

# Side-Channel Analysis of Lattice-Based Post-Quantum Cryptography: Exploiting Polynomial Multiplication

Catinca Mujdei, Arthur Beckers, Jose Bermudo, Angshuman Karmakar,  
Lennert Wouters and Ingrid Verbauwhede

imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium,  
[catinca\\_mujdei@yahoo.ca](mailto:catinca_mujdei@yahoo.ca), [firstname.lastname@esat.kuleuven.be](mailto:firstname.lastname@esat.kuleuven.be)

**Abstract.** Polynomial multiplication algorithms such as Toom-Cook and the Number Theoretic Transform are fundamental building blocks for lattice-based post-quantum cryptography. In this work we present correlation power analysis based side-channel analysis methodologies targeting *every* polynomial multiplication strategy for *all* lattice-based post-quantum key encapsulation mechanisms in the final round of the NIST post-quantum standardization procedure. We perform practical experiments on real side-channel measurements demonstrating that our method allows to extract the secret key from all lattice-based post-quantum key encapsulation mechanisms. Our analysis demonstrates that the used polynomial multiplication strategy can significantly impact the time complexity of the attack.

**Keywords:** post-quantum cryptography, side-channel analysis, number-theoretic transform, Toom-Cook

## 1 Introduction

The advent of quantum computers and their impact on the security of classical public-key cryptography (PKC) has increased research interest into post-quantum cryptography (PQC). The majority of the PQC research is being conducted in the context of the PQC standardization procedure lead by the National Institute of Standards and Technology (NIST) [NIS17]. NIST recently requested more research into the physical security of the PQC implementations as part of the, currently ongoing, final round of the standardization process [AASA<sup>+</sup>17].

New cryptographic primitives are first assessed cryptanalytically to ensure their theoretical security [ABD16, APS15, DDGR20]. A real-world implementation of a secure cryptographic primitive can still be vulnerable to implementation or physical attacks. Side-channel attacks are considered passive physical attacks in which the adversary is able to acquire side-channel information (e.g. power consumption, electromagnetic (EM) radiation, execution time, etc.) that is unintentionally produced by the implementation. This side-channel information can be exploited to extract secret information [KJJ99].

Lattice-based hard problems have a long history of being used for cryptographic constructions. Starting with Ajtai's seminal work on the short integer solution hard problem [Ajt96], and later with the introduction of NTRU [HPS98] and the learning with errors (LWE) hard problem [Reg04]. However, compared to PKC primitives such as RSA [RSA78] and elliptic-curve cryptography (ECC) [Mil86, Kob87], lattice-based cryptography received little attention in the context of physical security.

The current body of work related to side-channel analysis of PQC can be divided in two major classes. The first class of attacks use side-channels to perform chosen ciphertext attacks (CCA). Multiple works demonstrated that LWE and learning with rounding (LWR) schemes such as Kyber and Saber are vulnerable to CCA based side-channel attacks by exploiting leakage caused by the Fujisaki-Okamoto (FO) transformation [RRCB20, BDH<sup>+</sup>21, XPRO20]. NTRU based schemes using the FO transform or the Saito-Xagawa-Yamakawa transform have also been shown vulnerable to CCA based side-channel attacks [REB<sup>+</sup>21]. Most of these attacks use purposefully crafted ciphertext, such that the decrypted message depends on a single secret coefficient. By varying the ciphertext the secret coefficients can be retrieved using statistical tests (e.g. the t-test). On average a few thousand traces are sufficient to recover the full private key. All lattice based schemes have been shown vulnerable to CCA side-channel attacks. However, the ciphertexts used in most CCA based attacks will not decrypt to a valid message, and will therefore result in decryption failures. Limiting the maximum number of allowed decryption failures can prevent these attacks. For example, a device can impede CCA based side-channel attacks by deleting all key material after a certain amount of decryption failures have been observed.

The second class of attacks target the private key directly. This class of attack can use valid ciphertext inputs that do not lead to decryption failures. Primas et al. target the number theoretic transform (NTT) using profiled single trace attacks [PPM17], Pessl et al. later expanded on these attacks in [PP19]. NTRU, being the oldest variant of the lattice based finalists, has thus far received most attention when it comes to physical side-channel attacks. Huang et al. target the polynomial multiplication of NTRU prime using correlation power analysis (CPA), horizontal CPA, template attacks and simple power analysis (SPA) [HCY20]. Earlier works by Lee et al. [LSCH10] targeted NTRU using CPA, while Silverman et al. [SW07] performed timing attacks on NTRUEncrypt. Note that several variants, parameter sets and implementations exist for NTRU, the listed papers do not all target the most recent variant. In contrast, in this work we focus on the schemes, and their reference implementations, that are currently in the final round of the NIST PQC competition.

Interestingly, the current body of work related to the side-channel security of PQC is focused on CCA attacks and more advanced profiled attacks. In this work we investigate the applicability of classical non-profiled CPA approaches. Specifically, we provide a side-channel analysis of the polynomial multiplication implementations used by *all* lattice-based NIST PQC KEM finalists.

Concretely the contributions of this paper can be summarised as follows:

- We propose a non-profiled side-channel attack methodology targeting all the different polynomial multiplication algorithms used in lattice-based cryptography. We categorically show how these algorithms can be attacked using CPA. Our techniques are generic for the different multiplication algorithms.
- We demonstrate practical side-channel attacks using real power traces on lattice-based KEMs in the final round of NIST standardization procedure *i.e.* NTRU-KEM [CDH<sup>+</sup>19], Kyber [ABD<sup>+</sup>21], and Saber [BMD<sup>+</sup>20]. For each of these schemes there are multiple implementations relying on different multiplication algorithms. We exhaustively demonstrate our attack on all the reference implementations of these schemes.
- We compare the complexity of attacking the different schemes, multiplication methods and parameter choices using CPA. We show how the different multiplication strategies can have a significant impact on the data and time complexity of the attack.

## Outline

In Section 2 we provide background information on the studied lattice-based cryptosystems as well as the different strategies used for polynomial multiplication. In Section 3 we discuss how Toom-Cook-based polynomial multiplication can be attacked using CPA. Similarly, in Section 4 we apply our techniques to the NTT. In Section 5 we provide practical results, demonstrating key recovery using real-world side-channel measurements on all lattice-based KEMs in the final round of NIST standardization procedure. Finally, in Section 6, we discuss and summarise our results, and discuss potential countermeasures.

## 2 Preliminaries

This section provides general notation and background information on lattice-based KEMs. We also describe the Number Theoretic Transform and Toom-Cook polynomial multiplication strategies that are used by these lattice-based KEMs.

### 2.1 Notations

Throughout this paper, we denote by  $Z_q$  the ring of integers modulo  $q$  for a positive integer  $q$ . We consider rings of the form  $R_q = Z_q[x]/(x^n + 1)$ , with  $n$  a power of 2, of which the elements are denoted by lower-case letters, sometimes considering the variable  $x$  implicit; if the variable  $x$  of a polynomial  $a(x) \in R_q$  is specified, then we denote its  $i$ -th coefficient as  $a_i$ . Vectors and matrices are written in bold lower and bold upper case, respectively. Given a vector  $\mathbf{v}$ , we write the mean value of its coefficients as  $\bar{\mathbf{v}}$ . Sampling an element  $x$  from a distribution over some domain  $\mathbb{D}$  is denoted as  $x \leftarrow (\mathbb{D})$ .

### 2.2 Lattice-based cryptosystems

The lattice-based KEMs in the final round of the NIST PQC competition are based on two constructions. First, Kyber [ABD<sup>+</sup>21] and Saber [DKRV19], which are both variants of the public-key encryption (PKE) scheme attributed to Lyubashevsky, Peikert and Regev (LPR) [LPR10]. Secondly, NTRU-like schemes [CDH<sup>+</sup>19], which were first proposed by Hoffstein, Pipher and Silverman [HPS98]. Specific transforms are applied to these schemes to transform them from chosen-plaintext attack secure PKE to chosen-ciphertext attack (CCA) secure KEM.

We briefly cover these schemes here, and refer the reader to the original submission documents (e.g. [ABD<sup>+</sup>21, BMD<sup>+</sup>20, CDH<sup>+</sup>20]) for more details.

**LPR encryption scheme.** Regev’s *Learning With Errors* (LWE) problem [Reg05, Reg04] entails solving a system of “noisy” linear equations: given a fixed secret  $\mathbf{s} \in Z_q^k$ , an error  $e \in Z_q$  and a uniformly random  $\mathbf{a} \in Z_q^k$ , the goal is to recover  $\mathbf{s}$  from  $(\mathbf{a}, b = \mathbf{a} \cdot \mathbf{s} + e)$ . Secret and error sampling for LWE-based schemes is typically done over a centered binomial distribution  $\mathcal{B}_\mu$  for a positive even integer  $\mu$  [ADPS16]. Samples of  $\mathcal{B}_\mu$  lie within  $x \in \{-\frac{\mu}{2}, -\frac{\mu}{2} + 1, \dots, \frac{\mu}{2} - 1, \frac{\mu}{2}\}$  and are distributed according to the probability mass function

$$\Pr[x \in \mathcal{B}_\mu] = \frac{\mu!}{(\mu/2 + x)!(\mu/2 - x)!} 2^{-\mu}.$$

The LPR-scheme is based on the hardness of *Ring-LWE* (RLWE) [LPR10], an algebraic extension of LWE, with samples of the form  $(a, b = a \cdot s + e)$  for polynomials  $a, s, e$  all belonging to  $R_q = Z_q[x]/(x^n + 1)$ . The module variant *Module-LWE* (MLWE) [LS15] treats samples of the form  $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$  for  $\mathbf{A} \in R_q^{l \times l}$  and  $\mathbf{s}, \mathbf{e} \in R_q^l$ . *Learning With Rounding* (LWR) [BPR12] adds the random error term implicitly in the second component

of the LWE-sample (instead of explicitly adding it) by scaling down from  $q$  to a smaller modulus  $\rho$ . Such a sample is of the form  $\mathbf{a}, b = \frac{\rho}{q} \mathbf{a} \cdot \mathbf{s} \in \mathbb{Z}_q^k \times \mathbb{Z}_\rho$ . Ring as well as module variants also exist for LWR.  $\square$

Algorithm 1 describes key generation in an LPR-based scheme. The public key is a RLWE-sample  $(a, b)$  and is based on the secret key  $s$ . Algorithm 2 illustrates encryption of an  $n$ -bit message  $m$  (after mapping it to  $R_q$ ): two additional LWE-samples are generated and  $m$  is added to the most significant bits of the second sample. Decryption (see Algorithm 3) multiplies the first sample by the secret key, then subtracts it from the second sample. A final decoding step removes the randomness introduced by the error terms  $e$ ,  $e$  and  $e$  from  $m'$ , yielding  $m$ .

In this paper, we target the medium security parameter sets of the MLWE-based KEM Kyber ( $l = 3, n = 256, q = 3329$  and  $\mu = 4$ ) and MLWR-based KEM Saber ( $l = 3, n = 256, q = 2^{13}, \rho = 2^{10}$  and  $\mu = 8$ ). Nevertheless, the presented attacks can be easily extended to other security levels.

Algorithm 1 LPR key generation	Algorithm 2 LPR encryption: input $(pk, m)$
1: $a \leftarrow \mathcal{U}(R_q)$	1: $s, e, e' \leftarrow \mu(R_q)$
2: $s, e \leftarrow \mu(R_q)$	2: $b = a \cdot s + e$
3: $b = a \cdot s + e$	3: $v = b \cdot s + e' + \frac{q}{2} m$
4: return $(pk = (a, b), sk = s)$	4: return $ct = (b, v)$

Algorithm 3 LPR decryption: input $(ct, sk)$
1: $m = v - b \cdot s$
2: return $\text{Decode}(m)$

The chosen plaintext attack secure LPR-based PKEs can be converted to CCA-secure KEMs using a variant [HHK17] of the Fujisaki-Okamoto (FO) [FO99, FO13] transform. The transform can prevent attacks that make use of a so-called plaintext-checking oracle, which reveals information about the secret key by providing the decryption of specially-crafted ciphertexts. Figure 1 shows the decapsulation process. The input ciphertext  $c$  is first decrypted, after which the obtained plaintext is re-encrypted, resulting in  $c'$ . The shared key is not revealed when a decryption failure occurs (i.e.,  $c' \neq c$ ). In other words, an attacker can no longer gain information about the secret key as maliciously chosen ciphertexts are rejected during decapsulation. However, as shown by Ravi et al., this approach does not mitigate CCAs that make use of side-channel leakages [RRCB20].

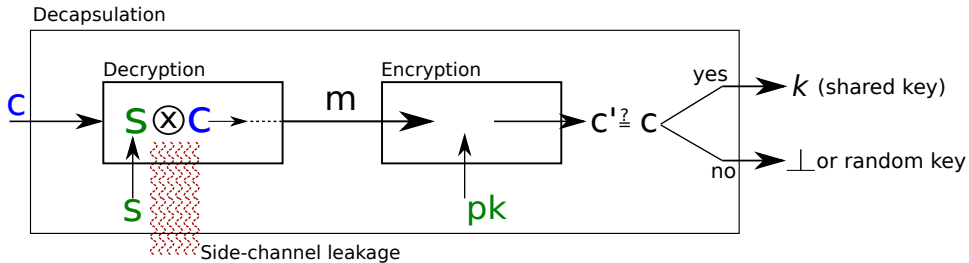


Figure 1: The decapsulation procedure with re-encryption stage using the FO-transform. The green and blue coloured components respectively denote what the attacker can or cannot control during a chosen-ciphertext attack.

**NTRU.** NTRU-like schemes consider polynomial rings  $\mathbb{Z}_3[x]/\Phi_n$ ,  $\mathbb{Z}_q[x]/\Phi_n$  and  $\mathbb{Z}_q[x]/(\Phi_1 \cdot \Phi_n)$ , where  $\Phi_1 = x - 1$  and  $\Phi_n = x^{n-1} + x^{n-2} + \dots + 1$ . Key generation, encryption and decryption are described in Algorithms 4, 6, and 5 respectively. The routines `Sample` and `Li ft` and sets  $L_r$  and  $L_m$  are explained in detail in [CDH<sup>+</sup>19]. The NTRU-submission to the NIST PQC standardization procedure is a combination of NTRUEncrypt and NTRU-HRSS-KEM, and proposes parameter sets for both `ntruhs` and `ntruhrss`. NTRU’s secret coefficients  $\{0, 1, 2\}$  are not binomially distributed: coefficients 0 occur with probability  $\frac{86}{256}$ , whereas 1 and 2 occur both with probability  $\frac{85}{256}$ .

Algorithm 4 NTRU key generation	Algorithm 5 NTRU decryption: input (ct, sk)
1: $f, g$ <code>Sample</code> () 2: $\hat{f}_q$ $f^{-1} \bmod (q, \Phi_n)$ 3: $h$ $3 \cdot g \cdot \hat{f}_q \bmod (q, \Phi_1 \cdot \Phi_n)$ 4: $h_q$ $h^{-1} \bmod (q, \Phi_n)$ 5: $\hat{f}_p$ $f^{-1} \bmod (3 \cdot \Phi_n)$ 6: return (pk = $h$ , sk = $(f, \hat{f}_p, h_q)$ )	1: if $c = 0 \bmod (q, \Phi_1)$ , return fail 2: $a$ $(c \cdot f) \bmod (q, \Phi_1 \cdot \Phi_n)$ 3: $m$ $(a \cdot \hat{f}_p) \bmod (3, \Phi_n)$ 4: $m$ <code>Li ft</code> ( $m$ ) 5: $r$ $((c - m) \cdot h_q) \bmod (q, \Phi_n)$ 6: <b>If</b> $(r, m) \in L_r \times L_m$ return $(r, m, 0)$ 7: return $(0, 0, 1)$
Algorithm 6 NTRU encryption: input (pk, $(r, m)$ )	
1: $m$ <code>Li ft</code> ( $m$ ) 2: $c$ $(r \cdot h + m) \bmod (q, \Phi_1 \cdot \Phi_n)$ 3: return ct = $c$	

The initial NTRU submissions [CDH<sup>+</sup>19] also used the FO-transform to achieve CCA-security. Instead, the final round submission of NTRU [CDH<sup>+</sup>20] uses the Saito-Xagawa-Yamakawa [SXY18] transform (Algorithm 5 line 1). In contrast to the FO-transform, the Saito-Xagawa-Yamakawa transform does not require re-encryption, and thus greatly improves the efficiency of the cryptographic scheme. In this work we use side-channel analysis to target the polynomial multiplication directly, this makes the presented attacks independent of the used transform.

## 2.3 Multiplication strategies

Lattice-based cryptosystems usually resort to either the Number Theoretic Transform ( $O(n \log n)$ ) [Pol71] or Toom-Cook/Karatsuba ( $O(n^{1+\epsilon})$ ,  $0 < \epsilon < 1$ ) [Too63, Coo66, KO62] for efficient multiplication of polynomials with  $n$  coefficients. Neither one of the two algorithms is unambiguously the fastest in any scenario, characterized by operand size and target processor architecture, since asymptotic complexities include constant factors [ACC<sup>+</sup>21]. The core idea of both multiplication strategies is multiplication by point evaluation. Given two  $n$ -coefficient polynomials  $a(x), b(x) \in \mathbb{R}_q$ , evaluated at  $2n - 1$  distinct points  $\{x_1, x_2, \dots, x_{2n-1}\} \subset \mathbb{Z}_q$ , we can compute the coefficients of  $(a \cdot b)(x)$  by interpolation of the values  $(a \cdot b)(x_i) = a(x_i)b(x_i)$ ,  $1 \leq i \leq 2n - 1$ .

### 2.3.1 Toom-Cook

The base case of Toom-Cook-based [Too63, Coo66] multiplication is that of *Karatsuba*, [KO62] which we illustrate for two linear polynomials  $a(x) = a_0 + a_1x$  and  $b(x) = b_0 + b_1x$ . Three evaluation points suffice, namely  $x_1 = 0$ ,  $x_2 = 1$  and  $x_3 = \omega$ . We then get  $a(0) = a_0$ ,  $a(1) = a_0 + a_1$ ,  $a(\omega) = a_0 + \omega a_1$ ,  $b(0) = b_0$ ,  $b(1) = b_0 + b_1$ ,  $b(\omega) = b_0 + \omega b_1$ . These point evaluations can be interpolated as  $(a \cdot b)(x) = a_0b_0 + (a_1b_0 + a_0b_1)x + a_1b_1x^2 = a(0)b(0) + (a(1)b(1) - a(0)b(0) - a(\omega)b(\omega))x + a(\omega)b(\omega)$ . This interpolation by point evaluation reduces the number of scalar multiplications from four  $(a_0b_0, a_1b_0, a_0b_1, a_1b_1)$  to three

$(a(0)b(0), a(1)b(1), a(\dots)b(\dots))$ . Karatsuba's strategy can be applied to polynomials of arbitrary (power-of-two) number of coefficients  $n$  by rewriting  $a(x) = a^0(x) + a^1(x)x^{\frac{n}{2}}$  and  $b(x) = b^0(x) + b^1(x)x^{\frac{n}{2}}$ , and similarly splitting the sub-polynomials  $a^0(x), a^1(x), b^0(x), b^1(x)$  until a threshold degree is reached. Below this threshold degree the naive schoolbook method is used for multiplication.

The method of  $w$ -way Toom-Cook generalizes the idea of Karatsuba by splitting  $rw$ -coefficient polynomials into  $w$  parts and performing the multiplication for  $r$  coefficients, after which the results are recombined. That is, given two such polynomials  $a(x)$  and  $b(x)$ , these are split as  $a(y) = a^0(x) + a^1(x) \cdot y + a^2(x) \cdot y^2 + \dots + a^{w-1}(x) \cdot y^{w-1}$  and  $b(y) = b^0(x) + b^1(x) \cdot y + b^2(x) \cdot y^2 + \dots + b^{w-1}(x) \cdot y^{w-1}$ , with  $y = x^r$ . The sub-polynomials  $a^0(x), \dots, a^{w-1}(x), b^0(x), \dots, b^{w-1}(x)$  are then multiplied together in a suitable way to facilitate interpolation. Multiplying the sub-polynomials is often done by applying a number of Karatsuba-layers, before resorting to schoolbook multiplication.

All Toom-Cook-based versions of Saber and NTRU use  $w = 4$ , with evaluation points  $x_1 = 0, x_2 = 1, x_3 = -1, x_4 = 2, x_5 = -2, x_6 = 3, x_7 = \dots$ . Saber further splits the resulting sub-polynomials by two Karatsuba-layers, after which a 16-coefficient schoolbook multiplication is performed. All NTRU-versions follow a similar structure, but with four Karatsuba-layers (except for ntruhs2048509, which uses only three) and varying schoolbook thresholds. To illustrate the 4-way Toom-Cook procedure, we consider two  $n$ -coefficient polynomials  $a(x)$  and  $b(x)$  that are split into four "equally large" polynomials:  $a(y) = a^0(x) + a^1(x) \cdot y + a^2(x) \cdot y^2 + a^3(x) \cdot y^3$  and  $b(y) = b^0(x) + b^1(x) \cdot y + b^2(x) \cdot y^2 + b^3(x) \cdot y^3$ , with  $y = x^{\frac{n}{4}}$ . Out of the seven  $\frac{n}{4}$ -coefficient multiplications, denoted by ( ), at seven evaluation points as described above, we consider the following four ones:

$$\text{mul}_1 = a(0) \cdot b(0) = a^0(x) \cdot b^0(x) \quad (1)$$

$$\begin{aligned} \text{mul}_2 &= a(2) \cdot b(2) \\ &= (a^0(x) + 2 \cdot a^1(x) + 4 \cdot a^2(x) + 8 \cdot a^3(x)) \\ &\quad (b^0(x) + 2 \cdot b^1(x) + 4 \cdot b^2(x) + 8 \cdot b^3(x)) \end{aligned} \quad (2)$$

$$\begin{aligned} \text{mul}_3 &= a(3) \cdot b(3) \\ &= (a^0(x) + 3 \cdot a^1(x) + 9 \cdot a^2(x) + 27 \cdot a^3(x)) \\ &\quad (b^0(x) + 3 \cdot b^1(x) + 9 \cdot b^2(x) + 27 \cdot b^3(x)) \end{aligned} \quad (3)$$

$$\text{mul}_4 = a(\dots) \cdot b(\dots) = a^3(x) \cdot b^3(x) \quad (4)$$

### 2.3.2 Number Theoretic Transform

The *Number Theoretic Transform* (NTT) can be regarded as a special form of the Fast Fourier Transform (FFT). Since it computes over  $Z_q$  instead of  $\mathbb{C}$ , no floating-point operations are required. The NTT also facilitates polynomial multiplication by some form of point evaluation. If

$$q \equiv 1 \pmod{n}, \quad (5)$$

then a primitive  $n$ th root of unity  $\omega \in Z_q$ , satisfying  $\omega^{n/2} \equiv -1 \pmod{q}$ , exists and  $x^n - 1$  can be factorized as  $\prod_{i=0}^{n-1} (x - \omega^i)$ . Given two polynomials  $a(x), b(x) \in R_q$  to be multiplied, the NTT is concerned with computing the point evaluations  $\text{NTT}(a(x)) = (a(0), \dots, a(n-1))$  and  $\text{NTT}(b(x)) = (b(0), \dots, b(n-1))$ . Multiplication of  $a(x)$  and  $b(x)$  modulo  $x^n - 1$  can then be computed as  $\text{NTT}^{-1}(\text{NTT}(a(x)) \cdot \text{NTT}(b(x)))$ , where ( ) denotes coefficient-wise multiplication.

If  $n = 2^k$ , a power of 2, then the NTT can be computed by iteratively factorizing the original  $x^n - 1$  "in half" down to linear degree. Implementation-wise, each factorization can be computed as  $Z_q[x] / (x^{n/2^i} - \omega^{n/2^i}) \cong Z_q[x] / (x^{n/2^i} + \omega^{n/2^i}) \times Z_q[x] / (x^{n/2^i} - \omega^{n/2^i}) : a_0(x) + a_1(x) \cdot$

$x^{n/2^{2^i-1}}$   $(a_0(x) - \cdot a_1(x), a_0(x) + \cdot a_1(x))$ , for some power of  $x$ , called a *twiddle factor*. The above mapping is termed the *Cooley-Tukey butterfly* [CT65], and its inverse (up to a factor  $\frac{1}{2}$ ) the *Gentleman-Sandén butterfly* [GS66].

**Negacyclic NTT:** Most LWE-based schemes multiply over a ring  $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ . To this end, we can exploit the fact that  $x^{2^n} - 1 = (x^n + 1)(x^n - 1)$ , hence any root of  $x^n + 1$  is also a root of  $x^{2^n} - 1$ . If a primitive  $2n$ th root of unity  $\omega_{2n}$  exists, then these shared roots can be written as the odd powers  $\omega_{2n}^{2j+1} = \omega_{2n} \cdot \omega_n^j$ , for  $j \in \{0, \dots, n-1\}$ , with  $\omega_n = \omega_{2n}^2$ . When multiplying two polynomials  $a(x)$  and  $b(x)$  over  $R_q$ , it suffices to compute the regular  $n$ -coefficient NTT on  $a(\omega_{2n} \cdot x)$  and  $b(\omega_{2n} \cdot x)$ . This procedure is called the *negacyclic NTT*, and it strengthens Condition (5) to

$$q \equiv 1 \pmod{2n}. \quad (6)$$

We assume from now on to be always working over rings of the form  $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ .

**Incomplete NTT:** For efficiency reasons, one might want to stop the recursive splitting prematurely after  $l$  (with  $l < k$ ) layers, so that the component-wise multiplication is of modular degree at least 2. The resulting *incomplete NTT* generalizes Condition (6) to  $q \equiv 1 \pmod{\frac{n}{2^{k-l-1}}}$ , so that a primitive  $\frac{n}{2^{k-l-1}}$ th root of unity exists.

**Modulus unsuitable for NTT:** Some authors of LWE-based schemes choose to use a modulus  $q$  not satisfying Condition (6), as is the case for Saber and NTRU. To facilitate NTT-based polynomial multiplication, we can represent the polynomial coefficients in  $[-\frac{q}{2}, \frac{q}{2})$  and choose a new modulus  $q'$  such that both  $q' > \frac{nq^2}{2}$  and Condition (6) hold [CHK<sup>+</sup>21]. After performing polynomial multiplication by the NTT over  $\mathbb{Z}_{q'}[x]$ , the resulting coefficients can then be reduced back to  $\mathbb{Z}_q[x]$ .

Among the lattice-based KEM finalists, NTRU and Saber were initially designed to use Toom-Cook/Karatsuba [HRSS17, KRSS, KMRV18, MKV20]-based polynomial multiplications. However, Chung et al. [CHK<sup>+</sup>21] later proposed NTT-based multiplication for these schemes using a larger modulus as described in this section. The most efficient implementations of Kyber have always used NTT based multiplication.

### 2.3.3 Montgomery reduction

When performing the NTT, we should be careful to avoid register overflow, which may occur when multiplying a polynomial coefficient with (a power of) a twiddle factor [LS19]. Therefore, we must modularly reduce intermediate NTT-values by  $q$  regularly enough. The *Montgomery modular multiplication* [Mon85] can be used to implement 32-bit modular reduction efficiently in constant-time [PP19], and is suitable for the 32-bit registers of the Cortex-M4 [CHK<sup>+</sup>21]. Montgomery reduction makes use of the *REDC algorithm*, which requires a value  $R > q$  such that  $\gcd(q, R) = 1$  and basic computations mod  $R$  are inexpensive. The algorithm provides an efficient way to compute  $TR^{-1} \pmod q$ , given  $T$ , if  $0 < T < Rq$  is satisfied, and requires modular reduction by  $R$  instead of  $q$ .

## 2.4 Correlation Power Analysis

The attacks covered in this paper use the Correlation Power Analysis (CPA) method introduced by Brier et al. [BCO04]. In essence the attacker will attempt to recover the cryptographic key using a divide-and-conquer approach. A target device implements the target cipher and produces some (unintentional) side-channel leakage that is recorded by the attacker. A power model (e.g. Hamming weight or Hamming distance) is combined with a known input and unknown secret key. The attacker will guess the secret key by computing the correlation (e.g. using Pearson correlation) between the hypothetical power consumption and real side-channel measurements.

In the remainder of this work we exploit side-channel leakage of the multiplication between the secret polynomial and a ciphertext polynomial during decryption, as indicated in Figure 1. Specifically, we target Line 1 in Algorithm 3 for LWE-based schemes and Line 2 in Algorithm 5 for NTRU-like schemes. We try to recover all  $n$  coefficients of the secret polynomials and we verify their correctness by decrypting the encryption of a known plaintext.

### 3 Correlation Power Analysis on Toom-Cook-based polynomial multiplication

In this Section we propose a CPA-approach targeting Toom-Cook-based multiplication in both Saber and NTRU. The attacks for both schemes follow the same main structure, but differ slightly because of the integer ring to which the secret polynomial coefficients belong, as well as the schoolbook threshold. This section starts by providing details on the targeted schoolbook multiplication implementation, followed by the general structure of the attack and scheme-specific details.

#### 3.1 Schoolbook multiplication implementation

We target the basic schoolbook multiplication proposed by Kannwischer et al. in [KRS18] which is used in multiple implementations of Saber and NTRU. The polynomial coefficients are all packed in pairs, which allows for efficient manipulation by the 32-bit ARMv7E-M instruction set. The latter includes parallel multiplications and instructions that can compute on combinations of low and high halfwords. Our attack targets the generic structure of the multiplication and is therefore independent of the target platform and instruction set. The implementation follows a parallelogram-like structure, as illustrated by Figure 2, to perform an 8-coefficient multiplication of a ciphertext polynomial  $a(x) = \sum_{i=0}^7 a_i x^i$  and secret polynomial  $b(x) = \sum_{i=0}^7 b_i x^i$ . Each column consists of all additive terms needed to compute  $c_i$ ,  $0 \leq i < 15$ , the coefficients of the resulting polynomial  $c(x) = \sum_{i=0}^{14} c_i x^i = a(x) \cdot b(x)$ . Each rectangle refers to the execution of one instruction. A rectangle containing two terms indicates that both operations are performed in parallel by a single instruction. The parallelogram-like structure can be generalized for an arbitrary number of coefficients of the multiplicands  $a(x)$  and  $b(x)$ . The order of operations is degree-dependent: instructions in the same column are normally executed one after another, whereas neighbouring columns might not be. We refer to the source code in the pqm4-library [KRSS] for the ordering for all relevant degrees.

We exploit the side-channel leakage that results from storing the coefficients  $c_i$ . Note that, for any  $m$ -coefficient polynomial multiplication, every secret coefficient  $b_i$ ,  $0 \leq i < m$ , is involved in one scalar multiplication per resulting coefficient  $c_i, c_{i+1}, \dots, c_{i+m-1}$ . It thus suffices to only target the computation of  $c_0, \dots, c_{m-1}$  to recover all coefficients  $b_i$ ,  $0 \leq i < m$ . Our strategy is then to guess each coefficient  $b_i$ ,  $0 \leq i < m$ , in that order, based on the value of  $c_i$ , so that we need to already know the values of  $b_0, \dots, b_{i-1}$ , taking into account the parallelogram-like structure.

Table 1 lists the parameter sets deployed by Saber and NTRU for Toom-Cook-based polynomial multiplication. The schoolbook thresholds vary from 10 to 16. The fourth column of the table also lists in brackets the number of sub-polynomials that each original polynomial splits into at the schoolbook level. It can be computed as  $\frac{n}{\text{schoolbook threshold}}$ . This is the number of schoolbook multiplications that we target.



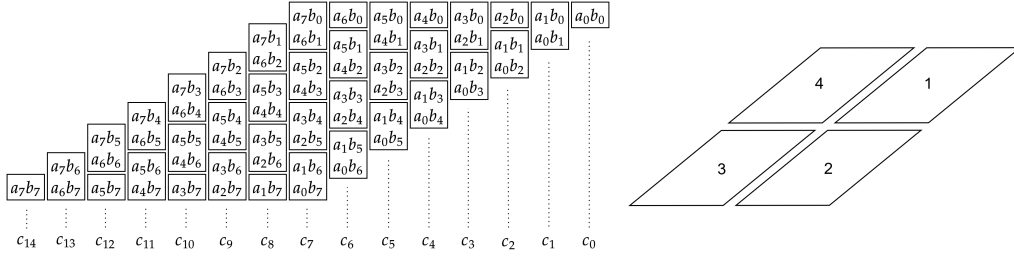


Figure 2: Left: An illustration of the operations executed during 8-coefficient polynomial multiplication  $c(x) = \sum_{i=0}^{14} c_i x^i = a(x) \cdot b(x)$  of arbitrary polynomials  $a(x) = \sum_{i=0}^7 a_i x^i$  and  $b(x) = \sum_{i=0}^7 b_i x^i$ . Right: Sub-division of 16-coefficient schoolbook multiplication into four 8-coefficient multiplication according to specific order as indicated by the numbers in the separate sub-parallellograms [KRS18].

Table 1: Saber and NTRU Toom-Cook parameter sets.

name	$n$	$q$	schoolbook threshold(s)
Saber	256	$1024 = 2^{10}$	16 (16)
ntruhs2048509	509	$2048 = 2^{11}$	16 (32)
ntruhs2048677	677	$2048 = 2^{11}$	10 (24), 11 (40)
ntruhrss701	701	$8192 = 2^{13}$	11 (64)
ntruhs4096821	821	$4096 = 2^{12}$	12 (8), 13 (56)

## 3.2 Generic structure of the known-ciphertext attack

### 3.2.1 Attacker model

We assume that the attacker has physical access to a device performing a decapsulation procedure using a long-term key. Additionally, he can submit as many valid ciphertexts as he wants. He is not required to know the output of the decapsulation, but passively records side-channel information.

### 3.2.2 General outline

The core idea of our attack is to target various executions of schoolbook multiplication, as many as there are sub-polynomials into which the original secret polynomial splits at the schoolbook threshold degree. The recovered sub-polynomials are combined to retrieve the original  $n$ -coefficient polynomial.

Recalling our previous discussion on 4-way Toom-Cook, we consider each of Equations (1), (2), (3) and (4), which all multiply two  $\frac{n}{4}$ -coefficient polynomials. These multiplications usually involve several Karatsuba-layers, followed by the targeted schoolbook multiplication at the threshold degree. For example, Saber uses  $n = 256$  with two Karatsuba-layers and schoolbook threshold degree 16. For this case the structure for Equation (1) is shown in Figure 3: there are 9 multiplications at the threshold degree, but it suffices to target the 4 (non-composite) ones led to by the full arrows, since the total of 64 secret coefficients  $b_{16}^0(x), b_{16:32}^0(x), b_{32:48}^0(x), b_{48}^0(x)$  can be recovered from these. Similarly, we recover the 64 coefficients of  $b^3(x)$  directly from Equation (4). We then combine our knowledge of  $b^0(x)$  and  $b^3(x)$  in order to recover  $b^0(x) + 2 \cdot b^1(x) + 4 \cdot b^2(x) + 8 \cdot b^3(x)$  from Equation (2), and  $b^0(x) + 3 \cdot b^1(x) + 9 \cdot b^2(x) + 27 \cdot b^3(x)$  from (3). Finally,  $b^1(x)$  and  $b^2(x)$  are recovered by solving a linear system.

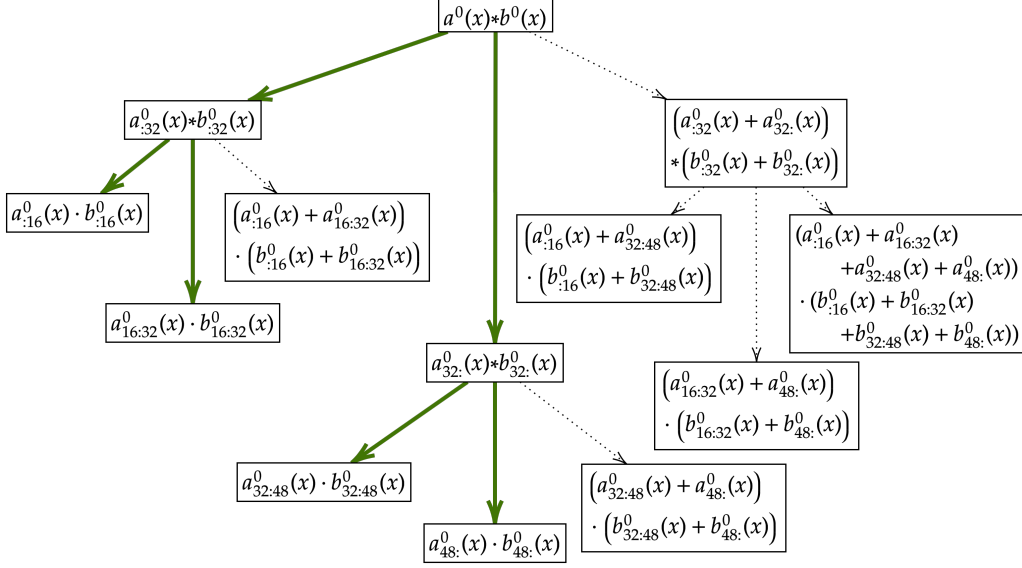


Figure 3: Splitting structure of 64-coefficient polynomials in Saber. The full green lines denote the multiplications exploited in our attack. Schoolbook multiplications are denoted by  $(\cdot)$ .

### 3.2.3 Attack on schoolbook multiplication

We assume to be given a set of  $n$ -coefficient ciphertext polynomials  $a(x) = \sum_{i=0}^{n-1} a_i x^i$  and their corresponding decapsulation side-channel measurements. We also assume that all of these side-channel measurements are captured from a device with a fixed secret polynomial  $b(x) = \sum_{i=0}^{n-1} b_i x^i$ . Our goal is to recover all coefficients  $b_{i \cdot m}, \dots, b_{(i+1) \cdot m - 1}$ , for all  $0 \leq i < \frac{n}{m}$ , when the schoolbook multiplication  $\sum_{j=0}^{m-1} a_{i \cdot m + j} x^j \cdot \sum_{j=0}^{m-1} b_{i \cdot m + j} x^j$  is executed. For ease of notation, we will from now on denote  $a_0 := a_{i \cdot m}, \dots, a_{m-1} := a_{(i+1) \cdot m - 1}$  and  $b_0 := b_{i \cdot m}, \dots, b_{m-1} := b_{(i+1) \cdot m - 1}$  for a certain chosen  $0 \leq i < \frac{n}{m}$ . We first perform a structured CPA-strategy, called *CPA from right to left*, on the schoolbook parallelogram, resulting in a set of guesses for  $b_0, b_1, \dots, b_{m-1}$ . Some of the secret coefficients are not recoverable using CPA. For example, when  $b_0 = \dots = b_i = 0$  for some  $0 \leq i < m$ , the intermediate target values  $c_0, \dots, c_i$  will always be zero, regardless of the input ciphertext coefficient. We propose an iterative approach to rectify incorrect guesses of the secret polynomial coefficients.

**CPA from right to left:** The name of this procedure refers to the direction in which we traverse the schoolbook multiplication parallelogram (recall the parallelogram from Figure 2). The core idea is, given  $b_0, \dots, b_{i-1}$  for some  $0 \leq i < m$ , to determine  $b_i$  as the value  $b_{\text{guess}}$  giving the maximum correlation between the side-channel traces and the Hamming weight of

$$c_i = a_i \cdot b_0 + \dots + a_1 b_{i-1} + a_0 b_{\text{guess}}. \quad (7)$$

If the resulting correlation coefficient is below a threshold  $r_{\text{min}}$ , we set  $b_{\text{guess}} = 0$  and the corresponding correlation to 0. The value of  $r_{\text{min}}$  is determined empirically and will depend on the target platform and measurement setup. As a baseline we recommend taking the average value of the correlation coefficient for the entire trace and multiplying this by a safety margin of 1.5. More details on setting  $r_{\text{min}}$  are provided in Section 5.2.

We chose to use a threshold  $r_{\text{min}}$  to help cope with two scenarios in which the

correlation coefficient for  $c_i$  can be close to zero. The first scenario occurs when  $b_0, \dots, b_i$  are truly 0, as there will not be any value for  $b_{\text{guess}}$  for which the resulting correlation coefficient is larger than  $r_{\text{min}}$ . If all correlation coefficients are below  $r_{\text{min}}$  it is likely that  $b_i$  is 0. Similarly,  $b_{\text{guess}}$  is likely to be correct if the resulting correlation coefficient is larger than  $r_{\text{min}}$ .

Secondly, if a wrong guess was made for one of the  $b_0, \dots, b_{i-1}$  then  $c_i$  yields a low correlation for all possible values of  $b_{\text{guess}}$ . Intuitively, if we incorrectly force our guess for coefficient  $b_i$  to be 0, then the ensuing guesses for  $b_j, i < j < m$  will be wrong as well (i.e. the error propagates). We propose one possible correction mechanism, which we refer to as the iterative approach, to detect and correct such invalid guesses.

**Iterative approach:** Given the regular structure of the multiplication the correlation coefficients for the different secret coefficients are expected to be of the same magnitude. If the correlation for  $b_{m-1}$  is significantly lower compared to when it would have been guessed correctly, then one of the  $b_i, 0 \leq i < m-1$ , was likely guessed incorrectly, and the error will propagate when guessing  $b_{i+1}, \dots, b_{m-1}$ . If guessing  $b_{m-1}$  results in a low correlation we impose the first values  $b_0, \dots, b_j$ , for some  $0 \leq j < m-1$ , to be set to an unrecoverable value. The range of unrecoverable values depends on the target scheme, details are provided in Sections 3.3 and 3.4 for Saber and NTRU respectively. Afterwards we run the CPA from right to left again (but only for  $b_{j+1}, \dots, b_{m-1}$ ). More specifically, for increasing value of  $j$ , we enumerate all combinations of values for  $b_0, \dots, b_j$  which we would not have been able to guess. This iterative process can be automated by setting a minimal correlation value  $t_{\text{min}}$  which needs to be attained by the correlation for  $b_{m-1}$ . The value of  $t_{\text{min}}$  is set in an empirical manner and will be discussed in Section 5.2. We repeat this iterative procedure until the correlation for  $b_{m-1}$  satisfies our condition of being larger than  $t_{\text{min}}$ .

We propose two different iterative approaches: one for  $b^0(x)$  and  $b^3(x)$  (“iterative approach without offset”), and one for  $b^0(x) + 2 \cdot b^1(x) + 4 \cdot b^2(x) + 8 \cdot b^3(x)$  and  $b^0(x) + 3 \cdot b^1(x) + 9 \cdot b^2(x) + 27 \cdot b^3(x)$  (“iterative approach with offset”). For further scheme-specific details of the iterative approaches, we refer to Sections 3.3 and 3.4.

Our choice of only checking the correlation for  $b_{m-1}$  is motivated by the observation of unavoidable error propagation once one of the  $b_i$  is wrongly guessed. Moreover, we observed that, if all coefficients  $b_0, \dots, b_{i-1}$  are correctly guessed, then the correlation for the correct value  $c_i$  will increase for increasing value of  $i$ . If after the iterative approach the correct key is still not recovered the attack should be repeated with either more traces or different values for  $r_{\text{min}}$  and  $t_{\text{min}}$ . Alternatively a more complex attack strategy can be developed that combines the CPA from right to left with a CPA from left to right.

### 3.3 Attack on Saber

In the case of Saber, the ciphertext coefficients  $a_0, \dots, a_{15}$  lie within the range  $\{0, 1, \dots, 1022, 1023\}$ , whereas the secret coefficients  $b_0, \dots, b_{15}$  lie within  $\{0, 1, 2, 3, 4, 1020, 1021, 1022, 1023\}$ . Saber deploys a threshold degree 16 for schoolbook multiplication, of which the structure is split into four equally large 8-coefficient multiplications as illustrated in Figure 2. We target sub-parallelogram 1 for coefficients  $b_0, b_1, \dots, b_7$ , as well as sub-parallelogram 2 for  $b_8, \dots, b_{15}$ . We operate according to the following procedure:

1. Perform CPA from right to left and iterative approach on sub-parallelogram 1, resulting in a set of guesses for  $b_0, b_1, \dots, b_7$ .
2. Perform CPA from right to left (without the iterative approach) on sub-parallelogram 2, resulting in a set of guesses for  $b_8, \dots, b_{15}$ .

3. Adjust the set of guesses from Step 1 based on the quality of the set of guesses from Step 2.

In the following paragraphs we clarify each of the necessary components for this procedure.

**Iterative approach:** For the scenario of “iterative approach without offset”, the cases  $b_0 = b_1 = \dots = b_{i-1} = 0$  and  $b_i \in \{1, 2, 4\}$  for some  $0 \leq i < 7$  will all give the same correlation for  $c_i$ . Recall from Equation (7) that  $c_i$  will have the same Hamming weight for all of these cases. It is possible that one of the above cases occurred for a certain  $0 \leq i < 7$  if correlation for  $b_7$  is not larger than  $t_{\text{min}}$ . If this correlation check fails indeed, then we try out the other two cases for the same value of  $i$ , to see whether the CPA-results for one of these might yield success instead. If  $t_{\text{min}}$  is still not attained, we next consider the cases

- $b_0 = b_1 = \dots = b_i = 0$  and  $b_{i+1} \in \{0, 1023\}$  for some  $0 \leq i < 8$ ,
- $b_0 = b_1 = \dots = b_{j-1} = 0$ ,  $b_j = 1023$ ,  $b_{j+1} = b_{j+2} = \dots = b_i = 0$  and  $b_{i+1} = 0$  for some  $0 \leq j < i < 7$ ,

which will cause the values of  $c_0, \dots, c_i$  to have zero variance (but not  $c_{i+1}$ ). Thus, in these cases, we are unable to guess  $b_0, \dots, b_i$ . The iterative approach enumerates these cases until an instance is found for which the correlation of  $b_7$  reaches  $t_{\text{min}}$ . We start the enumeration by setting  $b_0$  to either 0 or 1023. For any other value of  $b_0$ , the value of  $r_{\text{min}}$  would have been attained during the initial CPA-execution and  $b_7$  would have been guessed with a higher correlation than  $t_{\text{min}}$ . More generally, for increasing value of  $0 \leq i < j < 8$ , we first set  $b_i = 1023$ , and in case of failure, we also set  $b_{i+1} = b_{i+2} = \dots = b_j = 0$  for increasing value of  $0 < j < 8$  until  $t_{\text{min}}$  is attained. If the latter does not occur, we “permanently” set  $b_i$  to 0, and repeat the same procedure for  $i + 1$ .

We explain the procedure of “iterative approach with offset” for  $b^0(x) + 2 \cdot b^1(x) + 4 \cdot b^2(x) + 8 \cdot b^3(x)$ , which is analogous to the procedure for  $b^0(x) + 3 \cdot b^1(x) + 9 \cdot b^2(x) + 27 \cdot b^3(x)$ . For this iterative approach, we assume that all coefficients  $b^0(x)$  and  $b^3(x)$  have already been recovered. It thus remains to recover  $2 \cdot b^1(x) + 4 \cdot b^2(x)$ , of which the coefficients can take on any value  $2 \cdot r + 4 \cdot s$  with  $r, s \in \{0, 1, 2, 3, 4, 1020, 1021, 1022, 1023\}$ . We sort these values by decreasing probability, taking into account the centered binomial distribution of Saber’s secret coefficients. The value of  $b_0$  represents  $b_i^0 + 2 \cdot b_i^1 + 4 \cdot b_i^2 + 8 \cdot b_i^3$  for some  $i \in \{0, 16, 32, 48\}$  (recall Figure 3). We will set the value of  $2 \cdot b_i^1 + 4 \cdot b_i^2$  to be the first entry of the sorted list if the CPA from right to left did not result in a value for  $b_7$  with higher correlation than  $t_{\text{min}}$ . This process is repeated for all entries in the sorted list until a value for  $b_7$  with higher correlation than  $t_{\text{min}}$  is found. From our practical experiments it is clear that only changing the value of  $b_0$  will eventually make the CPA succeed. This iterative approach will only be successful if  $b^0(x)$  and  $b^3(x)$  have been recovered correctly, hence the need for a successful “iterative approach without offset”.

We only perform an iterative approach for the guesses of  $b_0, \dots, b_7$ . When guessing  $b_i$  for some  $8 \leq i < 15$ , we are computing  $c_i$ , for which we already have an offset  $a_7 b_{i-7} + \dots + a_{i-7} b_7$ . Practice has shown that these offsets will result in correct guesses for  $b_8, \dots, b_{14}$ , removing the need for an iterative approach. However, if the guesses for  $b_0, \dots, b_7$  are wrong, then the guesses for  $c_8, \dots, c_{14}$  will be wrong as well.

**Re-assessment of  $b_0, \dots, b_7$  based on  $b_8, \dots, b_{14}$ :** The CPA from right to left followed by the iterative approach proves to be effective when recovering  $b_0, \dots, b_7$ , unless the first few coefficients are zero. That is, if  $b_0 = b_1 = \dots = b_k = 0$ , for some  $0 \leq k < 7$ . In that case our technique will assign the correct value of  $b_{k+1}$  to  $b_0$ , the value of  $b_{k+2}$  to  $b_1$ , and so on. Moreover, the guesses for  $b_{7-k}, \dots, b_7$  will all be set to 0 since this value will each time give the highest correlation for Equation (7). This “shift” in the recovered values

for  $b_0, \dots, b_7$  can be detected by the values  $b_8, \dots, b_{14}$  not being recovered by the iterative approach. We reject the guesses for  $b_0, \dots, b_7$  if  $b_7$  was guessed to be 0 *and* the guess for  $b_{14}$  did not attain a correlation of at least  $t_{\text{min}}$ . We correct the guesses for  $b_0, \dots, b_7$  by “shifting them to the right”, that is, by setting our guess for  $b_{i+1}$  to our guess for  $b_i$ , for  $0 \leq i < 7$ , and our guess for  $b_0$  to 0, after which we again try to recover coefficients  $b_8, \dots, b_{14}$ . This shifting procedure is repeated until  $t_{\text{min}}$  is reached by  $b_{14}$ .

### 3.4 Attack on NTRU

For NTRU the secret coefficients are in the range  $\{0, 1, 2\}$ , whereas the ciphertext coefficients are in the range  $[0, q)$ . Ambiguity will arise if all coefficients of a sub-polynomial are either 0 and 1, or 0 and 2, as the Hamming weight for each  $c_i$  in Equation (7) will be the same for both cases. This is not an unlikely scenario because of the distribution of NTRU’s secret coefficients (see Section 2.2). Nevertheless, we can take this scenario into account when combining all our sub-polynomial guesses into the full polynomial. Depending on the value of  $n$ , different values for the schoolbook threshold are considered, as indicated by Table 1. We make a distinction between the several NTRU-versions based on whether the schoolbook threshold is less than 16 or equal to 16.

#### 3.4.1 ntruhs2048677, ntruhrss701 and ntruhs4096821

These three NTRU-versions have a threshold degree less than 16, for which the schoolbook multiplication consists of one sub-parallelogram instead of four. Additionally, the order of operations has an “overlapping” structure regarding the indices of the  $c_i$ , that is,  $c_1, c_3, c_5, \dots, c_{2(\text{threshold}-1)-1}, c_0, c_2, c_4, \dots, c_{2(\text{threshold}-1)}$  for thresholds 10, 11 and 12, and  $c_1, c_3, c_5, \dots, c_{2(\text{threshold}-1)-1}, c_{2(\text{threshold}-1)}, c_{2(\text{threshold}-1)-2}, \dots, c_4, c_2, c_0$  for threshold 13. An even number of zeros as the first coefficients (when  $b_0 = b_1 = \dots = b_i = 0$  for some odd positive integer  $i$ ) will not be recoverable as only one sub-parallelogram is involved in the schoolbook multiplication process. Nevertheless, as previously mentioned, we can recover from this scenario when combining all the sub-polynomial guesses. Specifically, we can use our guesses that have an even number of zeroes as their last coefficients, and try several combinations that are shifted by an even number.

**Iterative approach:** Given the secret range  $\{0, 1, 2\}$ , when correlating against  $c_0 = a_0 \cdot b_0$  for  $b^0(x)$  or  $b^3(x)$ , we either get zero variance for the Hamming weight of  $c_0$  when  $b_0 = 0$ , or indistinguishable variance when  $b_0 \in \{1, 2\}$  as  $c_0$  will in both cases always have the same Hamming weight. In case the correlation exceeds the value of  $r_{\text{min}}$ , we set our guess for  $b_0$  to 2 and we proceed with the CPA from right to left. If the value for the last coefficient  $b_{\text{threshold}-1}$  was not guessed with at least correlation  $t_{\text{min}}$ , we try out the value  $b_0 = 1$ . In case this fails again, we permanently set  $b_0$  to 0. This procedure is repeated for  $b_i$  for increasing value  $0 \leq i < \text{threshold}$  until  $t_{\text{min}}$  is attained. We extend this “iterative approach without offset” by rejecting guesses of which the locations within the traces do not correspond with the specific ordering of the operations.

The “iterative approach with offset” is the same as for Saber, except now the sorted list of values is much shorter so that the correct value will be found more quickly.

#### 3.4.2 ntruhs2048509

Since this version of NTRU uses the same schoolbook threshold as Saber, we can use the same overall strategy as applied in Saber, but using the iterative approach previously discussed for the other NTRU-versions.

## 4 Correlation Power Analysis on NTT-based polynomial multiplication

In this section, we propose a CPA-approach targeting the NTT, for which we again exploit the underlying schoolbook multiplication. The main two differences with the CPA on Toom-Cook are that the ranges of the secret coefficients are now greatly magnified through the NTT, whereas the schoolbook threshold degree is smaller. Table 2 lists the moduli, as well as the schoolbook threshold degree, deployed by the NTT-based implementations of Kyber, Saber and NTRU. Kyber is the only KEM to originally have an NTT-friendly modulus. Note that  $\text{NTT}_n$ , the size of the NTT, is chosen to efficiently facilitate an NTT, and is always at least as large as  $n$ .

Table 2: Kyber, Saber and NTRU NTT parameter sets.

name	$\text{NTT}_n$	$q$ (or $q'$ )	schoolbook threshold
Kyber768	256	3329	2
Saber	256	25166081	3
ntruhs2048509	1024	1043969	4
ntruhs2048677	1536	1389569	3
ntruhss701	1536	5747201	3
ntruhs4096821	1728	3365569	3

### 4.1 Schoolbook multiplication implementation

Even though most NTT-based multiplications used in different PQC schemes deploy unique NTT-structures, the implementation of the schoolbook multiplication is always similar. We illustrate the concept for threshold degree 3 (as in ntruhs4096821), which can be easily generalized for arbitrary degrees. Given are two polynomials in the NTT-domain  $\hat{a}(x) = \hat{a}_0 + \hat{a}_1x + \hat{a}_2x^2$  and  $\hat{b}(x) = \hat{b}_0 + \hat{b}_1x + \hat{b}_2x^2$ , defined over  $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^3 - \gamma)$  for some power of a twiddle factor  $\gamma$ . Modular multiplication of these two polynomials yields

$$\begin{aligned} \hat{a}(x) \cdot \hat{b}(x) &= (\hat{a}_0 + \hat{a}_1x + \hat{a}_2x^2) \cdot (\hat{b}_0 + \hat{b}_1x + \hat{b}_2x^2) \bmod x^3 - \gamma \\ &= (\hat{a}_0\hat{b}_0 + (\hat{a}_1\hat{b}_2 + \hat{a}_2\hat{b}_1)) + (\hat{a}_0\hat{b}_1 + \hat{a}_1\hat{b}_0 + \hat{a}_2\hat{b}_2)x + (\hat{a}_0\hat{b}_2 + \hat{a}_1\hat{b}_1 + \hat{a}_2\hat{b}_0)x^2 \\ &= \hat{c}_0 + \hat{c}_1x + \hat{c}_2x^2. \end{aligned} \tag{8}$$

For the source code implementing these computations, we refer to the pqm4-library [KRSS]. The computation of  $\hat{c}_0$  involves the addition  $\hat{a}_1\hat{b}_2 + \hat{a}_2\hat{b}_1$ , followed by a Montgomery multiplication with  $\gamma^{-1}$ , after which  $\hat{a}_0\hat{b}_0$  is added. The coefficient  $\hat{c}_1$  is computed in a similar manner. No Montgomery multiplication by  $\gamma^{-1}$  is needed when computing  $\hat{c}_2$ . For efficiency reasons involving the inverse NTT, one last Montgomery reduction is applied to the final values of all three  $\hat{c}_0$ ,  $\hat{c}_1$ ,  $\hat{c}_2$ , before being stored consecutively. We will target the Hamming weight of the first resulting coefficient being stored, and the Hamming distance between consecutively stored coefficients, the order of which depends on the target implementation.

### 4.2 Generic structure of the side-channel attack

#### 4.2.1 Attacker model

We assume the same attacker model as for our CPA on Toom-Cook-based multiplication. The only difference is that we can not simply use any random, valid ciphertexts for our attacks on NTRU's and Saber's NTT.

For NTRU, while targeting an  $m \times m$  schoolbook multiplication we would need to consider all possible combinations of the  $m$  secret values involved for each of the  $m$  resulting coefficients. We refer to the second line in Equation (8), that shows how to compute the  $m$  resulting values for the case of ntruhs4096821. The number of combinations  $(q)^m$  increases with the value of the modulus  $q$ . We propose a strategy that eliminates the need to guess these combinations for NTRU in Section 4.4. As Saber uses an even larger modulus  $q$  we will resort to guessing the secret coefficients byte per byte, which reduces our search space to  $0, 2^8$ . This attack strategy is explained in more detail in Section 4.5 and requires ciphertexts of which the NTT-coefficients can take on any 32-bit values.

#### 4.2.2 General outline

We are given a size- $\text{NTT}_n$  NTT that uses  $m$ -coefficient schoolbook multiplication. That is, after applying the necessary NTT-layers on two  $n$ -coefficient polynomials  $a(x)$  and  $b(x)$ , there are  $\frac{\text{NTT}_n}{m}$  schoolbook multiplications to perform of the form

$$\begin{aligned} & \hat{a}_{m-i} + \dots + \hat{a}_{m-i+m-1}X^{m-1} \cdot \hat{b}_{m-i} + \dots + \hat{b}_{m-i+m-1}X^{m-1} \pmod{(X^m - \omega^i)} \\ & = \hat{c}_{m-i} + \dots + \hat{c}_{m-i+m-1}X^{m-1} \end{aligned}$$

for some power of a twiddle factor  $\omega^i$  and  $0 \leq i < \frac{\text{NTT}_n}{m}$ . Our aim is to target each of these  $\frac{\text{NTT}_n}{m}$  schoolbook multiplications to retrieve all coefficients  $\hat{c}_{m-i+j}$ ,  $0 \leq i < \frac{\text{NTT}_n}{m}$ ,  $0 \leq j < m$ . To check the correctness of our guesses, we simply apply the inverse NTT and check whether the result matches the coefficients of  $c(x) = a(x) \cdot b(x) \pmod{X^n + 1}$ .

#### 4.2.3 Attack on schoolbook multiplication

As mentioned in Section 4.1, given a schoolbook multiplication characterized by some index  $0 \leq i < \frac{\text{NTT}_n}{m}$ , we target the storage of all resulting values  $\hat{c}_{m-i+j}$ ,  $0 \leq j < m$ . The final Montgomery reduction reduces the coefficients  $\hat{c}_{m-i+j}$  to the interval  $[-\frac{q}{2}, \frac{q}{2}]$ . As explained later in Section 4.6, it suffices to consider this same range for the coefficients  $\hat{b}_{m-i+j}$ . We select the value of  $\hat{b}_{m-i+j}$  with highest correlation for  $\hat{c}_{m-i+j}$ .

**Iterative approach:** As for the attack on Toom-Cook-based multiplication, we can use an iterative approach to correct invalid guesses. Only for relatively small moduli, such as  $q = 3329$  for Kyber, we do not explicitly implement an iterative approach, as we can already iterate over all possible combinations for  $(\hat{c}_{2i}, \hat{c}_{2i+1})$ ,  $0 \leq i < 128$ , in a reasonable amount of time. For larger moduli, as in the NTRU-based schemes, the range of possible values is much larger, so that we can no longer consider all combinations, as further explained in Section 4.4.

### 4.3 Attack on Kyber

Kyber is the only KEM in this paper using a relatively small NTT-modulus. Since  $n = \text{NTT}_n = 256 = 2^8$  and  $q = 3329 = 13 \cdot n + 1$ , Kyber can facilitate a 7-layer, incomplete NTT, followed by a 2-coefficient schoolbook multiplication. In the implementation, the two resulting coefficients  $\hat{c}_{2i} = \hat{a}_{2i}\hat{b}_{2i} + \hat{a}_{2i+1}\hat{b}_{2i+1}$  and  $\hat{c}_{2i+1} = \hat{a}_{2i}\hat{b}_{2i+1} + \hat{a}_{2i+1}\hat{b}_{2i}$ , for a  $0 \leq i < 128$  are stored by the same instruction instead of consecutively. We thus need to guess two coefficients at once within the range  $[-\frac{q}{2}, \frac{q}{2}]$ , implying a search over  $q^2 = 11082241$  combinations. Iterating over this number of combinations can still be done efficiently. We simply select the pair of values resulting in the largest correlation coefficient as our key guess.

## 4.4 Attack on NTRU

As opposed to Kyber, the NTT-moduli for all NTRU-versions are rather large, on the order of  $2^{20}$  (recall Table 2). It thus becomes computationally intensive, but not impossible, to try out all possible combinations of the secret coefficients when analyzing the schoolbook multiplication. For a schoolbook threshold  $m$ , recovering the secret coefficients  $\hat{b}_{m:i}, \dots, \hat{b}_{(m+1):i-1}$  requires making  $2^{20 \cdot m}$  key guesses, when correlating against the resulting  $\hat{c}_{m:i}, \dots, \hat{c}_{(m-1):i-1}$ , for any  $0 \leq i < \frac{\text{NTT}_n}{m}$ . This makes correlating against the Hamming weight of all possible key values nearly impossible.

An alternative approach simplifies the attack by choosing each ciphertext polynomial  $a(x)$  in such a way that only NTT-coefficients of the form  $\hat{a}_{m:i}, 0 \leq i < \frac{\text{NTT}_n}{m}$ , can be non-zero. This would imply that  $\hat{c}_{m+i+j} = \hat{a}_{m:i} \cdot \hat{b}_{m+i+j}$ , for all  $0 \leq j < m$ , hence we need not be concerned with any of the Montgomery multiplications by a power of a twiddle factor  $\omega$ . For such a set of NTT-coefficients with “regular” indices to be all zero, we can choose the corresponding original ciphertext coefficients to be zero. Such ciphertexts  $c$  will still pass the check in Line 1 of Algorithm 5. This approach will however lead to decryption failures which can be detected by the device under attack.

We illustrate the specifications of the attack for the case of ntruhs4096821. After carefully selecting the ciphertext coefficients, we correlate against the Hamming weight of the resulting coefficients, each time they are stored during execution. Depending on the NTRU-version, the number of consecutive storing operations varies. For example, ntruhs4096821 stores its resulting coefficients in groups of  $24 = 8 \cdot 3$  coefficients. The values  $\hat{c}_{24+i+j}$  are (mostly) stored sequentially for increasing value of  $0 \leq j < 24$ , so that we can correlate against the Hamming weight of  $\hat{c}_{24i}$  when guessing its value. We then correlate against the Hamming distance between  $\hat{c}_{24i}$  and  $\hat{c}_{24i+1}$  when guessing the latter coefficient, and so on, for all  $0 \leq i < 72$ . Hamming distance implies a dependency between two coefficients, in the sense that  $\hat{c}_{24i+1}$  can only be correctly guessed if  $\hat{c}_{24i}$  was correctly guessed, and so on. We exploit this characteristic in the iterative approach. These observations are of course implementation-specific: if a different implementation strategy stores (for example) intermediates, then our attack would be greatly simplified as we can target the individual  $\hat{c}_i$ 's.

**Iterative approach:** As the range  $[-\frac{q}{2}, \frac{q}{2}]$  can become quite large, we prematurely stop our brute-force search over this interval once we guess a value for each  $\hat{c}_{24i}, 0 \leq i < 72$ , with correlation exceeding a given threshold. The threshold can be established in a similar way as `r_min` and `t_min` in the CPA on Toom-Cook. We then continue by guessing  $\hat{c}_{24i+1}$ , based on our guess for  $\hat{c}_{24i}$ . If no value can be found for  $\hat{c}_{24i+1}$  with high enough correlation, we back-track to our guess for  $\hat{c}_{24i}$ , eliminate it, and sweep over the remaining range of  $[-\frac{q}{2}, \frac{q}{2}]$  to find another guess for  $\hat{c}_{24i}$ . We repeat this procedure until we find a guess for  $\hat{c}_{24i+1}$  with high enough correlation, and then we proceed analogously for  $\hat{c}_{24i+2}$ , and so on.

## 4.5 Attack on Saber

For Saber the modulus  $q$  is 25166081, roughly 10 times larger compared to the moduli for NTRU. The increased range  $[-\frac{q}{2}, \frac{q}{2}]$  of values for the secret coefficients decreases the efficiency of some of the previous CPA-approaches, as in the best-case scenario an attacker would have to iterate over roughly  $2^{20}$  key guesses. The exact number of key guesses that have to be enumerated depends on the implementation and leakage model. For the provided reference implementation we would have to recover four secret coefficient simultaneously, as the results of the  $4 \times 4$  schoolbook multiplication are stored consecutively.



In practice, this mean we would have to make roughly  $2^{80}$  guesses. To alleviate this issue we propose a different strategy that targets the scalar multiplication on a smaller level.

Given two 32-bit values  $\hat{a} = \hat{a}_{[3]} \cdot 2^{24} + \hat{a}_{[2]} \cdot 2^{16} + \hat{a}_{[1]} \cdot 2^8 + \hat{a}_{[0]}$  (known) and  $\hat{b} = \hat{b}_{[3]} \cdot 2^{24} + \hat{b}_{[2]} \cdot 2^{16} + \hat{b}_{[1]} \cdot 2^8 + \hat{b}_{[0]}$  (secret) to be multiplied. This byte notation simplifies the following statements. If we choose  $\hat{a}_{[0]} = \hat{a}_{[1]} = \hat{a}_{[2]} = 0$ , then

$$\hat{a} \cdot \hat{b} \bmod 2^{32} = \hat{a}_{[3]} \cdot \hat{b}_{[0]} \bmod 2^8.$$

The right-hand side of the latter equation can take on at most  $2^8$  values, so that we can guess  $\hat{b}_{[0]}$  in a reasonable amount of time. Next, if we choose  $\hat{a}_{[0]} = \hat{a}_{[1]} = 0$ , we observe that

$$\hat{a} \cdot \hat{b} \bmod 2^{32} = \hat{a}_{[3]} \cdot \hat{b}_{[0]} \cdot 2^8 + \hat{a}_{[2]} \cdot \hat{b}_{[1]} \cdot 2^8 + \hat{b}_{[0]} \quad \bmod 2^{16}.$$

Since we already know  $\hat{a}_{[3]}$ ,  $\hat{b}_{[0]}$  and  $\hat{a}_{[2]}$ , the right-hand side of the latter equation can again only take on at most  $2^8$  distinct values, so we can guess  $\hat{b}_{[1]}$ . In an analogous way, we recover  $\hat{b}_{[2]}$  and then finally  $\hat{b}_{[3]}$ .

Note that we previously correlated against store operations for all CPAs, whereas we now consider the 32 least significant bits of the results of 32-bit multiplications. We found the individual scalar multiplications in

$$\hat{a}_{4i+3} \cdot \hat{b}_{4i+2} + \hat{a}_{4i+2} \cdot \hat{b}_{4i+3} + \hat{a}_{4i+1} \cdot \hat{b}_{4i} + \hat{a}_{4i} \cdot \hat{b}_{4i+1}, 0 \quad i < 64, \quad (9)$$

to yield the largest correlation against the traces of all four computations in Saber's  $4 \times 4$  schoolbook multiplication. The eight NTT-coefficients are multiplied, and subsequently accumulated in the same order as shown in Equation (9). We execute the (accumulated) CPA-approach as explained earlier each time such a new scalar multiplication is performed. We use the Hamming-weight model when correlating against  $\hat{a}_{4i+3} \cdot \hat{b}_{4i+2}$ , and the Hamming-distance model for all ensuing operations. This attack strategy requires the attacker to submit invalid ciphertexts and can be detected by the device under attack if suitable countermeasures are implemented.

## 4.6 A note on the binomial distribution

Even if the polynomial coefficients of the original secret polynomial  $a(x)$  are restricted to a very small range, applying the NTT will greatly enlarge this range, of which the size is on the order of the modulus  $q$  (or  $q$ ) times the number of NTT-layers. The distribution appears to be bell-shaped, as shown in Figure 4 on the left for Saber. However, all intermediate results in the computation of the (inverse) NTT are equivalent mod  $q$  (or mod  $q$ ). Hence, we only need consider reduced secret NTT-values within, for example, the range  $-\frac{q}{2}, \frac{q}{2}$  (or  $-\frac{q}{2}, \frac{q}{2}$ ). We collapse the bell-shaped distribution to this interval in the rightmost plot in Figure 4. This plot shows that there is no distinctive distribution in the new range, so we will consider the distribution to be uniform. We therefore do not take into account the original centered binomial distribution of the secret polynomial coefficients in our CPA.

## 5 Results

This section covers the practical CPA results for both Toom-Cook and the NTT, as proposed in Sections 3 and 4, respectively. The attacks are implemented in Python and executed on a MacBook with 1.3 GHz Dual-Core Intel Core i5 processor. All pqm4-implementations were compiled using the arm-none-eabi-gcc compiler. The implementations are mostly written in C, but large parts of the Toom-Cook and NTT-based polynomial multiplication are written in assembly.

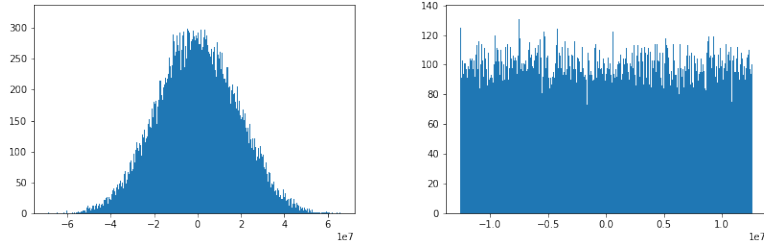


Figure 4: Histograms illustrating the distribution of the secret NTT-coefficients in Saber, un-reduced (left) and reduced to  $[-\frac{q}{2}, \frac{q}{2}]$  (right),  $10^5$  samples, 1000 bins.

## 5.1 Measurement set-up

Our attack results are based on the code provided by the `pqm4`-library [KRSS], a common framework for implementations and benchmarks of the NIST PQC finalists on the ARM Cortex-M4 microprocessor, which is equipped with the 32-bit ARMv7E-M architecture. This resource-constrained platform is often used for efficiency evaluation of cryptographic implementations due to its large enough memory to support public-key algorithms, while still being reasonably small and cheap [KMRV18].

Side-channel measurements were acquired from a NewAE CW308 UFO board combined with a STM32F415 (Cortex-M4) target board. The instantaneous power consumption was amplified using a Mini-Circuits ZFL-1000LN+ amplifier, low-pass filtered (20 MHz) and sampled by a Tektronix DPO7254C oscilloscope using a sample rate of 200 MS/s. The STM32F415 microcontroller was configured to use an 8 MHz external clock and 24 MHz core operating frequency. The external clock was provided by a signal generator that shared a reference clock with the oscilloscope.

## 5.2 CPA on Toom-Cook

The acquired side-channel traces cover the polynomial multiplication, but not the full decapsulation to reduce the runtime. The trace length increases with the value of  $n$ : we use traces of  $3.5 \cdot 10^5$  samples for  $n = 256$  (Saber) and  $5 \cdot 10^6$  samples for  $n = 821$  (ntruhs4096821).

To reduce the overall runtime of the attack, we first locate each schoolbook multiplication and segment the input trace. In this way the Pearson correlation coefficient can be calculated over short trace segments instead of the full trace. We locate a schoolbook multiplication by searching the moment in time where the first non-zero  $b_i$ ,  $0 < i < m$ , is involved in the computation of  $c_i$ . As  $b_0 = \dots = b_{i-1} = 0$ , there is only a small range of values that  $c_i = a_0 \cdot b_i$  can take on (depending on the range of  $b_i$ ), for each of which we correlate against the traces. The instance with highest correlation gives us a reference point in time at which the schoolbook multiplication is performed, and also provides us with a reference for selecting  $r_{\text{min}}$  and  $t_{\text{min}}$ . The value of  $r_{\text{min}}$  is set to a generous lower bound on these correlation values, for which we chose 0.05. For  $t_{\text{min}}$  we consider an average of the correlations. Regarding Saber and ntruhs2048509 (which both use a threshold degree 16) we set  $t_{\text{min}}$  to 0.4, whereas a more generous value 0.5 is considered for the other NTRU-based schemes.

**Saber and NTRU** Table 3 lists the minimum number of traces for the CPA on Toom-Cook for full recovery of a secret polynomial as well as the average runtimes. On average the required number of traces decreases for a smaller schoolbook threshold and for a lower

Table 3: Saber and NTRU Toom-Cook CPA runtimes.

name	minimum number of traces	average runtime (s)
Saber	200	4
ntruhs2048509	400	10
ntruhs2048677	100	23
ntruhrss701	200	32
ntruhs4096821	100	38

value of  $n$ . The average runtime increases with  $n$ . Moreover, the runtime is affected by the average duration of the iterative approach for each sub-polynomial, which will depend on the signal-to-noise ratio (SNR) of the measurements and the selected `r_min` and `t_min`.

The proposed approach can be easily adapted for other versions of Saber, such as the higher-security `Fi reSaber` that uses a reduced set of secret coefficients  $\{0, 1, 2, 3, 1021, 1022, 1023\}$ . In fact, this reduced set of secret coefficients would speed up the iterative approach resulting in lower overall runtime. For NTRU, the runtimes only consider a single full run of guessing all coefficients, iterative approach included. This does not include testing for combinations with “shifted zeros” for  $b^0(x)$  and  $b^3(x)$ , which would require new guesses for  $b^0(x) + 2 \cdot b^1(x) + 4 \cdot b^2(x) + 8 \cdot b^3(x)$  and  $b^0(x) + 3 \cdot b^1(x) + 9 \cdot b^2(x) + 27 \cdot b^3(x)$ . The number of such cases is relatively small, resulting in a minor impact on the overall runtime.

### 5.3 CPA on NTT

Note that guessing groups of NTT-coefficients can be easily parallelized, since individual groups can be guessed independently. The runtimes provided in Table 4 were obtained using sequential recovery without parallelization. The side-channel traces cover the polynomial multiplication, and the number of samples per trace ranges from  $2 \cdot 10^4$  for Kyber to  $5 \cdot 10^5$  for ntruhs4096821.

The following technique was used to reduce the number of time samples used during the CPA attack. Schoolbook multiplication is implemented in such a way that each of the  $m \times m$  schoolbook multiplications of two pairs of  $m$  NTT-coefficients is followed by the  $m \times m$  schoolbook multiplication of the next two pairs of  $m$  NTT-coefficients. Depending on the implementation, resulting coefficients can be stored after each  $m \times m$  multiplication, or they can be stored after  $j$  multiplications, for some  $j \geq 1$ . We consider all these  $j \cdot m$  coefficients being consecutively stored as a group. For each group, we can narrow down the approximate location within the measured traces of when the coefficients are being stored; we search for as many distinctive peaks in the variance as there are groups. Based on these variance peaks, we can restrict our CPA to much smaller ranges than the full trace domain. We illustrate our reasoning for the case of ntruhs4096821 with size-1728 NTT, schoolbook threshold degree 3, and eight  $3 \times 3$  schoolbook multiplications per group. We thus expect  $\frac{1728}{8 \cdot 3} = 72$  variance peaks, as shown in Figure 5, which plots the trace variance during the full schoolbook multiplication that takes place between the forward and inverse NTT.

**Kyber** Our approach requires 200 traces to recover all  $\text{NTT}_n$  coefficients. Guessing one pair of coefficients takes roughly 5 minutes on average. Here we used a brute-force approach where the coefficients resulting from the modular multiplication were guessed in their entirety. Therefore we only provided valid ciphertexts as inputs to the decapsulation.

**NTRU** We ran the CPA on NTRU’s NTT with each time 500 traces. Due to the long run-time we fixed the number of traces for each NTRU-version to 500. We attacked NTRU using the alternative approach proposed in Section 4.4 due to the high time complexity of performing an attack with only valid ciphertexts.

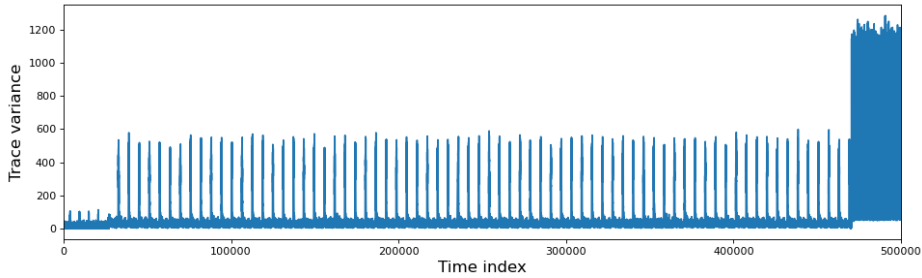


Figure 5: Variance plot of 500 traces of schoolbook multiplication in ntruhs4096821, showing 72 peaks.

We considered the following values for the iterative approach threshold: 0.3 for ntruhs2048509 and ntruhs4096821, and 0.5 for ntruhs2048677 and ntruhrss701. We clearly see the influence of the size of the NTT-modulus  $q$  on the runtime per group of coefficients in Table 3. The larger the value of  $q$ , the larger the range of values to brute-force over. Note that, unlike for Kyber, the runtimes also cover the iterative approach needed for corrections. Table 3 lists in the third column the number of elements contained in each independent group, as well as the number of groups in the fourth column.

Table 4: NTRU NTT CPA runtimes.

name	average running time per group (s)	number of elements per group	number of groups
ntruhs2048509	300	16 (4 sets of 4)	64
ntruhs2048677	282	3	512
ntruhrss701	1040	3	512
ntruhs4096821	9840	24 (8 sets of 3)	72

**Saber** Similar to NTRU we applied the alternative approach when attacking Saber (see Section 4.5). In contrast with NTRU we do not target the Hamming distance leakage of the full 32-bit state but target only 8 bits at a time. Targeting a smaller portion of the state reduces the SNR and results in a significant increase in the number of traces required to mount the attack. A minimum of 10000 traces was required to mount a successful attack. Given the reduced number of target bits, the correct guess is not always ranked highest. We therefore enumerate all possible values for  $\hat{b}_{[0]}$ , for each of these we then perform the proposed CPA to guess the remaining three bytes. Afterwards, we select the full value  $\hat{b} = \hat{b}_{[3]} \cdot 2^{24} + \hat{b}_{[2]} \cdot 2^{16} + \hat{b}_{[1]} \cdot 2^8 + \hat{b}_{[0]}$  giving the highest correlation. Guessing one of the four bytes takes about 20s on average, which we have to repeat  $4 \cdot 256$  per secret NTT-coefficient.

## 6 Conclusion

In this paper we propose a non-profiled side-channel attack methodology that targets polynomial multiplication. We apply our methodology to all lattice-based KEMs in the final round of the NIST PQC competition. In doing so we target both Toom-Cook-based multiplication and NTT-based multiplication. Both of these polynomial multiplication strategies perform point-wise multiplication at a threshold level before the results are interpolated. We demonstrate how the secret operands involved in these scalar multiplications can be recovered using correlation power analysis. Our experiments show that Toom-Cook-based polynomial multiplication is more straightforward to attack. This is

reflected in the large difference between the runtimes of the attacks targeting Toom-Cook implementations with the attacks on implementations of the NTT-based multiplication. While we demonstrate our methodology for Kyber, Saber and NTRU, it is also applicable to any other unprotected lattice-based scheme using Toom-Cook or the NTT for polynomial multiplication. The proposed attacks can be mitigated by using masking and hiding countermeasures [BDK<sup>+</sup>21, BGR<sup>+</sup>21].

## 7 Acknowledgements

This work was supported in part by Semiconductor Research Corporation (SRC).

## References

- [AASA<sup>+</sup>17] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody and Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the second round of the nist post-quantum cryptography standardization process. <https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf>, 2017. [Online; accessed 08-Oct-2021].
- [ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178. Springer, 2016.
- [ABD<sup>+</sup>21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber. algorithm specifications and supporting documentation. (round 3 submission). <https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf>, 2021. [Online; accessed 30-September-2021].
- [ACC<sup>+</sup>21] Erdem Alkim, Dean Yun-Li Cheng, Chi-Ming Marvin Chung, Hülya Evkan, Leo Wei-Lun Huang, Vincent Hwang, Ching-Lin Trista Li, Ruben Niederhagen, Cheng-Jhih Shih, Julian Wälde, and Bo-Yin Yang. Polynomial multiplication in NTRU prime comparison of optimization strategies on Cortex-M4. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):217–238, 2021.
- [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.
- [Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *In Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108. ACM, 1996.
- [APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.
- [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.
- [BDH<sup>+</sup>21] Shivam Bhasin, Jan-Pieter D’Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):334–359, 2021.
- [BDK<sup>+</sup>21] Michiel Van Beirendonck, Jan-Pieter D’Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of SABER. *ACM J. Emerg. Technol. Comput. Syst.*, 17(2):10:1–10:26, 2021.

- [BGR<sup>+</sup>21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021.
- [BMD<sup>+</sup>20] Andrea Basso, Jose Maria Bermudo Mera, Jan-Pieter D’Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Michiel Van Beirendonck, and Frederik Vercauteren. SABER: Mod-LWR based KEM (Round 3 Submission). <https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf>, 2020. [Online; accessed 30-September-2021].
- [BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012.
- [CDH<sup>+</sup>19] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU algorithm specifications and supporting documentation. Second PQC Standardization Conference, 2019, University of California, Santa Barbara, USA, 2019.
- [CDH<sup>+</sup>20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Tsunekazu Saito, Peter Schwabe, William Whyte, Keita Xagawa, Takashi Yamakawa, and Zhenfei Zhang. NTRU algorithm specifications and supporting documentation,. Second PQC Standardization Conference, 2019, University of California, Santa Barbara, USA, 2020. [Online; accessed 30-September-2021].
- [CHK<sup>+</sup>21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. NTT multiplication for ntt-unfriendly rings new speed records for saber and NTRU on Cortex-M4 and AVX2. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):159–188, 2021.
- [Coo66] S. A. Cook. *On the Minimum Computation Time of Functions*. PhD thesis, Harvard University, 1966. pp. 51-77.
- [CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.
- [DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.
- [DKRV19] Jan Pieter D’Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Mod-LWR based kem,. Second PQC Standardization Conference, 2019, University of California, Santa Barbara, USA, 2019.
- [FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO’ 99*, pages 537–554, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

- [FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, Jan 2013.
- [GS66] W. M. Gentleman and G. Sande. Fast fourier transforms: For fun and profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference, AFIPS '66 (Fall)*, page 563–578, New York, NY, USA, 1966. Association for Computing Machinery.
- [HCY20] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on NTRU prime. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):123–151, 2020.
- [HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.
- [HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory: Third International Symposium, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, pages 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
- [HRSS17] Andreas Hülsing, Joost Rijneveld, John M. Schanck, and Peter Schwabe. High-speed key encapsulation from NTRU. In 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*, pages 232–252. Springer, 2017.
- [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 Maria 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.
- [KO62] A. Karatsuba and Yu. Ofman. Multiplication of many-digital numbers by automatic computers. *Proceedings of USSR Academy of Sciences*, 145(7):293–294, 1962.
- [Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–203, 1987.
- [KRS18] 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. Cryptology ePrint Archive, Report 2018/1018, 2018. <https://epri nt. i ac r. org/2018/1018>,.
- [KRSS] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. <https://gi thub. com/mupq/pqm4>; Accessed 30-September-2021.



- [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, French Riviera, May 30 – June 3, 2010. Proceedings*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.
- [LS19] Vadim Lyubashevsky and Gregor Seiler. NTTRU: truly fast NTRU using NTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):180–201, 2019.
- [LSCH10] Mun-Kyu Lee, Jeong Song, Dooho Choi, and Dong-Guk Han. Countermeasures against power analysis attacks for the ntru public key cryptosystem. *IEICE Transactions*, 93-A:153–163, 01 2010.
- [Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426. Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.
- [MKV20] Jose Maria Bermudo Mera, Angshuman Karmakar, and Ingrid Verbauwhede. Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):222–244, 2020.
- [Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.
- [NIS17] NIST. Post-quantum cryptography standardization. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization>, 2017. [Online; accessed 10-Oct-2021].
- [Pol71] J. M. Pollard. The fast fourier transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.
- [PP19] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019.
- [PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In 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*, pages 513–533. Springer, 2017.
- [REB<sup>+</sup>21] Prasanna Ravi, Martianus Frederic Ezerman, Shivam Bhasin, Anupam Chattopadhyay, and Sujoy Sinha Roy. Generic side-channel assisted chosen-ciphertext attacks on streamlined NTRU prime. *IACR Cryptol. ePrint Arch.*, page 718, 2021.
- [Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on theory of computing*, STOC '05, pages 84–93. ACM, 2005.
- [RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.
- [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.
- [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.
- [SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551. Springer, 2018.
- [Too63] A.L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics-Doklady*, volume 7, pages 714–716, 1963. <http://toomandre.com/my-articles/engmat/MULT-E.PDF>.
- [XPRO20] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David F. Oswald. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber. *IACR Cryptol. ePrint Arch.*, page 912, 2020.