Power Analysis on NTRU Prime

Wei-Lun Huang, Jiun-Peng Chen and Bo-Yin Yang

Academia Sinica, Taiwan,
271828182euler@gmail.com, jpchen@ieee.org, byyang@iis.sinica.edu.tw

Abstract. This paper applies a variety of power analysis techniques to several implementations of NTRU Prime, a Round 2 submission to the NIST PQC Standardization Project. The techniques include vertical correlation power analysis, horizontal in-depth correlation power analysis, online template attacks, and chosen-input simple power analysis. The implementations include the reference one, the one optimized using **smladx**, and three protected ones. Adversaries in this study can fully recover private keys with one single trace of short observation span, with few template traces from a fully controlled device similar to the target and no *a priori* power model, or sometimes even with the naked eye. The techniques target the constant-time generic polynomial multiplications in the product scanning method. Though in this work they focus on the decapsulation, they also work on the key generation and encapsulation of NTRU Prime. Moreover, they apply to the ideal-lattice-based cryptosystems where each private-key coefficient comes from a small set of possibilities.

Keywords: Ideal Lattice Cryptography · Single-Trace Attack · Correlation Power Analysis · Template Attack · Simple Power Analysis · NTRU Prime

1 Introduction

Due to Shor's algorithm [Sho97], quantum computing is a potential threat to all public-key cryptosystems based on the hardness of integer factorization and discrete logarithms, including RSA [RSA78], Diffie-Hellman key agreement [DH76], ElGamal encryption [Gam85], and ECDSA [JMV01]. Recently, quantum computers have been estimated as arriving in 10 to 15 years [Sni16, WLYZ18], which identifies an urgent need for the examination and standardization of post-quantum cryptosystems. This need led to the NIST Post-Quantum Cryptography Standardization Project [Nat17]. The candidates are mostly based on lattices [BCD+16, DKL+18, BDK+18], error correction codes [McE78, BBC+18], multivariate quadratic equations [DS05], symmetric-key primitivies [BHH+15], and supersingular isogenies [JF11].

Unfortunately, quantum resistance is no guarantee of practical security. There has been a large amount of work on the implementation attacks against post-quantum cryptosystems. [TE15] provides a comprehensive collection of fault analysis and side-channel analysis on various post-quantum schemes. [EFGT17], [KAJ17], and [PSKH18] present more cutting-edge side-channel analyses on digital signatures. [EFGT17] applies electromagnetic analysis to BLISS, and achieves full key recovery from one single trace using integer linear programming. [KAJ17] features three zero-value attacks on supersingular isogeny Diffie-Hellman using refined power analysis. [PSKH18] proposes the correlation power

This work was supported by Taiwan Ministry of Science and Technology grant MOST108-2218-E-001-001 and 108-2221-E-001-008 and Academia Sinica Investigator Award AS-IA-104-M01. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Ministry of Science and Technology" (or other funding agencies). Date: 2019.10.15. Future updates including software at https://eprint.iacr.org/2019/100.

analysis on Rainbow and Unbalanced Oil-and-Vinegar, two digital signatures based on multivariate quadratic equations, to fully recover the secrets in use.

Our Contributions NTRU Prime [BCLvV16], a Round 2 submission to the NIST PQC Standardization Project, is based on ideal lattices. This submission contains two schemes: Streamlined NTRU Prime and NTRU LPRime. Streamlined NTRU Prime is a variant of the classic NTRU [HPS98], and NTRU LPRime shares a similar structure with NewHope [ADPS16, LPR10]. However, their reference implementations are not subject to the previous attacks against lattice-based schemes. These attacks target the implementations with data-dependent timing differences [SW07] and the ones which employ the operand scanning method [AKJ$^+$18, ATT$^+$18], sparse multiplication [LSCH10, KY12, WZW13, ZWW13, AKJ$^+$18, SMS19] or the NTT network [PPM17] for polynomial multiplications. In contrast (and somewhat unusually), the reference implementation of NTRU Prime is constant-time and generic, realizing polynomial multiplications with the product scanning method.

This paper applies a variety of power analysis techniques to several implementations of NTRU Prime. The techniques include vertical correlation power analysis (VCPA) [BCO04], horizontal in-depth correlation power analysis (HIDCPA) [CFG$^+$10], online template attacks (OTA) [BCP$^+$14], and chosen-input simple power analysis (CISPA) [KJJ99]. The implementations include the reference one [BCLvV16], one optimized using DSP instructions, and some protected ones [LSCH10]. Adversaries can fully recover private keys with one single trace, with few template traces and no *a priori* power model, or even with the naked eye.

We run the experiments on Cortex-M4-based STM32F3 [STM18] and STM32F4 [STM16] microcontrollers. Cortex-M4 is the prevalent platform for embedded post-quantum implementations and designs [KMRV18, HOKG18, OSPG18, SBG$^+$18, BKS19, KRS19, SMS19]. NIST has specified Cortex-M4 for benchmarking in its contest [TPG15]. Cortex-M4 also provides DSP instructions like **smuad(x)** and **smlad(x)** [ARM11], allowing the optimization of product scanning at the instruction level.

This work demonstrates the private-key recovery from the polynomial multiplication in decapsulation. However, because HIDCPA and OTA are single-(target-)trace attacks on random inputs, they are also able to reveal private keys from NTRU LPRime's key generation and the seeds of session keys from both schemes' encapsulations. The ideal-lattice-based cryptosystems where each private-key coefficient comes from a small set of possibilities [ADPS16, HRSS17, HPS$^+$17, KMRV18, BDK$^+$18] may well succumb to the approaches in this study.

Even if NTRU Prime optimizes its polynomial multiplications with Karatsuba's method [Kar63, WP06] and Toom's method [Too63, CA69], the approaches may remain effective. Karatsuba's method itself does not prevent the VCPA, the OTA, and the CISPA on lowest-level multiplications. If low-level schoolbook multiplications are sufficiently long, then HIDCPA works, too. Unfortunately, if the optimized version uses Toom-k as the first layer, the approaches can only reveal the first and last $1/k$ of private-key coefficients. How to adapt them to a fully optimized NTRU Prime [BCLvV16] in pursuit of full private-key recovery is worth further investigation.

Prior Work In the field of side-channel analysis on lattice-based encryption, [SW07] and [LSCH10] are the classics. [SW07] describes a timing attack against NTRUEncrypt exploiting the variation in the number of hash calls during decryption. [LSCH10] not only applies simple power analysis and correlation power analysis to a typical NTRU software implementation, but also provides the corresponding countermeasures.

More advanced techniques are introduced in [MNY17], [PPM17], [AKJ$^+$18], [ATT$^+$18], [BFM$^+$18], and [SMS19]. [MNY17] proposes at the architecture level an NTRUEncrypt

FPGA implementation resistant to first-order differential power analysis. [PPM17] exploits side-channel leakages in the Number Theoretic Transform, features full private-key recovery from one single trace, and breaks the masked implementations of some lattice-based schemes. [AKJ+18] performs single-trace power analysis on two versions of NTRU. [ATT+18] mounts horizontal differential power analysis on NewHope and Frodo to reveal private keys with >99% success rate from one single trace. [BFM+18] applies extend-and-prune template attacks to the challenges beyond the scope of [ATT+18]. [SMS19] implements an additive masking for NTRUEncrypt with little overhead using Cortex-M4 SIMD instructions, and performs a second-order attack against the additive masking.

In all the literature, there does not seem to be an attack against NTRU Prime, or against polynomial multiplication using the product scanning method in general. However, it is worth noting that [HMHW09] and [UW14] mount correlation power analysis on multi-precision integer multiplication using the product scanning method in ECDSA and optimal-Ate pairings, respectively. Also, [JB16] launches (repeated) single-trace correlation/clustering attacks against the operand-scanning field multiplications in elliptic curve scalar multiplication with precomputations, and claims its applicability to the product scanning method.

2 Preliminaries

2.1 NTRU Prime

NTRU Prime [BCLvV16] is a Round 2 candidate in the NIST Post-Quantum Cryptography Standardization Project [Nat17]. It features polynomial rings distinct from those of typical Ring-LWE-based cryptosystems and NTRU to avoid potential algebraic attacks. In NTRU Prime there are two key-encapsulation mechanisms based on ideal lattices: Streamlined NTRU Prime and NTRU LPRime.

Let $\mathbb{Z}/m\mathbb{Z}$ be represented by $(-m/2, m/2] \cap \mathbb{Z}$. For a given prime p and an arbitrary $m \in \mathbb{Z}^+$, R and R_m refer to $\mathbb{Z}[x]/(x^p - x - 1)$ and $(\mathbb{Z}/m\mathbb{Z})[x]/(x^p - x - 1)$, respectively. A polynomial is *small* if all of its coefficients belong to $\{-1, 0, 1\}$, and *weight-w* if exactly w of its coefficients are nonzero. If not specified, the following expressions for a polynomial of degree $n \in \mathbb{N}$ are interchangeable: $\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_n x^n = \sigma$. The terms above help describe Streamlined NTRU Prime and NTRU LPRime concisely.

Streamlined NTRU Prime has positive integer parameters p, q, and w: p and q are primes; $2p \geq 3w$; $q \geq 16w + 1$; $x^p - x - 1$ is irreducible in $(\mathbb{Z}/q\mathbb{Z})[x]$. The following introduction to its key generation, encapsulation, and decapsulation skips error detection, encoding, and decoding due to their irrelevance to this paper.

■ Key Generation

1. Pick randomly and uniformly a *small* polynomial $g \in R$ invertible in R_3.
2. Pick randomly and uniformly a *small weight-w* polynomial $f \in R$.
3. Set $h = (g/(3f) \text{ in } R_q)$ as the public key and $(f, (g^{-1} \text{ in } R_3))$ as the private key.

■ Encapsulation

1. Pick randomly and uniformly a *small weight-w* polynomial $r \in R$.
2. Hash r to obtain the session key K.
3. Obtain $c \in R$ by rounding each coefficient of $(h \times r \text{ in } R_q)$ to the nearest multiple of 3, and set c as the ciphertext.

■ **Decapsulation**

1. Obtain $r = (((c \times 3f \text{ in } R_q) \bmod 3) \times g^{-1} \text{ in } R_3)$.
2. Hash r to obtain the session key K.

NTRU LPRime has positive integer parameters p, q, w, δ, and I: p and q are primes; $8 \mid I$; $2p \geq 3w$; $q \geq 16w + 2\delta + 3$; $p \geq I$; $x^p - x - 1$ is irreducible in $(\mathbb{Z}/q\mathbb{Z})[x]$. Also, it includes the following five functions:

- *Hash*: a hash function producing two fixed-length strings: cipher key and session key from each I-bit string
- *Generator*: producing a polynomial $\in R_q$ from each seed string
- *Small*: producing a *small weight-w* polynomial $\in R$ from each cipher key
- *Top*: mapping each vector $\in (\mathbb{Z}/q\mathbb{Z})^I$ to a fixed-length string
- *Right*: mapping each string in the image of *Top* to a vector $\in (\mathbb{Z}/q\mathbb{Z})^I$ such that $\forall C \in (\mathbb{Z}/q\mathbb{Z})^I$, $\mathit{Right}(\mathit{Top}(C)) - C \in \{0, 1, \cdots, \delta\}^I$

The following introduction to its key generation, encapsulation, and decapsulation skips error detection, encoding, and decoding again due to their irrelevance to this paper.

■ **Key Generation**

1. Pick randomly and uniformly a seed S.
2. Pick randomly and uniformly a *small weight-w* polynomial $a \in R$.
3. Obtain $A \in R$ by rounding each coefficient of $(a \times \mathit{Generator}(S) \text{ in } R_q)$ to the nearest multiple of 3.
4. Set (S, A) as the public key and a as the private key.

■ **Encapsulation**

1. Pick randomly and uniformly an I-bit string $r = r_0 r_1 \cdots r_{I-1}$.
2. Obtain the cipher-key-session-key pair $(k, K) = \mathit{Hash}(r)$.
3. Obtain $b = \mathit{Small}(k)$.
4. Obtain $B \in R$ by rounding each coefficient of $(b \times \mathit{Generator}(S) \text{ in } R_q)$ to the nearest multiple of 3.
5. Obtain $C = (C_0, C_1, \cdots, C_{I-1})$: $C_j = ((b \times A \text{ in } R_q)_j + r_j \times (q-1)/2) \bmod q$.
6. Obtain $\tilde{C} = \mathit{Top}(C)$, and set $(B, \tilde{C})$ as the ciphertext.

■ **Decapsulation**

1. Obtain $r = r_0 r_1 \cdots r_{I-1}$: $r_j = 1\{(\mathit{Right}(\tilde{C})_j - (a \times B \text{ in } R_q)_j + 4w + 1) \bmod q < 0\}$.
2. Obtain the session key K from $(k, K) = \mathit{Hash}(r)$.

The polynomial multiplication in R_q in decapsulation is the operation of interest. In NTRU-like cryptosystems there are three common ways to realize such polynomial multiplications [LSCH10, KY12, WZW13, ZWW13, BCLvV16, AKJ+18, SMS19]. The two inputs of degree $< p$ here are the *small* private key f (or a) and the known ciphertext c (or B).

- **Operand Scanning Method** [HW11]: viewing $f \times c$ as the superposition of $f_i \times c$
- **Sparse Multiplication**: expressing f as $(\{i : f_i = 1\}, \{j : f_j = -1\})$ and substituting integer addition/subtraction for integer multiplication
- **Product Scanning Method** [HW11]: calculating $(f \times c)_i$ one by one

Table 1 shows the details. Conventional NTRU implementations favor the operand scanning method and sparse multiplication. Hence, these two have been intensively studied in the field of power analysis. In contrast, NTRU Prime adopts the product scanning method in its reference implementation. This method has received little attention in the literature, so this paper features a comprehensive set of power analysis on it.

Table 1: The Polynomial Multiplication in R_q in Decapsulation

Initialization: $e_i \leftarrow 0, \forall i \in \{0, 1, \cdots, (2p-2)\}$		
Operand Scanning Method	**Sparse Multiplication**	**Product Scanning Method**
	$\forall i$: $f_i = 1$	for $i = 0$ to $(p-1)$
for $i = 0$ to $(p-1)$	for $k = 0$ to $(p-1)$	for $j = 0$ to i
for $j = 0$ to $(p-1)$	e_{i+k} += c_k (mod q)	e_i += $f_{i-j} \times c_j$ (mod q)
e_{i+j} += $f_i \times c_j$ (mod q)	$\forall j$: $f_j = -1$	for $i = p$ to $(2p-2)$
	for $k = 0$ to $(p-1)$	for $j = (i-p+1)$ to $(p-1)$
	e_{j+k} −= c_k (mod q)	e_i += $f_{i-j} \times c_j$ (mod q)
Output: $e(x)$ mod $P(x) \Rightarrow P(x) = x^p - x - 1$ for NTRU Prime and $x^p \pm 1$ for NTRUEncrypt		

Although the experiments in this article only focus on the recovery of f from $(c \times 3f$ in $R_q)$ in Streamlined NTRU Prime and a from $(a \times B$ in $R_q)$ in NTRU LPRime, the formula $h = (g/(3f)$ in $R_q)$ assures the recovery of g in Streamlined NTRU Prime with the knowledge of public key. Furthermore, the single-(target-)trace attacks on random inputs in this article can easily adapt to the multiplications in encapsulation/key generation for the session-key/secret-key recovery.

2.2 Power Analysis

Side-channel analysis can break an implementation without breaking the underlying cryptosystem under its design assumptions. First, it collects side-channel leakages (such as execution time [Koc96], power consumption [KJJ99], and electromagnetic radiation [vE85]) from cryptographic devices. Then it identifies the relations between such leakages and the operations being executed or the intermediate values being processed. Finally, it employs a series of data processing, observation, and statistical analysis to reveal sensitive information about the cryptographic primitives in use.

Power analysis is a popular branch of side-channel analysis. The classic instances include simple power analysis (SPA) [KJJ99], correlation power analysis (CPA) [BCO04], and profiling attacks [CRR02]. This paper points out that NTRU Prime is subject to all of them. In general, power analysis consists of four steps [MOP07]. First, it targets specific intermediate values to decompose the entire key space into several tiny search spaces. Then it models the expected power consumption of the target device for these intermediate values. After power trace recording, it produces an optimal guess for each search space. Finally, sensitive information such as private keys is derived from these optimal guesses. In this context, ciphertexts are assumed accessible at low cost.

SPA, CPA, and profiling attacks follow different ways to model the expected power consumption and produce optimal guesses [MOP07]. SPA only cares about the target's data-dependent power characteristics that can be captured within one or few executions and identified with simple arithmetic or even visual inspection. The knowledge of such power characteristics usually requires deep understanding of the target device. CPA applies a simple power model to its target device (Hamming weight model for microcontrollers and Hamming distance model for FPGA). Then it decides on optimal guesses based on the Pearson correlation coefficients between expected power consumption and its counterpart in reality, for all candidates in the search space. Profiling attacks construct multivariate Gaussian distributions or typical power sample sequences from the measurements in its profiling stage. Then with these highly customized power models, it selects suitable statistical analysis to decide between optimal guesses.

3 Power Analysis on the Unprotected NTRU Prime

3.1 Vertical Correlation Power Analysis (VCPA)

This VCPA targets the e_{p-1} calculation in the product scanning method in Table 1. The only coefficient of e involving all the coefficients of f and c is e_{p-1}, so its calculation is highly controllable, and the power consumption is rich in data dependencies. Since the private key f is *small* and *weight-w*, that is $f = f_{b_1}x^{b_1} + f_{b_2}x^{b_2} + \cdots + f_{b_w}x^{b_w}, b_1 < b_2 < \cdots < b_w$, the e_{p-1} calculation contains w meaningful multiply-accumulate-reduce operations and thus w interesting internal states $e_{p-1,1}, e_{p-1,2}, \cdots, e_{p-1,w}$, as defined in Figure 1. This VCPA contains two stages: The first reveals $(b_w, b_{w-1}, f_{b_w}, f_{b_{w-1}})$ all at once, while the second reveals the other (b_k, f_{b_k}) sequentially. The first stage calculates $e_{p-1,2}$ from each hypothesis under examination, checking the similarity between the corresponding expected and real power consumptions. The second stage calculates each $e_{p-1,w-k+1}$ similarly.

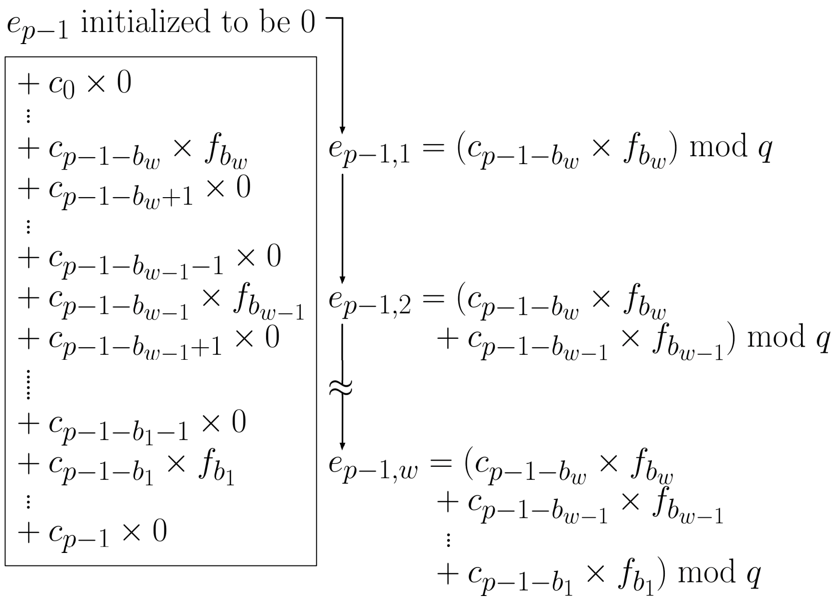

e_{p-1} initialized to be 0
$+ c_0 \times 0$
$\vdots$
$+ c_{p-1-b_w} \times f_{b_w}$
$+ c_{p-1-b_w+1} \times 0$
$\vdots$
$+ c_{p-1-b_{w-1}-1} \times 0$
$+ c_{p-1-b_{w-1}} \times f_{b_{w-1}}$
$+ c_{p-1-b_{w-1}+1} \times 0$
$\vdots$
$+ c_{p-1-b_1-1} \times 0$
$+ c_{p-1-b_1} \times f_{b_1}$
$\vdots$
$+ c_{p-1} \times 0$
$e_{p-1,1} = (c_{p-1-b_w} \times f_{b_w}) \bmod q$
$e_{p-1,2} = (c_{p-1-b_w} \times f_{b_w} + c_{p-1-b_{w-1}} \times f_{b_{w-1}}) \bmod q$
$\approx$
$e_{p-1,w} = (c_{p-1-b_w} \times f_{b_w} + c_{p-1-b_{w-1}} \times f_{b_{w-1}} \vdots + c_{p-1-b_1} \times f_{b_1}) \bmod q$

Figure 1: The Internal States of e_{p-1}

The following assumption helps accelerate this VCPA: The chosen power model suits the target device so well that in each search space, the correct hypothesis gives a correlation coefficient far away from those of the rest. Because every internal state corresponds to the same kind of multiply-accumulate-and-reduce, this assumption allows adversaries to set a fixed threshold shared by all the search spaces. During this VCPA, whenever the hypothesis being examined yields a coefficient that crosses the threshold, we will assume that this hypothesis is optimal in the current search space.

Furthermore, the assumption above ensures that there is a wide range of eligible thresholds, so adversaries can efficiently select a threshold leaving exactly one survivor per search space. Note that after computing a few correlation coefficients in the first stage, we should be able to identify the numerical gap between correct and incorrect hypotheses.

Algorithm 1 shows this VCPA in detail. It involves N independent multiplications of random ciphertexts c and a fixed secret key f. Each e_{p-1} calculation leads to a power trace of size L. Algorithm 1 employs the Hamming weight power model [KJJ99, MD99] by default due to its simplicity and prevalence in the microcontroller power modeling. Besides, Algorithm 1 by default considers higher correlation coefficients "better", and defines the optimal guess in a search space as the hypothesis with the "best" correlation coefficient. In this case, a correlation coefficient crosses the threshold if it is above the fixed threshold.

Algorithm 1 Vertical Correlation Power Analysis on NTRU Prime

```
Input:
    a random array of ciphertexts $c[N] \in ((3\mathbb{Z})[x] \cap R_q)^N$
    the transposed power-sample matrix $P[L][N]$ of the $e_{p-1}$ calculation
    THRESHOLD to sieve out the correct hypothesis
Output: a small weight-$w$ polynomial $f \in R$
 1: $f \leftarrow 0$
 2: for $j = 1$ to $(p-w+1)$ for $i = 0$ to $(j-1)$ do                  ▷ The First Stage
 3:     Among all the $(x, y, id_{w-1}) \in \{-1,1\}^2 \times \{1, 2, \cdots, L\}$,
 4:         identify the one giving the "best" correlation coefficient $\rho_{x,y,id_{w-1}}$ between
 5:         $P[id_{w-1}][0:N]$ and HammingWeight$((x \times c_i[0:N] + y \times c_j[0:N]) \bmod q)$
 6:         as $(x^*, y^*, id^*_{w-1})$
 7:     if $\rho_{x^*,y^*,id^*_{w-1}} >$ THRESHOLD then
 8:         $(f_{p-1-i}, f_{p-1-j}, i_{start}, wt) \leftarrow (x^*, y^*, (j+1), (w-2))$
 9:         $e_{p-1}[0:N] \leftarrow (x^* \times c_i[0:N] + y^* \times c_j[0:N]) \bmod q$       ▷ $e_{p-1,2}$
10:         break
11: for $i = i_{start}$ to $(p-1)$ do                                   ▷ The Second Stage
12:     Among all the $(x, id_{wt}) \in \{-1,1\} \times \{(id^*_{wt+1}+1), \cdots, L\}$,
13:         identify the one giving the "best" correlation coefficient $\rho_{x,id_{wt}}$ between
14:         $P[id_{wt}][0:N]$ and HammingWeight$((e_{p-1}[0:N] + x \times c_i[0:N]) \bmod q)$
15:         as $(x^*, id^*_{wt})$
16:     if $\rho_{x^*,id^*_{wt}} >$ THRESHOLD then
17:         $(f_{p-1-i}, wt) \leftarrow (x^*, (wt-1))$
18:         $e_{p-1}[0:N] \leftarrow (e_{p-1}[0:N] + x^* \times c_i[0:N]) \bmod q$        ▷ $e_{p-1,w-wt}$
19:     return $f$ once $wt = 0$
```

The approach takes probabilistic linear time in terms of p, if a $\Omega(1)$ lower bound τ for w/p exists in view of security concerns. In any case the second stage needs less than p iterations of testing $\pm c_i$. Assume the random variables $X_1 = (p - b_w)$ and $X_2 = (b_w - b_{w-1})$ satisfy the condition $E[X_1] = E[X_2] = p/w$. Then the first stage needs $E[(X_1 + X_2 - 2) \times (X_1 + X_2 - 1)/2 + X_1] < p/\tau$ iterations of testing $\pm c_i \pm c_j$ on average:

$$E[\frac{(X_1 + X_2 - 2) \times (X_1 + X_2 - 1)}{2} + X_1] < E[\frac{(X_1 + X_2)^2}{2}] < E[\frac{(X_1 + X_2) \times p}{2}] \leq \frac{p}{\tau}$$

There are practical considerations behind Algorithm 1. The first stage views f_{b_w} and $f_{b_{w-1}}$ as a pair to avoid the potential confusion between the access to c_i and the update of e_{p-1} from 0 to $e_{p-1,1}$. Also, to examine the hypotheses of high *a priori* probability first, the nested loop takes the form of "for $j = 1$ to $(p - w + 1)$ for $i = 0$ to $(j - 1)$ do" rather than "for $i = 0$ to $(p - w)$ for $j = (i + 1)$ to $(p - w + 1)$ do". This reduces both time consumption and the required number of traces. Moreover, this VCPA assumes the relative order of operations in an implementation to be the same as its counterpart in source code, so the second stage just cares about the samples after the latest id^*_{wt+1}.

Some may suggest that Algorithm 1 judge a correlation coefficient by its absolute value for the sake of generality. However, this makes the first stage prone to failure due to troublesome false positives: Every e_i is initially zero and for 32-bit microcontrollers (e.g. our target devices) $\text{HammingWeight}(x) \approx 32 - \text{HammingWeight}(-x)$, so $(b_{286}, b_{285}, -f_{b_{286}}, -f_{b_{285}})$ is a false positive which will lead to confusion during the first step. Similarly, under this general design, $(-f_{760}, -f_{759}, \cdots, -f_{761-m})$ is a false positive in the first block recovery of the HIDCPA in subsection 3.2. When deciding on the power model in use, adversaries can take into consideration their measurement setups so that they only need to check for either positive or negative correlation coefficients as hypotheses, not both.

3.2 Horizontal In-Depth Correlation Power Analysis (HIDCPA)

Algorithm 2 Horizontal In-Depth Correlation Power Analysis on NTRU Prime

```
Input:
    a random ciphertext $c \in ((3\mathbb{Z})[x] \cap R_q)$
    the power trace $P$ of the $e_{p-1} \cdots e_{p-2+l}$ calculation
    THRESHOLD to prune incorrect candidates
Output: a small weight-w polynomial $f \in R$
 1: $(startIdx, revKey) \leftarrow (0, [\,])$
 2: while $startIdx \le (p - m)$ do                ▷ The $f_{p-startIdx-(1,2,\cdots,m)}$ Recovery
 3:     $realArr \leftarrow [\,]$
 4:     for $i = 0$ to $(m-1)$ for $j = 0$ to $(l-1)$ do
 5:         $(x, y) \leftarrow ((p-1+j), (startIdx + i + j))$
 6:         $realArr$.append(the sample $\in P$ of $e_x \mathrel{+}= f_{x-y} \times c_y \pmod q$)
 7:     $(optGuess, bestCorr) \leftarrow candidateConstruct([\,])$
 8:     if $optGuess$ is empty then                ▷ Error Correction Mechanism
 9:         $startIdx \mathrel{-}= \lfloor m/2 \rfloor$ and $revKey \leftarrow revKey[: startIdx]$
10:     else
11:         $startIdx \mathrel{+}= m$ and $revKey \leftarrow revKey \,||\, optGuess$
12: $startIdx \leftarrow (p - m)$ and $revKey \leftarrow revKey[: startIdx]$
13: repeat Step 3 ... 7 and Step 11 for the final block recovery
14: return $revKey[::-1]$ after the exhaustive search removes tail errors
```

In pursuit of single-trace full private-key recovery, attackers may intuitively group every m consecutive coefficients (including zeros) of f into a block. The corresponding CPA relates each search space to m samples during the e_{p-1} calculation and avoids collecting samples vertically from different executions for hypothesis examination. Sadly, this naive CPA itself is impractical, but as we shall see below, we have solved three issues to make it work. Algorithm 2 and Algorithm 3 together detail this HIDCPA. Algorithm 3 by default employs the Hamming weight power model, considers higher correlation coefficients "better", and thus shares with Algorithm 1 the same definitions of which guess in a search space is optimal and how a correlation coefficient crosses the threshold.

Our HIDCPA reveals m coefficients of f at a time (depth $= m$) by observing the calculation of $e_{p-1}, e_p, \cdots, e_{p-2+l}$ in the product scanning method in Table 1 (breadth $= l \ll p$). During the e_i calculation, every (c_j, f_{i-j}) updates the e_i in memory with its multiply-accumulate-and-reduce. This HIDCPA targets such updates, locating $m \times l$ samples to check every m-coefficient hypothesis. It starts from the block $(f_{p-1}, f_{p-2}, \cdots, f_{p-m})$, generates eligible candidates recursively, and prunes incorrect candidates every n coefficients during the recursive construction. Should the pruning leave no survivor in the current block recovery, roll back the starting index of the block by $\lfloor m/2 \rfloor$. The next block recovery will then eliminate the error from the previous block recovery. Else set the optimal survivor as part of the final guess, and move the starting index forward by m. This HIDCPA iterates such block recovery until the block ends at f_0. Figure 2 diagrams an instance of the depth $m = 5$, the pruning period $n = 2$, and the breadth $l = 5$.

Algorithm 3 *candidateConstruct*

```
Input: the current block candidate $coeffs$
Output: (the "best" $m$-coefficient hypothesis $coeffs$ gives, the corresponding $\rho$)
 1: $lvl \leftarrow \text{len}(coeffs)$
 2: if $lvl = m$ then return $(coeffs, \rho_{realArr,idealArr})$
 3: if $lvl = 0$ then $IVArr \leftarrow [(revKey \bullet c[j : (j + startIdx)]) \bmod q$ for $j = 0$ to $(l-1)]$
 4: else if $n \mid lvl$ then                                        ▷ Candidate Pruning
 5:     if $\rho_{realArr[:lvl \times l],idealArr} <$ THRESHOLD then return $([\,], 0)$
 6: for $x$ in $\{-1, 0, 1\}$ do
 7:     for $j = 0$ to $j = (l-1)$ do
 8:         $IVArr$.append($(IVArr[(j-l)] + x \times c_{startIdx+lvl+j}) \bmod q$)
 9:     $idealArr \leftarrow \text{HammingWeight}(IVArr[l:])$
10:     $(guess_x, \rho_x) \leftarrow candidateConstruct(coeffs \,||\, [x])$
11:     $IVArr$.pop() for $l$ times
12: return $(guess_{x^*}, \rho_{x^*})$, where $\rho_{x^*} = \max\{\rho_{-1}, \rho_0, \rho_1\}$
```

Despite its resemblance to horizontal CPA [CFG+10], HIDCPA focuses on the depth with the breadth being auxiliary. Candidate pruning and error correction together make for an efficient HIDCPA with surprising parameter sets (e.g. $(m, l) = (67, 5)$). Because l is small, this HIDCPA requires far shorter observation span than horizontal CPA does. We describe how the three main features of our HIDCPA above solve the main practical problems encountered.

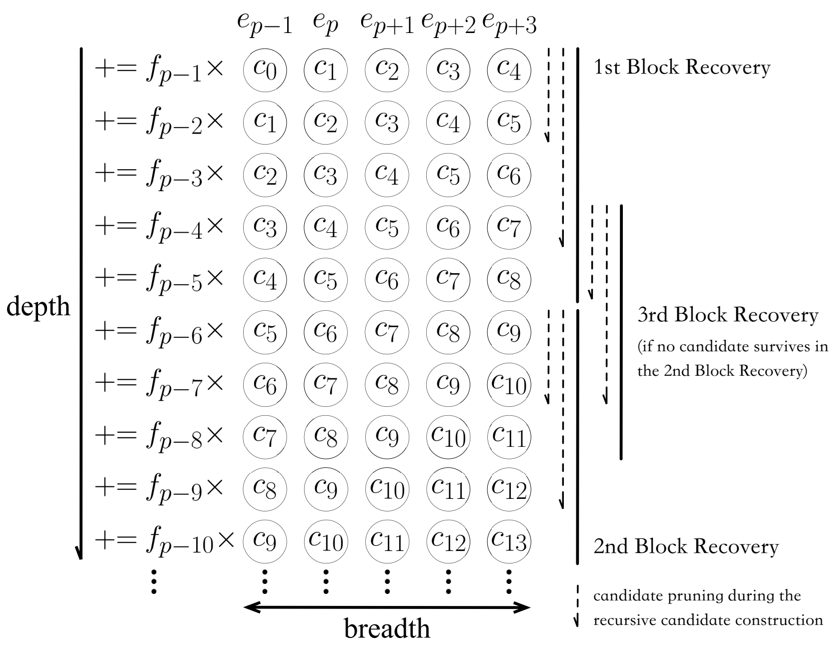

e_{p-1} e_p e_{p+1} e_{p+2} e_{p+3}
$+= f_{p-1}\times$ c_0 c_1 c_2 c_3 c_4
$+= f_{p-2}\times$ c_1 c_2 c_3 c_4 c_5
$+= f_{p-3}\times$ c_2 c_3 c_4 c_5 c_6
$+= f_{p-4}\times$ c_3 c_4 c_5 c_6 c_7
$+= f_{p-5}\times$ c_4 c_5 c_6 c_7 c_8
$+= f_{p-6}\times$ c_5 c_6 c_7 c_8 c_9
$+= f_{p-7}\times$ c_6 c_7 c_8 c_9 c_{10}
$+= f_{p-8}\times$ c_7 c_8 c_9 c_{10} c_{11}
$+= f_{p-9}\times$ c_8 c_9 c_{10} c_{11} c_{12}
$+= f_{p-10}\times$ c_9 c_{10} c_{11} c_{12} c_{13}
depth
breadth
1st Block Recovery
3rd Block Recovery
(if no candidate survives in the 2nd Block Recovery)
2nd Block Recovery
candidate pruning during the recursive candidate construction

Figure 2: The HIDCPA with $m = 5$, $n = 2$, and $l = 5$

1. If m is too small, too many candidates would fit the measurements well. However, in the naive CPA, the search complexity grows exponentially in m by a factor of 3. In practice the naive CPA is doomed to fail with $m < 20$, yet when $m \geq 20$, it encounters search spaces of size $\geq 3^{20} \approx 2^{32}$.

   Our solution a la subsection 3.1 is to prune the candidate list whenever n new coefficients are added during the recursive construction of block candidates. To be precise, we choose a fixed threshold for correlation coefficients, and check the current candidate whenever its size reaches a multiple of n. Prune the current candidate if its correlation coefficient fails to cross the threshold, along with all its descendants.

2. Since m is large, if we make an error during recovery towards the end of a block, the naive CPA may well not detect it due to the smallness of its influence on the correlation coefficients for the current block. However, we will know the error as soon as we start the next block: It has all block candidates pruned.

   As described above, our solution is for attackers to "roll back" by half a block — decrement the *startIdx* of the block by $\lfloor m/2 \rfloor$ — when all candidates are pruned. By starting this new block, which contains the last half of the previous block and the first half of the next block, we ensure that the troublesome recovery error would be near the middle of this new block, and thus its influence becomes noticeable.

   This feature will not detect a recovery error in the last block. Thus, attackers will check the last few uncertain coefficients via exhaustive search (in Streamline NTRU Prime, h and the smallness of g give (f, g); in NTRU LPRime, knowing (S, A) gives a). Two tricks serve to accelerate the search:

   - Apply the threshold not only to the entire block but also solely to the last n coefficients each time we prune candidates.
   - Apply smaller m to the last few block recoveries.

3. This HIDCPA observes the calculation of $e_p, e_{p+1}, \cdots, e_{p-2+l}$ and enables us to check each m-coefficient hypothesis with $m \times l$ samples in the single trace. This makes our approach as effective as a VCPA of $m \times l$ traces, increases the numerical gap between correct and incorrect guesses, and improves the efficiency of the candidate pruning.

   Note that the naive CPA only examines the e_{p-1} calculation, mapping an m-coefficient hypothesis to only m samples. Thus, it is as effective only as a VCPA of m traces. Take $(m, l) = (67, 5)$ as an example. It is easy to see that there are situations where 67 samples are not sufficient for VCPA while 335 samples are.

3.3 Online Template Attack (OTA)

The correlation-based approaches in previous subsections require the assumptive use of simple power models. Traditional template attacks generate refined power models, but they demand numerous template traces in the profiling stage and heavy computational power. Fortunately, [BCP+14] proposes a way out: online template attacks. Such approaches originally target elliptic curve cryptography. Thanks to online template generation, they achieve single-(target-)trace full private-key recovery with only one template trace per secret scalar bit. Though [BCP+14] claims that the transfer of OTA to other cryptographic algorithms is nontrivial, this paper presents an OTA against NTRU Prime here. Algorithm 4 shows its control flow.

This OTA targets the e_{p-1} calculation in the product scanning method in Table 1, and it works as follows: First, attackers acquire one single "target trace" from the target device. They partition the target trace into p n-dimensional "target vectors" of power samples, each corresponding to the operation $e_{p-1} \mathrel{+}= f_{p-1-i} \times c_i \pmod{q}$, respectively for $i = 0, 1, \cdots, (p-1)$. Second, attackers reveal all f_{p-1-i} iteratively with the knowledge of c and $f_{p-1}, f_{p-2}, \cdots, f_{p-i}$. They collect three "template traces", extracting "template vectors" for the operations $e_{p-1} \mathrel{+}= x \times c_i \pmod q, \forall x \in \{-1, 0, 1\}$. Right after the online template generation, they measure the similarity between the i^{th} target vector and each of the three template vectors. The template vector resembling the target vector the most makes its hypothesis attackers' optimal guess. Empirically, the Euclidean distance measure suffices to distinguish the correct hypothesis.

Algorithm 4 Online Template Attack on NTRU Prime

```
Input:
    a ciphertext $c \in ((3\mathbb{Z})[x] \cap R_q)$
    the power trace $P$ of the $e_{p-1}$ calculation
Output: a *small weight-w* polynomial $f \in R$
1: Partition $P$ into $P_0, P_1, \cdots, P_{p-1}$        ▷ $P_i$ for $e_{p-1} \mathrel{+}= f_{p-1-i} \times c_i \pmod q$
2: $e_{p-1} \leftarrow 0$
3: for $i = 0$ to $(p-1)$ do
4:     for $x$ in $\{-1, 0, 1\}$ do
5:         $T_x \leftarrow$ the template vector of $(e_{p-1} + x \times c_i) \bmod q$
6:     $f_{p-1-i} \leftarrow x^*$, where $\|P_i - T_{x^*}\|_2 = \min\{\|P_i - T_{-1}\|_2, \|P_i - T_0\|_2, \|P_i - T_1\|_2\}$
7:     $e_{p-1} \mathrel{+}= f_{p-1-i} \times c_i \pmod q$
8: return $f$
```

The idea above needs further elaboration to reach a practical implementation: How to correctly partition the target trace? How to collect template traces and extract the template vectors?

Attackers can notice with the naked eye a pattern repeated p times in the target trace. The peak indices in adjacent copies of the pattern give n. The OTA assumes that attackers fully control a device of the same type as the target one. On this device, they experiment with different ciphertexts of identical prefixes and the same private key. Superposing the corresponding traces, they identify where the divergence begins and meanwhile the boundaries between target vectors. For example, the two traces in Figure 3 diverge at Index 11819; the two peaks A and B sit at Index 11860 and Index 11972, respectively. These statistics together imply $(n, \text{boundaries})$ to be $(112, \{59, 171, \cdots, 11819, \cdots, 85179, 85291\})$. The black dotted lines in Figure 3 represent the boundaries, which further indicate the correct partition of the target trace.

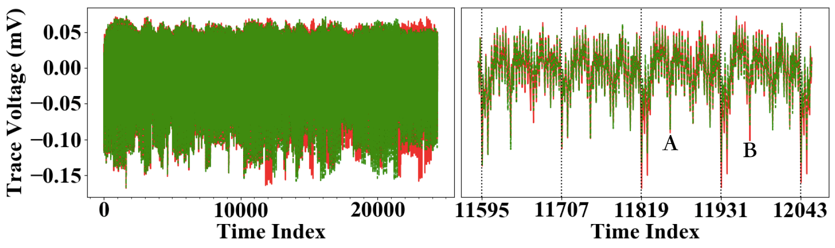


Figure 3: Target Trace Partitioning in the OTA

On the fully controlled device, attackers can run the e_{p-1} calculation with known private keys f^* and arbitrary ciphertexts c^* [MOP07]. Besides, the knowledge of c and $f_{p-1}, f_{p-2}, \cdots, f_{p-i}$ assures the knowledge of the e_{p-1} "before" the operation $e_{p-1} \mathrel{+}= f_{p-1-i} \times c_i \pmod q$. The selection of $c_0^*, c_1^*, \cdots, c_{p-1}^*$ based on the e_{p-1} before the operation and $f_{p-1}^*, f_{p-2}^*, \cdots, f_0^*$ guarantees the existence of the wanted template vector in each template trace. Knowing how to partition power traces correctly, attackers can easily locate the vector. Such online template generation is highly efficient because eligible c^* are numerous and easy to derive.

If the fully controlled device allows illegitimate f^*, a chosen-input attack with offline template generation becomes a smarter choice. This new OTA is just slightly different from the old one: First, $c_0 = c_1 = \cdots = c_{p-1}$ and $3 \mid c_0$. Second, set $f^* = \pm(x + x^3 + x^5 + \cdots + x^{p-2})$ and $c^* = c + (c_1^* - c_0)x$, where $c_1^* = c_0$ or $(c_0 \times (-w) \bmod q)$. The first feature boosts the reusability of template vectors since it limits e_{p-1} to the multiples of c_0 (in $\mathbb{Z}/q\mathbb{Z}$). In addition, $f_i = -1$ and $f_i = 1$ are randomly and uniformly distributed, so in most cases e_{p-1} only takes its value from few multiples. Thanks to the second feature, four executions suffice to generate all the template vectors this new OTA needs in the case of $(p, q, w) = (761, 4591, 286)$. Table 2 shows how this works. The template vectors for e_{p-1} += $(-1) \times c_i$, e_{p-1} += $0 \times c_i$, and e_{p-1} += $1 \times c_i$ in $\mathbb{Z}/q\mathbb{Z}$ are denoted as [−], [×], and [+], respectively. "e_{p-1} before the operation" is expressed as $c_0 \times t \bmod q$ for some $t \in \{-w, -w+1, \cdots, w\}$.

Table 2: The Mapping between (f^*, c^*) and Template Vectors

c_1^* \ f^*	$x + x^3 + x^5 + \cdots + x^{p-2}$	$-x - x^3 - x^5 - \cdots - x^{p-2}$
$c_0 \times (-w) \bmod q$	[+] for $t = -w, \cdots, -1, 0$ [×] for $t = -w, \cdots, -1, 0$	[×] for $t = 0, 1, \cdots, w$ [−] for $t = 0, 1, \cdots, w$
c_0	[+] for $t = 1, 2, \cdots, w$ [×] for $t = 1, 2, \cdots, w$	[×] for $t = -w, \cdots, -2, -1$ [−] for $t = -w, \cdots, -2, -1$

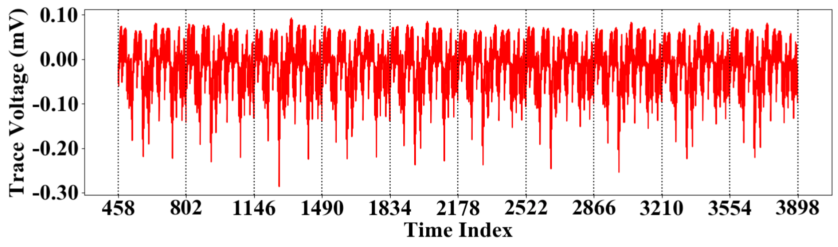
3.4 Experiments and Results

The three approaches above are applied to the reference C implementation of NTRU Prime [BCLvV16, KRSS18] on STM32F303RCT7 [STM18] and STM32F415RGT6 [STM16], two Cortex-M4-based STM32 boards, to validate their efficacy. As what the NTRU Prime submission [BCLvV16] recommends for Streamlined NTRU Prime, the target implementation sets $(p, q, w) = (761, 4591, 286)$. Figure 4 presents the power patterns of multiply-accumulate-and-reduces on both target devices. It aligns the power patterns vertically using correlation methods and separates each horizontal copy with black dotted lines. Each copy corresponds to e_i += $f_{i-j} \times c_j \pmod q$ in the product scanning method in Table 1.

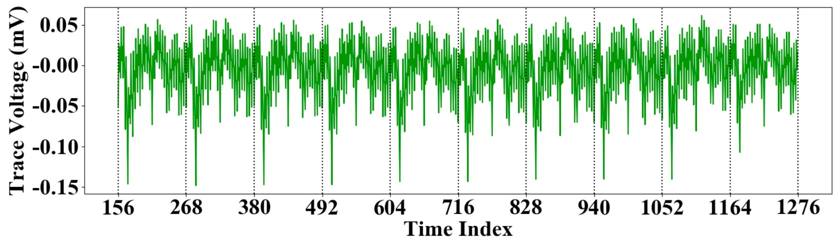
All the experiments here are based on ChipWhisperer-Lite Two-Part Version [O'F16]. A control board clocks the target boards at 7.38MHz and samples their power consumptions at 29.54MS/s. The program ChipWhisperer Capture [O+18] retrieves power samples from the control board, storing power traces and input data. While the HIDCPA and the OTA are programmed in Python 3.6.1, the VCPA is programmed in C++ in pursuit of high performance. They all run on a MacBook Air.

Note that in our experiments, correlation-based approaches check for negative correlation coefficients as hypotheses and set negative thresholds. The reason behind this design is that ChipWhisperer-Lite Two-Part Version [O'F16] inserts a resistor between the target device and its power supply, measuring the voltage after the resistor to quantify the target's power consumption. As a result, the more power the target device consumes, the lower the readings.

We have tried alternatives like inverting the classic Hamming weight power model or reversing the voltage and ground lines during power measurement using the Semi-Rigid cable with SMA(F) connector from Jyebao Co., Ltd. [JYE19]. However, they do not reflect as well as the original choice how each factor in the measurement setups influences the design of correlation attacks.



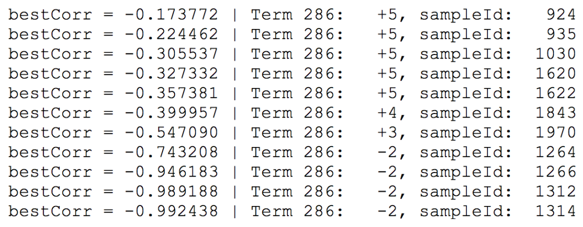
(a) F3 Power Pattern



(b) F4 Power Pattern

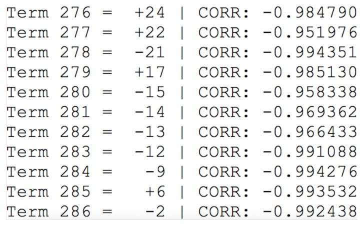
Figure 4: The Power Patterns of Both Target Devices

The experiment for the VCPA contains 10 trials on the F3 board. Each trial involves an independent key generation. The VCPA adopts -0.90 as its threshold because the Hamming weight power model is stunningly compatible with STM32 boards. The C++ program only needs 50 traces to completely reveal each of the 10 secret keys within less than 8 seconds. Figure 5 and Figure 6 are the screenshots of an example trial. The f_{b_1} recovery in this trial starts with the monomial $+x^5$. It searches from higher-order monomials to lower-order ones and from smaller sample indices to larger ones, updates its guess with the (monomial, sampleId) of the "best" known correlation coefficient, and finally outputs $-x^2$ as its answer. At the end of this trial, the program lists all the 286 (b_i, f_{b_i}): $\cdots, (b_{11}, f_{b_{11}}) = (24, 1), \cdots, (b_2, f_{b_2}) = (6, 1), (b_1, f_{b_1}) = (2, -1)$.



```
bestCorr = -0.173772 | Term 286:   +5, sampleId:   924
bestCorr = -0.224462 | Term 286:   +5, sampleId:   935
bestCorr = -0.305537 | Term 286:   +5, sampleId:  1030
bestCorr = -0.327332 | Term 286:   +5, sampleId:  1620
bestCorr = -0.357381 | Term 286:   +5, sampleId:  1622
bestCorr = -0.399957 | Term 286:   +4, sampleId:  1843
bestCorr = -0.547090 | Term 286:   +3, sampleId:  1970
bestCorr = -0.743208 | Term 286:   -2, sampleId:  1264
bestCorr = -0.946183 | Term 286:   -2, sampleId:  1266
bestCorr = -0.989188 | Term 286:   -2, sampleId:  1312
bestCorr = -0.992438 | Term 286:   -2, sampleId:  1314
```

Figure 5: The f_{b_1} (i.e., Term 286) Recovery



```
Term 276 =  +24 | CORR: -0.984790
Term 277 =  +22 | CORR: -0.951976
Term 278 =  -21 | CORR: -0.994351
Term 279 =  +17 | CORR: -0.985130
Term 280 =  -15 | CORR: -0.958338
Term 281 =  -14 | CORR: -0.969362
Term 282 =  -13 | CORR: -0.966433
Term 283 =  -12 | CORR: -0.991088
Term 284 =   -9 | CORR: -0.994276
Term 285 =   +6 | CORR: -0.993532
Term 286 =   -2 | CORR: -0.992438
```

Figure 6: The Result of the VCPA

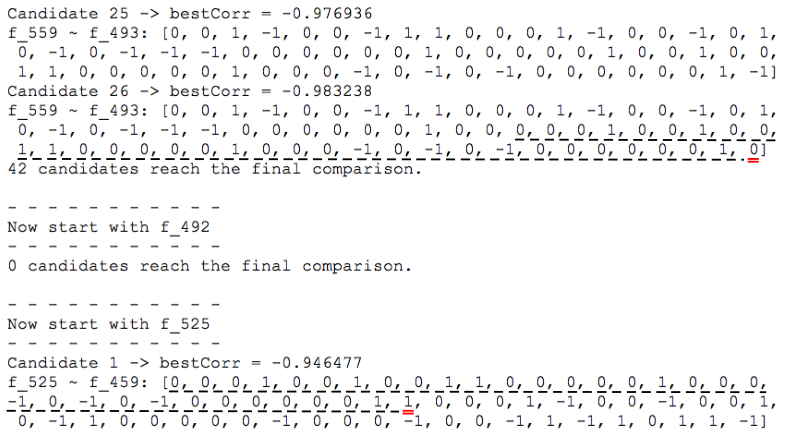
Here are some interesting observations from the example trial.

- -0.90 as the negative threshold is a nice choice for this experiment: 134 out of the 285 CORR are lower than -0.99, 201 lower than -0.98, and 262 lower than -0.95. The "worst" CORR is -0.914526 (Term 156: $+x^{329}$).

- The range of eligible thresholds is wide in this experiment: Globally, the "best" correlation coefficient the wrong hypotheses can give is -0.707664 in the first stage (Term 001: $-x^{757}$ and Term 002: $-x^{755}$) and -0.658716 in the second stage (Term 213: $+x^{177}$), so the range is roughly 0.2.

- Dynamic threshold, a more general design, seems worth trying: Locally, the difference between the "best" correlation coefficients from the optimal guess and the second-best hypothesis for (b_k, f_{b_k}) is 0.547713 on average and 0.294544 at worst (Term 118: -0.925785 from $+x^{421}$ and -0.631241 from $+x^{422}$).

The experiment for the HIDCPA sets (m, n, l) as $(67, 6, 5)$ and adopts -0.95 as the threshold in its candidate pruning. Figure 7 and Figure 8 come from an example trial on the F3 board. In this trial the HIDCPA takes around 3.5 minutes to reveal $f_{760}, f_{759}, \cdots, f_{10}$ and leaves $f_9, f_8, \cdots, f_0$ for exhaustive search. In the $f_{559} \cdots f_{493}$ recovery, 42 67-coefficient hypotheses survive candidate pruning, and the 26^{th} survivor becomes the HIDCPA's optimal guess. Unfortunately, this guess is partially incorrect since the next block recovery yields no result. Thus, the start of block candidates rolls back from f_{492} to f_{525} so that the HIDCPA can review its answers to $f_{525}, f_{524}, \cdots, f_{493}$ (underlined in Figure 7) and meanwhile make its first meaningful guess of $f_{492}, f_{491}, \cdots, f_{459}$. The comparison between Candidate 26 and Candidate 1 in Figure 7 shows that the $f_{559} \cdots f_{493}$ recovery only fails at f_{493} (double-underlined in red).



```
Candidate 25 -> bestCorr = -0.976936
f_559 ~ f_493: [0, 0, 1, -1, 0, 0, -1, 1, 1, 0, 0, 0, 1, -1, 0, 0, -1, 0, 1,
 0, -1, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, -1, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 1, -1]
Candidate 26 -> bestCorr = -0.983238
f_559 ~ f_493: [0, 0, 1, -1, 0, 0, -1, 1, 1, 0, 0, 0, 1, -1, 0, 0, -1, 0, 1,
 0, -1, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, -1, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0]
42 candidates reach the final comparison.

- - - - - - - - - - -
Now start with f_492
- - - - - - - - - - -
0 candidates reach the final comparison.

- - - - - - - - - - -
Now start with f_525
- - - - - - - - - - -
Candidate 1 -> bestCorr = -0.946477
f_525 ~ f_459: [0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
-1, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, -1, 0, 0, -1, 0, 0, 1,
 0, -1, 1, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 1, -1, 1, 0, 1, 1, -1]
```

Figure 7: The Error Correction Mechanism in the HIDCPA

11 line segments of slope 1 constitute the curve in Figure 8. The recursive construction between candidate pruning results in the constant slope, while candidate pruning itself leads to the sudden vertical drops between line segments. Each pruning reduces the number of block candidates by a factor of $3^{5.68113}$ on average. Thus, the slope of the envelope is 0.0531444, and block candidates appear to triple per 19 coefficients in the block.

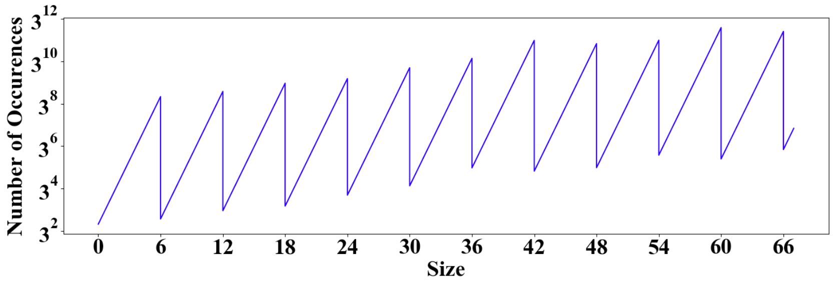


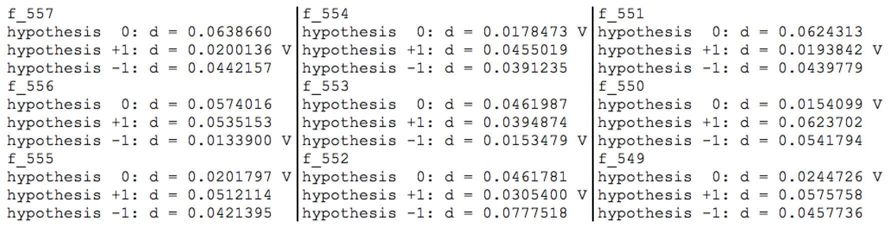
Figure 8: Size Distribution of Block Candidates in the HIDCPA Recursive Construction

The breadth l plays an important role in the HIDCPA's accuracy and efficiency. The $f_{760} \cdots f_{694}$ recovery is a complete success in the case $l = 5$, while it fails at f_{694} in the cases $l = 4$ and $l = 3$. When $l < 3$, no hypothesis survives the candidate pruning in the $f_{760} \cdots f_{694}$ recovery. The $f_{760} \cdots f_{10}$ recovery takes around 8 minutes in the case $l = 4$ and more than 2 hours in the case $l = 3$.

Here are some interesting observations from the example trial.

- There is no (obvious) upper bound on the negative threshold in the HIDCPA: If the threshold goes too negative, no hypothesis can survive the candidate pruning. Indeed, the optimal guess does not always give the "best" correlation coefficients during the candidate pruning, so adversaries should sacrifice a little efficiency for broader effectiveness by selecting less extreme thresholds. Sadly, the HIDCPA's time consumption rapidly grows as the negative threshold elevates. For example, switch the threshold from -0.95 to -0.90, and the example trial now takes 17 minutes to finish the first block recovery and 261 minutes to reveal $f_{760}, f_{759}, \cdots, f_{10}$.
- Note that each of the coefficients $f_{l-2}, f_{l-1}, \cdots, f_0$ corresponds to less than l samples. Since Algorithm 2 and Algorithm 3 just provide high-level descriptions for the HIDCPA, attackers should take care of this detail when implementing the last block recovery. Some may therefore worry about the accuracy of the last block recovery and the efficiency of the subsequent exhaustive search. We conduct some extra trials to relieve such concerns. According to the trials with $n = 6$, $l = 5$, and $m \in \{67, 61, 55, 49, 43, 37, 31, 25\}$, the block recovery with $m \times l$ samples nearly makes no error on average. In the worst case ($m = 43$), one out of 150 revealed coefficients is wrong. Assume the exhaustive search for $f_0, f_1, \cdots, f_{l-2}$ is inevitable. The full private-key recovery needs the additional search for f_{l-1} on average, so the entire search roughly terminates in $(0.5 * 3^l) \approx 122$ rounds.

Note that correlation attacks may well fail in their example trials if they judge a correlation coefficient only by its absolute value: In the VCPA's first stage, the correct guess (756, 755, 1, -1) gives -0.992131 while its deceptive counterpart (756, 755, -1, 1) gives +0.986618 as their "best" correlation coefficients. In the HIDCPA's first block recovery, the correct guess gives -0.987827 while its additive inverse gives +0.984279. In either case, the numerical gap in between is very small.



```
f_557                              |f_554                              |f_551
hypothesis  0: d = 0.0638660       |hypothesis  0: d = 0.0178473 V     |hypothesis  0: d = 0.0624313
hypothesis +1: d = 0.0200136 V     |hypothesis +1: d = 0.0455019       |hypothesis +1: d = 0.0193842 V
hypothesis -1: d = 0.0442157       |hypothesis -1: d = 0.0391235       |hypothesis -1: d = 0.0439779
f_556                              |f_553                              |f_550
hypothesis  0: d = 0.0574016       |hypothesis  0: d = 0.0461987       |hypothesis  0: d = 0.0154099 V
hypothesis +1: d = 0.0535153       |hypothesis +1: d = 0.0394874       |hypothesis +1: d = 0.0623702
hypothesis -1: d = 0.0133900 V     |hypothesis -1: d = 0.0153479 V     |hypothesis -1: d = 0.0541794
f_555                              |f_552                              |f_549
hypothesis  0: d = 0.0201797 V     |hypothesis  0: d = 0.0461781       |hypothesis  0: d = 0.0244726 V
hypothesis +1: d = 0.0512114       |hypothesis +1: d = 0.0305400 V     |hypothesis +1: d = 0.0575758
hypothesis -1: d = 0.0421395       |hypothesis -1: d = 0.0777518       |hypothesis -1: d = 0.0457736
```

Figure 9: The Result of the OTA

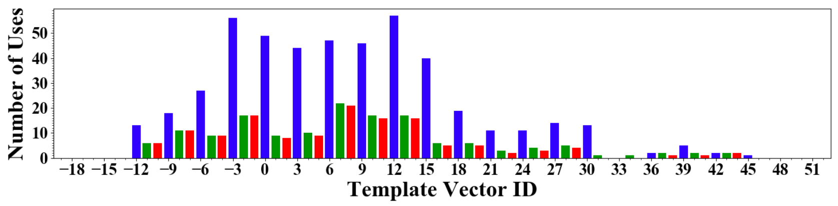


Figure 10: The Number of Uses of Each Template Vector

The experiment for the OTA mounts a chosen-input attack with offline template generation on the F4 board. The trial in Figure 9 sets $c_0 = 2046$ and shares the same secret with the example trial for the HIDCPA. Finding out the boundaries $\{44, 156, \cdots, 85164\}$, the OTA then takes 0.5 seconds to achieve full private-key recovery. Figure 10 shows how many times each template vector gets used in this trial. It labels all the template vectors with their $(3t + f_{p-1-i})$, where t follows the definition for Table 2. The red, blue, and green bars are respectively for $f_{p-1-i} = -1$, 0, and 1. Figure 10 implies that the trial only needs 60 template vectors, namely those for $t = -4, -5, \cdots, 15$. Moreover, because $w = 286 \ll (2p/3) \approx 508$, most of the blue bars are much higher than their adjacent counterparts in red and green. To summarize, Figure 10 proves the reusability claim in subsection 3.3.

4 Power Analysis on the Protected NTRU Prime

4.1 Software Countermeasures

There are three common software countermeasures for NTRU-like cryptosystems, with the prototypes first introduced in [LSCH10]. All designed for the polynomial multiplication in R_q, these countermeasures are compatible with NTRU Prime and the product scanning method in Table 1:

Countermeasure 1: the random initialization of e_i

Countermeasure 2: the randomized access to (c_j, f_{i-j}) pairs

Countermeasure 3: a first-order masking scheme

Countermeasure 1 assigns a random integer $m_i \in \mathbb{Z}/q\mathbb{Z}$ to each e_i in the initialization stage, and removes all the m_i using modular subtractions after the product scanning method. The polynomial multiplication in Countermeasure 2 receives one more argument $Perm[p]$: a random permutation of $\{0, 1, \cdots, (p-1)\}$. During the e_i calculation, the program iterates from $j = 0$ to $j = (p-1)$, adding the appropriate $f_{i-Perm[j]} \times c_{Perm[j]}$ to e_i. Countermeasure 3 can be briefly expressed as:

Input: $m_f, m_c \leftarrow$ random masks $\in R_q$; $\bar{f} = f + m_f$; $\bar{c} = c + m_c$

Output: $m_d = -m_f \times m_c$; $\bar{d} = d + m_d = f \times c - m_f \times m_c$

Algorithm: $D_1 = \bar{f} \times \bar{c}$; $D_2 = \bar{f} \times m_c$; $D_3 = m_f \times \bar{c}$; $\bar{d} = D_1 - D_2 - D_3$

The operations above are in R_q, and every multiplication follows the product scanning method in Table 1.

For the first defensive strategy, the increase in time consumption is negligible. The third one takes just twice as long to complete the entire computation: D_2 only depends on the two masks m_f, m_c and the private key f. Thus, NTRU Prime can compute D_2 ahead in its key schedule, and an update of D_2 is unnecessary until a regeneration of private key or mask pair [MOP07]. All these defensive strategies can protect NTRU Prime from the analyses in section 3. However, this does not mean they are invincible. As shown later in this section, Countermeasure 1 and Countermeasure 2 are subject to chosen-input simple power analysis. Improper implementations of Countermeasure 3 are at the risk of horizontal correlation power analysis [ATT+18] and online template attacks. Luckily, to the best of our knowledge, there is still no efficient attack against NTRU Prime with both ciphertexts and private keys masked. Despite its large overhead, the masking scheme is the software countermeasure this paper recommends the most.

4.2 Chosen-Input Simple Power Analysis (CISPA)

The introduction of the CISPA starts with Countermeasure 2 as the victim due to the simplicity of its CISPA implementation. Although the counterpart on Countermeasure 1 is a bit more sophisticated, the idea behind remains the same: Observe the victim's $e_0 \cdots e_{p-1}$ calculation with ciphertexts of only one/two nonzero coefficients, and reveal f according to the discontinuities in each of the few collected power traces.

The CISPA on Countermeasure 2 sets $c = c_0$, where $3 \mid c_0$ and $c_0 \neq 0$, acquires one single trace from its target device, and partitions the full trace roughly into p partial traces corresponding to the e_i calculation, where $i \in \{0, 1, \cdots, (p-1)\}$. In each partial trace exists at most one discontinuity, and its existence indicates the change of e_i's value during the calculation. There are two types of such discontinuities: One signals $f_i = 1$ and the other $f_i = -1$. Attackers then classify the partial traces with the naked eye (and simple arithmetic if necessary) into three categories: *No Discontinuity*, *Discontinuity I*, and *Discontinuity II*. In our experiment, the major difference between the two types of discontinuities lies in the degree of their voltage drops.

The f_i recovery directly follows the categorization. To be specific, *No Discontinuity* implies $e_i = 0$ throughout the calculation, so $f_i = 0$. Either *Discontinuity I* or *Discontinuity II* implies $e_i = c_0$ at the end of the calculation (i.e., $f_i = 1$), and the other implies $e_i = -c_0$ (i.e., $f_i = -1$). Figure 11 shows part of the full trace corresponding to the $e_{745} \cdots e_{760}$ calculation ($p = 761$), and labels each partial trace with the category it belongs to. "X" stands for *No Discontinuity*, "I" *Discontinuity I*, and "II" *Discontinuity II*. These labels imply $f = \pm(x^{759} - x^{758} - x^{754} + x^{753} + x^{751} + x^{750} + x^{746} + x^{745} + \cdots)$. The error detection mechanisms in NTRU Prime [BCLvV16] help find out which is correct.

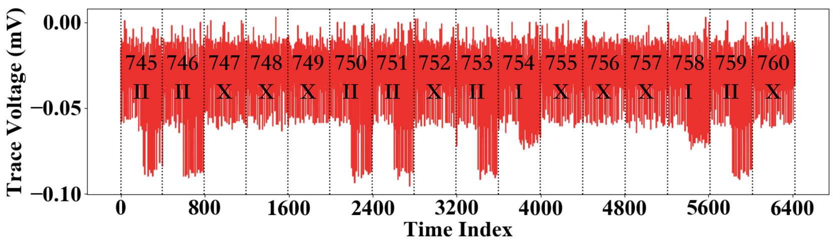


Figure 11: The CISPA on Countermeasure 2

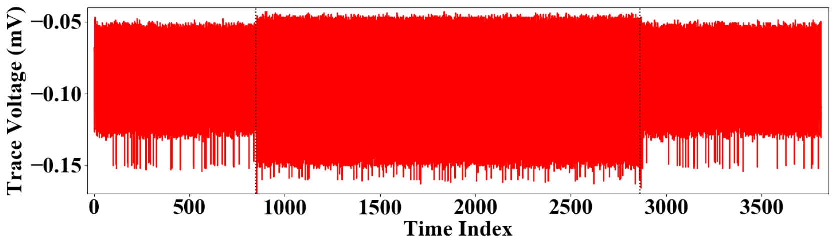
Even if each e_i calculation follows an independent random permutation, this CISPA still works. An *ad hoc* solution is to shuffle all the accesses to (c_j, f_{i-j}) pairs regardless of the e_i calculation in which they are involved. Unfortunately, this solution is not a choice for resource-constrained implementations due to its need of high entropy.

The CISPA on Countermeasure 1 contains two stages. The first stage checks if f_i is nonzero, $\forall i \in \{0, 1, \cdots, (p-1)\}$, and the second stage further divides the nonzero f_i into two sets, one for $f_i = 1$ while the other for $f_i = -1$. The first stage resembles the CISPA on Countermeasure 2, but it only cares whether the partial trace under examination is continuous or not. The corresponding f_i is nonzero if and only if discontinuity exists.

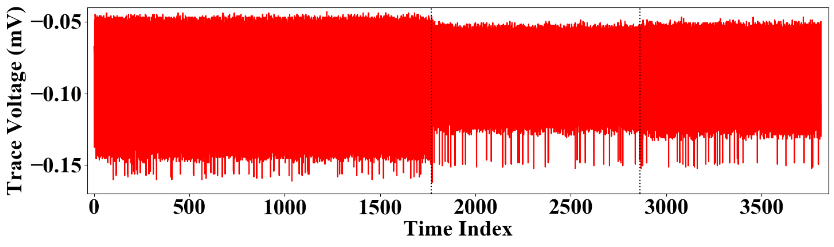
The second stage focuses on the e_{p-1} calculation in the product scanning method in Table 1 with the following $(w-1)$ ciphertexts. Note that the $b_1, b_2, \cdots, b_w$ here follow the definition in subsection 3.1:

$$c = c_0 x^{p-1-b_{\lfloor w/4 \rceil}} + c_0 x^{p-1-b_k}, \forall k > \lfloor w/2 \rceil$$
$$c = c_0 x^{p-1-b_{\lfloor 3w/4 \rceil}} + c_0 x^{p-1-b_k}, \forall k \leq \lfloor w/2 \rceil \text{ and } k \neq \lfloor w/4 \rceil$$

As a result, each power trace in this stage gets divided by two discontinuities into three parts. $f_{b_k} \neq f_{b_{\lfloor w/4 \rceil}}$ (or $f_{b_{\lfloor 3w/4 \rceil}}$) if and only if the first part shares the same pattern with the last part. Figuratively, $f_{b_1}, f_{b_2}, \cdots, f_{b_w}$ cluster into two groups. The error detection mechanisms in NTRU Prime [BCLvV16] reveal f by finding out the group for $f_i = 1$. Figure 12 shows two power traces from this stage. $c = 1149x^{573} + 1149x^{170}$ gives the upper one, and $c = 1149x^{573} + 1149x^{354}$ the lower one. In the first stage attackers already know $b_{\lfloor 286/4 \rceil} = 187$, $b_{\lfloor 286 \times 3/4 \rceil} = 590$, and $b_{\lfloor 286/2 \rceil + 5} = 406$. Now they further know $(f_{187}, f_{590}, f_{406}) = (1, -1, 1)$ or $(-1, 1, -1)$.



(a) $f_{590} \neq f_{187}$



(b) $f_{406} = f_{187}$

Figure 12: The CISPA on Countermeasure 1: the second stage

While attackers may come across the practical problems below implementing the CISPA on Countermeasure 1, solutions always exist. First, a seemingly continuous (partial) trace from Countermeasure 1 can actually contain a subtle discontinuity. Since the e_i initialization is random, a few more executions with the same ciphertext suffice to avoid the miss. Second, discontinuities at the beginning and the end of a (partial) trace are difficult to detect with the naked eye. Therefore, $c = c_0$ in the first stage should be replaced by $c = c_0x^d$ if it is hard to notice the discontinuities at the first and the last d multiply-accumulate-reduce operations in a partial trace. The knowledge of public key enables the brute-force search of $f_0, f_1, \cdots, f_{d-1}$ and $f_{p-d}, \cdots, f_{p-2}, f_{p-1}$. Third, the CISPA requires legitimate ciphertexts to make use of the error detection mechanisms in NTRU Prime [BCLvV16] and find out which of the two final candidates is correct. Luckily, such ciphertexts are available at low cost in the context of power analysis [MOP07].

Though the two CISPA require long observation span and a strict form of chosen inputs, they need no power model, few observations, and low sampling frequency. Besides, their underlying assumption is natural: The operations $e_i \mathrel{+}= f_{i-j} \times c_j \pmod{q}$ with $c_j = 0$ have similar power patterns for a fixed e_i and $f_{i-j} \in \{-1, 0, 1\}$ and distinct power patterns for (three) different e_i. The three e_i are from $\{-c_0, 0, c_0\}$ in Countermeasure 2 and $\mathbb{Z}/q\mathbb{Z}$ in Countermeasure 1.

The two CISPA may well remain successful in noisy settings. They target not only specific characteristics (e.g. voltage drops) but the general change of power pattern in a (partial) trace after one certain multiply-accumulate-and-reduce (or **smladx**, in section 5). The attacks work as long as adversaries can discern such changes from the normal variation due to electronic noise and categorize them into only two classes. Thanks to the strict form of chosen inputs, power pattern stays uniform before and after the targeted operation, so adversaries can focus on any changes happening there. Hence, a device whose signal-to-noise ratio is low enough to hide such changes would be immune to many other, if not most, power attacks. Such a device is not among the targets that power analysis typically cares about.

4.3 Additional Remarks

Some may argue that Countermeasure 3 remains secure against efficient power analysis even if it only masks private keys (Variant K) or ciphertexts (Variant C). Unfortunately, neither of the variants is secure. Though originally designed for Frodo and NewHope, the horizontal CPA in [ATT+18] directly applies to Variant K, independently revealing $\bar{f}$ and m_f. The knowledge of ciphertexts remains useful because input ciphertexts, which are accessible to the public, participate in the polynomial multiplications without any disguise. Variant C cannot hide the $b_1, b_2, \cdots, b_w$ defined in subsection 3.1, and online template attacks may further reveal f_{b_k} by jointly considering $\bar{c}$ and m_c. The few possibilities of f_{b_k} render the template vectors of eligible hypotheses in a round mutually distinguishable. In contrast, the original version of Countermeasure 3 survives every attack in this paper.

4.4 Experiments and Results

The experiments here are designed to confirm the CISPA's effectiveness on Countermeasure 1 and Countermeasure 2. The settings are almost the same as those in subsection 3.4, except that the CISPA allows much lower sampling frequencies. The CISPA samples the power consumptions of Countermeasure 1 on the F3 board and Countermeasure 2 on the F4 board at 434.39kS/s and 115.38kS/s, respectively. Figure 12 and Figure 11 are the very results of these experiments.

The analysis on the first target requires higher sampling frequency due to the following two reasons: First, if e_i is randomly initialized, its (and other relevant intermediate values') Hamming weight may not change drastically after meaningful multiply-accumulate-reduce operations (nonzero f_{i-j} and c_j). Second, although the F3 board is a bit more compatible with the Hamming weight power model than the F4 board, its data-dependent component in power consumption is far less significant than the F4 counterpart.

Here are the performance statistics. On the F4 board, Countermeasure 1 requires 16044870 clock cycles to complete the ($f \times c$ in R_q) calculation, Countermeasure 2 25233239 clock cycles, Countermeasure 3 31986334 clock cycles, and the unprotected version 15973389 clock cycles. The next section presents an optimized version. As we shall see below, the optimized NTRU Prime remains subject to the power analyses in section 3 and section 4.

5 Power Analysis on more Optimized NTRU Prime

5.1 NTRU Prime Optimized Using smladx

Table 3: The Two Optimizations at the Instruction Level

Boundary Conditions: $f_{-1} = c_p = 0$		
Reference Implementation	**The First Optimization**	**The Second Optimization**

for $i = 0$ to $(p-1)$
  for $j = 0$ to i
    $e_i = (e_i + f_{i-j} \times c_j) \bmod q$
for $i = p$ to $(2p-2)$
  for $j = (i-p+1)$ to $(p-1)$
    $e_i = (e_i + f_{i-j} \times c_j) \bmod q$
	for $i = 0$ to $(p-1)$
	  for $j = 0$ to i
	    $e_i \mathrel{+}= f_{i-j} \times c_j$
	  $e_i = e_i \bmod q$
	for $i = p$ to $(2p-2)$
	  for $j = (i-p+1)$ to $(p-1)$
	    $e_i \mathrel{+}= f_{i-j} \times c_j$
	  $e_i = e_i \bmod q$
		for $i = 0$ to $(p-1)$
		  $\forall j \in \{0, 2, \cdots, (i-1)$ or $i\}$
		    $e_i \mathrel{+}= (f_{i-j} \times c_j + f_{i-j-1} \times c_{j+1})$
		  $e_i = e_i \bmod q$
		for $i = p$ to $(2p-2)$
		  $\forall j \in \{(i-p+1), (i-p+3), \cdots, (p-2)$ or $(p-1)\}$
		    $e_i \mathrel{+}= (f_{i-j} \times c_j + f_{i-j-1} \times c_{j+1})$
		  $e_i = e_i \bmod q$

The polynomial multiplication in R_q is side-channel informative and computationally intensive, so the majority of optimizations for NTRU-like cryptosystems focus on this operation [SDC09, BCLvV16, HRSS17, DWZ18, KRS19]. The pursuit of faster implementations is not only for performance improvement but also for side-channel leakage suppression.

Compared with the previous versions, the NTRU Prime in this section contains two optimizations at the instruction level on the product scanning method in Table 1. First, its e_i calculation only needs one reduction in $\mathbb{Z}/q\mathbb{Z}$, which follows a series of multiply-and-accumulates. In contrast, the old version [BCLvV16] demands that each multiply-accumulate operation stay strictly in $\mathbb{Z}/q\mathbb{Z}$. Second, every two **smlabb**s in the assembly are replaced with one **smladx**. **smlabb** adds the product of one (c_j, f_{i-j}) pair to e_i at a time, while **smladx** adds the products of two consecutive (c_j, f_{i-j}) pairs [ARM11]. Table 3 presents the implementations of the polynomial multiplication in R_q before and after each optimization.

After the first modification, the multiplication takes 5837648 clock cycles, 36.55% of the original running time. After the second modification, the running time further decreases to 2947368 clock cycles, 50.49% of the previous one.

5.2 The Transfer of Power Analyses

In theory, VCPA, HIDCPA, OTA, and CISPA are easily adaptable to the optimized NTRU Prime. Compared with Algorithm 1, the new VCPA takes into account the possibility that both f_{i-j} and f_{i-j-1} involved in the same **smladx** are nonzero. Therefore, it may reveal $(b_w, b_{w-1}, b_{w-2} = (b_{w-1} - 1), f_{b_w}, f_{b_{w-1}}, f_{b_{w-2}})$ in the first stage and $(b_k, f_{b_k}, b_{k-1} = (b_k - 1), f_{b_{k-1}})$ at a time in the second stage. Just like the **smlabb** of (c_j, f_{i-j}) before, every **smladx** of $(c_j, f_{i-j}, c_{j+1}, f_{i-j-1})$ here updates the e_i in memory. Following the spirits of Algorithm 2 and Algorithm 3, the new HIDCPA checks $2m$-coefficient hypotheses and reveals $2m$ coefficients of f at a time with the $m \times l$ samples corresponding to the above memory updates. Also, the new HIDCPA has the starting index of the block roll back by $\lfloor m/2 \rfloor \times 2$ rather than $\lfloor m/2 \rfloor$ in its error correction mechanism.

The new OTA partitions the target trace into $\lceil p/2 \rceil$ target vectors, each corresponding to the operation $e_{p-1} \mathrel{+}= (f_{p-1-j} \times c_j + f_{p-2-j} \times c_{j+1})$ in Table 3, where $j = 0, 2, \cdots, (p-2)$ or $(p-1)$. It then reveals all (f_{p-1-j}, f_{p-2-j}) iteratively: In each round the new OTA prepares nine template vectors of the operation $e_{p-1} \mathrel{+}= (a \times c_j + b \times c_{j+1})$, where $(a, b) \in \{-1, 0, 1\}^2$. The online template generation relies on the knowledge of c and the recovery results for $f_{p-1}, f_{p-2}, \cdots, f_{p-j}$. The CISPA in subsection 4.2 is directly applicable, and we recommend the instance on Countermeasure 2 due to its simplicity.

In practice, attackers may find it frustrating during the adaptation. The load/store instructions at the assembly level contribute to many highly data-dependent components in microcontrollers' power consumption [MOP07]. Unfortunately, the substitution of **smladx** for **smlabb** cuts in half the number of such instructions available for the f recovery. As a result, in correlation-based approaches the numerical gaps between correct and incorrect hypotheses shrink.

For the new OTA and the CISPA, it worsens the situation that $e_{p-1} \mathrel{+}= (f_{p-1-j} \times c_j + f_{p-2-j} \times c_{j+1})$ takes much fewer clock cycles than the aggregate of $e_{p-1} \mathrel{+}= f_{p-1-i} \times c_i$ and $e_{p-1} \mathrel{+}= f_{p-2-i} \times c_{i+1}$ in subsection 3.3. The fewer data-dependent samples per f_i the OTA observes from each target vector, the more susceptible the OTA is to electronic noise and the lower the OTA's resolution. The CISPA suffers the same setbacks.

Accordingly, the new VCPA requires more traces, and the new HIDCPA demands smaller m, smaller n, and larger l. Now that each (f_{p-1-j}, f_{p-2-j}) corresponds to 9 clock cycles, far less than the 56 clock cycles in the unoptimized case, the new OTA may well fail in practice. Despite its direct applicability the CISPA no longer tolerates extremely low sampling frequencies.

5.3 Experiments and Results

The experiments below apply the new HIDCPA in subsection 5.2 and the CISPA in subsection 4.2 to the optimized NTRU Prime in subsection 5.1 on the F4 board. The experiment settings, if not specified, are the same as those in subsection 3.4 and subsection 4.4, including how to judge a correlation coefficient.

The new HIDCPA changes (m, n, l) to $(7, 3, 10)$. The example trial in Figure 13 shares the same secret with that in Figure 7. In this trial the new HIDCPA takes around 9.5 minutes to reveal $f_{760}, f_{759}, \cdots, f_9$ and leaves $f_8, f_7, \cdots, f_0$ for exhaustive search. The CISPA samples the target's power consumption at 14.769MS/s, and Figure 14 is the very result. The labels in Figure 14 imply $f = \pm(x^{759} - x^{758} - x^{754} + x^{753} + x^{751} + x^{750} + x^{746} + x^{745} + \cdots)$, the same secret as that in Figure 11. Furthermore, attackers can know from the target's assembly that the discontinuities here result from the updates of e_i in memory. If so, the knowledge of $c = 2046$ and the assumptive use of the Hamming weight power model together make $f = x^{759} - x^{758} + \cdots$ the only possibility.

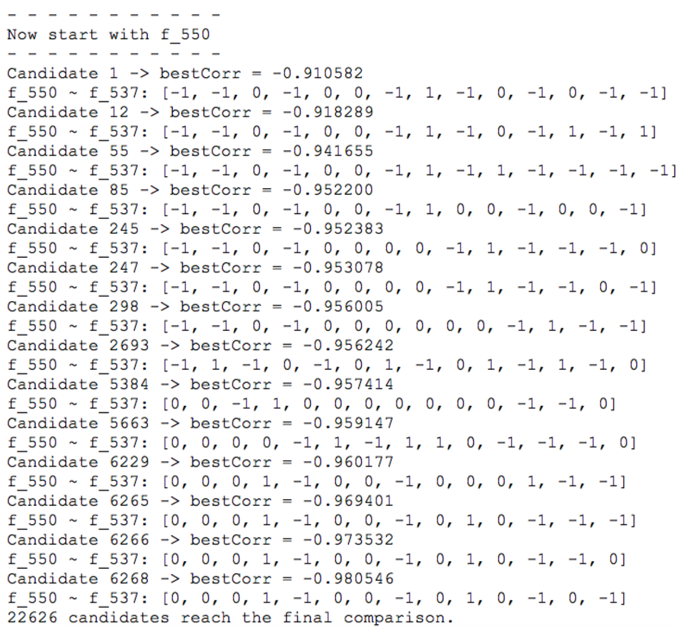

```
- - - - - - - - - - - -
Now start with f_550
- - - - - - - - - - - -
Candidate 1 -> bestCorr = -0.910582
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, -1, 1, -1, 0, -1, 0, -1, -1]
Candidate 12 -> bestCorr = -0.918289
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, -1, 1, -1, 0, -1, 1, -1, 1]
Candidate 55 -> bestCorr = -0.941655
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, -1, 1, -1, 1, -1, -1, -1, -1]
Candidate 85 -> bestCorr = -0.952200
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, -1, 1, 0, 0, -1, 0, 0, -1]
Candidate 245 -> bestCorr = -0.952383
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, 0, 0, -1, 1, -1, -1, -1, 0]
Candidate 247 -> bestCorr = -0.953078
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, 0, 0, -1, 1, -1, -1, 0, -1]
Candidate 298 -> bestCorr = -0.956005
f_550 ~ f_537: [-1, -1, 0, -1, 0, 0, 0, 0, 0, 0, -1, 1, -1, -1]
Candidate 2693 -> bestCorr = -0.956242
f_550 ~ f_537: [-1, 1, -1, 0, -1, 0, 1, -1, 0, 1, -1, 1, -1, 0]
Candidate 5384 -> bestCorr = -0.957414
f_550 ~ f_537: [0, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0]
Candidate 5663 -> bestCorr = -0.959147
f_550 ~ f_537: [0, 0, 0, 0, -1, 1, -1, 1, 1, 0, -1, -1, -1, 0]
Candidate 6229 -> bestCorr = -0.960177
f_550 ~ f_537: [0, 0, 0, 1, -1, 0, 0, -1, 0, 0, 0, 1, -1, -1]
Candidate 6265 -> bestCorr = -0.969401
f_550 ~ f_537: [0, 0, 0, 1, -1, 0, 0, -1, 0, 1, 0, -1, -1, -1]
Candidate 6266 -> bestCorr = -0.973532
f_550 ~ f_537: [0, 0, 0, 1, -1, 0, 0, -1, 0, 1, 0, -1, -1, 0]
Candidate 6268 -> bestCorr = -0.980546
f_550 ~ f_537: [0, 0, 0, 1, -1, 0, 0, -1, 0, 1, 0, -1, 0, -1]
22626 candidates reach the final comparison.
```

Figure 13: The HIDCPA on the Optimized NTRU Prime

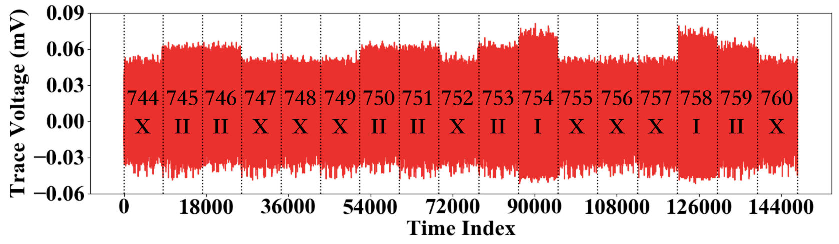


Figure 14: The CISPA on the Optimized NTRU Prime

6 Conclusion

This paper features multiple power analysis approaches on the product scanning method specialized for NTRU-like cryptosystems. We test the attacks on NTRU Prime, a Round 2 candidate in the NIST Post-Quantum Cryptography Standardization Project. Experiments are run using the reference implementation, an implementation further optimized using SIMD instructions, and implementations featuring common protective measures. Every approach achieves full private-key recovery for both schemes in NTRU Prime.

The VCPA is fast and extensible to larger parameters. One single trace of short observation span suffices for the HIDCPA to reveal dozens of f_i at a time, quickly and reliably. The OTA needs no *a priori* power model but three template traces per f_i from a fully controlled device similar to the target, one single target trace, and little computational resources. Attackers can uncover private keys of the protected NTRU Prime with the naked eye using the CISPA.

The approaches in this study focus on the polynomial multiplication in R_q in decapsulation. Nonetheless, the single-(target-)trace attacks on random inputs, namely the HIDCPA and the OTA, also apply to ($a \times \mathit{Generator}(S)$ in R_q) in NTRU LPRime's key generation, ($h \times r$ in R_q) in Streamlined NTRU Prime's encapsulation, and ($b \times A$ in R_q) in NTRU LPRime's encapsulation. This paper recommends both operand polynomials in such multiplications be masked since (the original version of) Countermeasure 3 is the sole survivor throughout the article.

Other ideal-lattice-based cryptosystems are likely susceptible to these approaches if their private-key coefficients are from a small set of possibilities. As the number of possible coefficients increases, the OTA and the CISPA require higher-resolution measurements, and the HIDCPA becomes computationally impractical.

The approaches potentially break even more optimized NTRU Prime. If the polynomial multiplications use multi-level Karatsuba ending with schoolbook multiplications between two n-long polynomials, these approaches can still target the schoolbook multiplications of the form:

$$(\sum_{i=0}^{n-1} f_{kn+i}x^{kn+i}) \times (\sum_{j=0}^{n-1} c_{kn+j}x^{kn+j}), \forall k \in \{0, 1, \cdots, \lceil p/n \rceil - 1\}$$

In other words, they target the schoolbook multiplications related to the multiplications between two upper halves or two lower halves in the multi-level Karatsuba. In theory, this adaptation leads to full private-key recovery. If Toom-k is used instead of Karatsuba as the first layer, this adaptation can only reveal the first and last $1/k$ of private-key coefficients. How to adapt these approaches so as to be generally successful against a mix of Toom and Karatsuba multiplications is an interesting follow-up question.

Finally, it is possible that parts of these techniques are adaptable to implementations which use the operand scanning method as the bottom layer, especially when the fixed operand is the small polynomial f and the scanning operand is the generic-looking c.

References

[AC18] Carlisle Adams and Jan Camenisch, editors. *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*, volume 10719 of *Lecture Notes in Computer Science*. Springer, 2018.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A New Hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.

[AKJ$^+$18] Soojung An, Suhri Kim, Sunghyun Jin, HanBit Kim, and HeeSeok Kim. Single trace side channel analysis on NTRU implementation. *Applied Sciences*, 8(11):2014, 2018.

[ARM11] ARM Holdings. *Cortex-M4 Devices Generic User Guide*, August 2011.

[ATT$^+$18] Aydin Aysu, Youssef Tobah, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. Horizontal side-channel vulnerabilities of post-quantum key exchange protocols. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2018, Washington, DC, USA, April 30 - May 4, 2018*, pages 81–88. IEEE Computer Society, 2018.

[BBC$^+$18] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAkem: A post-quantum key encapsulation mechanism based on QC-LDPC codes. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*, pages 3–24. Springer, 2018.

[BCD$^+$16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018. ACM, 2016.

[BCLvV16] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. *IACR Cryptology ePrint Archive*, 2016:461, 2016.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BCP$^+$14] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2014.

[BDK$^+$18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018.

[BFM$^+$18] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Assessing the feasibility of single trace power analysis of frodo. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 216–234. Springer, 2018.

[BHH+15] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 2015.

[BKS19] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. Memory-efficient high-speed implementation of kyber on cortex-m4. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings*, volume 11627 of *Lecture Notes in Computer Science*, pages 209–228. Springer, 2019.

[CA69] Stephen A Cook and Stål O Aanderaa. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142:291–314, 1969.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175, 2005.

[DWZ18] Wei Dai, William Whyte, and Zhenfei Zhang. Optimizing polynomial convolution for ntruencrypt. *IEEE Trans. Computers*, 67(11):1572–1583, 2018.

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1857–1874. ACM, 2017.

[FH17] Wieland Fischer and Naofumi Homma, editors. *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*. Springer, 2017.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[HMHW09] Michael Hutter, Marcel Medwed, Daniel M. Hein, and Johannes Wolkerstorfer. Attacking ecdsa-enabled RFID devices. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 519–534, 2009.

[HOKG18] James Howe, Tobias Oder, Markus Krausz, and Tim Güneysu. Standard lattice-based key encapsulation on embedded devices. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):372–393, 2018.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[HPS$^+$17] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, volume 10159 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2017.

[HRSS17] Andreas Hülsing, Joost Rijneveld, John M. Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In Fischer and Homma [FH17], pages 232–252.

[HW11] Michael Hutter and Erich Wenger. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 459–474. Springer, 2011.

[JB16] Kimmo Järvinen and Josep Balasch. Single-trace side-channel attacks on scalar multiplications with precomputations. In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2016.

[JF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011.

[JMV01] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.

[JYE19] JYEBAO RF & Microwave. JYEBAO RF & Microwave - your best RF product provider!, 2019.

[KAJ17] Brian Koziel, Reza Azarderakhsh, and David Jao. Side-channel attacks on quantum-resistant supersingular isogeny Diffie-Hellman. In Adams and Camenisch [AC18], pages 64–81.

[Kar63] Anatolii Karatsuba. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596, 1963.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KMRV18] Angshuman Karmakar, Jose M. Bermudo Mera, Sujoy Sinha Roy, and Ingrid Verbauwhede. Saber on ARM cca-secure module lattice-based key encapsulation on ARM. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):243–266, 2018.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KRS19] Matthias J. Kannwischer, Joost Rijneveld, and Peter Schwabe. Faster multiplication in \mathbb z_2^m[x] on cortex-m4 to speed up NIST PQC candidates. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 281–301. Springer, 2019.

[KRSS18] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: post-quantum crypto library for the ARM Cortex-M4, 2018.

[KY12] Abdel Alim Kamal and Amr M. Youssef. A scan-based side channel attack on the NTRUEncrypt cryptosystem. In *Seventh International Conference on Availability, Reliability and Security, Prague, ARES 2012, Czech Republic, August 20-24, 2012*, pages 402–409. IEEE Computer Society, 2012.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

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

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[MD99] Thomas S. Messerges and Ezzy A. Dabbish. Investigations of power analysis attacks on smartcards. In Scott B. Guthery and Peter Honeyman, editors, *Proceedings of the 1st Workshop on Smartcard Technology, Smartcard 1999, Chicago, Illinois, USA, May 10-11, 1999.* USENIX Association, 1999.

[MNY17] Moustafa Mahmoud, Mouna Nakkar, and Amr Youssef. A power analysis resistant FPGA implementation of NTRUEncrypt. In *Microelectronics (ICM), 2017 29th International Conference on*, pages 1–4. IEEE, 2017.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007.

[Nat17] National Institute of Standards and Technology. Post-quantum cryptography, 2017.

[O+18] Colin O'Flynn et al. ChipWhisperer - the complete open-source toolchain for side-channel power analysis and glitching attacks, 2018.

[O'F16] Colin O'Flynn. ChipWhisperer-Lite (CW1173) two-part version, 2016.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical cca2-secure and masked ring-lwe implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):142–174, 2018.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Fischer and Homma [FH17], pages 513–533.

[PSKH18] Aesun Park, Kyung-Ah Shim, Namhun Koo, and Dong-Guk Han. Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations - rainbow and UOV -. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):500–523, 2018.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SBG+18] Markku-Juhani O. Saarinen, Sauvik Bhattacharya, Óscar García-Morchón, Ronald Rietman, Ludo Tolhuizen, and Zhenfei Zhang. Shorter messages and faster post-quantum encryption with round5 on cortex M. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers.*, volume 11389 of *Lecture Notes in Computer Science*, pages 95–110. Springer, 2018.

[SDC09] Xiaoyu Shen, Zhenjun Du, and Rong Chen. Research on NTRU algorithm for mobile java security. In Keqiu Li, Geyong Min, Yongxin Zhu, Meikang Qiu, and Wenyu Qu, editors, *International Conference on Scalable Computing and Communications / Eighth International Conference on Embedded Computing, ScalCom-EmbeddedCom 2009, Dalian, China, September 25-27, 2009*, pages 366–369. IEEE Computer Society, 2009.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[SMS19] Thomas Schamberger, Oliver Mischke, and Johanna Sepúlveda. Practical evaluation of masking for ntruencrypt on ARM cortex-m4. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 253–269. Springer, 2019.

[Sni16] Brian Sniffen. Akamai faster forward crypto at scale, 2016.

[STM16] STMicroelectronics. *Datasheet - STM32F415xx STM32F417xx*, September 2016. Revision 8.

[STM18] STMicroelectronics. *Datasheet - STM32F303xB STM32F303xC*, October 2018. Revision 14.

[SW07] Joseph H. Silverman and William Whyte. Timing attacks on NTRUEncrypt via variation in the number of hash calls. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, volume 4377 of *Lecture Notes in Computer Science*, pages 208–224. Springer, 2007.

[TE15] Mostafa Taha and Thomas Eisenbarth. Implementation attacks on post-quantum cryptographic schemes. *IACR Cryptology ePrint Archive*, 2015:1083, 2015.

[Too63] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

[TPG15] Hannes Tschofenig and Manuel Pegourie-Gonnard. Performance of state-of-the-art cryptography on ARM-based microprocessors. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, United States, July 2015.

[UW14] Thomas Unterluggauer and Erich Wenger. Practical attack on bilinear pairings to disclose the secrets of embedded devices. In *Ninth International Conference on Availability, Reliability and Security, ARES 2014, Fribourg, Switzerland, September 8-12, 2014*, pages 69–77. IEEE Computer Society, 2014.

[vE85] Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269–286, 1985.

[WLYZ18] Yuanhao Wang, Ying Li, Zhangqi Yin, and Bei Zeng. 16-qubit IBM universal quantum computer can be fully entangled. *npj Quantum Information*, 4(1):46, 2018.

[WP06] André Weimerskirch and Christof Paar. Generalizations of the Karatsuba algorithm for efficient implementations. *IACR Cryptology ePrint Archive*, 2006:224, 2006.

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

[ZWW13] Xuexin Zheng, An Wang, and Wei Wei. First-order collision attack on protected NTRU cryptosystem. *Microprocessors and Microsystems - Embedded Hardware Design*, 37(6-7):601–609, 2013.