UDC 003.26:519.688

Bit-sliced Algorithm for the 512-point Number Theoretic Transform

Illia Kripaka[1], Andrii Fesenko[1]

[1]*National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Institute of Physics and Technology*

Abstract

A method for computing the 512-digit number theoretic transform used in the Vershyna digital signature scheme, employing bitwise digit operations, is proposed. The correctness of the developed algorithm and its efficient constant-time performance have been proven.

The obtained results indicate that the proposed approach is adaptive and can be applied to computations with other polynomials. This enables its easy integration into various cryptosystems to ensure protection against side-channel attacks. The proposed method does not require changes to the digital signature scheme itself, introducing modifications only to the polynomial multiplication function.

Introduction

In recent years, post-quantum algorithms have been actively researched and implemented to protect against potential attacks involving quantum computers, which will leverage quantum computing to solve mathematical problems. Given the prospective ability of quantum computers to efficiently solve certain computationally hard problems for classical computers, NIST, in turn, launched a competition in 2017 to standardize future post-quantum algorithms [1].

The third round of the competition concluded with the selection of the following schemes and mechanisms: Dilithium, Kyber, FALCON, and SPHINCS+. The first three of these employ algebraic operations on lattices, which, in turn, rely on the operation of multiplying large polynomials together.

In addition to the third round, NIST also announced a fourth round, in which other algorithms that do not use lattice arithmetic will be considered. The main reason is the desire to diversify the algorithms.

Despite the fact that the mathematical foundation of the standardized lattice-based schemes is considered reliable, practical implementations, especially polynomial multiplication operations, often become the target of side-channel attacks.

Optimizations aimed at improving performance can inadvertently create vulnerabilities that attackers can exploit to leak secret data. During algorithm optimization, specialized parameters are often employed, which, on one hand, enhance their performance, while on the other, may open additional opportunities for side-channel attacks by adversaries.

The Number Theoretic Transform (NTT), in turn, is a critical component of most of the aforementioned schemes, and therefore protecting its implementation from potential data leaks also plays an important role.

1. Existing algorithms for polynomial multiplication and features of the number theoretic transform

For polynomial multiplication, algorithms such as the following are used, for example.

- **Standard multiplication algorithm** or referred to in the literature as the "schoolbook" algorithm.
  This algorithm has a complexity of $O(n^2)$ and uses standard polynomial multiplication followed by modulo reduction [2].
- **Karatsuba/Tom-Cook algorithm.**
  Historically, Anatoly Karatsuba and Andrei Tom with help of Stephen Cook developed almost the same algorithm, only at different points on the planet. This algorithm uses the "divide and conquer" strategy for multiplying two numbers, which makes it

possible to calculate the multiplication between significantly smaller numbers, which makes it efficient. The complexity of Karatsuba's algorithm is $O(n^{1.58})$ [3] due to the decomposition of the number into 2 factors. The complexity of Toom-Cook algorithm is $O(n^r)$, where k is the number of smaller parts and $r = \log_k 2k - 1$ [4, 2].

- **Fast Fourier Transform (FFT) algorithm.** In general, Discrete Fourier Transform (DFT) is used to process signals and decoupling them into components, but, according to its properties, in the context of polynomial multiplication, DFT can be used to directly calculate the product. This method of calculation will have the same speed as "schoolbook" multiplication, namely $O(n^2)$. The idea for creating a more efficient version of the algorithm was proposed as early as 1805 [5] in an unpublished work by Friedrich Gauss, but the initiative to create an efficient algorithm was taken up in 1965, when Cooley and Tukey proposed their algorithm, which runs in $O(n \cdot \log n)$ [6]. By its nature, FFT operates on floating-point numbers, which in some situations can cause errors, while NTT operates on integers relative to operations in the field. Most schemes utilizing lattices tend to use NTT.

The number theoretic transform is a specialized form of the discrete Fourier transform that that operates using modular arithmetic. Unlike the DFT, which works with complex numbers, the NTT relies on integer modulo arithmetic. This property makes it suitable for error-free and efficient computations in cryptography and polynomial multiplication. The efficiency of the NTT is achieved through the use of algorithms similar to those employed in the FFT: the Cooley-Tukey algorithm, the Gentleman-Sande algorithm, and others.

In NTT calculations, cyclotomic polynomials of the form $x^n + 1$ and $x^n - 1$ are typically used. Thus, the algorithm efficiently utilizes its resources depending on whether cyclic or negative convolution is employed, with certain differences.

In the modern NIST standards Kyber, Dilithium, and the Ukrainian national standard DSTU 9212:2023 (Vershyna digital signature scheme), the most commonly used form of cyclotomic polynomials is $x^n + 1$, which is employed to construct negative convolution and provides certain advantages in implementation.

Similar to the DFT, the number theoretic transform consists of the forward NTT and the inverse INTT. Structurally, the NTT and INTT algorithms are similar, differing mainly in constants and the direction of coefficient rotation.

The general procedure for multiplying polynomials $a(x), b(x) \in R_q$ with coefficients $a(x) = (a[0], a[1], \ldots, a[n-1])$ and $b(x) = (b[0], b[1], \ldots, b[n-1])$ is performed in three stages. The multiplication of polynomials $a \cdot b \in R_q$ is defined as $c = \text{INTT}(\text{NTT}(a) \odot \text{NTT}(b))$, where $\odot$ denotes element-wise multiplication of the polynomials.

Polynomial multiplication is typically performed in rings of the form $\mathbb{Z}_q[x]/(x^n + 1)$ or $\mathbb{Z}_q[x]/(x^n - 1)$, corresponding to positive or negative convolution, respectively.

- In the case of positive convolution, the NTT will take the following form:

$$\hat{a}_j = \sum_{i=0}^{n-1} a_i \omega^{ij} \pmod q, \quad j = 0, \ldots, n-1.$$

  Correspondingly, the inverse transform $INTT$ takes the form:

$$a_i = n^{-1} \sum_{j=0}^{n-1} \hat{a}_j \omega^{-ij} \pmod q, \quad i = 0, \ldots, n-1.$$

- In the case of negative convolution, the NTT will look the same as in the case of positive convolution, up to multiplication by the coefficient ψ:

$$\hat{a}_j = \sum_{i=0}^{n-1} a_i \psi^i \omega^{ij} \pmod q, \quad j = 0, \ldots, n-1.$$

  Correspondingly, the inverse transform $INTT$ takes the form:

$$a_i = n^{-1} \psi^{-i} \sum_{j=0}^{n-1} \hat{a}_j \omega^{-ij} \pmod q, \quad i = 0, \ldots, n-1.$$

The most commonly used version of NTT is the negative convolution, in accordance with the NIST agency standards Kyber, Dilithium, and the national standard DSTU 9212:2023. Therefore, we will further use the field $\mathbb{F}_q$ and the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where $\mathbb{Z}_q[x] = (\mathbb{Z}/q\mathbb{Z})[x]$.

It is assumed that n must be a square of some number, and the modulus q must be prime such that $q \equiv 1 \pmod{2n}$. Thus, the field $\mathbb{F}_q$ will contain an n-th root of unity ω and its square root $\psi : \psi^{2n} = 1$, which implies $\omega^n = 1 \pmod{q}$.

2. Attacks on existing implementations of the Number Theoretic Transform and possible approaches to mitigating side-channel attacks

Side-channel attacks on NTT exploit physical information leaks during the computation of the transform: algorithm execution time, power consumption, electromagnetic radiation, and others. The additional data can be used to construct attacks on cryptographic schemes that rely on NTT. Let us consider the existing side-channel attacks on the number theoretic transform:

- **Overdetermined system method** [7].
  This attack involves constructing a system of inequalities that constrain the possible values of unknown coefficients when a portion of the coefficients in the private key is known. This approach significantly reduces the search space and enables efficient recovery of the key.
- **Method using SIS algorithms** [7].
  It has been shown that by employing methods for solving SIS problems on lattices, one can successfully recover vectors s from multiplication by the matrix $As = t$. Multiplications of this kind are actively used to generate the private key and to form the signature.
- **Dimension reduction using SIS algorithms** [7]
  This key recovery strategy embeds the NTT transformation matrix into an SIS search problem. In general, it optimizes the SIS-based attack by reducing the rank of the lattice used for the search. Instead of working with the full n-dimensional problem, this method selects only a subset of equations corresponding to known coefficients, effectively mapping the problem into a space of significantly lower dimension.
- **Attacks on possible variable shuffling** [8].
  It has been demonstrated that simple simple methods of shuffling the order in which coefficients are selected for NTT computation may not provide sufficient protection against side-channel attacks, since adversaries can exploit differences in variable distributions to recover secret keys with a finite number of observations.
- **Soft analytical side-channel attacks on NTT** [9].
  Soft analytical side-channel attacks mathematically model the butterfly method network used in NTT as a factor graph, in which side-channel leakages obtained through template matching are integrated as initial probabilistic distributions – soft information – for intermediate variables. The core of the attack is an analytical stage that efficiently solves the resulting marginalization problem for this graph using the belief propagation algorithm to recover the most probable values of the secret coefficients. The boundary problem refers to the use of probabilistic models to represent uncertainty in the system and calculate boundary distributions for individual variables.
- **Single-Trace Attacks on NTRU via NTT** [10].
  In the study of the security of the NTRU cryptosystem implementation, it was shown that the device's power consumption can be sufficient to compromise it using the number theoretic transform. These attacks analyze power consumption leakages from the device; the authors also confirm the possibility of extracting the secret key using only a single measurement (single-trace attack) and describe a new template attack to demonstrate the vulnerability in general.

There exist various strategies and approaches to counter side-channel attacks. The main countermeasures are the following methods.

- Physical protection at the hardware level — electromagnetic shielding of components.
- Implementation of algorithms that operates in constant time — this helps avoid attacks, especially under constrained computational

resources. Studies [11, 12] have employed different techniques, demonstrating the feasibility of such implementations.

- Vector masking – this introduces entropy into intermediate values so that they do not correlate with traces left during device usage. Such transformations have been described in articles [7, 13, 14, 15].
- Hiding information using the Chinese Remainder Theorem — this allows coefficients to be chosen randomly and added before certain operations, enabling them to be easily removed later after applying modular reduction. Such protection was presented in the work [16].
- Shuffling to complicate data processing — this ensures that additional information obtained from reading extra data about the device's operation cannot be trivially mapped to the corresponding nodes in the factor graph, provided there is sufficient entropy. Such transformations were described in the works [17, 14].
- Bitwise segmentation — this involves reducing all possible operations to bitwise ones that are executed in constant time. This type of technique was first developed to improve the performance of the DES algorithm [18].

3. Bit-slicing algorithm of the NTT-256 transform

Bit-slicing in software is an implementation strategy that enables the construction of constant-time algorithms, where execution time depends solely on the data size. Algorithms implemented using this approach demonstrate resistance to side-channel attacks, particularly those exploiting leaks through cache memory and execution timing [19].

The key concept lies in decomposing the computational function down to elementary bitwise logical operations (AND, XOR, OR, NOT), analogous to designing a combinational logic circuit at the hardware level. Since these operations — like combinational circuits — are independent, they provide significant opportunities for efficient parallelization of computations.

Let us consider two different existing bit-slicing algorithms of the NTT-256 transform.

1) **Algorithm proposed in [20].**
This approach provides a formal description of standard bit shuffling. From equation 1, it can be seen that each term in the sum extracts the j-th bit of index i and places it in the symmetric position $(nS-1-j)$ in the output index.

$$\begin{gathered} b[i] = \text{PolyBitReversal}(a)[i] = \\ = a[\text{BitReversal}(i)], \\ \text{where} \quad \text{BitReversal}(i) = \\ = \sum_{j=0}^{nS-1} (((i \gg j)\&1) \ll (nS-1-j)) \\ \text{and} \quad nS = \log(n). \end{gathered} \tag{1}$$

2) **Standard shuffling used in the NIST standard [21].**
An alternative method involves using a standard procedure in which each index must be mapped to its reverse binary representation. It is assumed that for this purpose, $\log n$ bits are taken into account, where n is the length of the polynomial used for computing the NTT.

The next critical step is data transposition, since bitwise segmentation aims to process a sequence of bits in "layers", similar to the operation of a hardware combinational circuit, which requires appropriate reorganization of input data and reassembly at the output.

In a 32-bit implementation, transposition must convert 32 n-bit numbers into n 32-bit numbers.

The core computational unit of the algorithm is the butterfly method, applied in subsequent steps. It requires the implementation of:

- 32-bit addition and subtraction with modular reduction;
- 32-bit multiplication combined with the Barrett reduction algorithm..

In general, the butterfly method takes three input values — $in1$, $in2$, ω, and computes the output values $out1$, $out2$ as follows:

$$\begin{aligned} out1 &= in1 + in2 \cdot \omega, \\ out2 &= in1 - in2 \cdot \omega. \end{aligned}$$

After shuffling and transposing the input data, the algorithm performs 5 steps using the butterfly method, accompanied by additional data operations. This is necessary because, in the initial

stages, extensive shuffling of the results obtained from the butterfly method must occur.

These steps are accompanied by large-scale shuffling operations on intermediate results. On the sixth computation stage, the data is aggregated into four 64-digit polynomials, from which, using two additional steps, the correct computation of the NTT-256 transform can be obtained.

4. Construction of a new bit-sliced algorithm for the NTT-512 transform in the Vershyna digital signature scheme

Since bit-slicing is considered an effective auxiliary approach to counter side-channel attacks, there arose a need to adapt the polynomial multiplication method implemented in the Vershyna digital signature scheme.

New bit-sliced algorithm

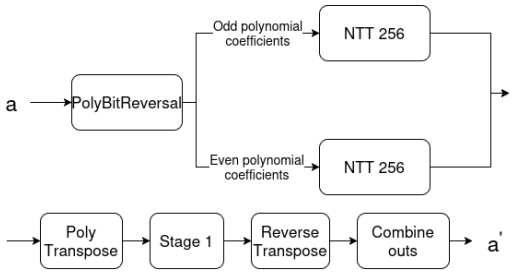

Figure 1: Diagram for the bit-sliced computation of the 512-coefficient NTT transform.

The algorithm scheme is demonstrated in Figure 1.

Algorithm 1 (New bit-sliced algorithm for computing NTT-512)**.**

Preparation:

Precomputed transposed values:

- ω_{512}*, which is the 512th root of unity;*
- $\omega_{256} = \omega_{512}^2$*;*
- $\psi_{256} = \sqrt[2]{\omega_{256}}$ *and* $\psi_{512} = \sqrt[2]{\omega_{512}}$.

Input*: Polynomial* $a(x) \in R_q$.

Output*: Polynomial* $\hat{a}(x) \in R_q$*, such that* $\hat{a} = NTT(a)$.

Algorithm steps*:*

1) *Application of shuffling for a 512-point polynomial.*
2) *Decomposition of the polynomial into coefficients with even and odd indices. These sets are passed to the corresponding identical functions from the 256-bit NTT algorithm for further processing from [20].*
3) *Transposition of outputs according to Figure 3 to convert them into a format suitable for a bitwise-segmented butterfly round using 32-bit blocks.*
4) *Performing one butterfly round, which includes the parallel application of eight butterfly computation blocks on different input values, according to Figure 2.*
5) *Performing inverse transposition to convert the results from bit-sliced format back to the standard polynomial representation with coefficients.*
6) *Correct merging of the two output arrays to form the final 512-point polynomial, where the values from one array are placed in odd positions and those from the other array are placed in the corresponding even positions.*

Proof of correctness

Theorem 1. *Algorithm 1 for computing NTT-512 runs in constant time.*

Proof. To prove constant-time execution, it is necessary to show that the total number of operations and the execution time of the algorithm do not depend on the values of the input data.

- **Initial shuffling of the polynomial.**
  Shuffling, according to the equation 1, runs in constant time, since shuffling a polynomial requires only knowing the corresponding indices to which each element should be moved. The calculation of the index itself also occurs independently of the numerical values.
- **Direct and inverse transposition of 32-bit 32-point polynomials.**
  The transposition operation works with fixed-size data, changing only their representation. For this algorithm, it is only necessary to move a fixed number of bits from thirty-two 32-bit numbers into another format. The inverse algorithm works similarly, only transposing thirty-two 32-bit arbitrary numbers into thirty-two 32-point polynomials.

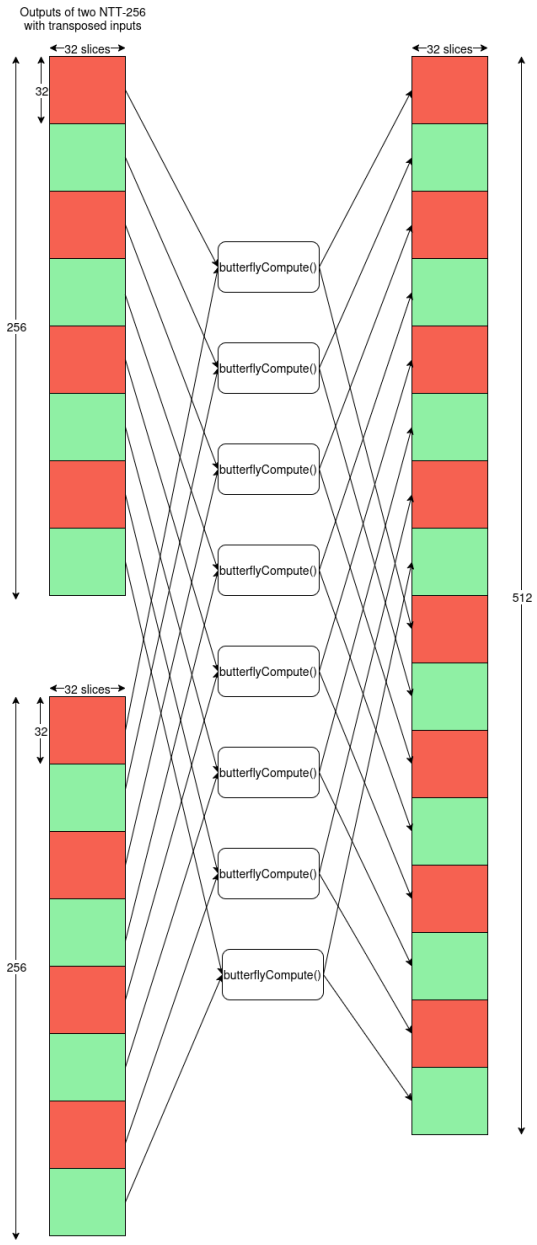

Figure 2: Diagram for the first step using the butterfly method for the 512-coefficient NTT transform.

- **Multiplication by a predetermined transposed number.**
  The implementation of multiplication through bit-sliced combinational circuits means that the operation reduces to a set of bitwise logical operations, where the corresponding bits are taken and combined. As a result, the answer is returned in the format of one bit of information. Additionally, the execution flow of such operations is fixed and does not contain conditional branches that depend on the data.

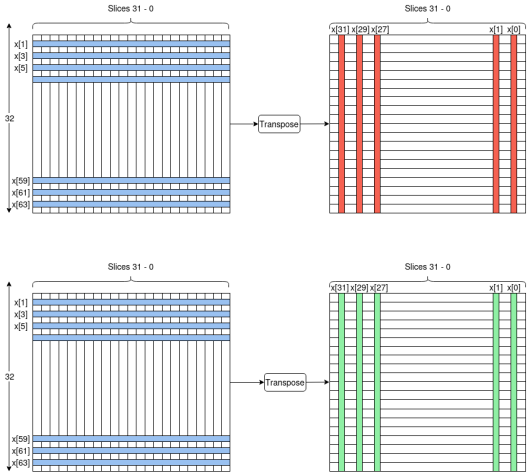

Figure 3: Diagram for transposing 32-coefficient polynomials, used by the butterfly method for the 512-coefficient and 256-coefficient NTT transform.

- **Butterfly method.**
  Computation using the butterfly method is performed according to article [20] and uses SIMD operations. Accordingly, such an algorithm will run in constant time independently of the input data.
  The butterfly method calculation is performed according to the article [20] and uses SIMD operations. Accordingly, such an algorithm will run in constant time regardless of the input data.
- **Combining outputs.**
  The final merging of results is performed in constant time. This stage involves only copying coefficients to their final positions using known indices, which is a fixed-time operation independent of the data values.

Theorem 2. *Algorithm 1 for calculating the NTT-512 transformation is correct.*

Proof. To prove the correctness of the algorithm, we need to analyze the steps of the NTT-512 transformation algorithm.

- **Initial shuffling.**
  Shuffling is performed in a standard manner, according to equation 1, which does not affect the correctness of the algorithm.
- **Separation and calculation of two NTT-256 transformations.**
  The algorithm involves splitting the polynomial into even and odd coefficients to feed them as input to two independent NTT-256 calculations. This decomposition is correct because the Cooley-Tukey algorithm

has an iterative structure, allowing it to be divided into partial computations using smaller NTTs based on the size of the input data. This principle aligns with the theoretical derivation of the algorithm [22].

- **Transposition of the output after computing two NTT-256 transforms.**
  The results of the NTT-256 calculations are provided in standard coefficient representation along with numbers in the usual computer format, making transposition important in this context. It converts the standard number representation into a format convenient for use with the combinational circuit of the butterfly method [20].
- **Butterfly method for NTT-512 transform.**
  One of the final butterfly transformations after transposing the outputs from the NTT-256 transformations is correct according to the Cooley-Tukey algorithm [22], which allows for the iterative structure of the algorithm and its division into steps.
- **Inverse transposition.**
  Inverse transposition is correct because it converts bit values back into meaningful numbers. Transposition was used here to transform numbers into bit representation to enable the use of combinational circuits for the butterfly method [20].
- **Combining outputs.**
  The final operation of combining outputs moves elements from the two computed 256-element arrays into the corresponding positions for the 512-degree polynomial, according to a similar implementation in [20].

Conclusions

A bit-sliced number theoretic transformation algorithm for 256-coefficient polynomials is examined, and a computation method for 512-coefficient polynomials using bit-slice operations is proposed. The correctness of the constructed algorithm and its constant time execution are proven. The proposed method does not require changes to the structure of the digital signature scheme and is limited only to modifying the polynomial multiplication function. The obtained results open the possibility of adapting the algorithm for various types of polynomials.

References

[1] National Institute of Standards and Technology (NIST), "Post-quantum cryptography project," 2025. Accessed: 2025-03-04.

[2] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms, Volume 2*. Addison-Wesley Professional, 2014.

[3] A. A. Karatsuba and Y. P. Ofman, "Multiplication of many-digital numbers by automatic computers," in *Doklady Akademii Nauk*, vol. 145, pp. 293–294, Russian Academy of Sciences, 1962.

[4] R. E. Crandall and C. Pomerance, *Prime numbers: a computational perspective*, vol. 2. Springer, 2005.

[5] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast fourier transform," *Archive for History of Exact Sciences*, vol. 34, no. 3, pp. 265–277, 1985.

[6] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[7] Z. Qiao, Y. Liu, Y. Zhou, M. Shao, and S. Sun, "When NTT meets SIS: Efficient side-channel attacks on dilithium and kyber." Cryptology ePrint Archive, Paper 2023/1866, 2023.

[8] P. Pessl, "Analyzing the shuffling side-channel countermeasure for lattice-based signatures." Cryptology ePrint Archive, Paper 2017/033, 2017.

[9] F. Custers, A. Hülsing, C. Cloostermans, and J. Renes, *Soft Analytical Side-Channel Attacks on the Number Theoretic Transform for Post-Quantum Cryptography*. PhD thesis, 2022.

[10] T. Rabas, J. Buček, and R. Lórencz, "Single-trace side-channel attacks on ntru implementation," *SN Computer Science*, vol. 5, no. 2, p. 239, 2024.

[11] E. Kupcová and M. Drutarovský, "Number-theoretic transform with constant time computation for embedded post-quantum cryptography," *Acta Electrotechnica et Informatica*, vol. 22, no. 4, pp. 30–37, 2022.

[12] J. Bertels, H. V. L. Pereira, and I. Verbauwhede, "FINAL bootstrap acceleration

on FPGA using DSP-free constant-multiplier NTTs." Cryptology ePrint Archive, Paper 2025/137, 2025.

[13] R. C. Rodriguez, E. Valea, F. Bruguier, and P. Benoit, "Hardware implementation and security analysis of local-masked NTT for CRYSTALS-kyber." Cryptology ePrint Archive, Paper 2024/1194, 2024.

[14] P. Ravi, R. Poussier, S. Bhasin, and A. Chattopadhyay, "On configurable SCA countermeasures against single trace attacks for the NTT - a performance evaluation study over kyber and dilithium on the ARM cortex-m4." Cryptology ePrint Archive, Paper 2020/1038, 2020.

[15] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal, "Masking kyber: First- and higher-order implementations." Cryptology ePrint Archive, Paper 2021/483, 2021.

[16] D. Heinz and T. Pöppelmann, "Combined fault and DPA protection for lattice-based cryptography." Cryptology ePrint Archive, Paper 2021/101, 2021.

[17] G. Assael, P. Elbaz-Vincent, and G. Reymond, "Improving single-trace attacks on the number-theoretic transform for cortex-m4," in *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 111–121, 2023.

[18] E. Biham, "A fast new des implementation in software," in *Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4*, pp. 260–272, Springer, 1997.

[19] T. Taubert, "Bitslicing, an introduction: Data orthogonalization for cryptography," August 2018. Accessed: 2025-04-25.

[20] R. Singh, S. Islam, B. Sunar, and P. Schaumont, "An end-to-end analysis of emfi on bit-sliced post-quantum implementations," 2022.

[21] National Institute of Standards and Technology, "FIPS 204: Module-Lattice-Based Digital Signature Standard," August 2024. Federal Information Processing Standards Publication 204.

[22] W. Cochran, J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, G. Maling, D. Nelson, C. Rader, and P. Welch, "What is the fast fourier transform?," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 45–55, 1967.