IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 285–310. DOI:10.46586/tches.v2022.i4.285-310

Single-Trace Side-Channel Attacks on the Toom-Cook: The Case Study of Saber

Yanbin Li[1], Jiajie Zhu[1], Yuxin Huang[1], Zhe Liu[2,3] and Ming Tang[4]

[1] Nanjing Agricultural University, Nanjing, China, yanbinli@njau.edu.cn
[2] Zhejiang Lab, Hangzhou, China
[3] Nanjing University of Aeronautics and Astronautics, Nanjing, China, zhe.liu@nuaa.edu.cn
[4] Wuhan University, Wuhan, China, m.tang@126.com

Abstract. The Toom-Cook method is a well-known strategy for building algorithms to multiply polynomials efficiently. Along with NTT-based polynomial multiplication, Toom-Cook-based or Karatsuba-based polynomial multiplication algorithms still have regained attention since the start of the NIST's post-quantum standardization procedure. Compared to the comprehensive analysis done for NTT, the leakage characteristics of Toom-Cook have not been discussed. We analyze the vulnerabilities of Toom-Cook in the reference implementation of Saber, a third round finalist of NIST's post-quantum standardization process. In this work, we present the first single-trace attack based on the soft-analytical side-channel attack (SASCA) targeting the Toom-Cook. The deep learning-based power analysis is combined with SASCA to decrease the number of templates since there are a large number of similar operations in the Toom-Cook. Moreover, we describe the optimized factor graph and improved belief propagation to make the attack more practical. The feasibility of the attack is verified by evaluation experiments. We also discuss the possible countermeasures to prevent the attack.

Keywords: post-quantum cryptography · Saber KEM · Toom-Cook · side-channel attack · deep learning

1 Introduction

The impending threat of Shor's algorithm [Sho99] to conventional public-key cryptographic algorithms has prompted interest in alternate algorithms that are resistant to quantum computers. The National Institute of Standards and Technology (NIST) started the Post-Quantum Cryptography Standardization Project in 2016 [NIS]. The process for post-quantum cryptography is currently in the third round. The remaining candidates are the seven finalists and the eight alternates for Public Key Encryption (PKE), Key-Encapsulation Mechanism (KEM) and Digital Signature (DS) schemes [AASA+20]. Among the finalists for KEMs, 3 out of 4 finalists are latticed-based.

The lattice-based schemes are split into Learning With Errors (LWE)-based schemes and NTRU-based schemes. Their security relies on the ideal lattices and learning with errors problems. However, their implementations have shown vulnerability against side-channel attacks (SCAs) in the context of PKE, KEM or DS [ATT+18, EFGT17, OSPG18]. Side-channel attacks aim to establish the relationship between the detectable leakages from physical devices and sensitive information. NIST has clarified the resistance against SCAs should be considered as an essential criterion for the Post-Quantum Cryptography Standardization Project [AASA+20].

There has been a lot of work focus on the risks of lattice-based cryptographic systems to side-channel attacks. Several works exploit vulnerabilities of different operations, including

Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

but not limited to polynomial multiplication [PPM17, HCY19, XPSR+21, AKJ+18], message encoding/decoding [ACLZ20, RBRC22, NDGJ21], Gaussian sampler [Pes16, KH18], or the Fujisaki-Okamoto (FO) transform [RSRCB20, BDH+21]. In particular, the polynomial multiplication has attracted more attention for the vulnerability of long-term secret key. Huang et al. performed various power attacks on NTRU Prime [HCY19]. The horizontal DPA was proposed to reduce the traces required for the attack on implementations of NewHope and Frodo on FPGA [ATT+18]. A more generic long-term key recovery attack was proposed by Ravi et al. on several lattice-based PKE and KEMs [RSRCB20]. However, it needed a large number of traces for full key recovery. Xu et al. constructed special ciphertexts to classify secret coefficients in Kyber with a lower number of traces [XPSR+21].

An extreme case of these attacks is the single-trace attack, where the adversary recovers the sensitive information with a single measurement. The single-trace attack was viewed as the worst-case side-channel security evaluation, and many publications researched the single-trace vulnerabilities of KEMs. In CHES 2017, a single-trace attack was performed on the NTT-based polynomial multiplication [PPM17], and it was improved in the later works [PP19, HHP+21]. Other works on the single-trace side-channel attack have targeted the sampling [EFGT17, KAA21, AKP+], rejection procedure [FDK20], and message encoding/decoding [SKL+20, ACLZ20].

Among the three third round finalists of KEMs, NTT-based polynomial multiplication is used in the Kyber scheme [BDK+18], while NTRU-Prime [BCLVV16] and Saber [DKSRV18] use Toom-Cook-based [Too63, CA69] and Karatusba-based [KO62] polynomial multiplications. NTT-based polynomial multiplication is ubiquitously used in schemes for multiplications are low time complexity. While the inherent design constraints of NTT-based polynomial multiplication, Toom-Cook-based and Karatsuba-based polynomial multiplication with well chosen parameters and optimal technologies fare well against their NTT-based counterparts [BMKV20]. Recent works showed that the schemes could benefit from transforming into a larger prime modulus for NTRU-Prime/Saber [CHK+21, ACC+21]. However, Toom-Cook is still an alternative to NTT in many post-quantum cryptographic algorithms, and its security against side-channel attacks has never been analyzed.

The family of Toom-Cook methods (Karatsuba can be shown to be similar to a Toom-Cook-2-way algorithm) is a well-known strategy for building algorithms to multiply dense univariate polynomials efficiently. It has been applied to optimize the implementations of RSA, ElGamal, Diffie-Hellman [SV93], high-speed ECC Processors [DLG19], improving the efficiency of McEliece cryptosystem [RU01], big number arithmetic [GT15]. We aim to discuss the security of Toom-Cook in Saber in the two aspects. Firstly, the specification and reference implementation of Saber still uses the Toom-Cook in round 3. The research on achieving more efficient polynomial multiplication has been continuous and uninterrupted (both on NTT and Toom-Cook). For example, Mera et al. proposed an optimized Toom-Cook implementation in Saber [BMKV20] (later surpassed by [CHK+21, ACC+21]). On the other hand, the NTT-based implementation was attacked in a known way, while the resistance of Toom-Cook is still unknown. We exhibit that this approach is also vulnerable to single-trace attacks.

The existing works mainly perform divide-and-conquer attacks on the Karatsuba and schoolbook. One original contribution is that we analyze the characteristics of Toom-Cook and MLWE and exhibit the challenge for divide-and-conquer attacks on the Toom-Cook in Saber. Another existing work on attacking NTT used the SASCA [PPM17], which was also utilized to attack Keccak [KPP20]. We show how to construct the factor graphs in the Toom-Cook, more importantly, optimize the factor graphs and algorithms to make the SASCA more practical. In addition, the deep learning-based power analysis into SASCA to improve the attack efficiency. To the best of our knowledge, we are the first to analyze the vulnerability of Toom-Cook against side-channel attacks.

The remainder of the paper is organized as follows. Section 2 describes the background of Toom-Cook, Karatusba and Saber. Section 3 introduces the vulnerabilities analysis of Toom-Cook multiplication in Saber. We then describe our attack on Toom-Cook in detail and provide various optimizations to decrease the complexity of the attack in Section 4. The evaluations and experiments are provided in Section 5. Furthermore, Section 6 provides the discussion. Section 7 concludes the paper.

2 Preliminaries

2.1 Saber key-encapsulation mechanism

Saber is a third round finalist post-quantum key-encapsulation mechanism candidate [VBDK+21]. The security is based on the hardness of Module Learning with Rounding problem (MLWR).

Let $\mathbb{Z}_q$ be the ring of integers modulo a positive integer q and quotient polynomial ring $R_q(x)$ is defined as $\mathbb{Z}_q(x)/(x^n+1)$. The symbol l determines the dimension of the underlying lattice problem. The positive integers q, p, and T are the moduli involved in the scheme and are chosen to be powers of 2, in particular $q = 2^{\epsilon_q}$, $p = 2^{\epsilon_p}$ and $T = 2^{\epsilon_T}$ with $\epsilon_q > \epsilon_p > \epsilon_T$. Setting parameter p and T to higher values results in lower security, but lower failure probability. Its implementation resists the timing side-channel attacks [GBHLY16, KRVV19]. The symbol $\gg$ represents the bitwise right shift operation. This type of operation can be extended to the coefficients in polynomials. Three constants (h, h_1, h_2) are used to replace rounding operations with a simple bit shift.

The decryption of PKE is shown in Algorithm 1 [DKSRV18]. Saber.KEM is transformed from Saber.PKE using the Fujisaki-Okamoto (FO) transform. It would be harmful to security when $p \nmid q$ introduces bias in the generated keys. To avoid this noise, Saber designers choose p and q as a power of two, i.e. 2^{10} and 2^{13} respectively.

Algorithm 1 *Saber.PKE.Dec*

```
Input: $c = (c_m, b')$, $s$
Output: $m'$
1: $v = b'^T(s \mod p) \in R_p$
2: $m' = ((v - 2^{\epsilon_p - \epsilon_T} c_m + h_2 \ mod\ p) \gg (\epsilon_p - 1) \in R_2$
3: return $m'$
```

2.2 Toom-Cook & Karatsuba multiplication

The most straightforward method to compute the multiplication result between two n-degree polynomials is the schoolbook multiplication [NG21]. The Karatsuba algorithm is a divide-and-conquer approach to implementing the polynomial multiplication. Supposing there are two n-degree polynomials $A(x)$ and $B(x)$. The polynomial $A(x)$ can be split into two $n/2$-degree polynomials $a_h(x)$ and $a_l(x)$, where $A(x) = a_h(x) \cdot x^{n/2} + a_l(x)$.

$$\begin{aligned} a_h(x) &= a_{n-1} \cdot x^{n/2-1} + \cdots + a_{n/2+1} \cdot x + a_{n/2} \\ a_l(x) &= a_{n/2} \cdot x^{n/2-1} + \cdots + a_1 \cdot x + a_0 \end{aligned} \tag{1}$$

The polynomial $B(x)$ is also split into two $n/2$-degree polynomials $b_h(x)$ and $b_l(x)$. Then the multiplication $C(x) = A(x) \cdot B(x)$ can be calculated by Karatsuba algorithm.

$$\begin{aligned}
C(x) &= A(x) \cdot B(x) \\
&= (a_h(x) \cdot x^{n/2} + a_l(x)) \cdot (b_h(x) \cdot x^{n/2} + b_l(x)) \\
&= a_h(x) \cdot b_h(x) \cdot x^n + (a_h(x) \cdot b_l(x) + a_l(x) \cdot b_h(x))x^{n/2} + a_l(x) \cdot b_l(x) \\
&= a_h(x) \cdot b_h(x) \cdot x^n + ((a_h(x) + a_l(x)) \cdot (b_h(x) + b_l(x)) \\
&- (a_h(x) \cdot b_h(x) + a_l(x) \cdot b_l(x))) \cdot x^{n/2} + a_l(x) \cdot b_l(x)
\end{aligned} \tag{2}$$

Toom-Cook-4-way analogously splits the operands into 4 coefficients. Assuming the two multiplicand polynomials are $A(x) = a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_0$ and $B(x) = b_{n-1} \cdot x^{n-1} + b_{n-2} \cdot x^{n-2} + \cdots + b_0$. Consider the parameter $n = 256$ and $k = 4$, they can be written as follows:

$$\begin{aligned}
A(x) &= A3 \cdot x^{64 \cdot 3} + A2 \cdot x^{64 \cdot 2} + A1 \cdot x^{64} + A0 \\
B(x) &= B3 \cdot x^{64 \cdot 3} + B2 \cdot x^{64 \cdot 2} + B1 \cdot x^{64} + B0
\end{aligned} \tag{3}$$

where $A3 = a_{255} \cdot x^{63} + \cdots + a_{192}$, $A2 = a_{191} \cdot x^{63} + \cdots + a_{128}$, $A1 = a_{127} \cdot x^{63} + \cdots + a_{64}$, $A0 = a_{63} \cdot x^{63} + \cdots + a_0$. The coefficients of $B(x)$ are defined similarly. For simplicity, we define $x^{64} = y$ and the above equations are written as

$$\begin{aligned}
A(y) &= A3 \cdot y^3 + A2 \cdot y^2 + A1 \cdot y + A0 \\
B(y) &= B3 \cdot y^3 + B2 \cdot y^2 + B1 \cdot y + B0
\end{aligned} \tag{4}$$

The polynomials are at $2k - 1 = 7$ points of y, i.e. $y = p_0, \cdots, y = p_6$. There are no fixed values for the points. The points are selected the small values to decrease the computation complexity. Generally, they can use $p_0 = 0, p_1 = 1/2, p_2 = -1/2, p_3 = 1, p_4 = -1, p_5 = 2, p_6 = \infty$. $A(p_0), \cdots, A(p_6), B(p_0), \cdots, B(p_6)$ are calculated. The second step is multiplication, i.e. $C(p_i) = A(p_i) \cdot B(p_i)$ for all $i \in [0, 6]$.

The product polynomial $C(y) = C_6 \cdot y^6 + C_5 \cdot y^5 + \cdots + C_0$. The last step is interpolation to calculate the coefficients from $C(p_i)$.

$$\begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_6 \end{bmatrix} = \begin{bmatrix} (p_0)^0 & (p_0)^1 & \cdots & (p_0)^6 \\ (p_1)^0 & (p_1)^1 & \cdots & (p_1)^6 \\ \vdots & \vdots & \ddots & \vdots \\ (p_6)^0 & (p_6)^1 & \cdots & (p_6)^6 \end{bmatrix}^{-1} \cdot \begin{bmatrix} C(p_0) \\ C(p_1) \\ \vdots \\ C(p_6) \end{bmatrix} \tag{5}$$

Finally, $C(x)$ can be reconstructed as $C(x) = C_6 \cdot x^{64 \cdot 6} + C_5 \cdot x^{64 \cdot 5} + \cdots + C_0$. In Saber, the Toom-Cook-based and Karatsuba-based multiplication are combined to implement $256 \cdot 256$ polynomial multiplication, which we will discuss in detail in Section 3.1.

2.3 Soft-Analytical Side-Channel Attacks

Soft-Analytical Side-Channel Attacks (SASCA) aims at exploiting more leakages than the single attack point used in classical divide-and-conquer attacks for AES [VCGS14]. It combines template matching with Belief Propagation on the factor graph of cryptographic implementation. It first performs a template attack on the target intermediate variables and retrieves the probabilities. For the intermediate variable x_i, one gets probabilities conditioned on the observable leakage ℓ, i.e., $Pr(x_i = x|\ell)$, where x runs through all of the possible values of x_i, ℓ is the observed side-channel leakage.

The adversary can construct a factor graph of the cryptographical algorithm and their implementation. The factor graph models the relationships among the intermediate variables. After adding the probabilities into the graph, the adversary performs the belief propagation algorithm to determine marginal probability distributions for the subkey. Next, we explain the factor graph and belief propagation more thoroughly.

Definition 1 (*Factor graphs* [KFL01])**.** *A factor graph is a bipartite graph representing the factorization of a function. Factor graph is comprised of variable node for each variable x_i, factor nodes for each function f_j and edge-connecting variable node x_i to factor node f_j if x_i is an argument of f_j.*

The set of factor nodes can be separated into two subsets. The first type of factor reflects the relationships among the variables in the implementation. For example a cryptographic operation $\mathrm{OP}(x_{i_1}, x_{i_2})$, the factor can be represented as

$$f_i(x_{i_1}, x_{i_2}, x_{i_3}) = \begin{cases} 1 & if\ \ \mathrm{OP}(x_{i_1}, x_{i_2}) = x_{i_3} \\ 0 & otherwise \end{cases} \tag{6}$$

The second subset describes the probabilities of the variables by observable side-channel leakages ℓ. These factors are non-deterministic and can be represented as $f_i(x_i) = Pr(x_i = x|\ell)$.

Based on these rules, the common arithmetic circuit of cryptographic implementation can be constructed by the adversary. The variables correspond to the variable nodes and the factors describe the relationship among the variables. In the original SASCA work, it showed how to construct a factor graph for AES (http://point-at-infinity.org/avraes) [VCGS14].

The belief propagation algorithm is a message-passing algorithm on graphical models such as factor graphs. It was initially proposed to compute the marginalization of a function efficiently.

Each variable node represents one variable x_n in the factor graph, and the factor node represents one factor f_m. We denote n, m as variable and factor indices, respectively. Let x_m be the variables that factor f_m depends on, $\mathrm{x}_{m\backslash n}$ denotes the set of variables in x_m without x_n, and v_n represents the value of the domain of x_n. We denote $\mathcal{M}(x_n)$ as the neighbors of the variable node x_n, and $\mathcal{N}(f_m)$ denotes the neighbor variables that the factor f_m depends on. These notations are the same with the indices n', m'.

The message passing includes from variables to factors ($u_{x_n \to f_m}$) and from factors to variables ($u_{f_m \to x_n}$):

Messages from variable to factor:

$$u_{x_n \to f_m}(v_n) = \prod_{m' \in \mathcal{M}(x_n) \backslash m} u_{f_{m'} \to x_n}(v_n) \tag{7}$$

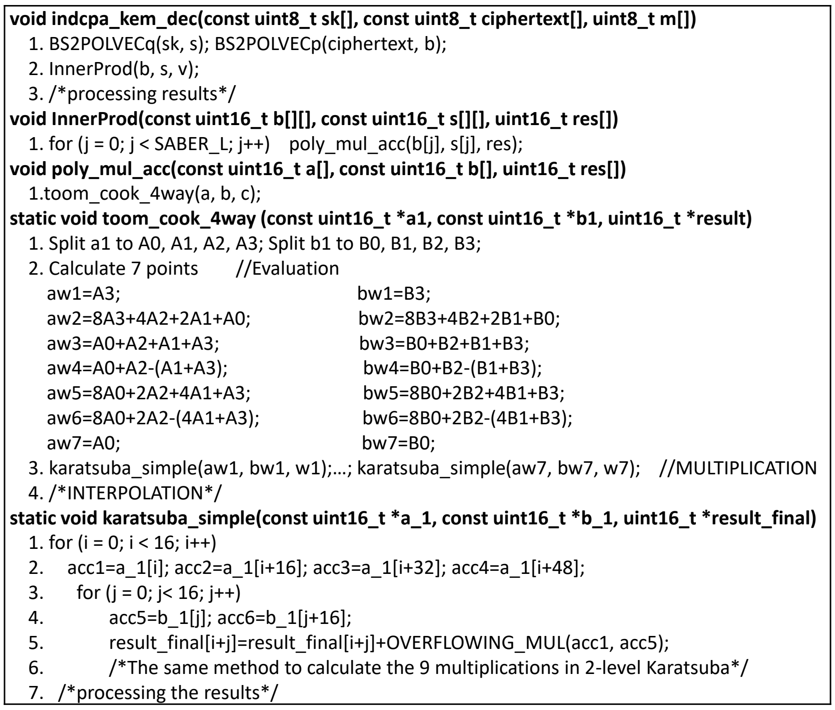
Messages from factor to variable:

$$u_{f_m \to x_n}(v_n) = \sum_{\mathrm{x}_{m\backslash n}} (f_m(\mathrm{x}_{m\backslash n}, v_n) \prod_{n' \in \mathcal{N}(f_m) \backslash n} u_{x_{n'} \to f_m}(v_{n'})) \tag{8}$$

In a factor graph of cryptographic implementation, there have two types of factors. The one corresponds to the variables of implementation while the another describes the priori knowledge of the variables acquired through template attacks on side-channel leakages. The BP applies the above message rules to all the nodes and factors. Finally, the probabilities of sensitive nodes (i.e. key) are obtained with the iteration of propagation.

3 Vulnerabilities Analysis of Toom-Cook Multiplication

We analyze the leakage characteristics of Toom-Cook adopted by Saber using divide-and-conquer attacks in this section. Firstly, we describe the implementation of Toom-Cook multiplication. Then we exploit the vulnerabilities in the implementation and illustrate the challenges for the traditional attacks.



```
void indcpa_kem_dec(const uint8_t sk[], const uint8_t ciphertext[], uint8_t m[])
   1. BS2POLVECq(sk, s); BS2POLVECp(ciphertext, b);
   2. InnerProd(b, s, v);
   3. /*processing results*/
void InnerProd(const uint16_t b[][], const uint16_t s[][], uint16_t res[])
   1. for (j = 0; j < SABER_L; j++)    poly_mul_acc(b[j], s[j], res);
void poly_mul_acc(const uint16_t a[], const uint16_t b[], uint16_t res[])
   1.toom_cook_4way(a, b, c);
static void toom_cook_4way (const uint16_t *a1, const uint16_t *b1, uint16_t *result)
   1. Split a1 to A0, A1, A2, A3; Split b1 to B0, B1, B2, B3;
   2. Calculate 7 points        //Evaluation
      aw1=A3;                              bw1=B3;
      aw2=8A3+4A2+2A1+A0;                  bw2=8B3+4B2+2B1+B0;
      aw3=A0+A2+A1+A3;                     bw3=B0+B2+B1+B3;
      aw4=A0+A2-(A1+A3);                   bw4=B0+B2-(B1+B3);
      aw5=8A0+2A2+4A1+A3;                  bw5=8B0+2B2+4B1+B3;
      aw6=8A0+2A2-(4A1+A3);                bw6=8B0+2B2-(4B1+B3);
      aw7=A0;                              bw7=B0;
   3. karatsuba_simple(aw1, bw1, w1);…; karatsuba_simple(aw7, bw7, w7);    //MULTIPLICATION
   4. /*INTERPOLATION*/
static void karatsuba_simple(const uint16_t *a_1, const uint16_t *b_1, uint16_t *result_final)
   1. for (i = 0; i < 16; i++)
   2.     acc1=a_1[i]; acc2=a_1[i+16]; acc3=a_1[i+32]; acc4=a_1[i+48];
   3.       for (j = 0; j< 16; j++)
   4.           acc5=b_1[j]; acc6=b_1[j+16];
   5.           result_final[i+j]=result_final[i+j]+OVERFLOWING_MUL(acc1, acc5);
   6.           /*The same method to calculate the 9 multiplications in 2-level Karatsuba*/
   7.   /*processing the results*/
```

Figure 1: Pseudo code of reference implementation of Saber.PKE.Dec().

3.1 Polynomial Multiplication in Saber

During the multiplication, the polynomials are of degree 256. To perform a $256 \cdot 256$ polynomial multiplication $C(x) = A(x) \cdot B(x)$, it adopts the Toom-Cook-4-way split the $A(x)$ and $B(x)$. The step transforms the multiplication of two 256-degree polynomials to 7 multiplications of 64-degree polynomials. Then split the polynomials into 9 multiplications of degree 16 through 2-levels of Karatsuba multiplication. Finally, the polynomials perform the schoolbook multiplication.

Currently, the polynomials multiplication of Saber reference C implementation is implemented based on the Toom-Cook. The full implementation is a very complex process. We simplify it for clarity, as shown in Figure 1. The IND-CPA decryption receives a *ciphertext* and the secret key *sk*. The polynomial multiplication is performed by b and s, which are transformed from ciphertext and the secret key respectively by data conversion algorithms *BS2POLVECp/BS2POLVECq*. The data conversion algorithms map a byte string to a vector in the ring.

After the call of *InnerProd* and *poly_mul_acc* functions, the *toom_cook_4way* is conducted to implement multiplication, which is our mainly target in analysis. The *toom_cook_4way* function takes as input $a1, b1$ (256 coefficients) and outputs the multiplication result. The 256-degree polynomial can be split into 4 polynomials $B3, B2, B1, B0$ of degree 64. During the evaluation, $bw1, \cdots, bw7$ can be obtained by substituting the points value $y = p_0, \cdots, y = p_6$ into the Equation (4), shown as Equation (9).

$$\begin{aligned}
bw1 &= B3 \\
bw2 &= 8 \cdot B3 + 4 \cdot B2 + 2 \cdot B1 + B0 \\
bw3 &= B0 + B2 + B1 + B3 \\
bw4 &= B0 + B2 - (B1 + B3) \\
bw5 &= 8 \cdot B0 + 2 \cdot B2 + 4 \cdot B1 + B3 \\
bw6 &= 8 \cdot B0 + 2 \cdot B2 - (4 \cdot B1 + B3) \\
bw7 &= B0
\end{aligned} \tag{9}$$

Similarly to $a1$, $aw1, \cdots, aw7$ can be obtained in the same way. The complexity of multiplication between polynomials of degree 64 is still unacceptable to implementation. Therefore, it further split 64-degree polynomial to 4 polynomials of degree 16, for example splitting $bw1$ into $bw1\_3, bw1\_2, bw1\_1, bw1\_0$, where $bw1 = bw1\_3 \cdot x^{48} + bw1\_2 \cdot x^{32} + bw1\_1 \cdot x^{16} + bw1\_0$. The 9 polynomials of degree 16 can be obtained using the 2-level Karatsuba algorithm according to Section 2.2.

$$\begin{aligned}
bw1_1 &= bw1\_3 \\
bw1_2 &= bw1\_2 \\
bw1_3 &= bw1\_3 + bw1\_2 \\
bw1_4 &= bw1\_1 \\
bw1_5 &= bw1\_0 \\
bw1_6 &= bw1\_1 + bw1\_0 \\
bw1_7 &= bw1\_3 + bw1\_1 \\
bw1_8 &= bw1\_2 + bw1\_0 \\
bw1_9 &= bw1\_3 + bw1\_2 + bw1\_1 + bw1\_0
\end{aligned} \tag{10}$$

3.2 Vulnerability Analysis

Side-channel attacks threaten cryptographic algorithms through the physical information such as timing, power, and electromagnetic emissions. In the view of timing leakage, a powerful source code analyzer [FGL+18] can be exploited to perform the static analysis to track the sensitive information in the code. The goal of the tool is to detect the cache-timing leakages on the code level. The candidates submitted to the first round of the PQC project have been analyzed by this tool, which showed Saber was one of the submissions to be correctly protected against timing attacks [FGL+18]. In Saber, all moduli are powers of 2 and allow the explicit modular reduction to be removed. The lack of modular reduction also implies that Saber is naturally constant time, making it resistant to cache timing attacks.

There are many leakage assessment methodologies to evaluate the detectable leakage in the power/electromagnetic analysis. The general method of Welch's t-test evaluates the leakage independent of any power model and sensitive value [GGJR+11], which was improved by Becker et al. and renamed Test Vector Leakage Assessment (TVLA) [BCDM+13]. It has been applied to detecting the leakage of lattice-based algorithms such as Kyber, Saber, NewHope and LAC. The polynomials transformed from the secret key are marked as the sensitive variables, as dotted arrows in Figure 2. The sensitive variables obtained from the input secret key will eventually be multiplied with the ciphertext coefficients, which leads to side-channel leakage inevitably. Next, we analyze the challenges and limitations of Toom-Cook in Saber to classical divide-and-conquer attacks in the view of the attack.

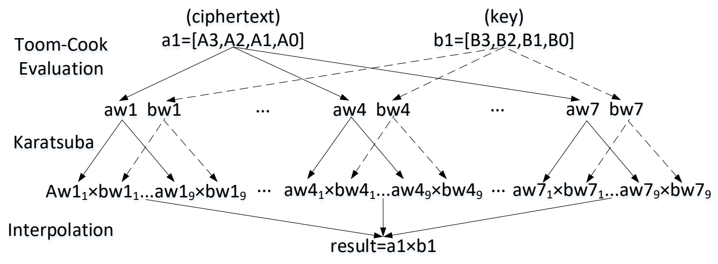


Figure 2: The dataflow of Toom-Cook multiplication in Saber.

Incomplete key recovery. The schoolbook multiplication is vulnerable to SCA [ATT+18, SMS19] since its intermediate values depend on the known ciphertext and unknown secret key. Huang et al. demonstrated the private-key recovery from the Karatsuba multiplication in NTRU Prime [HCY19]. Karatsuba's method itself can not resist against the vertical power analysis attacks on lowest-level multiplications. However, the application of Toom-Cook makes these attacks fail to recover full private-key, as considered in [HCY19] *"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 in pursuit of full private-key recovery is worth further investigation".* According to Equation (9), the lowest-level multiplication of $bw1 \times aw1$ and $bw7 \times aw7$ can be used to recover $B3$ and $B0$ while the other coefficients are hardly recovered straightforwardly.

Indistinguishable guessing keys. A critical factor of successful power/electromagnetic analysis is distinguishing the statistical analysis results of the correct key from the other wrong keys. In general, divide-and-conquer attacks can use the correlation, differential, or mutual information between the power/electromagnetic samples and the Hamming weight of the intermediate value in the cryptographic algorithm. Xu et al. focused on the output of multiplication in Kyber and mounted a chosen-ciphertext SPA attack on polynomial multiplication using few traces [XPSR+21]. Kyber executes the Montgomery reduction ($mod \pm q$) by $fqmul()$ after the multiplication. The coefficients of the secret-key range from -2 to 2 and are obtained from the binomial distribution. The adversary can distinguish the coefficient values$(-2, -1, 0, 1, 2)$ by choosing different ciphertexts, while the Hamming weights of the output of $fqmul()$ can be divided into different classes. However, in Saber, the integer moduli are powers of 2 so that there is no need for explicit modular reduction. The last-level multiplication is implemented by the $OVERFLOWING\_MUL$ function.

$$\begin{aligned} &OVERFLOWING\_MUL(s_{coeff}, b_{coeff}) \\ &= ((uint16\_t)((uint32\_t)(s_{coeff}) * (uint32\_t)(b_{coeff}))) \end{aligned} \tag{11}$$

where s_{coeff} denotes the coefficient of the secret key and b_{coeff} denotes the coefficient of the ciphertext b. In Saber, the possible values of $s_{coeff} \in \{0, 1, 2, 3, 8188, 8189, 8190, 8191\}$ and the coefficient of ciphertext satisfies $0 \leq b_{coeff} \leq 2^{10}$. Under Hamming weight leakage model, the power sample is approximately relating to the HW (Hamming weight) of the output of $OVERFLOWING\_MUL$. We calculate the Pearson's correlation coefficient among the HW vectors corresponding to different guessing keys for $b_{coeff} \in [0, 2^{10}]$, as shown in Table 1. It is obvious that all elements of multiplication results are zero for $s_{coeff} = 0$, which is not available to the correlation calculation.

It can be seen that there are many similar correlation coefficients, even completely equal values. For example, the output of $OVERFLOWING\_MUL$ for an arbitrary b_{coeff} and $s_{coeff} = 2$ can be viewed as a one-bit left shift of b_{coeff}. Without modular reduction, the

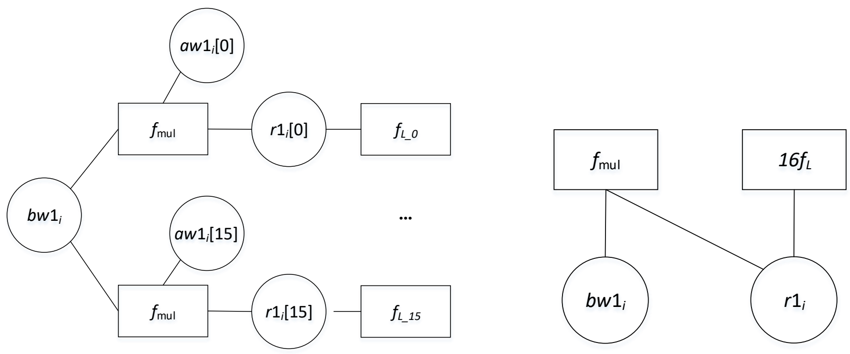
Table 1: The Pearson's correlation coefficient among different guessing keys

s_{coeff}		guessing key						
		1	2	3	8188	8189	8190	8191
correct key	1	1	1	0.48	0.75	0.14	0.75	0.74
	2	1	1	0.48	0.75	0.14	0.75	0.74
	3	0.48	0.48	1	0.33	0.79	0.33	0.33
	8188	0.75	0.75	0.33	1	0.42	0.99	0.99
	8189	0.14	0.14	0.79	0.42	1	0.42	0.43
	8190	0.75	0.75	0.33	0.99	0.42	1	0.99
	8191	0.74	0.74	0.33	0.99	0.43	0.99	1

HW of the output under $s_{coeff} = 2$ is equal to the HW under $s_{coeff} = 1$. The differences between other guessing keys are also not obvious, making it difficult for the attack to distinguish the correct key.

4 Single-trace attack on Toom-Cook

In the single-trace attack, it is crucial to extract as much information per trace as possible. SASCA constructs a graphical model for a cryptographic algorithm. The factor graph contains the information of intermediate variables in the attacked cryptographic algorithm and its specific implementation. The posterior distributions provided by performing Template Attack (TA) [CRR03] on all the intermediate variables are added as function nodes to the factor graph. It propagates the information by the Belief Propagation (BP) algorithm [MMK+04] to determine marginal probability distributions for subkeys.



(a) Original SFG (b) Bayes-based SFG

Figure 3: Schoolbook multiplication with factor graph representation (SFG)

4.1 SASCA on the Toom-Cook

In this section, we describe how to construct a factor-graph targeting such generic Toom-Cook software implementations.

In a factor graph, each variable node (represented by a circle) is connected to a factor node (represented by a rectangle) by an edge if the factor depends on it. As seen in Section 3, the variables are interconnected after Toom-Cook&Karatsuba transformation. The secret key s is split into $bw1, bw2, \ldots, bw7$ of degree 64, multiplying $aw1, aw2, \ldots, aw7$

obtained by the ciphertext. It further splits the polynomial into four polynomials of degree 16 and transforms to decrease the complexity. For example, $aw1_1, \dots, aw1_9$ and $bw1_1, \dots, bw1_9$ deduced from $aw1$ and $bw1$ perform the $16 \cdot 16$ schoolbook polynomial multiplications, respectively.

The factor graph corresponding to the example of schoolbook polynomial multiplication of $bw1_i$ and $aw1_i$ is illustrated in Figure 3 (a), named as SFG for simplicity.

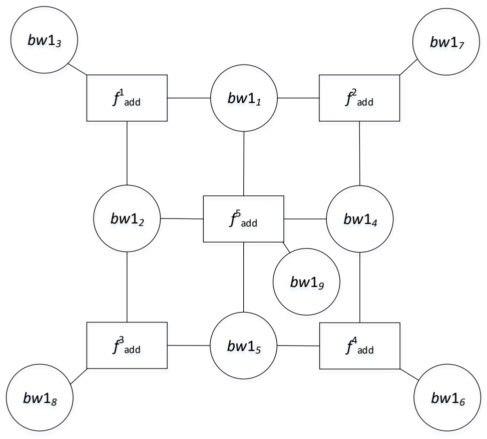


Figure 4: Factor graph corresponding to Karatsuba (KFG).

The factor graph represents variables and factor nodes by circles and squares, respectively. The factor nodes are further split into two groups. The first group of factors is characterized by side-channel information. Its purpose is to add the side-channel information, i.e., the results of the template matching. During the schoolbook multiplication of $bw1_i$ and $aw1_i$, the value of $aw1_i$ is known according to a known ciphertext. The attack can observe the side-channel information during the execution of the multiplication operation to obtain the posterior distribution of the multiplication result $r1_i$. Thus the first type of factor corresponds to the a priori knowledge of the multiplication results obtained by side-channel leakages, for example, $f_{L\_0} = Pr(r1_i[0]|L\_0)$, where L_0 represents the observed leakage of $r1_i[0]$. It is the same with other operands, so $f_{L\_0}, f_{L\_1}, \dots, f_{L\_15}$ are connected to the multiplication results $r1_i[0], r1_i[1], \dots, r1_i[15]$, respectively.

The second group of factors is modeling the relationships between the variables nodes in the schoolbook. We add the multiplication as factor nodes f_{mul}, which can be defined as:

$$f_{mul}(x, y, z) = \begin{cases} 1 & if\ z = OVERFLOWING\_MUL(x, y) \\ 0 & otherwise \end{cases} \tag{12}$$

where $OVERFLOWING$ is defined in Equation (11).

In the Karatsuba multiplication, $bw1_1, \dots, bw1_9$ are obtained from $bw1$ based on Equation (10). Figure 4 depicts the Karatsuba factor graph (KFG) of this operation. The nodes $bw1_1, \dots, bw1_9$ are connected to the corresponding SFG, i.e. $SFG_{bw1_1}, \cdots, SFG_{bw1_9}$, that is, a total of 9 SFG. Based on Equation (10), $bw1_1, \dots, bw1_9$ and $aw1_1, \dots, aw1_9$ are generated to perform the 2-level Karatsuba algorithm. Since $bw1_1, \dots, bw1_9$ are related to the secret key while $aw1_1, \dots, aw1_9$ are all known under a known ciphertext, we only construct the factor graph for the relationships among $bw1_1, \dots, bw1_9$. For example, $bw1_1 = bw1\_3, bw1_2 = bw1\_2, bw1_3 = bw1\_3 + bw1\_2 \, mod \, q$, there has $bw1_3 = bw1_1 + bw1_2 \, mod \, q$. The factor f_{add} describes how variables nodes inside a 2-levels

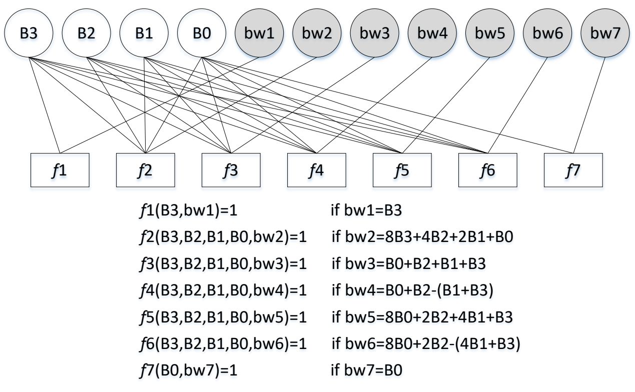

B3
B2
B1
B0
bw1
bw2
bw3
bw4
bw5
bw6
bw7
f1
f2
f3
f4
f5
f6
f7
f1(B3,bw1)=1 if bw1=B3
f2(B3,B2,B1,B0,bw2)=1 if bw2=8B3+4B2+2B1+B0
f3(B3,B2,B1,B0,bw3)=1 if bw3=B0+B2+B1+B3
f4(B3,B2,B1,B0,bw4)=1 if bw4=B0+B2-(B1+B3)
f5(B3,B2,B1,B0,bw5)=1 if bw5=8B0+2B2+4B1+B3
f6(B3,B2,B1,B0,bw6)=1 if bw6=8B0+2B2-(4B1+B3)
f7(B0,bw7)=1 if bw7=B0

Figure 5: Factor graph corresponding to Toom-Cook evaluation (TFG).

of the Karatsuba algorithm are related:

$$f_{add}(bw1_1, bw1_2, bw1_3) = \begin{cases} 1 & if\ bw1_3 = bw1_1 + bw1_2\, mod\, q \\ 0 & otherwise \end{cases} \tag{13}$$

It is the same with other variables. The difference is the number of input nodes for $bw1_9 = bw1\_3 + bw1\_2 + bw1\_1 + bw1\_0$, corresponding to the factor $f_{add}(bw1_1, bw1_2, bw1_4, bw1_5, bw1_9)$. It can be found that there are many shortest cycles in the KFG, which are not beneficial to the efficiency of belief propagation. The next section will show how to avoid it and improve belief propagation.

The last step is to formalize the implementation of the Toom-Cook evaluation. During the Toom-Cook-4-way evaluation, $bw1, \cdots, bw7$ can be obtained by substituting the seven points values, as Equation (9). The nodes $bw1, \cdots, bw7$ are connected to its corresponding KFG, i.e. $KFG_{bw1}, \cdots, KFG_{bw7}$. We defined the factors $f_1, \cdots, f_7$ to describe the relationships among variables during the evaluation of different points, shown in Figure 5.

The various factor graphs above construct the complete factor graph for implementing Toom-Cook. The relationships among the different factor graphs are described in Figure 6.

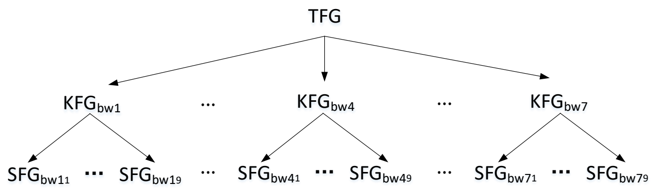


Figure 6: The construction of the full algorithm.

There are $16 \cdot 2 + 1 = 33$ variable nodes in one SFG and 9 in one KFG. The overall graphs include $33 \cdot 9 \cdot 7 + 4 = 2083$ variable nodes. The $16 \cdot 9 \cdot 7 = 144$ leakage factors f_L modeled the observed side-channel leakage of multiplication in a trace. It requires close to ten million templates ($2^{16} \cdot 144$) overall. Moreover, the BP algorithm is used to determine the marginal probabilities of the secret key $B3, B2, B1, B0$ by iteratively executing the message propagation computation. It can be found that many short loops degrade the BP performance in the factor graph.

4.2 Improving Practical Single-Trace Attacks

This section shows how to address these problems and perform single-trace attack on the Toom-Cook more practical. Firstly, we decrease the templates number by deep neural networks. Secondly, we prove the factor graph can be optimized by merging tracks based on Bayes' algorithm. Then, the update rules are improved for the short loops during belief propagation.

4.2.1 Decreasing the Number of Templates

One straightforward way to decrease required templates is to switch to Hamming weight templates. For the large bit width of the intermediate value in the algorithm, we profile templates for Hamming weights instead of every possible value. It performs a template matching and, therefore, gives a vector of probabilities conditioned on the leakage l. We take node $r1_i[0]$ in SFG as an example. It has

$$f_{L\_0} = Pr(r1_i[0] = v|l) \tag{14}$$

where v represents the value of $r1_i[0]$ and l represents the observable leakage. If it profiles the templates of Hamming weight of the intermediate value, the probabilities of Hamming weight are assigned to the factor nodes. Let $HW()$ denote the Hamming weight function. There has

$$f_{L\_0} = Pr(HW(r1_i[0]) = HW(v)|l) \tag{15}$$

It can be found that the class number of each node is decreased by using Hamming weight templates. However, it still requires $7 * 144 * 17 = 17,136$ templates. The existing work of power analysis showed that neural network technology could build an efficient model that is at least as effective as a standard template model without any preprocessing operations.

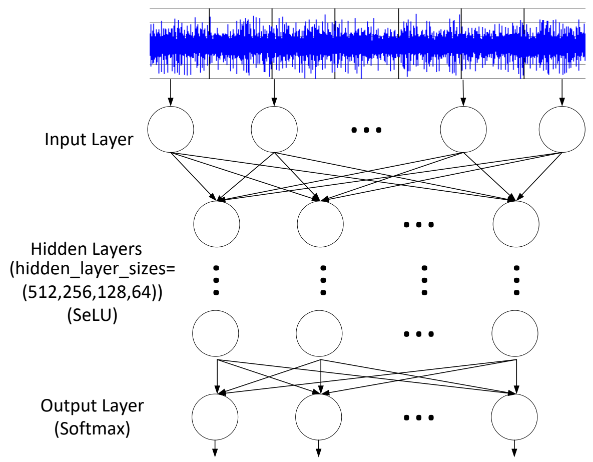


Figure 7: The MLP Architecture in our attack.

Many works have validated the efficiency of different NNs on side-channel analysis, such as MLP [DGD+19, RAD20], VGG16 [BPS+20] and ResNets [ZS20]. Zaid et al. also proposed an efficient CNN architecture and showed that the networks do not need to be very complex to perform well in the side-channel context [ZBHV19]. We aim to utilize a suitable model with a good balance between training time and effect. Our MLP architecture is based on the model proposed in [RAD20], and hyperparameters are trained to fit the requirements for the target implementation.

In the profiling phase, we train the network using the Adam optimizer [KB14] and a learning rate of 10^{-2}. The inputs of our network are the power traces, and it outputs the distribution over the class labels. The training label is set as the Hamming weight of the intermediate values in Toom-Cook multiplication. The activation function for hidden layers is SeLU, and that for output layers is Softmax. Softmax is a nonlinear function, mainly used for the classifier output of multi-class classification, which produces a distribution over the class labels. In some scenes, the output is used as the probability since it reflects the difference among various classes [KSH12, SWT14, TYRW14]. In the deep learning-based side-channel context, it also uses the output of NN into the multi-trace Bayes' probability to distinguish the secret [ZBHV19]. The outputs are assigned to the factor nodes f_L instead of the probabilities in Equation (15). The MLP model is shown in Figure 7. The input layer contained inputs corresponding with the sample points of each schoolbook and output layer had the same number neurons as the Hamming weights species to predict target value $y^{'}$.

$$\begin{aligned} y^{'} &= f_s(\sum_{l=1}^{N_1} w_{jl}^{'} x_j^{'} - \sigma_l^{'}) \\ x_j^{'} &= f_s(\sum_{i=1}^{N_2} w_{ij} x_i - \sigma_j) \end{aligned} \tag{16}$$

where f_s denotes the nonlinear activation function, j denotes hidden layer j-th neuron, l denotes output layer l-th neuron, N_1 denotes the number of neurons in output layer, N_2 is the number of neurons in a hidden layer, σ denotes the threshold of the neuron, w_{ij} denotes weights between i-th and j-th neuron and $x_i^{'}$ denotes the neurons output. The result of this classification is a probability vector of the Hamming weight prediction. The samples of total schoolbook in one trace are send to the input of the network. Each trace segment in the training set is assigned its label as the Hamming weight of multiplication result to train the network. The normalization of the trace is applied to the input of the network. Besides, we select the same number of training traces for each class of Hamming weight to avoid overfitting training.

4.2.2 Factor Graph Optimization

The memory cost of a factor graph is influenced by the number of nodes and edges. Generally, the larger graph requires more memory. The factor graph of schoolbook multiplication contains 16 same factors f_{mul}, as Figure 3 (a). Based on the Sum-Product algorithm in the message-update rules, the marginal probability of variable $bw1_i$ is

$$\begin{aligned} p(bw1_i) &= v_{f_{mul}\to bw1_i}(bw1_i)\ldots v_{f_{mul}\to bw1_i}(bw1_i) \\ &= \sum_{aw1_i[0], r1_i[0]} (f_{mul}(aw1_i[0], r1_i[0], bw1_i) \cdot u_{aw1_i[0]\to f_{mul}}(aw1_i[0]) \\ &\cdot u_{r1_i[0]\to f_{mul}}(r1_i[0])) \\ &\ldots \sum_{aw1_i[15], r1_i[15]} (f_{mul}(aw1_i[15], r1_i[15], bw1_i) \cdot u_{aw1_i[15]\to f_{mul}}(aw1_i[15]) \\ &\cdot u_{r1_i[15]\to f_{mul}}(r1_i[15])) \end{aligned} \tag{17}$$

whereas the variables $aw1_i[0], \ldots, aw1_i[15]$ denote the ciphertexts which are known to adversary. There has $u_{aw1_i[0]\to f_{mul}}(aw1_i[0]) = 1$. It performs a template matching the multiplication result and therefore gets a vector of conditional probabilities on the leakage

$l_0, \dots, l_{15}$, i.e. $u_{r1_i[0] \to f_{mul}}(r1_i[0]) = f_{L\_0} = p(r1_i[0]|t_0)$. Thus,

$$\begin{aligned} p(bw1_i) &= \sum_{aw1_i[0], r1_i[0]} (f_{mul}(aw1_i[0], r1_i[0], bw1_i) \cdot f_{L\_0}) \\ &\dots \sum_{aw1_i[15], r1_i[15]} (f_{mul}(aw1_i[15], r1_i[15], bw1_i) \cdot f_{L\_15}) \\ &= \sum_{aw1_i[0], r1_i[0]} (f_{mul}(aw1_i[0], r1_i[0], bw1_i) \cdot p(r1_i[0]|t_0)) \\ &\dots \sum_{aw1_i[15], r1_i[15]} (f_{mul}(aw1_i[15], r1_i[15], bw1_i) \cdot p(r1_i[15]|t_{15})) \\ &= p(bw1_i|t_0) \cdot p(bw1_i|t_1) \dots p(bw1_i|t_{15}) \end{aligned} \tag{18}$$

In template attacks, a single trace is usually not enough to recover the key with high confidence in practice. The posterior probabilities of the key candidates are calculated from multiple traces based on Bayes' theorem [OM06].

$$p(bw1_i|t_0, \dots t_{15}) = \frac{(\prod_j p(t_j|bw1_i))p(bw1_i)}{\sum_l ((\prod_j p(t_j|bw1_i^l))p(bw1_i^l))} \tag{19}$$

The sum of probabilities of the candidates is equal to 1, i.e. $\sum_l (p(bw1_i^l))$. Based on Bayes' rule, the marginal probability can be represented as

$$p(bw1_i) = p(bw1_i|t_0) \cdot p(bw1_i|t_1) \dots p(bw1_i|t_{15}) = \frac{(\prod_j p(t_j|bw1_i)p(bw1_i))}{\prod_j ((\sum_l p(t_j|bw1_i^l))p(bw1_i^l))} \tag{20}$$

Since the initial probability of $bw1_i$ is an uniformed value, the denominator in the above formula does not influence the ranking among the key candidates. The Bayes' probability can be converted into the marginal probability by normalization.

$$\begin{gathered} p(bw1_i) = p(bw1_i|t_0) \cdot p(bw1_i|t_1) \dots p(bw1_i|t_{15}) = p(bw1_i|t_0, \dots t_{15}) \cdot \mathcal{C} \\ \mathcal{C} = \frac{\sum_l ((\prod_j p(t_j|bw1_i^l))p(bw1_i^l)) \prod_j p(bw1_i)}{\prod_j ((\sum_l p(t_j|bw1_i^l))p(bw1_i^l))} \end{gathered} \tag{21}$$

Figure 3 (b) shows the simplified graph according to the merging trick. In contrast with the factor graph representation in Figure 3 (a), which includes 33 variable nodes, the new one consists of only 2 variable nodes in one SFG. Moreover, the number of variable nodes in the overall graphs drops from 2083 to 130, reducing time and memory complexities for the following belief propagation.

4.2.3 Improving Belief Propagation

In this section, we analyze the factor graph on the performance of belief propagation. Based on this, we improve the belief propagation steps for the special case during the attack.

The way of SASCA to describe the implementation and leakage is similar to decoding factor graphs using a BP algorithm in the Low-Density Parity Check Code (LDPC) [GGSB20]. LDPC error correction decoders are widely used in communication systems for their strong performance. The factor graph can be transformed into the Tanner graph.

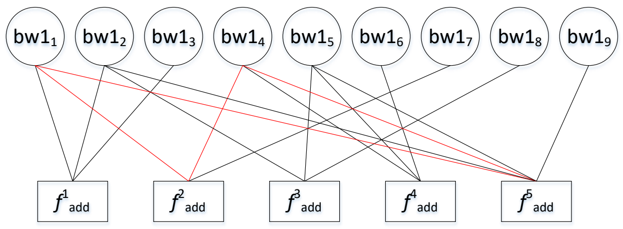


Figure 8: Tanner graph associated with KFG.

It is viewed as a graphical representation for LDPC codes. The nodes correspond to the variable nodes and factor nodes. The KFG in Figure 4 can be represented as a Tanner graph as Figure 8.

In LDPC, short cycles especially, cycles of length 4, influence the performance using the BP algorithm [CH06]. In this Tanner graph, there are also existing many circles of length 4, for example, the red lines in Figure 8. To deal with these short cycles, we first detect them with the parity-check matrix. The parity checks are used to check the errors in the received codeword, called the parity-check matrix. The parity-check matrix H of the above graph equals:

$$H = \begin{bmatrix} \mathbf{1} & \mathbf{1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 1 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

Firstly, it can straightly identify the cycles of length 4 in the matrix. It indicates that the cycle of length 4 occurs when two rows have ones in two same column locations in the parity-check matrix. It can be found that four cycles of length 4 are all connected to factor f^5_{add}, as shown in the bold numbers in the parity-check matrix H.

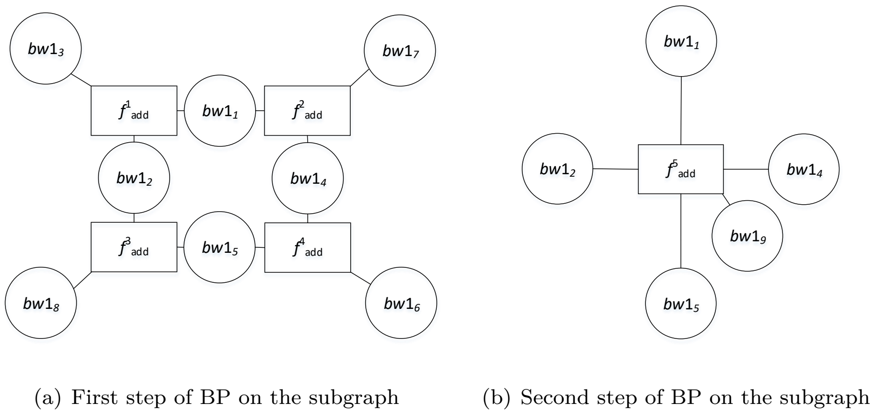


Figure 9: The BP strategy on the KFG with cycles of length 4.

To avoid those shortest cycles of length 4, we split the BP in KFG into two steps. It can be found that the removal of factor f^5_{add} in the parity-check matrix H can avoid all cycles of length 4. Thus we first perform the standard BP algorithm on the subgraph, as shown in Figure 9.(a). Note that there are also exist cycles in the subgraph.

However, the influence of a large cycle is slight on the BP. Then, the marginal probabilities $p(bw1_1), p(bw1_2), p(bw1_4), p(bw1_5)$ are input to the initial distribution in the second subgraph, as shown in Figure 9.(b). In this step, it computes the joint distribution by

$$p(bw1_9) = \sum_{bw1_1, bw1_2, bw1_4, bw1_5} f^5_{add}(bw1_1, bw1_2, bw1_4, bw1_5, by1_9) \tag{23}$$

Finally, the marginal probabilities of $bw1_1, bw1_2, bw1_4, bw1_5, bw1_9$ are updated. Compared to the original BP on KFG, this method splits a whole loopy BP into two steps and executes in sequential, which mitigates the performance degradation.

5 Attacks on the Saber case

In this section, we perform the SASCA on the Saber case. In particular, it uses performance and success rate to show the efficiency of optimization methods including factor graph optimization, improving belief propagation and decreasing the number of templates.

5.1 Evaluation

We evaluate our attack method using leakage simulations in various settings. The target is the C reference implementation of Saber as submitted to the NIST contest[1]. The experiments are run on an Intel i7-10510U (2.3GHz). The simulation traces are generated by the Hamming weight (HW) with an additive Gaussian noise model. The simulated leakage is $L = HW(v) + \mathcal{N}(0, \sigma)$, where $\mathcal{N}(0, \sigma)$ represents a Gaussian distribution with zero mean and standard deviation σ. In this section, we analyze the improvement of our methods under different noise levels ($\sigma = 2, 5, 10$).

For each simulation, we generate 30 samples for each intermediate variable, and there have 144 intermediate variables of multiplication. The 30 samples are generated to simulate the continuous leakage in the practical scene over a period. The same samples do not influence the comparative evaluation of different methods. Thus, it has 4320 samples in one trace. We use 50,000 traces for training and perform the template matching to obtain the corresponding conditional probabilities. For each noise level, we perform 100 experiments and compute the averaged results.

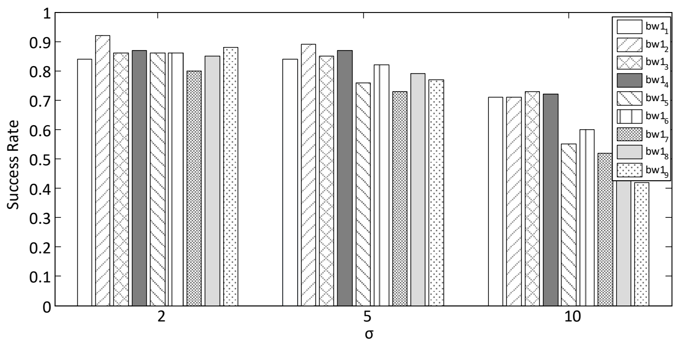


Figure 10: Success rates of attacking $bw1_1, \ldots, bw1_9$ in Bayes-based SFG under $\sigma = 2, 5, 10$. The success rates with Bayes-based SFG are the same with the original SFG.

We first evaluate the effects of the factor graph optimization. The coefficient $b1$ of the secret key is split into $bw1, \ldots, bw7$ during Toom-Cook evaluation. Each element splits into

[1] https://www.esat.kuleuven.be/cosic/pqcrypto/saber/software.html

9 coefficients during Karatsuba evaluation. For example, $bw1$ is split into $bw1_1, \dots, bw1_9$. Thus these coefficients are our target during the attack. Based on Figure 3, we perform BP on the original SFG and obtain the marginal probabilities of $bw1_1, \dots, bw1_9$. Then we perform BP on the Bayes-based SFG. The 100 independent traces are performed to compute the success rate. The success rates under original SFG and Bayes-based SFG are the same, shown in Figure 10. The success rates tend to decrease with the increasing noise level.

We also evaluate the whole factor graph under the two types of SFG. The execution time during the BP on the factor graph is recorded to compare the efficiency of the two SFGs. We average the results from 100 experiments to decrease the influence of noise. Table 2 shows the performance metrics of Bayes-based SFG and original SFG under a certain noise level ($\sigma = 2$). The execution time difference among different subgraphs is due to the various scale of the distribution corresponding to the space of possible values. Based on Equation (9), it can calculate the space of each variable, denoted as $N_{bw1} = 8, N_{bw2} = 624, N_{bw3} = 65, N_{bw4} = 65, N_{bw5} = 624, N_{bw6} = 624, N_{bw7} = 8$. The higher complexity of distribution costs more time during BP. From Table 2, Bayes-based SFG requires less time than the original SFG while maintaining the same success rate as expected. The time needed to the same success rate reduces to less than 1/2 for original attacks.

Table 2: The efficient of Bayes-based SFG compared to original SFG. (Time measured in seconds.)

metric	method	$bw1$	$bw2$	$bw3$	$bw4$	$bw5$	$bw6$	$bw7$	sum
success rate	Original SFG	0.86	0.88	0.83	0.88	0.87	0.87	0.86	0.86
	Bayes-based SFG	0.86	0.88	0.83	0.88	0.87	0.87	0.86	0.86
time	Original SFG	1.88	4.12	1.86	2.30	3.71	3.79	2.43	20.08
	Bayes-based SFG	0.10	2.68	0.47	0.49	2.66	2.81	0.09	9.30

In order to evaluate the efficiency of improved BP, we perform the original BP and the improved BP on the same KFG. Note that there exist cycles in the KFG, which degrade the performance of the BP algorithm. The attack targets are the coefficients $bw1\_3, bw1\_2, bw1\_1, bw1\_0$ corresponding to the nodes $bw1_1, bw1_2, bw1_4, bw1_5$ in the KFG, as Equation (10). We first perform the original BP to calculate the marginal probabilities. The number of iteration is set to 5 since the depth of the graph is not large. We attack 100 traces with the independent key to compute the success rate. The same case is also set to improved BP algorithm. A summary of the results obtained after the two BP algorithms is given in Table 3. As predicted above, the cycles in the factor graph significantly impact attacking performance. The attack's success rate with improved BP outperforms the original BP while taking less time.

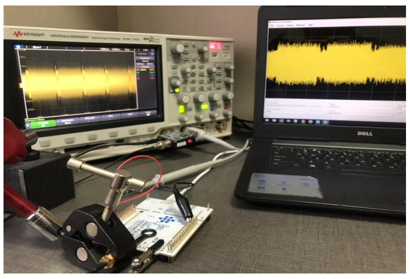
Table 3: The success rate of our attacks on the KFG with improved BP algorithm.

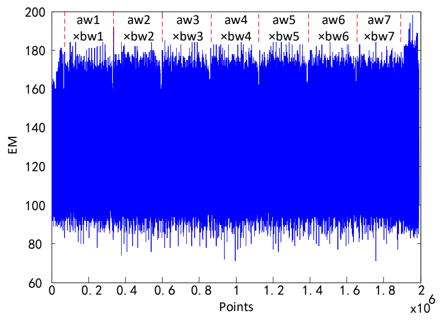
metric	success rate					
noise	2		5		10	
method	Original	Improved BP	Original	Improved BP	Original	Improved BP
$bw1\_3$	0.84	0.94	0.81	0.95	0.71	0.81
$bw1\_2$	0.92	0.94	0.80	0.94	0.71	0.80
$bw1\_1$	0.86	0.97	0.68	0.97	0.73	0.87
$bw1\_0$	0.87	0.94	0.67	0.95	0.72	0.78
metric	time in seconds					
noise	2		5		10	
method	Original	Improved BP	Original	Improved BP	Original	Improved BP
time	0.12	0.07	0.18	0.07	0.13	0.06

5.2 Analyzing a real device

The practical attack is performed with the electromagnetic radiation measurement of the STM32F103RB (ARM Cortex-M3) micro-controller on the STM32 Nucleo-64 board. It was also used by many other implementations of LWE encryption. The target consists of the C reference implementation submitted to the NIST contest. The STM32CubeIDE and the ST programmer from STMicroelectronics are used to compile and program the device.

A Langer RF2 near-field probe is placed in proximity to the chip. The electromagnetic signals are sampled through a digital oscilloscope DSOX3024T from KEYSIGHT. Besides, a PA303 Amplifier is used to pre-treatment signals. The setup is depicted in Figure 11(a). We record the leakage traces at a sampling rate of 100MSa/s and the bandwidth limit of 200MHz. Figure 11(b) presents the EM patterns of multiplications on the target device. The leakage is located at line 3 of *toom_cook_4way*() procedure (Figure 1), where the decoded ciphertexts are multiplied by the secret key. Each operation contains 144 schoolbook multiplications, implemented by the MUL instruction. The leakage of MUL instruction is exploited to perform the SASCA.


(a) Measurement setup.



(b) EM trace.

Figure 11: (a)Experimental setup demonstrating the measurement oscilloscope, PC and target board. (b)The measured EM trace of implementation.

To magnify the leakage of multiplication results, we choose the 16 ciphertexts which distinguish HW difference among different key classes, similar with the chosen ciphertext on attacking Kyber [XPSR+21]. The training set consists of 90,000 traces captured for the chosen ciphertexts. We use a unified deep learning model to profile all the leakage of multiplications. The MLP architecture is shown in Figure 7. Each trace segment in the training set is assigned its label as the Hamming weight of multiplication result to train the network. We set the learning rate to 10^{-2} and the number of epochs to 1000 with a batch size of 4096. The normalization of the trace is applied to the input of the network.

After training the network, we use the profiled model to attack the target trace. The target trace segment is sent into the network to predict the Hamming weight of the intermediate value. Thus the probability distribution under 17 classes (0~16) is obtained. These probabilities are assigned to the side-channel leakage nodes f_L in the SFG factor graph. In this type factor graph, $aw1_i[0] \sim aw1_i[15]$ are the ciphertext values, while the $bw1_i$ is the unknown node. Since the efficiency has been proved and evaluated, it performs BP on the Bayes-based SFG.

To compared to the traditional template attack, we also perform the template attack on the same profiling measurements. We use the trace set to build Hamming weight templates for each multiplication. The points-of-interest (POIs) used for template building are determined with the correlation coefficients. We have also set different numbers of POIs in our attacks, i.e. $50, 80, 100$. Then the obtained probabilities are used to perform the BP on our factor graph. We perform 100 experiments with independent attacking traces. The success rates of attacking $bw1\_3, bw1\_2, bw1\_1, bw1\_0$ are shown as the grey histograms in Figure 12.

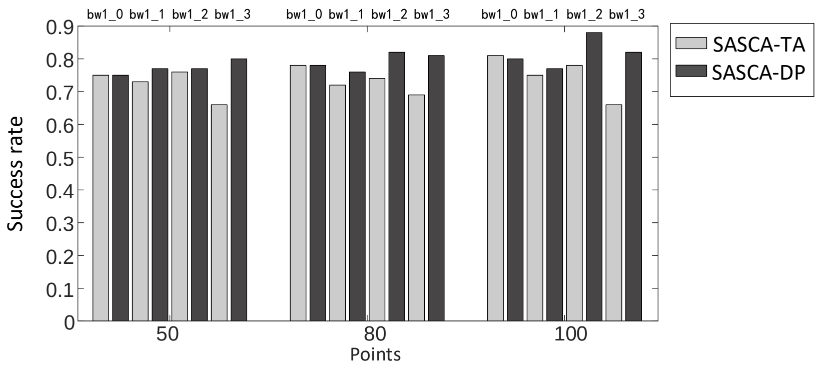


Figure 12: Success rates of practical attacks. The results of SASCA with the template attack (SASCA-TA) are indicated as the grey histograms while SASCA with deep learning power attack (SASCA-DP) are the black.

During the deep learning-based attack, the parameters are the same with template attacks. After BP, it can recover the coefficients of the secret key. The success rates are shown as the black histograms in Figure 12. We can observe that the deep learning-assisted SASCA achieves a higher success rate than the template attack. Meanwhile, it simplifies the profile procedure other than a large number of templates in the template attacks.

6 Discussions

We establish that single-trace attacks on the Toom-Cook. The method can be applied to other cryptographic algorithms using Toom-Cook as a polynomial multiplication. We now discuss the attack scenario and some possible countermeasures.

6.1 Scenario and limitation

A straight application of KEMs (NTRU, Saber) is the authenticated key exchange protocol. New protocol standards, including TLS 1.3, are increasingly advocating and mandating PKE/KEMs that utilize ephemeral key pairs to achieve the notion of perfect forward secrecy [SFG20].

The key exchange at the beginning of a TLS session involves one keygen, one encapsulation, and one decapsulation in the post-quantum TLS key exchange schemes [SSW20, BBCT22]. It makes session key recovery more attractive for recovering the real-time encrypted information [RBRC20]. Some implementations reuse the keys at the cost of limited forward secrecy. It usually restricts the update rules. For example, Microsoft Windows TLS library Schannel caches keys for two hours [CNE+14]. It limits the number of traces required by DPA/CPA and makes the single-trace attack more threatening.

Like template attacks, SASCA needs profiling and the ability to configure keys during the profiling phase. Since the complexity of building the profiles takes more time than DPA/CPA, it requires many traces to create a good template. The POI (Points-of-Interest) for each attack point aligns with the observable leakage is necessary for a successful attack. In addition, one needs to know the implementation to construct a factor graph in the attack. We limit our attack to a single device. A more threatening attack can be extended to the cross-device scene. It needs more complicated profiling than on a single device. There are many works to conquer the variation of devices, including multi-device traces training [DGD+19], principle comportment analysis [GDD+19], and frequency transformation [ZSX+20]. In the future, we will extend our method to the cross-device context based on the related technologies.

6.2 Potential defenses

There already exist masking implementations for the lattice schemes [VBDK+21, BGR+21]. Masking was proposed to resist the DPA and lead to interference in single-trace attacks. The randomized intermediate variables in the scheme destroy the relationship between the sensitive information and side-channel leakage.

However, the SASCA on NTT can also attack the masked implementation [PPM17, PP19]. The masking splits the secret key into two shares before decryption, and it simply recovers each share individually and adds them up to obtain the unmasked input. Due to the linearity of the polynomial multiplication and addition, masking is a natural fit for lattice-based schemes [RSRVV15, VBDK+21]. In these schemes, the private key s can be split into two shares s', s'', which satisfy $s = s' + s'' \mod q$. Then, the polynomial multiplications, additions and other operations can be computed on each share individually. In the last step, the decoding module is designed carefully to output two shares of the message. The targets are the polynomial multiplications between the two shares and the ciphertext b' at the beginning of the first stage. Thus it performs the SASCA on the NTT implementation of $b'^T s'$ and $b'^T s''$ to recover s', s'' individually [PPM17, PP19]. The principle of attacking masking is also suitable for the single-trace attack on Toom-Cook.

Shuffling is one of the hiding countermeasures and is viewed as an effective countermeasure against algebraic side-channel attacks. It can use Fisher-Yates algorithm [FY53] to decrease the SNR during the execution. To seek higher security, it can integrate other countermeasures, for example, clock jitter, instruction shuffling, and dummy operations [CK09, VCMKS12, BPS+20]. How to protect the implementation with lightweight countermeasure is one of our future work.

7 Conclusion

In this work, we investigate the security of the Toom-Cook multiplication concerning single-trace attacks. We show how to apply SASCA when targeting Toom-Cook in Saber and additionally adapt the deep learning power attacks to decrease the tremendous amount of multivariate templates of traditional SASCA. More concretely, we prove that the marginal probabilities of the Sum-Product algorithm in the message-update rule are equal to the posterior probabilities based on Bayes' theorem in the factor graph of schoolbook multiplications. The nodes in the graph representation of the schoolbook results in the Toom-Cook can be merged by the Bayes' theorem. We also improve the BP algorithm to eliminate the influence of short cycles that frequently appears in the factor graph of Toom-Cook. These technologies are also applicable to other cryptographic schemes that perform Toom-Cook algorithm.

8 Acknowledgements

This work was supported by the National Key R&D Program of China (Grant No.2020AAA0 107703), the National Natural Science Foundation of China (Grant No.62072247, 62132008 and 61972295) and the High level personnel project of Jiangsu Province (JSSCBS20210274).

References

[AASA$^+$20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, Jul. 2020.

[ACC$^+$21] Amin Abdulrahman, Jiun-Peng Chen, Yu-Jia Chen, Vincent Hwang, Matthias J. Kannwischer, and Bo-Yin Yang. Multi-moduli ntts for saber on cortex-m3 and cortex-m4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):127–151, Nov. 2021.

[ACLZ20] Dorian Amiet, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. Defeating newhope with a single trace. In *International Conference on Post-Quantum Cryptography*, pages 189–205. Springer, Cham, 2020.

[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), 2018.

[AKP$^+$] Furkan Aydin, Emre Karabulut, Seetal Potluri, Erdem Alkim, and Aydin Aysu. Reveal: Single-trace side-channel leakage of the seal homomorphic encryption library. pages 1527–1532.

[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)*, pages 81–88. IEEE, Washington, DC, USA, 2018.

[BBCT22] Daniel J Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. OpenSSLNTRU: Faster post-quantum TLS key exchange. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, 2022.

[BCDM$^+$13] G Becker, J Cooper, E De Mulder, G Goodwill, J Jaffe, G Kenworthy, et al. Test vector leakage assessment (tvla) derived test requirements (dtr) with aes. In *International Cryptographic Module Conference*, 2013.

[BCLVV16] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine Van Vredendaal. Ntru prime. *IACR Cryptology ePrint Archive*, 2016:461, 2016.

[BDH$^+$21] Shivam Bhasin, Jan-Pieter DAnvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and defending masked polynomial comparison for lattice-based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):334–359, Jul. 2021.

[BDK$^+$18] Joppe 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)*, pages 353–367. IEEE, London, UK, 2018.

[BGR$^+$21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):173–214, Aug. 2021.

[BMKV20] 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 Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):222–244, Mar. 2020.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[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.

[CH06] Kyuhyuk Chung and Jun Heo. Improved belief propagation (bp) decoding for ldpc codes with a large number of short cycles. In *2006 IEEE 63rd Vehicular Technology Conference*, volume 3, pages 1464–1466. IEEE,Melbourne, VIC, Australia, 2006.

[CHK+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 Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):159–188, Feb. 2021.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 156–170. Springer, Berlin, Heidelberg, 2009.

[CNE+14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *23rd USENIX security symposium (USENIX security 14)*, pages 319–335. USENIX Association, San Diego CA, Aug. 2014.

[CRR03] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28. Springer, Berlin, Heidelberg, 2003.

[DGD+19] Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. X-deepsca: Cross-device deep learning side channel attack. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[DKSRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, Cham, 2018.

[DLG19] Jinnan Ding, Shuguo Li, and Zhen Gu. High-speed ecc processor over nist prime fields applied with toomccook multiplication. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(3):1003–1016, 2019.

[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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1857–1874, 2017.

[FDK20] Apostolos P Fournaris, Charis Dimopoulos, and Odysseas Koufopavlou. Profiling dilithium digital signature traces for correlation differential side channel attacks. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 281–294. Springer, Cham, 2020.

[FGL$^+$18] Adrien Facon, Sylvain Guilley, Matthieu Lec'Hvien, Alexander Schaub, and Youssef Souissi. Detecting cache-timing vulnerabilities in post-quantum cryptography algorithms. In *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, pages 7–12. IEEE, 2018.

[FY53] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research.* Hafner Publishing Company, 1953.

[GBHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 323–345. Springer, Berlin, Heidelberg, 2016.

[GDD$^+$19] Anupam Golder, Debayan Das, Josef Danial, Santosh Ghosh, Shreyas Sen, and Arijit Raychowdhury. Practical approaches toward deep-learning-based cross-device power side-channel attack. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(12):2720–2733, 2019.

[GGJR$^+$11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GGSB20] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):209–238, Aug. 2020.

[GT15] Torbjrn Granlund and Gmp Development Team. *GNU MP 6.0 Multiple Precision Arithmetic Library.* Samurai Media Limited, London, GBR, 2015.

[HCY19] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on ntru prime. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):123–151, Nov. 2019.

[HHP$^+$21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked cca2 secure kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):88–113, Aug. 2021.

[KAA21] Emre Karabulut, Erdem Alkim, and Aydin Aysu. Single-trace side-channel attacks on ω-small polynomial sampling: With applications to ntru, ntru prime, and crystals-dilithium. In *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 35–45. IEEE, 2021.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KFL01] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[KH18] Suhri Kim and Seokhie Hong. Single trace analysis on constant time cdt sampler and its countermeasure. *Applied Sciences*, 8(10):1809, 2018.

[KO62] Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences, 1962.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):243–268, Jun. 2020.

[KRVV19] Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Pushing the speed limit of constant-time discrete gaussian sampling - a case study on the falcon signature scheme. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[MMK$^+$04] David JC MacKay, David JC Mac Kay, et al. Information theory, inference, and learning algorithms. *IEEE Transactions on Information Theory*, 50(10):2544–2545, 2004.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked ind-cca secure saber kem implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):676–707, Aug. 2021.

[NG21] Duc Tri Nguyen and Kris Gaj. Optimized software implementations of crystals-kyber, ntru, and saber using neon-based special instructions of armv8. In *Proceedings of the NIST 3rd PQC Standardization Conference (NIST PQC 2021)*, 2021.

[NIS] NIST. Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[OM06] Elisabeth Oswald and Stefan Mangard. Template attacks on masking—resistance is futile. In *Topics in Cryptology – CT-RSA 2007*, pages 243–256. Springer, Berlin, Heidelberg, 2006.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Gneysu. Practical cca2-secure and masked ring-lwe implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):142–174, Feb. 2018.

[Pes16] Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *Progress in Cryptology – INDOCRYPT 2016*, pages 153–170. Springer, Cham, 2016.

[PP19] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In *Progress in Cryptology – LATINCRYPT 2019*. Springer, Cham, 2019.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. volume 10529, pages 513–533, 2017.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. Scaul: Power side-channel analysis with unsupervised learning. *IEEE Transactions on Computers*, 69(11):1626–1638, 2020.

[RBRC20] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. Drop by drop you break the rock-exploiting generic vulnerabilities in lattice-based pke/kems using em-based physical attacks. *Cryptology ePrint Archive*, 2020.

[RBRC22] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On exploiting message leakage in (few) nist pqc candidates for practical message recovery attacks. *IEEE Transactions on Information Forensics and Security*, 17:684–699, 2022.

[RSRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based pke and kems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):307–335, Jun. 2020.

[RSRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 683–702. Springer, Berlin, Heidelberg, 2015.

[RU01] T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[SFG20] Douglas Steblia, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in tls 1.3. Technical report, Internet-Draft draft-ietf-tls-hybrid-design-00. Internet Engineering Task, 2020.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[SKL$^+$20] Bo-Yeon Sim, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. Single-trace attacks on message encoding in lattice-based kems. *IEEE Access*, 8:183175–183191, 2020.

[SMS19] Thomas Schamberger, Oliver Mischke, and Johanna Sepulveda. Practical evaluation of masking for ntruencrypt on arm cortex-m4. In *Constructive Side-Channel Analysis and Secure Design*, pages 253–269. Springer International Publishing, Cham, 2019.

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, page 1461C1480. Association for Computing Machinery, New York, NY, USA, 2020.

[SV93] M. Shand and J. Vuillemin. Fast implementations of rsa cryptography. In *Proceedings of IEEE 11th Symposium on Computer Arithmetic*, pages 252–259, 1993.

[SWT14] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[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.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[VBDK+21] Michiel Van Beirendonck, Jan-Pieter Danvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of saber. *ACM Journal on Emerging Technologies in Computing Systems*, 17:1–26, Apr. 2021.

[VCGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *Advances in Cryptology – ASIACRYPT 2014*, pages 282–296. Springer, Berlin Heidelberg, 2014.

[VCMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *Advances in Cryptology – ASIACRYPT 2012*, pages 740–757. Springer, Berlin, Heidelberg, 2012.

[XPSR+21] Zhuang Xu, Owen Michael Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Transactions on Computers*, pages 1–1, 2021.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019.

[ZS20] Yuanyuan Zhou and François-Xavier Standaert. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks. *Journal of Cryptographic Engineering*, 10(1):85–95, 2020.

[ZSX+20] Fan Zhang, Bin Shao, Guorui Xu, Bolin Yang, Ziqi Yang, Zhan Qin, and Kui Ren. From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.