test. As explained above, we assume that the attacker is weak in a sense that he cannot profile the device under test (DUT). In other words, he cannot perform any supervised attack. The attacker has access to the DUT that implements NTRU. Moreover, he knows the algorithms and the arithmetic that are implemented. He can only probe physical signals of a limited number of `Decrypt` executions. Also, he knows that the implementation integrates two classic countermeasures. The first one consists in the implementation of a blind decryption, by masking the ciphertext, in order to prevent the possibility of a chosen ciphertext attack. The second one is a rotation of the secret polynomial in order to randomize the operation order at each execution of the `Decrypt` algorithm. To simulate an experimental situation, the DUT is a STM32F407 discovery board with CPU running at the maximal frequency of 168MHz. The side-channel leakage used by the attacker is electromagnetic radiations (EM).

Sketch of the attack. Considering this attacker model, we perform a non-profiled side-channel attack on NTRU. The attack is performed during the first polynomial multiplication within the `Decrypt` algorithm and aims at retrieving a secret polynomial f. The secret coefficients belong to $\{0, 1, q-1\}$, where q is a power of 2. The attack is performed in two steps: 1) finding the secret key rotations by locating the $q-1$ coefficients; 2) distinguishing between the 0 and 1 coefficients. The first step is done by using a clustering algorithm in order to identify the coefficients with value $q-1$. We use the property that these coefficients have a high Hamming weight. Afterwards, we perform dot products on the clustering results in order to retrieve the rotation indices. The second step uses another property in order to distinguish between 0's and 1's. We use the fact that a 0 coefficient does not change the value of the accumulator during the polynomial multiplication. In practice, it might mean that the CPU handles identical data and instructions during a part of two consecutive coefficient products (this phenomenon is called a collision).

Results. We provide a complete analysis of this side-channel attack against NTRU descapsulation. In particular, we present algorithms to make our approach fully reproducible. It is done in a realistic scenario with a protected implementation running on a non restricted (in terms of frequency) microcontroller. The attacker model is particularly weak (no profiling, no ciphertext forging). It is therefore representative of the kind of threat model that the developers must consider when designing secure embedded implementations. Despite lightweight SCA countermeasures that are wanted to be efficient (performance-wise) yet effective against weak attackers, the attack succeeds in 45 EM traces in average. To succeed, the attack exploits the particular structures of NTRU that we will exhibit later on (key distribution, polynomial arithmetic).

Organization

Section 2 introduces the NTRU `Decrypt` algorithm. The targeted polynomial multiplication algorithm is presented as well as the implemented countermea-

sures that are supposed to protect (to a certain extent) this polynomial multiplication against side-channel analysis. Finally, section 3 contains the details of the attack and provides the results of the experiments.

2 NTRU Description and Implemented Countermeasures

In this section, we present the NTRU `Decrypt` algorithm that we evaluate against side-channel attacks in Section 3. We first introduce notations and then describe the reference implementation of NTRU `Decrypt` with a focus on the targeted operation. Finally, we describe the implemented countermeasures that we take into account in our analysis.

2.1 NTRU Algorithm and notations

Notations. For any integer $q \geq 1$, we note $\mathbb{Z}_q = \mathbb{Z}/(q)$. For three integers $p, q, n \geq 1$, we define R_q and S_p the following two polynomial rings: $R_q = \mathbb{Z}_q[x]/(\phi_1\phi_n)$ (with $\phi_1\phi_n = x^n - 1$) and $S_p = \mathbb{Z}_p[x]/(\phi_n)$ (with $\phi_n = 1 + x + \ldots + x^{n-1}$). An element in R_q (resp. S_p) is a polynomial of degree at most $n-1$ (resp. $n-2$) with coefficients in $\mathbb{Z}_q$ (resp. $\mathbb{Z}_p$). For any polynomial f, we denote by $f[i]$ or f_i the coefficient associated with the monomial x^i.

NTRU finalist of NIST competition. NTRU [10] is a Key Encapsulation Mechanism (KEM) based on the NTRU problem which is often assimilated to a lattice-based one. The first version of NTRU was presented in 1996 in [20]. Over the years, several variants of the NTRU cryptosystem were proposed in order to improve the original one. The series of improvements finally led to the variants submitted to the NIST competition. The implementation supporting the application to the NIST competition is used as a reference for our work. The NTRU submission contains 3 routines: `KeyGen`, `Encrypt` and `Decrypt`. These routines can differ slightly depending on the security parameters and the chosen configuration (HRSS or HPS). However, these small differences do not change the overall structure of the algorithms. As the other ideal lattice KEM proposed in the NIST call, the NTRU submission performs polynomial arithmetic. The polynomials are defined over R_q or S_p with $n \in \{509, 677, 701, 821\}$, $q \in \{2048, 4096, 8192\}$ and $p = 3$ depending on the security level.

Decrypt routine. In this work, we focus on retrieving some information about the long term secret polynomials. These secret data are used in the subroutines `KeyGen` and `Decrypt`. However, the `KeyGen` subroutine is generally run only once in the life cycle of an embedded device. That is not the case for the `Decrypt` subroutine. Moreover, `KeyGen` can be done during the production in factory, therefore preventing a possible side-channel attack during the secret key generation. For these reasons, the present attack focuses on a side-channel analysis on `Decrypt`. Alg. 1 describes `Decrypt` as presented in the reference specification [10]. The secret key is made of (f, f_p, h_q) where f, f_p are secret polynomials

Algorithm 1 `Decrypt`$((f, f_p, h_q), c)$

Require: (secret) keys: f, f_p, h_q, ciphertext: c
1: if $c \not\equiv 0 \pmod{(q, \phi_1)}$ return $(0, 0, 1)$
2: $a \leftarrow (c \cdot f) \mod (q, \phi_1\phi_n)$
3: $m \leftarrow (a \cdot f_p) \mod (3, \phi_n)$
4: $m' \leftarrow \text{Lift}(m)$
5: $r \leftarrow ((c - m') \cdot h_q) \mod (q, \phi_n)$
6: if $(r, m) \in \mathcal{L}_r \times \mathcal{L}_m$ **return** $(r, m, 0)$
7: else **return** $(0, 0, 1)$

and h_q is the inverse of the public polynomial. The Lift operation depends on the NTRU variant (HPS or HRSS). As it is not important for the attack, we omit its definition.

2.2 Targeted algorithmic setting and operation

Algorithm 2 `poly_Rq_mul`(c, f)

Require: c, f, where $\deg(c) = \deg(f) = n - 1$
Ensure: $a = c \times f \mod (x^n - 1)$
1: **for** $k = 0$ to $n - 1$ **do**
2: $\quad a[k] \leftarrow 0$
3: $\quad$ **for** $i = 1$ to $n - k - 1$ **do** $a[k] \leftarrow a[k] + c[k + i] \times f[n - i]$
4: $\quad$ **for** $i = 0$ to k **do** $a[k] \leftarrow a[k] + c[k - i] \times f[i]$
5: **return** a

In order to demonstrate the efficiency of our attack and without loss of generality, we choose the following setting: $n = 509$, $q = 2^{11}$ (`ntruhps2048509` configuration). The results of the present attack can be straightforwardly generalized to any other setting. The side-channel analysis performed in this article targets the first polynomial multiplication at line 2 of Algorithm 1. This choice is motivated by the fact that this operation is performed on a secret polynomial. Therefore, this polynomial multiplication is highly sensitive and requires a thorough study of its potential sources of leakage. Algorithm 1 describes the decryption algorithm at a high level but our attack is performed against the reference implementation. Thus, we need to detail more precisely how the sensitive operations are defined in the source code. The same attack mounted against another implementation should therefore be adapted to its specificities.

The polynomial multiplication $a \leftarrow (c \cdot f) \mod (q, \phi_1\phi_n)$ is done using the function `poly_Rq_mul`(c, f) which is described in Algorithm 2. In the reference C implementation, the inputs are `uint16_t` arrays that represent polynomials and where each element corresponds to a coefficient. The modular reduction modulo q is not performed in this algorithm, but later on the final result.

2.3 Countermeasures

Let's recall that our attacker model implies that only non-profiled side-channel analysis is possible. Some countermeasures are integrated within the NTRU implementation as explained in this section. The reference implementation of NTRU does not claim to be secure against side-channel attacks. As mentioned previously, in order to investigate more thoroughly the vulnerability to side-channel analysis, our objective is to evaluate a more secure implementation, yet still efficient (a typical designer's constraint). In such a case, the considered attacker is not as powerful as in the previous works. Hereafter, we propose two countermeasures targeting such an attacker.

- The `poly_Rq_mul` algorithm is protected with a shuffling countermeasure which aims to randomize the order of the secret coefficients at every iteration.
- Blind decryption by masking the input ciphertext so as to prevent chosen ciphertext attacks.

These countermeasures do not modify the functions and the arithmetic of the reference implementation. All in all, the attack considers executions of the following operation:

$$\texttt{poly\_Rq\_mul}\left(c_1, \textsc{Rotate}\left(f\left(x\right), r\right)\right)$$

where r (resp. c_1) is uniformly sampled in $[0, n-1]$ (resp. R_q) at each decryption, without access to their value. Rotate and c_1 are defined below.

Polynomial Rotation as an Efficient Shuffling. Randomizing the secret coefficients' order is one way to secure an implementation against side-channel attacks. However, this countermeasure can be costly in terms of required entropy and may need to modify the high level polynomial arithmetic. To completely randomize a polynomial, it requires at least n random numbers ($n-1$ being the polynomial degree) when using Knuth's algorithm. Moreover, some optimization of polynomial multiplication such as Karatsuba or Toom-Cook algorithms cannot be applied with a random order of the coefficients. One cheaper alternative consists in rotating the coefficients of the secret polynomials. The rotation ensures a kind of randomization at the cost of one random generation per polynomial and without modifying the polynomial arithmetic. More precisely, let $f(x) = f_0 + f_1 x + \ldots + f_{n-1}x^{n-1}$ be a secret polynomial, then:

$$\textsc{Rotate}(f(x), r) = f_{n-r} + \ldots + f_{n-1}x^{r-1} + f_0 x^r + \ldots + f_{n-r-1}x^{n-1}$$

By considering that the underlying algebraic structure of NTRU is $\mathbb{Z}[x]/(x^n-1)$, this countermeasure is just a multiplication of f by x^r. The rotation is done prior to the execution of `poly_Rq_mul` algorithm and it is refreshed at each call to `Decrypt`. The implementation of `poly_Rq_mul` does not require modification to be applied to rotated polynomials.

Blind Decryption. Chosen ciphertext attacks combined with side-channel attacks are devastating (*e.g.* [28, 18]). The ciphertext shape is generally chosen in order to amplify the side-channel leakages during a computation with the secret key. In order to avoid these attacks, we mask the ciphertext at the beginning of the `Decrypt` algorithm. As a result, the ciphertext c is split into two random variables (aka shares) c_1 and c_2 such that $c = c_1 + c_2 \mod q$. Afterwards, we compute $a_1 \leftarrow (c_1 \cdot f) \mod (q, \phi_1\phi_n)$, $a_2 \leftarrow (c_2 \cdot f) \mod (q, \phi_1\phi_n)$ and we obtain $a = (c \cdot f) \mod (q, \phi_1\phi_n) = a_1 + a_2 \mod q$. Even if the attacker forges a malicious ciphertext, the secret polynomial is multiplied to random/unknown shares. Let's also recall that, as shown in [18], lattice based cryptosystems could be attacked with chosen ciphertexts even if their secret key is masked. Thus, it is a necessity to mask the ciphertext. This masking operation could require as much entropy as masking the secret key f but it does not introduce new potential critical sources of leakage and it can be implemented with less effort.

3 Side-Channel Analysis

In the following section, we target NTRU HPS with parameters $n = 509$ and $q = 2^{11} = 2048$. This attack can be directly applied to any other parameter set. The dataset and scripts are available at `github.com/ANSSI-FR/scantru.git`.

3.1 Target

We focus our analysis on the blind decryption. In particular, the masked ciphertext countermeasure provides the attacker only with side-channel leakages of the multiplication of the secret f by a random value[1].

Thus, we target the rotation countermeasure applied to the product $c \times f$ within the decryption routine (see Alg. 2). This polynomial product is implemented with a classic convolution product. Beside the rotation countermeasure which consists in shuffling the secret f through a multiplication with a random monomial x^r, the `for` loops at lines 3 and 5 of Alg. 2 might also be starting at a random index without having to change the attack. In order to defeat these kinds of countermeasures, we focus on some specific operations. More precisely, we consider the following targeted instructions: the first iteration of the outermost `for` loop (line 1 of Alg. 2).

For the sake of simplicity, we shall be referring to the coefficients $q-1 = 2^{11}-1$ of f as -1. For the experiment, we generated the following data:

- 200 random keys in $(\mathbb{Z}_3[x])/(x^{509} - 1)$ with coefficients in $\{-1, 0, 1\}$;
- 100 random ciphertexts in $(\mathbb{Z}_{2^{11}}[x])/(x^{509} - 1)$ for each key.

[1] If the computation of the multiplications by the two shares are done sequentially, the attacker might obtain twice more leakages per decryption, therefore reducing the required number of attack traces. Since we assume a weak attacker model, we consider only one multiplication per trace in our experiments.

Then, we got 100 EM traces of the targeted instructions for each key. Let's note that the knowledge of the ciphertexts is not necessary to mount the attack, in accordance with our attacker model (impossible ciphertext forgery).

3.2 Setup and pre-processing

The original implementation comes from the reference package of NTRU [10]. We have added a rotation function of the secret polynomial. This feature does not change the arithmetic used in the reference implementation. The attacked operation is implemented within the KEM decapsulation. In order to ease the acquisitions and post-processing, a trigger is placed at the beginning of the product cf. In a more realistic attack setup, let's note that the patterns of the polynomial product could be identified and used to start the acquisition.The DUT is a STM32F407 discovery board with clock settings that are tuned for the CPU to run at the maximal frequency, that is 168MHz. The implementation was compiled with `-O3` optimization flag. We stress that this optimization flag does not impact the countermeasures that are considered here (coefficient rotation and random ciphertext). The acquisitions of electromagnetic radiations (EM) are made with an EM probe Langer and a Lecroy Waverunner oscilloscope at 20Gs/s and 2GHz of bandwidth. Each trace contains approximately $700,000$ samples and captures a period of time that lasts $\sim 35\mu s$ (corresponding to the first iteration of the `for` loop at line 1 of Alg. 2).s

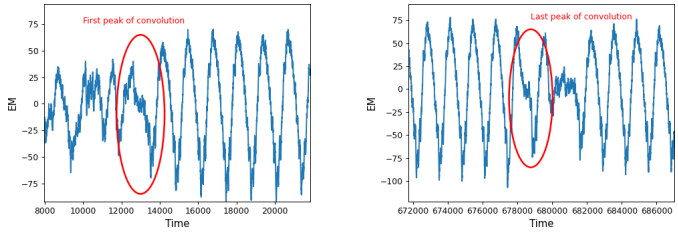


Fig. 1: Details of EM traces of targeted instructions. Red ellipsis point to first (left) and last (right) peaks to be discarded.

A code analysis shows that the targeted instructions are parted in two `for` loops. Given f and c, we expect the EM peaks to be related to the following products: (loop 1) $c_1 \cdot f_{(508+r)}, c_2 \cdot f_{(507+r)}, \ldots, c_{508} \cdot f_{(1+r)}$, (loop 2) $c_0 \cdot f_r$.

Figure 1 shows the beginning and the end of the attacked operations. We can notice that the first and last peaks appear to be quite different than the others, but similar to each other. Since these differences might affect the clustering results, they are discarded. The rotation countermeasure allows the attacker to remove some outliers (*e.g.* difformed peaks) because with several traces it is possible to get EM peaks for each coefficient anyway.

To summarize, the recorded peaks correspond to the iterations $i = 2$ to $i = n - 1$ of the `for` loop at line 3 of Alg. 2. Even though there is almost no jitter, a more accurate alignment is realized by using the Pearson correlation.

The assembly code corresponding to the acquired EM peaks is shown in Fig. 2. This code can be obtained from the open source implementation. It can help the attacker to mount the attack, even though it is not mandatory.

```
.L3:
 ldrh    fp, [r5], #2          @ loading ci
 ldrh    r4, [r9, #-2]!        @ loading fj
 smlabb  r3, r4, fp, r3        @ rk <- rk+ci*fj
 uxth    r3, r3                @ 16 to 32 bit copy
 cmp     r5, r10               @ loop counter check
 strh    r3, [r0]              @ store in r
 bne     .L3                   @ loop back if not done
```

Fig. 2: Assembly code of targeted first inner `for` loop of Alg. 2

3.3 Defeating the rotation countermeasure

The rotation countermeasure may be defeated if we find a way to (partially) identify the EM peaks that correspond to the processing of coefficients -1 of the secret key f. The reason why these coefficients are expected to help us through our attack comes from the following remark and hypothesis.

- *Remark:* The coefficients $-1 = q - 1 \mod q$, with $q = 2^{11}$ in the attacked setup, are encoded with Hamming weight (HW) equal to 11, whereas coefficients 0 (resp. 1) are encoded with HW zero (resp. HW one).
- *Hypothesis:* The target device leaks in Hamming weight.

Henceforth, the strategy to defeat the countermeasure consists in identifying the time samples where the HW of the coefficients leaks in EM, allowing a partial recovery of the coefficients -1. These partial recoveries, for every trace, might be enough to shift them back in a synchronized position. When analyzing the code in figure 2, we expect this leakage to happen during the `ldrh` instruction that loads the coefficient of secret polynomial f for instance.

Non-profiled leakage assessment From previous remarks we expect to be able to distinguish -1 from 0 and 1 more easily than 0 from 1. Let's note that, since q is a power of two, if the coefficients -1 are encoded in two's complement using the CPU register size (for instance $2^{32}-1$ in a 32-bit architecture, or $2^{16}-1$ when using a type such as `uint16_t`), then they are encoded with a maximal Hamming weight (HW). This would reinforce the assumed leakage property and would be therefore strongly in favor of the attack strategy we develop. In our experiments, we choose $q = 2^{11}$. Other possible settings correspond to $q = 2^{12}$ and $q = 2^{13}$. Thus, we stress that our attack might even be more efficient in such settings due to a higher Hamming weight for the -1 coefficients.

k-means clustering. A clustering algorithm is an unsupervised machine learning technique which groups data into a given number of sets according to a given similarity metric. With a k-means clustering [24], the metric is the Euclidean distance to the centroids (means of the clusters), and the variance gives a measure of the spread of the clusters.

It is possible to explore different clustering techniques, such as fuzzy [15] or hierarchical clustering [26], and to investigate their impact on the efficiency of the attack. However, the k-means algorithm is sufficient for our purpose. An advantage of this technique is that it scales up well on the size of the dataset. Practically, a k-means algorithm, where k is the chosen number of clusters, is fed with (parts of) the EM peak traces of all the 507 coefficients of each rotated f at the same time. It gathers the traces in k different sets, each one of them being associated with a so-called *label* that lies in the set $\{0, 1, \ldots, k-1\}$. In order to identify the leakage, we perform a k-means clustering [27] at every time sample and gather the results. We expect that, where the leakage appears to be, the peaks corresponding to -1 will be gathered in one cluster and the peaks for 0 and 1 will be grouped in a second cluster. The credibility of this hypothesis can be first assessed by using the *elbow method*.

Elbow method to determine an optimal number of clusters k. The elbow heuristic allows to determine an optimal number of clusters. The goal is to detect the number of clusters from which an overfitting phenomenon might start.

The metric used for this heuristic is called distortion which is the overall sum of the distances between the samples and the center of their own clusters. Thus, it is a measure of how spread the clusters are. A large decrease of the distortion means that adding an extra cluster allows to significantly decrease the overall variance (distance from the elements to their cluster centers). Whenever the distortion starts decreasing less, it means that adding an extra cluster becomes not as relevant when attempting to decrease the overall variance. This heuristic is tested for each time sample[2]. Since all the results suggest that 2 clusters is always an optimal choice, we carry on with the adopted strategy.

Non-profiled leakage assessment. In total, 15×507 peaks are used for this leakage assessment (*i.e.* the first 15 traces from one key). We use a classical metric in clustering which is the Davies-Bouldin index (`DBI`) [14]. This metric gives an insight of the compactness of the clusters respectively to their respective distance. For two clusters $\mathcal{C}_0$ and $\mathcal{C}_1$ that form a partition of the data, it can be expressed as follows:

$$\texttt{DBI} = \frac{\delta_0 + \delta_1}{2 \cdot d(\mu_0, \mu_1)} \in [0, +\infty)$$

where $\delta_j = \mid \mathcal{C}_j \mid^{-1} \cdot \sum_{x \in \mathcal{C}_j} d(x, \mu_j)$ is the average distance of the cluster points to the centroid μ_j (*e.g.* d is Euclidean distance), also known as cluster diameter. It measures the ratio between the cluster sizes and the distance between the cluster centroids. A lower `DBI` is preferred since it indicates more distinct clusters.

[2] `www.scikit-yb.org/en/latest/api/cluster/elbow.html` for the python module.

In order to gain higher confidence in the selection of points of interest, we combine DBI with another metric in order to gather more clues about the best area to select. Since the k-means clustering where $k = 2$ is assumed to be able to distinguish quite well between -1 and $0/1$ coefficients, we can expect that one cluster to be roughly twice bigger than the other one (since each coefficient is chosen uniformly randomly in $[-1, 0, 1]$). We then use the following metric (cluster size ratio), that we expect to be as close to zero as possible:

$$\texttt{CSR} = \left| \frac{\text{size of smallest cluster}}{\text{size of biggest cluster}} - \frac{1}{2} \right| \in \left[0, \frac{1}{2}\right]$$

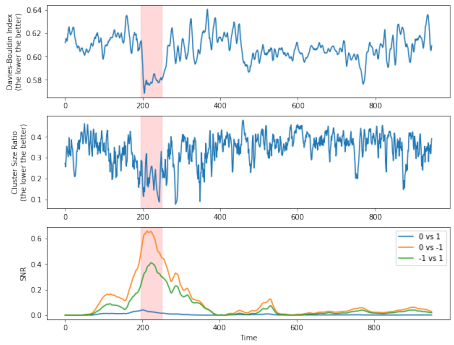


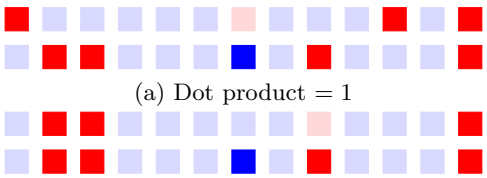
Fig. 3: Non-profiled leakage assessment, with area of interest (in red), performed on the collection of EM peaks of all the targeted coefficients. Top: DBI (the lower, the better). Middle: CSR (the lower, the better). Bottom: SNR with true labels (can only be performed when knowing the keys).

To sum up, a combination of DBI and CSR is used in order to detect points of interest without the knowledge of the true labels (*i.e.* coefficient values). In practice, our heuristic is as follows. First of all, for each time sample, we perform one clustering per time sample (with 2 clusters as suggested by the *elbow heuristic*) on all the extracted EM peaks. From the clustering labels, we calculate the corresponding DBI and CSR metrics. Second of all, we select areas combining expected metric values (low DBI and CSR). The results are displayed in Figure 3 (top and middle). One area is highlighted by the expected metrics behaviors. It combines both low DBI and low CSR. The area delimited by the red surface is chosen in order to mount the first phase of the attack. To demonstrate the validity of such approach, the figure 3 (bottom) shows the true Signal-to-Noise ratio SNR and the selected area of interest (between red dotted lines). The SNR is a classical metric in SCA. It requires to know the true labels to calculate it. In our case, it basically measures how bigger is the signal carried by the coefficient values (0, 1, -1) comparatively to the noise in the EM signal. Its formula is given by: $\texttt{SNR} = \mathbb{V}_{\mathbf{f}}\left(\mathbb{E}_{\mathbf{L}}(\mathbf{L} \mid \mathbf{f})\right) / \mathbb{E}_{\mathbf{f}}\left(\mathbb{V}_{\mathbf{L}}(\mathbf{L} \mid \mathbf{f})\right)$ where $\mathbf{L}$ is the random variable that represents the EM signal, and $\mathbf{f}$ is the variable that corresponds to the processed coefficients of f.

Let's notice that the figure 3 (bottom) also shows why a single trace SPA attack such as the one described in [1] is not possible. With a `SNR` lower than 1, it is difficult to distinguish the -1's with high confidence in a single trace. Even though the average EM activity is clearly different when processing a -1 rather than 0 or 1, the impact of the signal variance (noise) must be decreased by averaging enough traces to reveal which coefficients are equal to -1.

Retrieving random rotations by targeting the -1's

Clustering the traces and padding the vectors of labels. A k-means clustering with $k = 2$ is performed on the area of interest selected during the leakage assessment. For instance, labels 1 are assigned to the smallest cluster that should correspond to the coefficients -1, and 0 to the other cluster. For each trace, the corresponding vector of labels can be padded with 0's to compensate peaks that would have been discarded during the attack. As shown in Figure 1, the first and last peaks are not kept for all the traces. Therefore, all the vectors of labels are padded with two 0's.



(a) Dot product = 1

(b) Dot product = 3

Fig. 4: Dot product of clustering labels for two traces (top and bottom). Bottom: labels for a reference trace (fixed). Top: labels of a trace to be re-aligned to the ref. trace; they are rotated to find the best candidate (*i.e.* largest dot product result). Red squares: coefficients -1, blue squares: coefficients in $\{0, 1\}$. More vivid colors: clustering labels 1, light tone colors: labels 0.

Aligning vectors of labels and retrieving rotation indices. For each rotated key, we use the partially/erroneously recovered patterns of -1 coefficients provided by the clustering in order to find the secret rotation indices.

The strategy is to use the labels to re-align the traces. As shown in Figure 4, it is usually possible to find the rotation index between two traces. This index is expected to match with the rotation index of the vector of labels which provides the largest dot product with a reference vector of labels.

The tool used to re-align the shifted coefficients is a dot product (other tools like L1 distance were tried out without noticeable difference). This approach might be sensitive to the labels/trace used as a reference. The reference trace was chosen as one that minimizes the `CSR` metric (*i.e.* with a ratio (number of coeff. -1):(number of coeff. 0/1) being close to the expected ratio $1 : 2$). If the results are not satisfying, *i.e.* too many traces are required to recover the key,

we can simply choose another reference trace, randomly permute the data and try again. Each vector of labels is then compared to these reference vector. If they share of a big pattern, it can be detected with a peak when calculating the cyclical convolutions between the vectors of labels. In such case, we assume that it is likely that the two traces have been correctly aligned. Figures 5 display the differences between a trace for which we find the rotation index with high confidence, and one for which it is uncertain. The abscissa of the sharp peak points to the plausible rotation index to correctly re-align the two traces. When the scenario of Fig. 5 (right) comes up, we discard the corresponding labels/peak traces for the rest of the attack since it is most likely to add some noise in case the rotation index is not found.

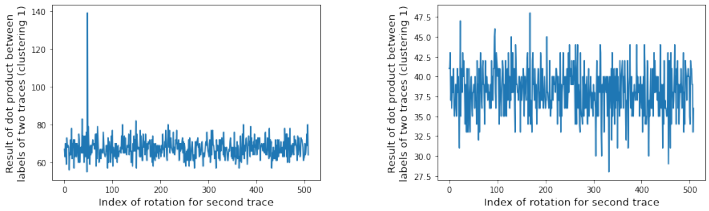


Fig. 5: Synchronizing two traces with (noisy) clustering labels. Left: a sharp peak = right re-alignment. Right: no sharp peak = uncertainty (trace discarded).

During the attack, this strategy allows to retrieve 100% of the rotation indices on the kept traces. Moreover, less than 10% of the traces were discarded during this process. Eventually, the kept traces are re-aligned with the reference trace which itself contains a random rotation. At the very end of the attack, this global rotation will be found with a simple brute force search.

Algorithm 3 Heuristic to locate the -1's and re-align the traces

1: Leakage assessment to identify an area of interest
2: k-means with $k = 2$ on this area of each EM peak
3: Pick a reference vector of labels $\mathbf{v}_{ref}$
4: Pad vectors of labels with 0's to match size $n = 509$
5: **for** each vector of labels **do**
6: Search for rotation providing a match with $\mathbf{v}_{ref}$
7: If no good match, discard the labels/trace for the rest of the attack
8: Apply majority vote to the aligned labels
9: **return** the assumed indices $\mathcal{I}_{-1}$ of coefficients -1, and $\mathcal{I}_{01} := [0, n) \setminus \mathcal{I}_{-1}$ the assumed indices of coefficients 0 and 1

Majority vote. Once most of the rotations are found, we can resynchronize the EM peaks and associate them to the right coefficient indices. Eventually, a majority vote is used on the synchronized labels to determine the right labels, *i.e.* which one of the following sets the key coefficients belong to: $\{-1\}$ or $\{0, 1\}$.

When the true keys are known, it is possible to check out the efficiency of this strategy. With around 20 traces, it is possible to retrieve almost 100% of the -1 coefficients. Eventually, we get a set of indices $\mathcal{I}_{-1}$ which is assumed to correspond with the locations of coefficients -1. By complementarity, the indices $\mathcal{I}_{01} = [0, n) \setminus \mathcal{I}_{-1}$ are those assumed to match with the locations of coefficients 0 and 1. The whole heuristic is summarized in Alg. 3.

3.4 Last phase of the attack: discriminating between 0 and 1

Initial (unsuccessful) tactics: clustering The initial strategy was to carry on with the clustering approach. Hopefully, it would be possible to distinguish between 0 and 1 by exploiting the same time window even though it is expected to require more traces in this case (because of close HW values). A clustering associated to a majority vote was then performed on the same area of interest. As expected with this approach, it was not possible to retrieve the keys with the given number of traces. The differences on the EM peaks between coefficients equal to 0 and 1 are too small for the attack to succeed in less than 100 traces. With known keys, this fact can be confirmed by analyzing the `SNR` between the true values. Figure 3 (bottom) shows that the `SNR` between classes 0 and 1 is around ten times smaller than the one between -1 and 0 or between -1 and 1. With several hundred of traces such approach might succeed. However, another approach is chosen in order to identify the coefficients 0 with less traces.

Changing tactics: finding collisions. If we perform a clustering on the EM peaks then the discrimination between 0 and 1 coefficients is more difficult. Nevertheless, we use the following crucial remark in order to identify more quickly the coefficients equal to 0.

- *Remark:* when processing a coefficient $f_i = 0$, the accumulator (`r3` reg. in fig. 2) remains unchanged: $\texttt{acc} \leftarrow \texttt{acc} + 0 \cdot c_{509-i} = \texttt{acc}$.
- *Hypothesis:* some parts of the EM peak (corresponding to the instructions `uxth` and `strh` in Fig. 2) should be similar to the preceding peak when corresponding to a coefficient 0, and differently when corresponding to a coefficient 1 or -1. This is called a collision between two consecutive peaks.

In other words, when processing a coefficient 0, we expect the difference in EM activity between the corresponding peak and the previous one to be quite low at a certain point in time. In such case, the difference is expected to be low since the same instructions (`uxth` and `strh`) and data (`r3`) are processed by the CPU.

Leakage assessment. In order to use the previous remarks, we perform a non-profiled leakage assessment by using a k-means clustering with $k = 2$ at every time sample on the difference of a peak and its predecessor for the whole set of EM traces. The Davies-Bouldin metric was used unsuccessfully. This may be explained by the fact that the cluster corresponding to a coefficient $-1/1$ might be of bigger variance/diameter (the value contained in `r3` register between two

consecutive peaks varying randomly, while it is not the case for a 0 coefficient). Overall, we combine two metrics in order to identify an area of interest:

- `CSR` metric (the smaller, the better): we expect one cluster for the 0's (smallest cluster $\mathcal{C}_0$), and another one for the $-1/1$'s (biggest cluster $\mathcal{C}_1$).
- Distance between the centroids $\|\mathbb{E}(\mathcal{C}_1) - \mathbb{E}(\mathcal{C}_0)\|$ (the higher, the better): since we expect the EM difference to be noticeably higher in average when processing $-1/1$, we target areas with very distant centroids.

Figure 6 shows the result. We can notice that it is possible to target areas that combine low `CSR` and high distance between centroids. Moreover, these areas of interest are located in the second part of the peaks. It means that they should indeed correspond to the targeted instruction `uxth` and `strh` in Fig. 2.

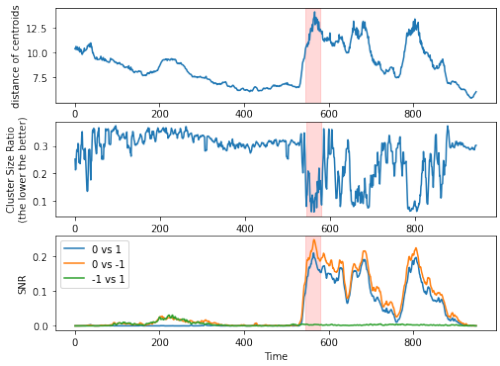


Fig. 6: Non-profiled Leakage assessment with area of interest (in red). Top: distance of centroids (the higher, the better). Middle: `CSR` (the lower, the better). Bottom: `SNR` with true labels (can only be performed when knowing the keys).

Classification heuristic based on L1-distance of consecutive EM peaks. The chosen heuristic (see Alg. 4) that allows to label the targeted coefficients (*i.e.* the ones assumed to be either 0 or 1 at this point) is simple. All the couples of traces corresponding to an EM peak, supposedly corresponding to 0 or 1 thanks to previous clustering, and its predecessor are gathered. In order to combine several time samples, we consider the L1-distances (sum of absolute difference) between these couples of peaks over the area of interest. Indeed, we do not want the differences from each time sample to compensate each other. Then, for each targeted coefficient, we average all the obtained distances. Other distances were tested without giving better results. Eventually, the classification is based on the absolute difference in EM activity. The L1-distance (on the selected area of interest) between two consecutive peaks will be smaller, in average, when the processed coefficient is 0. Hence, the heuristic simply consists in comparing the distances to the global average of all the calculated distances. If the average for one coefficient is lower than this global average, then it is assumed to be 0, otherwise it is labeled as 1. Let's note that one might run a k-means clustering over

the selected area of interest (the norm of the centroids would indicate the right labeling: 0 for the smallest centroid, 1 for the largest one). Even though it was tested, it did not improve the classification comparatively to the simple heuristic described in algorithm 4. Another method would consist in finding collisions through high correlation coefficients between consecutive peaks around the area of interest (see *e.g.* [31] for such kind of approach in a power analysis context). Given the low $\texttt{SNR}$, this approach did not allow us to retrieve the 0's.

One final remark about the attack phase is that we needed, for the attack to succeed, to consider the L1-distance on the area of interest whereas the leakage assessment was performing well on the raw difference at each time sample. When running the attack on the average of the raw differences, it was not possible to retrieve the keys. By calculating some $\texttt{SNR}$s, we can notice that it is actually the absolute difference that makes the 0's leaking more (see Figure 6 (bottom) for the detailed SNR). Surprisingly, the $\texttt{SNR}$ for the raw difference enables a better discrimination of the -1's. Even though the overall heuristic works, this shows the limitations of the non-profiled leakage assessment which remains a difficult task in such non-profiled SCA.

Finally, the Algorithm 4 summarizes the heuristic for this phase of the attack.

Algorithm 4 Heuristic to distinguish 0's from 1's

Require: n_t attack traces with N EM peaks each, a set of indices $\mathcal{I}_{01}$ for coefficients supposedly in $\{0, 1\}$

Leakage assessment to identify an area of interest $\mathcal{A}$
for each key coefficient f_i with $i \in \mathcal{I}_{01}$ **do**
    $\{(\mathbf{p}^{(i)}_{j_1-1}, \mathbf{p}^{(i)}_{j_1}), \ldots, (\mathbf{p}^{(i)}_{j_{n_i}-1}, \mathbf{p}^{(i)}_{j_{n_i}})\}$ collection of the EM peaks corresponding to f_i and their predecessors
    $m_i \leftarrow \mathbb{E}_{h\in[1,n_i]}\{\texttt{dist}_{|\mathcal{A}}(\mathbf{p}^{(i)}_{j_h-1}, \mathbf{p}^{(i)}_{j_h})\}$ average of the (L1-)distances on the area of interest $\mathcal{A}$
$m \leftarrow \mathbb{E}(\{m_i \mid i \in \mathcal{I}_{01}\})$
for each key coefficient f_i **do**
    Classification criteria: if $m_i \leq m$ then $f_i \leftarrow 0$ else $f_i \leftarrow 1$ # *based on hyp. that smaller difference in EM activity corresponds to coeff* 0
return the assumed locations $\mathcal{I}_0$ and $\mathcal{I}_1$ of coefficients 0 and 1 respectively

3.5 Summary of attack and results

Alg. 5 sums up the attack. The traces are processed with sets of traces of increasing size. Among the 200 keys, the average and standard deviation of the number of traces to retrieve all the coefficients with no error are resp. $\mu \sim 45$ and $\sigma \sim 10$, with a min. of 23 and a max. of 69. Fig. 7 (left) shows the average number of retrieved coefficients given the size of the attack trace set.

Residual entropy. As we might find the rotation indices up to a global rotation index (the one from the first reference trace) thanks to the first clustering, $\log_2(n)$ bits of entropy remain. By assuming that for the final key we retrieve only $n - x$ coefficients, the residual entropy of the final brute-force correction is bounded by

the following formula (searching from 1 to x errors; one erroneous coefficient can be replaced by 2 values): $\log_2(n) + \sum_{i=1}^{x} \binom{n}{i} 2^i$. If x becomes too large, using a lattice approach could be more efficient (see [13]). Fig. 7 (right) shows that with less than 20 traces, the security of the implementation drops below 100 bits.

Algorithm 5 NON-PROFILED SIDE-CHANNEL ATTACK ON NTRU

Require: attack traces

1: Pre-processing traces (peak selection, alignment)
2: Apply Alg. 3 to get $\mathcal{I}_{-1}$ and $\mathcal{I}_{01}$, supposedly being the locations (up to a global rotation) of key coefficients -1 and $0/1$ respectively
3: Apply Alg. 4 with $\mathcal{I}_{01}$ and corresponding traces to get $\mathcal{I}_0$ and $\mathcal{I}_1$, supposedly being the locations (up to the same global rotation) of key coefficients 0 and 1 respectively
4: Build (rotated) candidate out of indices $\mathcal{I}_{-1}$, $\mathcal{I}_0$ and $\mathcal{I}_1$
5: Brute force search for last global rotation and potential remaining few incorrect coefficients

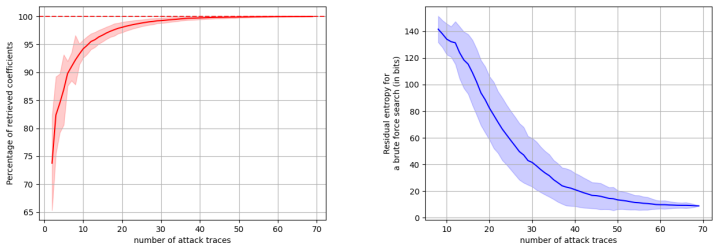


Fig. 7: Results on 200 keys: average ± 1 standard deviation. Left: percentage of coefficients retrieved. Right: residual entropy (in bits) for brute force correction.

Last remarks. We stress that it should be possible to improve the results by tuning the heuristics better. For instance, shuffling the set of traces can lead to different, sometimes better, results. Moreover, the clustering outliers were not treated as such. An outlier is an element that lies unusually farther to the centroid of its own cluster than the majority of the elements. These outliers might potentially be mislabeled, therefore adding noise in the results of majority votes when trying to locate the -1. Let's also notice that, since the absolute difference in EM activity between consecutive peaks leaks less than the value of the coefficients themselves (see Fig. 3 (bottom) vs Fig. 6 (bottom)), it wouldn't be more advantageous to use the 0's to find the rotation indices.

Yet the heuristic is perfectible, it was possible to show that such a non profiled side-channel attack against a implementation of NTRU which embeds some low-cost countermeasures works well with pretty few traces against a realistic setup.

4 Conclusion

In this paper, we show how a non-profiled side-channel analysis can defeat a somewhat secure (secret key rotation) NTRU `Decrypt` algorithm. In particular,

we assume a weak attacker model which cannot profile the targeted device nor use any chosen ciphertext. Despite this restrictive environment, we show that the number of required traces to recover the key is so low that it makes our attack practical. NTRU (and its alternate NTRUPrime) was a finalist of the NIST PQC competition, mostly because of its efficiency and well studied theoretical security. Our work shows that it might be challenging to design countermeasures, even against weak attackers, without noticeably impacting the efficiency of the primitive. Masking the secret key might be a solution but it would decrease the intrinsic efficiency and increase the RAM consumption by at least a factor 2 [12, 6]. The security overhead and the requirement of additional security can be scaled down by reusing contemporary secure co-processors to improve the polynomial arithmetic, or by developing new PQC specific secure hardware accelerators.

More generally, we believe the academic community and the cryptography industry will be able to provide innovative solutions in order to make embedded post-quantum cryptographic implementations both secure and efficient.

References

1. An, S., Kim, S., Jin, S., Kim, H., Kim, H.: Single Trace Side Channel Analysis on NTRU Implementation. Applied Sciences **8**(11) (2018)
2. ANSSI: Technical position paper - ANSSI views on the Post-Quantum Cryptography transition (2022)
3. Askeland, A., Rønjom, S.: A Side-Channel Assisted Attack on NTRU. Presented at the Third PQC Standardization Conference and published in IACR Cryptol. ePrint Arch. (2021)
4. Bernstein, D.J., Brumley, B.B., Chen, M.S., Chuengsatiansup, C., Lange, T., Marotzke, A., Peng, B.Y., Tuveri, N., van Vredendaal, C., Yang, B.Y.: NTRU Prime (2021)
5. Bernstein, D.J., Brumley, B.B., Chen, M., Tuveri, N.: OpenSSLNTRU: Faster post-quantum TLS key exchange. IACR Cryptol. ePrint Arch. p. 826 (2021), to appear in USENIX 2022
6. Bos, J.W., Gourjon, M., Renes, J., Schneider, T., van Vredendaal, C.: Masking kyber: First- and higher-order implementations. Cryptology ePrint Archive, Report 2021/483 (2021)
7. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Cryptographic Hardware and Embedded Systems - CHES 2004. vol. 3156, pp. 16–29. Springer (2004)
8. BSI: Migration zu Post-Quanten-Kryptografie - Handlungsempfehlungen des BSI (2020)
9. CACR: National Cryptographic Algorithm Design Competition (2018), available at `https://www.cacrnet.org.cn/site/content/838.html`
10. Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., M.Schank, J., Schwabe, P., Whyte, W., Zhang, Z.: NTRU (2021)
11. Cloudflare: The TLS Post-Quantum Experiment (2019)
12. Coron, J.S., Gérard, F., Montoya, S., Zeitoun, R.: High-order polynomial comparison and masking lattice-based encryption. Cryptology ePrint Archive, Report 2021/1615 (2021)

13. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference. vol. 12171, pp. 329–358. Springer (2020)
14. Davies, D. L. and Bouldin, D. W.: A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979)
15. Dunn, J.C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics **3**(3), 32–57 (1973)
16. Google: The Chromium Projects – The Chromium Projects (2019)
17. Greuet, A.: Smartcard and Post-Quantum Crypto (2021), available at `https://csrc.nist.gov/Presentations/2021/smartcard-and-post-quantum-crypto`
18. Hamburg, M., Hermelink, J., Primas, R., Samardjiska, S., Schamberger, T., Streit, S., Strieder, E., van Vredendaal, C.: Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 88–113 (2021)
19. Heyszl, J., Ibing, A., Mangard, S., Santis, F.D., Sigl, G.: Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8419, pp. 79–93. Springer (2013). https://doi.org/10.1007/978-3-319-08302-5“6
20. Hoffstein, J., Pipher, J., Silverman, J.: NTRU: A ring-based public key cryptosystem. ANTS 1998 **1423** (1998)
21. Huang, W., Chen, J., Yang, B.: Power Analysis on NTRU Prime. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 123–151 (2020)
22. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Advances in Cryptology - CRYPTO '96. vol. 1109, pp. 104–113. Springer (1996)
23. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Advances in Cryptology - CRYPTO '99. vol. 1666, pp. 388–397. Springer (1999)
24. MacQueen, J.: Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967). (1967)
25. Moody, D.: Post-Quantum Cryptography NIST's Plan for the Future (2016)
26. Nielsen, F.: Hierarchical Clustering, pp. 195–211 (02 2016)
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
28. 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)
29. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. **26**(5), 1484–1509 (Oct 1997)
30. Sim, B.Y., Kwon, J., Lee, J., Kim, I.J., Lee, T., Han, J., Yoon, H., Cho, J., Han, D.G.: Single-Trace Attacks on the Message Encoding of Lattice-Based KEMs. Cryptology ePrint Archive, Report 2020/992 (2020)
31. Zheng, X., Wang, A., Wei, W.: First-order collision attack on protected NTRU cryptosystem. Microprocessors and Microsystems **37**(6), 601–609 (2013)