Efficient Error Detection Methods for the Number Theoretic Transforms in Lattice-Based Algorithms

Mohamed Abdelmonem[1,2] [0009−0000−0706−9252], Lukas Holzbaur[2] [0000−0002−8048−3051], Håvard Raddum[1] [0000−0001−9779−5986], and Alexander Zeh[2] [0000−0002−5044−4940]

[1] Simula UiB, Bergen, Norway {mohameda,haavardr}@simula.no
[2] Infineon Technologies AG, Munich, Germany
{lukas.holzbaur,alexander.zeh}@infineon.com

Abstract. The Number Theoretic Transform (NTT) is a crucial component in many post-quantum cryptographic (PQC) algorithms, enabling efficient polynomial multiplication. However, the reliability of NTT computations is an important concern, especially for safety-critical applications. This work presents novel techniques to improve the fault tolerance of NTTs used in prominent PQC schemes such as Kyber, Dilithium, and Falcon. The work first establishes a theoretical framework for error detection in NTTs, exploiting the inherent algebraic properties of these transforms. It derives necessary and sufficient conditions for constructing error-detecting vectors that can identify single faults without the need for costly recomputation. For the Dilithium scheme, the work further advances the state-of-the-art by developing the first algorithm capable of detecting up to two maliciously placed faults. The proposed error detection methods are shown to reduce the number of required multiplications by half, leading to significant improvements in computational efficiency compared to existing single error-detecting algorithms. Concrete implementations for Kyber, Dilithium, and Falcon demonstrate the practicality and effectiveness of the error-detection scheme.

Keywords: Error Detection · Fault Countermeasures · Lattice-Based Cryptography · Number Theoretic Transform · Post-Quantum Cryptography

1 Introduction

The advent of quantum computing poses a significant challenge to the security of current cryptographic systems. Traditional public-key cryptography, which underpins much of today's digital security infrastructure, relies on the computational difficulty of problems such as integer factorization and the discrete

The work of M. Abdelmonem, L. Holzbaur and A. Zeh was supported by the project PARFAIT funded by the BMBF. The majority of the work was done while M. Abdelmonem was with Infineon Technologies AG.

logarithm problem. Algorithms such as RSA are fundamental to secure communications but are vulnerable to quantum attacks by Shor's algorithm [18], which can solve these problems in polynomial time on a sufficiently large quantum computer. In particular, long-lived products such as automotive microcontrollers, as well as sensitive data with long shelf life, need to be secured now against the future threat of quantum computers.

This threat has catalyzed the development of Post-Quantum Cryptography (PQC), which aims to establish secure cryptographic protocols in the face of quantum computing capabilities. In 2016, the National Institute of Standards and Technology (NIST) announced a competition to standardize PQC alternatives, and in July 2022, they selected four algorithms. Three of these four algorithms are from the class of lattice-based schemes, i.e., they rely on complex problems over lattices for their security. The lattices in these schemes are represented by elements of a polynomial ring, and arithmetic over this ring, therefore, plays a crucial role in their execution. The Number Theoretic Transform (NTT) is used to speed up this arithmetic, particularly the multiplication of polynomials. This approach is a generalization of the Fast Fourier Transform (FFT), which has long been established in fields such as signal processing. Since a significant part of the computational complexity of these algorithms lies in the NTT, it is a prime candidate for hardware acceleration, and many papers in the literature have proposed such accelerators [4, 7, 14, 13, 15].

While considerable efforts have been made to provide fast and lean NTT accelerators, the issue of fault tolerance has received little attention. The importance of this is described in [16], where the authors identify a critical vulnerability in the NTT, which enables practical key recovery and message recovery attacks on Kyber KEM, as well as existential forgery and verification bypass attacks on the Dilithium signature scheme.

In safety-critical applications, such as those in the industrial sectors, error resilience is an essential feature that the hardware must provide. This can be achieved by recomputation techniques as presented in [17]. On the other hand, these techniques are traditionally price-sensitive, so the additional chip area required for these features should be minimal. However, current approaches that guarantee error detection, such as those introduced by Sarker et al. [17], impose a large area (or latency) overhead as they rely on recomputation.

This paper addresses this gap by investigating and developing new methods for error-resilient NTT computation. By exploiting insights from existing error detection techniques in FFT computation and adapting them to the NTT context, this work aims to improve the fault tolerance of lattice-based algorithms without incurring prohibitive overheads.

1.1 Related Work

In [17], the authors introduced the first error detection architecture for the NTT. Their technique is based on recomputation, which ensures the detection of any number of computational faults. However, it does not address errors during data

loading and storage and significantly increases the computational complexity, requiring $N/2 \log N$ additional multiplications, where N is the ring degree.

Since the hardware architecture of the NTT and the FFT are quite similar as they both rely on a network of so-called butterfly operations, many well-studied techniques used to detect faults in the FFT can be applied to the NTT as well.

In 1988, Jou and Abraham [11] introduced the first algorithm-based fault tolerance (ABFT) scheme, which does not rely on recomputation. Instead, they encode the inputs and decode the outputs of the FFT. Afterwards, they compare these two results. If there is no error, the encoded inputs and decoded outputs should be equal; if there is an error, they should not. The authors use the sum of a normal Discrete Fourier Transform (DFT) and a rotated DFT as the encoder and derive the decoder accordingly. Based on that, the authors in [1] present their similar error detection algorithm on the NTT. It requires only N additional multiplications. Despite this efficiency, it lacks proven error detection, making it less reliable and not able to prevent faults in malicious attacker models.

In [19] and [20], the authors use weighted checksums for encoding and decoding, which means they multiply the outputs by an error-detection vector, multiply the inputs by the FFT of that vector, and compare the results. This method guarantees the detection of single errors with low overhead. The technique presented in [3] for the NTT is similar to that. Here the authors use polynomial evaluation and interpolation to protect the computation of the NTT against fault injection attacks. It requires the execution of $2N - 1$ additional multiplications, guarantees the detection of at least one fault, and offers the potential for probabilistic detection of additional faults if these faults are to occur randomly. The method described in [3] can be seen as a special case of the method presented in this work for single error detection. However, it requires twice the number of multiplications and is only introduced in the case of a single error, whereas our method can be extended to detect multiple errors efficiently.

In [10], the authors provide methods for the protection of arithmetic operations in lattice-based cryptography, including the NTT, against side-channel and fault attacks. They do this by using the redundant number representation (RNR) described in [21], which introduces redundancy by expanding the modulus q. Its effectiveness and cost depend heavily on the hardware architecture. Unlike the probabilistic detection of this method, our approach guarantees the detection of up to two faults.

Table 1: Comparison of different NTT Error Detection Techniques.

Method	**Error Detection**	**Mult**
[17]	Calculation errors guaranteed, load/store errors not	$N/2 \log N$
[1]	Probabilistic	N
[3]	One guaranteed	$2N$
This work	One guaranteed	N
	Two guaranteed	$2.5N$

1.2 Contribution

Our error detection technique extends concepts used in [19] and [20] for the FFT to the NTT. This paper shows that single error-detection vectors exist for NTT with complete splitting, as used in Dilithium and Falcon, and NTT without complete splitting, as used in Kyber. We have also provided a concrete choice for these error-detecting vectors, which are more general than the one presented in [3]. Our choice of error-detecting vectors also requires fewer multiplications, namely N instead of $2N$. Furthermore, our work introduces the necessary and sufficient conditions that guarantee the detection of two errors in Theorem 4. Using this, we introduced the first non-recomputation-based technique that can detect every error that results from injecting faults on two different wires in the NTT network for Dilithium while requiring only $2.5N$ additional multiplications, even if these faults are maliciously placed. This balance of efficiency and reliability makes our method particularly suitable for applications in fault-sensitive environments. Similar to [3], our method is also compatible with masking and shuffling countermeasures. Table 1 provides a comparative analysis of various NTT error detection techniques.

1.3 Notation

In this paper, we use the ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$, where q is a prime number satisfying $q \equiv 1 \mod 2N$ and N being a power of two. The following notation is used throughout this document:

- **Scalars** are denoted by lowercase italic letters, e.g., a, b, c.
- **Vectors** are denoted by bold lowercase letters, e.g., $\mathbf{v}$, $\mathbf{w}$, $\mathbf{x}$.
- **Matrices** are denoted by bold uppercase letters, e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$.
- **Polynomials** are denoted by lowercase roman letters, e.g., $\mathrm{f}(X)$, $\mathrm{g}(X)$, $\mathrm{h}(X)$.

The coefficients in $\mathbb{Z}_q$ of the polynomial $\mathrm{f}(X) = f_0 + f_1 X + \ldots + f_{N-1} X^{N-1} \in R_q$ are $f_0, f_1, \ldots, f_{N-1}$. Additionally, a polynomial $\mathrm{f}(X)$ is assumed to define a vector $\mathbf{f}$ by its coefficients, i.e., $\mathbf{f} = \begin{pmatrix} f_0 & f_1 & \cdots & f_{N-1} \end{pmatrix}$.

2 Preliminaries

2.1 Number Theoretic Transform

The NTT is a powerful technique for polynomial multiplication that exploits the properties of modular arithmetic to achieve efficient computation. The essence of the NTT is to decompose the multiplication task into operations on lower-degree polynomials, ideally scalars, and then combine these results to obtain the final product.

Let $R_q = \mathbb{Z}_q[X]/(X^N + 1)$, as defined in Section 1.3. Then the negative cyclic NTT (which is commonly used in lattice-based algorithms since it can be

implemented efficiently) of a polynomial $\mathrm{f} \in R_q$ is defined by its evaluation at the odd powers of the $2N$th root of unity ω_{2N}

$$\begin{aligned} \mathsf{NTT}(\mathrm{f}) &:= \left(\hat{f}_0\ \hat{f}_1\ \hat{f}_2 \cdots\ \hat{f}_{N-1}\right) && (1)\\ &:= \left(\mathrm{f}(\omega_{2N}^1)\ \mathrm{f}(\omega_{2N}^3)\ \mathrm{f}(\omega_{2N}^5) \cdots\ \mathrm{f}(\omega_{2N}^{2N-1})\right). && (2) \end{aligned}$$

This definition allows us to look at the NTT as a linear transform similar to the DFT, where

$$\hat{f}_j = \sum_{i=0}^{N-1} \omega_{2N}^{(2j+1)i} f_i. \tag{3}$$

Given the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & \omega_{2N}^1 & \omega_{2N}^2 & \dots & \omega_{2N}^{N-1} \\ 1 & \omega_{2N}^3 & \omega_{2N}^{3\cdot 2} & \dots & \omega_{2N}^{3\cdot(N-1)} \\ 1 & \omega_{2N}^5 & \omega_{2N}^{5\cdot 2} & \dots & \omega_{2N}^{5\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_{2N}^{(2N-1)} & \omega_{2N}^{(2N-1)\cdot 2} & \dots & \omega_{2N}^{(2N-1)\cdot(N-1)} \end{pmatrix} \tag{4}$$

the NTT of a vector $\mathbf{f} \in \mathbb{Z}_q^N$ can written as

$$\hat{\mathbf{f}} = \mathbf{f}\mathbf{A}.$$

The inverse Number Theoretic Transform (INTT) is the interpolation polynomial defined by the evaluation points $(\omega_{2N}^1, \hat{f}_1), (\omega_{2N}^3, \hat{f}_3), \dots, (\omega_{2N}^{2N-1}, \hat{f}_{2N-1})$. It can also be written as a linear transform

$$f_i = \frac{1}{N} \sum_{j=0}^{N-1} \omega_{2N}^{-(2i+1)j} \hat{f}_j. \tag{5}$$

It is well known that the product of two polynomials $\mathrm{h} = \mathrm{f} \cdot \mathrm{g} \in R_q$ can be calculated using the NTT and INTT

$$\mathrm{h} = \mathsf{INTT}\left[\mathsf{NTT}(\mathrm{f}) \circ \mathsf{NTT}(\mathrm{g})\right], \tag{6}$$

where $\circ$ is component-wise multiplication. In addition to that, the NTT can be calculated similarly to the FFT with a Cooley-Tukey algorithm [6], making it run in quasi-linear time. This means that the product can be calculated in time $\mathcal{O}(N \log N)$ instead of $\mathcal{O}(N^2)$.

This is achieved by using the ring isomorphism

$$\begin{aligned} &\mathbb{Z}_q[X]/\left(X^{\frac{N}{2^l}} - \omega_{2N}^{\frac{N}{2^l}}\right) \longrightarrow \mathbb{Z}_q[X]/\left(X^{\frac{N}{2^{l+1}}} - \omega_{2N}^{\frac{N}{2^{l+1}}}\right) \times \mathbb{Z}_q[X]/\left(X^{\frac{N}{2^{l+1}}} + \omega_{2N}^{\frac{N}{2^{l+1}}}\right), \\ &\mathrm{p}(X) \mapsto \left[\mathrm{p}(X) \mod \left(X^{\frac{N}{2^{l+1}}} - \omega_{2N}^{\frac{N}{2^{l+1}}}\right), \mathrm{p}(X) \mod \left(X^{\frac{N}{2^{l+1}}} + \omega_{2N}^{\frac{N}{2^{l+1}}}\right)\right], \end{aligned}$$

for $l = 0, \ldots, \log_2(N) - 1$, since

$$\mathrm{p}(X) \mod X^{N/2^{l+1}} - \omega_{2N}^{N/2^{l+1}} = \sum_{i=0}^{N/2^{l+1}-1} (p_i + \omega_{2N}^{N/2^{l+1}} p_{i+N/2^{l+1}}) X^i \qquad (7)$$

and

$$\mathrm{p}(X) \mod X^{N/2^{l+1}} + \omega_{2N}^{N/2^{l+1}} = \sum_{i=0}^{N/2^{l+1}-1} (p_i - \omega_{2N}^{N/2^{l+1}} p_{i+N/2^{l+1}}) X^i. \qquad (8)$$

hold. Calculations (7) and (8) can be done simultaneously in one single butterfly unit as displayed in Figure 1. This leads to a butterfly network consisting of $\log N$ layers of butterfly operations where the output of one layer is the input to the next, starting with the input vector and ending with the output vector, as shown in Figure 2. Each layer consists of $N/2$ butterfly units, so to calculate a complete NTT $N/2 \log_2 N$ butterfly operations are needed.

Note that this split is possible because ω_{2N} is a $2N$th root of unity. If we are given only an Nth root of unity ω_N, then the last split is impossible. In this case we end up with $N/2$ linear polynomials

$$\mathrm{f}(X) \mod X^2 - \omega_N^{2j+1}, \qquad j = 0, 1, \ldots, N/2 - 1. \qquad (9)$$

Equation (6) can still be applied here, but instead of doing N scalar multiplications, we have to perform $N/2$ multiplications of linear polynomials followed by reductions modulo the quadratic polynomials in (9). This leads to the following well-known result that an incomplete NTT can be calculated with two half-sized complete NTTs.

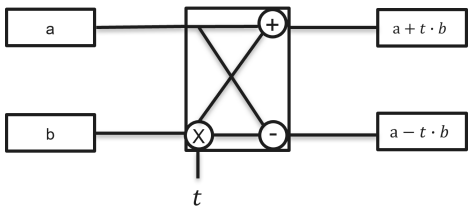

Fig. 1: Cooley-Tukey Butterfly.

Lemma 1. *Let* $\mathrm{f}(X)$ *be an* N*-degree polynomial, and* $\mathrm{f}^{(even)}(X)$ *and* $\mathrm{f}^{(odd)}(X)$ *be* $N/2$*-degree polynomials with the even/odd coefficients of* $\mathrm{f}(X)$

$$\mathrm{f}^{(even)}(X) = f_0 + f_2 X^2 + \cdots + f_{N-2} X^{N-2},$$
$$\mathrm{f}^{(odd)}(X) = f_1 X + f_3 X^3 + \cdots + f_{N-1} X^{N-1}.$$

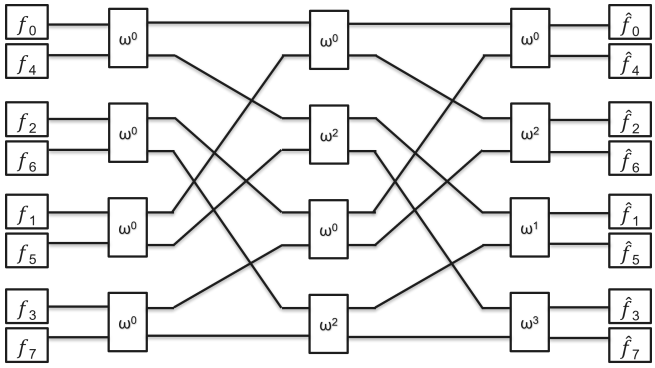

Fig. 2: Cooley-Tukey Butterfly Network for $N = 8$.

Let $\hat{\mathbf{f}}^{(lin)}$ *and* $\hat{\mathbf{f}}^{(const)}$ *be the linear/constant part of the output of the incomplete NTT*

$$\hat{\mathbf{f}} = \hat{\mathbf{f}}^{(lin)} X + \hat{\mathbf{f}}^{(const)}$$

Then using half-sized complete NTTs

$$\hat{\mathbf{f}}^{(const)} = \mathit{NTT}(\mathrm{f}^{(even)}) \quad \textit{and} \quad \hat{\mathbf{f}}^{(lin)} = \mathit{NTT}(\mathrm{f}^{(odd)})$$

holds.

Proof. We have

$$\begin{aligned}
\hat{\mathbf{f}} =& \mathrm{f}(X) \mod X^2 - \omega_{2N}^j \\
=& \sum_{i=0}^{N/2-1} f_i^{(even)} X^{2i} \mod X^2 - \omega_{2N}^j + \sum_{i=0}^{N/2-1} f_i^{(odd)} X^{2i+1} \mod X^2 - \omega_{2N}^j \\
=& \sum_{i=0}^{N/2-1} f_i^{(even)} X^{2i} \mod X^2 - \omega_{2N}^j + X \sum_{i=0}^{N/2-1} f_i^{(odd)} X^{2i} \mod X^2 - \omega_{2N}^j \\
=& \mathsf{NTT}(\mathrm{f}^{(even)}) + X\mathsf{NTT}(\mathrm{f}^{(odd)}).
\end{aligned}$$

□

2.2 NIST Standards

Dilithium [2] is a general-purpose lattice-based digital signature scheme based on the Module Learning with Errors (M-LWE) problem. It has been selected for standardization by NIST under the name ML-DSA [8]. The ring modulus is $q = 8380417$, and the ring degree is $N = 256$. Signatures are generated through several polynomial multiplications, typically arranged in matrix form.

The security levels of Dilithium are determined by the size of these matrices, with larger matrices providing stronger security.

Falcon [9] is another digital signature algorithm NIST will also standardize for scenarios where Dilithium signature sizes may be too large. Falcon uses a ring modulus $q = 12289$ and offers two levels of security: Falcon-I with a ring degree of $N = 512$ and Falcon-V with a ring degree of $N = 1024$. The NTT is used in the key generation, signature generation, and signature verification routines.

Kyber [5] is a lattice-based key encapsulation mechanism based on the M-LWE problem. It has been selected for standardization by NIST under the name ML-KEM [12]. The ring modulus is $q = 3329$, and the ring degree is $N = 256$. In contrast to Dilithium and Falcon, the ring used in Kyber has no $2N$th root of unity but an Nth root of unity. However, as we showed in the previous chapter, this allows us to do an incomplete NTT, which can be done with two NTTs of size $N = 128$.

3 Error Detection

In this section, we introduce our error detection method, beginning with a description of the fault and attacker model. Based on this model, we present our single fault detection technique for complete NTTs and explain its applicability to the Kyber cryptographic scheme. Additionally, we highlight how our method relates to the similar approach presented in [3]. Following this, we describe the first non-recomputation-based technique capable of detecting two errors for Dilithium and conjecture that they also exist for Kyber and Falcon, even under a strong attacker model. Since the only difference between the NTT and the inverse NTT is that ω is replaced by its inverse and an additional scaling factor of $1/N$, all the concepts we will present in the following sections can be adapted to the inverse NTT as well.

3.1 Fault Model

As discussed in Section 2.1, the NTT is computed in multiple layers, each consisting of a fixed number of butterfly operations. These layers are typically computed sequentially, with intermediate results stored and loaded from memory. We start with some assumptions on our fault model, which are the same as in [11, 19, 20]. We assume that additive errors in loading, storing, or calculating a value can occur within a single NTT network butterfly unit, as shown in Figure 3. The errors can occur either due to random hardware faults or error injection by malicious attackers. Errors of type $\{2, 5\}$ can be considered equivalent to errors of type $\{8\}$, and errors of type $\{3, 6\}$ equivalent to errors of type $\{9\}$. Therefore, we only need to consider errors occurring at each butterfly's inputs and outputs. We also distinguish between two attacker models. A weak attacker can only inject faults at random locations, and a strong attacker can control the location and the error value.

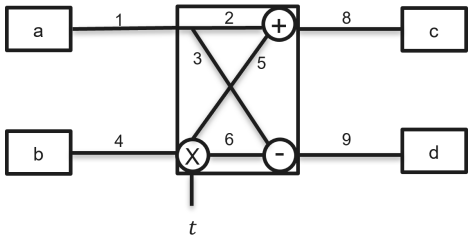

Fig. 3: Illustration of the Fault Model.

An NTT network of dimension N consists of $\log_2 N$ layers. The inputs to the 0th layer are $f_0, \ldots, f_{N-1}$, while $\hat{f}_0, \ldots, \hat{f}_{N-1}$ form the outputs of the $(\log_2 N)$th layer. Each layer contains N wires. For each layer $l = 0, \ldots, \log_2 N$, we can express the wth wire uniquely as

$$w = \frac{N}{2^l}\mu_1 + \mu_2, \tag{10}$$

where $\mu_1 \in \{0, \ldots, 2^l - 1\}$ and $\mu_2 \in \{0, \ldots, {}^N/_{2^l} - 1\}$. This representation is advantageous as it illustrates the propagation of an error through the network until it reaches the output layer, as seen in Figure 4. If an error happens at the line (l, w), it will affect all outputs with index

$$m = 2^l m_1 + m_2 \tag{11}$$

where $m_2 = \mu_1$ and $m_1 \in \{0, \ldots, {}^N/_{2^l} - 1\}$. We now derive the exact value that

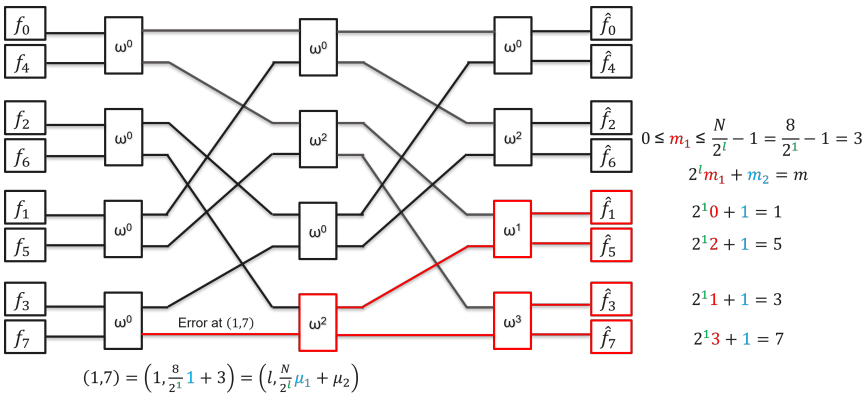

Fig. 4: Error Propagation in the NTT Network. Error occurring in $(l, \frac{N}{2^l}\mu_1 + \mu_2)$ affects all outputs where $m \mod 2^l = \mu_1$.

a propagated error has on the output layer.

Definition 1. *For a given* $(l, w) \in \{0, \ldots, \log N\} \times \{0, \ldots, N-1\}$ *the* ***error function*** $\mathbf{E}^{(l,w)}(e) = [E_0^{(l,w)}(e), \ldots, E_{N-1}^{(l,w)}(e)]$*, is defined for each* $e \in \mathbb{Z}_q$ *by the expression*

$$E_m^{(l,w)}(e) = E_{2^l m_1 + m_2}^{(l, \frac{N}{2^l}\mu_1 + \mu_2)}(e) = \begin{cases} e \sum_{r=0}^{\frac{N}{2^l}-1} \omega_{2N}^{(2(2^l r + \mu_1)+1)\mu_2} & \text{if } m_2 = \mu_1, \\ 0 & \text{if } m_2 \neq \mu_1, \end{cases} \tag{12}$$

where $(l, m_1, m_2, \mu_1, \mu_2)$ *are defined as in* (10) *and* (11)*.*

We proceed to demonstrate that (12) is the error at the output layer after one fault has been injected. We show this by applying the same techniques used in [19] on the FFT.

Theorem 1. *A fault* $e \in \mathbb{Z}_q$ *injected on line* $(l, \frac{N}{2^l}\mu_1 + \mu_2)$ *will cause the error vector* $\mathbf{E}^{(l,w)}(e)$*, i.e.,*

$$\mathit{NTT}_{faulty}(\boldsymbol{f}) = \mathit{NTT}(\boldsymbol{f}) + \mathbf{E}^{(l,w)}(e). \tag{13}$$

Proof. Consider the mth output of the NTT given by

$$\hat{f}_m = \sum_{n=0}^{N-1} f_n \omega_{2N}^{n\cdot(2m+1)}, \quad \text{for } 0 \leq m < N.$$

Decomposing m and n as

$$m = 2^l m_1 + m_2,$$
$$n = \frac{N}{2^l} n_1 + n_2,$$

with $0 \leq m_2, n_1 < 2^l$ and $0 \leq m_1, n_2 < {}^N\!/_{2^l}$, we can express $\hat{f}_m$ as

$$\begin{aligned}
\hat{f}_m =& \hat{f}_{2^l m_1 + m_2} \\
=& \sum_{n=0}^{N-1} f_n \omega_{2N}^{n(2(2^l m_1 + m_2)+1)} \\
=& \sum_{n_2=0}^{\frac{N}{2^l}-1} \sum_{n_1=0}^{2^l-1} f_{\frac{N}{2^l}n_1 + n_2} \omega_{2N}^{(2(2^l m_1 + m_2)+1)(\frac{N}{2^l}n_1 + n_2)} \\
=& \sum_{n_2=0}^{\frac{N}{2^l}-1} \left[\sum_{n_1=0}^{2^l-1} f_{\frac{N}{2^l}n_1 + n_2} \omega_{2N}^{(2(2^l m_1 + m_2)+1)(\frac{N}{2^l}n_1)} \right] \omega_{2N}^{(2(2^l m_1 + m_2)+1)n_2} \\
=& \sum_{n_2=0}^{\frac{N}{2^l}-1} \underbrace{\left[\sum_{n_1=0}^{2^l-1} f_{\frac{N}{2^l}n_1 + n_2} \omega_{2N}^{(2m_2+1)\frac{N}{2^l}n_1} \right]}_{\frac{N}{2^l}m_2 + n_2 \text{ output of the } l\text{th stage}} \omega_{2N}^{(2(2^l m_1 + m_2)+1)n_2}.
\end{aligned}$$

Let $\hat{\mathbf{f}}'$ be the NTT output with an error $e \in \mathbb{Z}_q$ in line $(l, \frac{N}{2^l}\mu_1 + \mu_2)$. Then it will affect only the outputs whose indices have the form $2^l m_1 + \mu_1$. This gives

$$\hat{f}'_{2^l m_1 + m_2} = \hat{f}_{2^l m_1 + m_2},$$

when $\mu_1 \neq m_2$ and when $\mu_1 = m_2$ holds:

$$\begin{aligned}
\hat{f}'_{2^l m_1 + m_2} = &\sum_{n_2=0}^{\frac{N}{2^l}-1} \left[\sum_{n_1=0}^{2^l-1} f_{\frac{N}{2^l} n_1 + n_2} \omega_{2N}^{(2m_2+1)\frac{N}{2^l} n_1} + \right] \omega_{2N}^{(2(2^l m_1 + m_2)+1) n_2} \\
&+ e \sum_{r=0}^{\frac{N}{2^l}-1} \omega_{2N}^{(2(2^l r + \mu_1)+1)\mu_2} \\
= &\hat{f}_{2^l m_1 + m_2} + e \sum_{r=0}^{\frac{N}{2^l}-1} \omega_{2N}^{(2(2^l r + \mu_1)+1)\mu_2} \\
= &\hat{f}_m + E_m^{(l,w)}(e).
\end{aligned}$$

□

It should be noted that in the context of our fault model, a single fault is defined as a single faulty butterfly unit. An alternative attack vector could be to falsify the value of a single twiddle factor, which would be loaded on multiple butterflies, or a zeroization of all twiddle factors, using a single targeted attack, as demonstrated in [16]. In our fault model, this would result in multiple faulty butterflies. We will show in Section 3.3 that, in this case, errors will be detected with $1 - 1/q$ probability.

3.2 Single Error Detection

Error detection is achieved by encoding the NTT's input, decoding the output, and then comparing the two results. We need to show that if no error has occurred, the encoded input and decoded output will match; conversely, if a single error has occurred, they will not.

Let the output of the NTT be denoted by the vector $\hat{\mathbf{f}} = \left(\hat{f}_0\ \hat{f}_1 \cdots\ \hat{f}_{N-1}\right)^T$, and let the error detecting vector be $\mathbf{a} = \left(a_0\ a_1 \cdots\ a_{N-1}\right)$. We compute the error detection value over the output as

$$C_{\text{out}} := \mathbf{a}\hat{\mathbf{f}}. \tag{14}$$

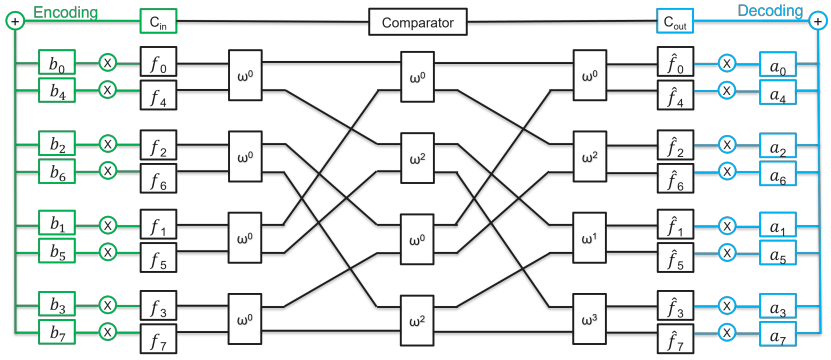

Fig. 5: Proposed error detection technique in which the inputs and outputs of the NTT are pointwise multiplied by vectors **b** and **a**, respectively. The resulting sums are then compared.

To determine the error detection value for the input, we substitute the definition of the NTT for $\hat{\mathbf{f}}$

$$C_{\text{out}} = \underbrace{\begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \end{pmatrix} \begin{pmatrix} 1 & \omega_{2N}^{(2\cdot 0+1)\cdot 1} & \cdots & \omega_{2N}^{(2\cdot 0+1)\cdot (N-1)} \\ 1 & \omega_{2N}^{(2\cdot 1+1)\cdot 1} & \cdots & \omega_{2N}^{(2\cdot 1+1)\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_{2N}^{(2\cdot (N-1)+1)\cdot 1} & \cdots & \omega_{2N}^{(2\cdot (N-1)+1)\cdot (N-1)} \end{pmatrix}}_{=:\begin{pmatrix} b_0 & b_1 & \cdots & b_{N-1} \end{pmatrix}} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix}. \tag{15}$$

We define the input to the comparator as:

$$C_{\text{in}} := \mathbf{bf}. \tag{16}$$

This way, we guarantee that if no error occurs, $C_{\text{in}} = C_{\text{out}}$ holds. The question is whether $C_{\text{in}} \neq C_{\text{out}}$ holds for any **a** when an error occurs. In the following theorem, we provide necessary and sufficient conditions on the choice of **a** so that no single error can go undetected.

Theorem 2. *An error $e \in \mathbb{Z}_q \backslash \{0\}$ occurring at line $(l, \frac{N}{2^l}\mu_1 + \mu_2) \in \{0, \ldots, \log_2 N\} \times \{0, \ldots, N-1\}$ is detectable with a vector $\mathbf{a} = (a_0, \ldots, a_{N-1}) \in \mathbb{Z}_q^N$ if and only if the sum $S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\boldsymbol{a})$ defined by*

$$S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\boldsymbol{a}) := \sum_{r=0}^{\frac{N}{2^l}-1} \omega_{2N}^{(2(2^l r+\mu_1)+1)\mu_2} a_{2^l r+\mu_1} \tag{17}$$

is nonzero.

Proof. From Theorem 1 we know that $\hat{f}'_{2^l m_1+m_2}$, the mth output the faulty NTT with $m = 2^l m_1 + m_2$ defined as in (11), is given by

$$\hat{f}'_m = \hat{f}_m + E_m^{(l,w)}(e).$$

Applying the definitions of C_{out}, C_{in} and taking their difference, we obtain

$$\begin{aligned} C_{out} &= \sum_{m=0}^{N-1} \hat{f}'_m a_m \\ &= \sum_{m=0}^{N-1} \hat{f}_m a_m + E_m^{(l,w)}(e) a_m \\ &= \sum_{m=0}^{N-1} \hat{f}_m a_m + e \sum_{r=0}^{\frac{N}{2^l}-1} \omega_{2N}^{(2(2^l r+\mu_1)+1)\mu_2} a_{2^l r+\mu_1} \\ &= \sum_{m=0}^{N-1} \hat{f}_m a_m + e \cdot S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\mathbf{a}) \end{aligned}$$

and

$$\begin{aligned} C_{in} &= \sum_{m=0}^{N-1} f_m b_m \\ &= \sum_{m=0}^{N-1} \hat{f}_m a_m. \end{aligned}$$

Then we obtain:

$$C_{out} - C_{in} = e \cdot S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\mathbf{a}).$$

Therefore, an error is detectable if and only if this difference is nonzero, which is equivalent to the condition

$$S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\mathbf{a}) \neq 0.$$

□

The subsequent theorem provides conditions under which an error-detecting vector exists and describes a recursive method for constructing such a vector.

Theorem 3. *If the inequality $q > 2N - 1$ holds, then there exists at least one single error-detecting vector.*

Proof. We describe a constructive method for building such a vector that always works as long as $q > 2N - 1$. The idea is to choose the first term in every

$S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\mathbf{a})$ such that the sum is nonzero. This will give us a number of restrictions on each element a_i. These restrictions are kept track of with sets $\mathbb{B}_i$ that will contain values that a_i can not take. In the end, we can simply pick values for a_i that avoid $\mathbb{B}_i$ for $0 \leq i \leq N-1$. We explain the process in more detail.

Starting with $l = \log_2 N$ and hence $\mu_2 = 0$, we have N sums (one for each $\mu_1 \in \{0, \ldots, N-1\}$), consisting of only one term

$$S^{\log_2 N}_{\mu_1}(\mathbf{a}) = \omega^0_{2N} a_{\mu_1}.$$

By choosing all the elements of $\mathbf{a}$ to be nonzero, $S^{\log_2 N}_{\mu_1}(\mathbf{a})$ will be nonzero, hence we initialize all the sets $\mathbb{B}_{\mu_1} = \{0\}$ for $0 \leq \mu_1 \leq N-1$.

In the next step, $l = \log_2 N - 1$, we also have $N-1$ sums, but each sum has two terms, where the first term is always a_{μ_1}, with $\mu_1 \in \{0, \ldots, N/2 - 1\}$

$$S^{\log_2 N-1}_{2\mu_1+\mu_2}(\mathbf{a}) = \omega^{(2\mu_1+1)\mu_2}_{2N} a_{\mu_1} + \omega^{(2(\frac{N}{2}+\mu_1)+1)\mu_2}_{2N} a_{\frac{N}{2}+\mu_1}.$$

We now fix a_i for $\frac{N}{2} \leq i \leq N-1$, to any value not in $\mathbb{B}_i$, i.e., anything nonzero. For the remaining indices, we now get two additional constraints

$$a_{\mu_1} \neq -\frac{\omega^{(2(2^l+\mu_1)+1)\mu_2}_{2N} a_{\frac{N}{2}+\mu_2}}{\omega^{(2\mu_1+1)\mu_2}_{2N}},$$

for $\mu_2 = 0, 1$. These two values are added to $\mathbb{B}_{\mu_1}$ for $0 \leq \mu_1 \leq N/2 - 1$. This process can be continued with $l = \log_2 N - 2$, which will fix a_i for $N/4 \leq i \leq N/2 - 1$ and add four values to $\mathbb{B}_{\mu_1}$ for $0 \leq \mu_1 \leq N/4 - 1$.

Finally, for $l = 0$ (and hence $\mu_1 = 0$), we get N forbidden values added to $\mathbb{B}_0$, in addition to the $\sum_{l=1}^{\log_2 N} \frac{N}{2^l}$ values that were added in previous stages, and all other a_i's have been fixed. All the values a_0 must avoid might be different, so the maximum size of $\mathbb{B}_0$ is

$$\sum_{l=0}^{\log_2(N)} \frac{N}{2^l} = 2N - 1, \tag{18}$$

and all other $\mathbb{B}_i$ will have smaller maximum sizes. So when $q > 2N - 1$, we are guaranteed that it will always be possible to choose $a_i \in \mathbb{Z}_q \setminus EX_i$ for all $i = 0, 1, \ldots, N-1$. □

Theorem 2 proves that if the conditions are fulfilled, suitable error detection vectors exist, thereby showing that it is possible to guarantee error detection for a wide selection of parameters. In practice, other vectors might be more suitable for implementation. In particular, for Kyber, Dilithium, and Falcon, it suffices to set $\mathbf{b} = \begin{pmatrix} 1 & 1 & \ldots & 1 \end{pmatrix}$ and $\mathbf{a} = \mathbf{b}\mathbf{A}^{-1}$, because in this case $S^l_{\frac{N}{2^l}\mu_1+\mu_2}(\mathbf{a}) \neq 0$ holds for all l, mu_1, μ_2, as we will demonstrate in Section 4.1.

Application to Kyber Using Lemma 1, we can apply our error detection method on Kyber as well by simply checking the two half-sized NTTs. That means our error-detecting vector has length $N/2$ but has to be used twice.

Error Detection through Evaluation and Interpolation In [3], the authors have derived an error detection technique based on polynomial evaluation and interpolation. This work has been done independently and during the same time as ours. The advantage of our scheme, aside from the reduction in the number of multiplications required, is that the general description of the method allows for an efficient extension to more than single errors, as shown in the next section. This technique can be seen as a special case of ours because the evaluation of the polynomial $\mathrm{f} \in R_q$ at $u \in \mathbb{Z}_q$ is simply a checksum over the input coefficients of the NTT

$$w := \mathrm{f}(u) = \sum_{i=0}^{N-1} f_i u^i. \tag{19}$$

Using Lagrangian interpolation, they do another checksum over the outputs of the NTT

$$w' := \mathrm{f}(u) = \sum_{i=0}^{N-1} \hat{f}_i L_i(u), \tag{20}$$

where $L_i(x)$ is the ith Lagrange polynomial.

By setting

$$\mathbf{b} = \begin{pmatrix} b_0 & b_1 & \cdots & b_{N-1} \end{pmatrix} = \begin{pmatrix} u^0 & u^1 & \cdots & u^{N-1} \end{pmatrix}, \tag{21}$$

and

$$\mathbf{a} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \end{pmatrix} = \begin{pmatrix} L_0(u) & L_1(u) & \cdots & L_{N-1}(u) \end{pmatrix}, \tag{22}$$

for $i = 0, 1, \ldots, N-1$, we can see that their method is a special case of ours.

3.3 Double Error Detection

Since the NTT is linear, it is easy to see that the output error of two faults in the butterfly network is the same as the sum of the output errors of two single faults. That means if we apply the same error-detecting technique as in Chapter 3, we get

$$C_{out} - C_{in} = S_w^l(\mathbf{a})e_1 + S_\gamma^\lambda(\mathbf{a})e_2,$$

for two errors $e_1, e_2 \in \mathbb{Z}_q \backslash \{0\}$ that occur on the lines (l, w) and (λ, γ). The error will not be detected if

$$e_2 = -\frac{S_w^l(\mathbf{a})e_1}{S_\gamma^\lambda(\mathbf{a})}.$$

If e_2 is random and independent from e_1, the probability for that is $1/q$. This probability also holds for the case of more than two errors. Depending on the application, this might be sufficient. However, for settings requiring higher assurance of error detection or considering strong attackers with the ability to select the positions and values of the errors, stronger guarantees are required. If we want to detect any two errors with 100% probability, we must use two checksums, as shown in Figure 6.

In the following theorem, we derive conditions on the two error-detecting vectors $\mathbf{a}$ and $\boldsymbol{\alpha}$ so that two errors are always detected.

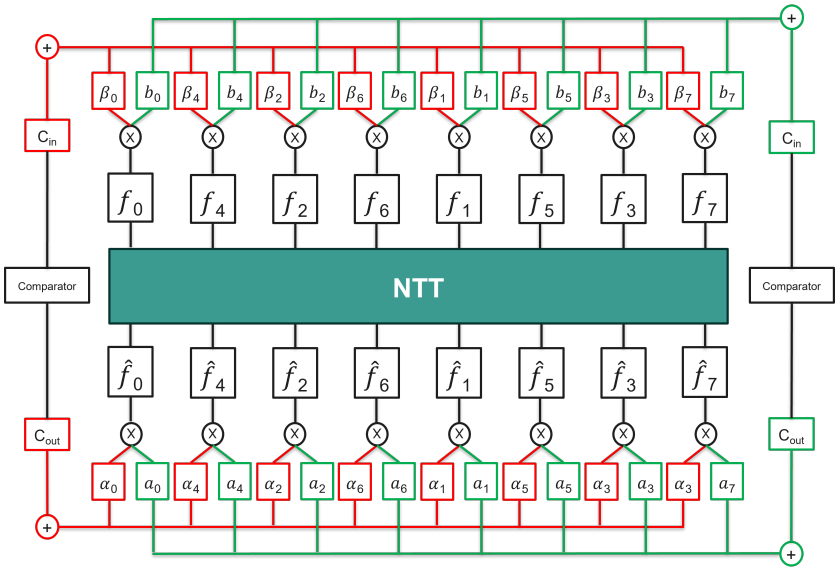

Fig. 6: Proposed error detection method for identifying two errors. The inputs are first pointwise multiplied by vectors $\mathbf{b}$ and $\boldsymbol{\beta}$, and their sums are computed. These sums are then compared to the sum of the pointwise multiplication of the NTT output with vectors $\mathbf{a}$ and $\boldsymbol{\alpha}$.

Theorem 4. *Two errors e_1 and e_2 occurring at lines (l, w) and (λ, γ) are detectable with two vectors $\mathbf{a} = (a_0, \ldots, a_{N-1}), \boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_{N-1}) \in \mathbb{Z}_q^N$ if and only if $S_w^l(\mathbf{a}) \neq 0$ and $S_\gamma^\lambda(\boldsymbol{\alpha}) \neq 0$ and*

$$S_w^l(\boldsymbol{\alpha}) S_\gamma^\lambda(\mathbf{a}) - S_w^l(\mathbf{a}) S_\gamma^\lambda(\boldsymbol{\alpha}) \neq 0. \tag{23}$$

Proof. If $(l, w) = (\lambda, \gamma)$, then the two errors can be regarded as one single error and will then be detected using only one vector. So, for the rest of the proof, we assume that the lines are distinct.

The errors will be detected if and only if

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \neq \begin{pmatrix} C_{out}^1 - C_{in}^1 \\ C_{out}^2 - C_{in}^2 \end{pmatrix}.$$

We know

$$\begin{aligned} \begin{pmatrix} C_{out}^1 - C_{in}^1 \\ C_{out}^2 - C_{in}^2 \end{pmatrix} &= \begin{pmatrix} S_w^l(\mathbf{a}) e_1 + S_\gamma^\lambda(\mathbf{a}) e_2 \\ S_w^l(\boldsymbol{\alpha}) e_1 + S_\gamma^\lambda(\boldsymbol{\alpha}) e_2 \end{pmatrix} \\ &= \begin{pmatrix} S_w^l(\mathbf{a}) & S_\gamma^\lambda(\mathbf{a}) \\ S_w^l(\boldsymbol{\alpha}) & S_\gamma^\lambda(\boldsymbol{\alpha}) \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}. \end{aligned}$$

Since $(e_1, e_2) \neq (0, 0)$ we know that

$$\begin{pmatrix} C^1_{out} - C^1_{in} \\ C^2_{out} - C^2_{in} \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

if and only if

$$\begin{aligned} &\begin{vmatrix} S^l_w(\mathbf{a}) & S^\lambda_\gamma(\mathbf{a}) \\ S^l_w(\boldsymbol{\alpha}) & S^\lambda_\gamma(\boldsymbol{\alpha}) \end{vmatrix} \\ =&S^l_w(\boldsymbol{\alpha}) S^\lambda_\gamma(\mathbf{a}) - S^l_w(\mathbf{a}) S^\lambda_\gamma(\boldsymbol{\alpha}) \neq 0. \end{aligned}$$

□

Let

$$\begin{aligned} \mathbf{S} &:= \left(S^0_0(\mathbf{a})\ S^0_1(\mathbf{a}) \cdots S^0_{N-1}(\mathbf{a})\ S^1_0(\mathbf{a}) \cdots\ S^1_{N-1}(\mathbf{a}) \cdots\ S^{\log_2 N}_0(\mathbf{a}) \cdots\ S^{\log_2 N}_{N-1}(\mathbf{a}) \right), \\ \boldsymbol{\sigma} &:= \left(S^0_0(\boldsymbol{\alpha})\ S^0_1(\boldsymbol{\alpha}) \cdots S^0_{N-1}(\boldsymbol{\alpha})\ S^1_0(\boldsymbol{\alpha}) \cdots\ S^1_{N-1}(\boldsymbol{\alpha}) \cdots\ S^{\log_2 N}_0(\boldsymbol{\alpha}) \cdots\ S^{\log_2 N}_{N-1}(\boldsymbol{\alpha}) \right), \end{aligned}$$

and

$$\mathbf{D} := (D(\mathbf{S}, \boldsymbol{\sigma})_{ij}) := (\mathbf{S}_i \boldsymbol{\sigma}_j - \mathbf{S}_j \boldsymbol{\sigma}_i) \in \mathbb{Z}_q^{N(\log_2 N+1) \times N(\log_2 N+1)}.$$

Now, (23) translates to $D(\mathbf{S}, \boldsymbol{\sigma})_{ij} \neq 0$ for $i \neq j$. Note that $\mathbf{D}$ is antisymmetric, meaning $D(\mathbf{S}, \boldsymbol{\sigma})_{ij} = -D(\mathbf{S}, \boldsymbol{\sigma})_{ji}$. That means

$$\binom{N(\log_2 N + 1)}{2} = \frac{(N(\log_2 N + 1) - 1) N(\log_2 N + 1)}{2}$$

inequalities have to be satisfied. Each of these conditions that are satisfied corresponds to a pair of wires in the NTT network and assures that any possible fault injection in these two wires will always be detected.

Similar to Theorem 3, we present here a sufficient condition that guarantees the existence of an error-detecting vector pair $\mathbf{a}$ and $\boldsymbol{\alpha}$ that satisfies the conditions in Theorem 4.

Theorem 5. *When $q > (2N-1)N(\log_2 N + 1)$ there exists at least one pair $(\boldsymbol{a}, \boldsymbol{\alpha})$ such that $\boldsymbol{a}$ and $\boldsymbol{\alpha}$ will detect any two errors in the NTT computation.*

Proof. Let $s = |\mathbf{S}| = |\boldsymbol{\sigma}|$ be the number of values in $\mathbf{S}$ and $\boldsymbol{\sigma}$, that is

$$s = N(\log_2 N + 1).$$

The total number of distinct determinants given by pairs of these values is given by:

$$\binom{s}{2} = \frac{s(s-1)}{2}.$$

These represent the number of distinct wire pairs to be tested for the determinant condition.

Now, fix $\boldsymbol{\alpha}$ as done in the proof of Theorem 3. Fixing $\boldsymbol{\alpha}$ will also determine all values in $\boldsymbol{\sigma}$. We proceed to select the values in $\mathbf{a}$, making sure to avoid the banned values in $\mathbb{B}_i$ for each i. When fixing the a_i's as explained in the proof of Theorem 3, the values in $\mathbf{S}$ also become fixed. To satisfy the determinant condition we must fix the a_i such that all determinants $\mathbf{S}_i\boldsymbol{\sigma}_j - \mathbf{S}_j\boldsymbol{\sigma}_i$ are also non-zero. Therefore, values for a_i that would make any determinant zero must also be added to $\mathbb{B}_i$.

As in the proof of Theorem 3, the last value to fix is a_0, and this is the value that must meet the largest number of constraints. So, in the following, we focus on selecting a_0. Among the s sums in $\mathbf{S}$, there are $2N-1$ sums that start with a_0, as established in Theorem 3. The values for each of these $2N-1$ sums form determinants with the $s-1$ values in $\boldsymbol{\sigma}$, so the number of determinants that include a_0 is $(2N-1)(s-1)$. In the worst case, this leads to the same number of unique values to be added to $\mathbb{B}_0$. From the proof of Theorem 3, we know that there are up to $2N-1$ additional values in $\mathbb{B}_0$ to avoid that, make sure each $\mathbf{S}_i$ is non-zero. So in total we have

$$|\mathbb{B}_0| \leq (2N-1)(s-1) + (2N-1) = (2N-1)N(\log_2 N + 1).$$

As long as q is larger than this bound, we can always select a value for a_0 that avoids making any $\mathbf{S}_i$ or any determinant 0. □

In Section 4.2, we will demonstrate that the bound in this theorem is not sharp enough to construct error-detecting vectors for Kyber and Falcon, because the ring modulus q is too small for these schemes. For Dilithium, we will use this construction to give a pair of error-detecting vectors that can detect up to two errors.

4 Implementation

This section introduces effective implementations for our error detection techniques. An NTT without error detection requires $\frac{N \log_2 N}{2}$ multiplications. In order to apply our error detection method with one checksum, we have to multiply the inputs of the NTT with $\mathbf{b}$ and the outputs with $\mathbf{a}$, which both require N multiplication. So we would need $2N$ extra multiplications. If we want to use two checksums, we would need $4N$ extra multiplications. However, by choosing as many multipliers equal to one as possible, we can demonstrate that we only need N extra multiplication for detecting a single error and $2.5N$ extra multiplication for detecting two errors.

4.1 One Checksum

Let $\mathbf{b} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$, we can show that the corresponding $\mathbf{a}$ can always detect one fault by calculating all $S_w^l(\mathbf{a})$ and confirming they are all nonzero for Kyber, Dilithium, and Falcon parameters. The exact vectors for guaranteed single error detection are given in the appendix.

4.2 Two Checksums

For our two error detection methods, we use the same $\mathbf{a}$ and choose

$$\alpha_{N-1} = \alpha_{N-2} = \cdots = \alpha_{\frac{N}{2}-1} = 1.$$

We choose the remaining α_i recursively as described in the proof of Theorem 5, so that $D(\mathbf{S}, \boldsymbol{\sigma})_{ij}$ is nonzero for all $i \neq j$. After that, we compute $\boldsymbol{\beta} := \boldsymbol{\alpha}\mathbf{A}$. Since half of the elements of $\boldsymbol{\alpha}$ are one, we only need $1.5N$ multiplications more compared to the one checksum case, meaning in total, we need $2.5N$ extra multiplications. The condition in Theorem 5 is satisfied for Dilithium. We provided $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for Dilithium in the appendix as well.

For Kyber and Falcon, it is not feasible to find values for low indices of $\boldsymbol{\alpha}$ using this method because this condition is not met. That means we can only guarantee two fault detection on a certain percentage of all wire pairs. As long as this percentage is not a 100%, meaning there is at least one wire pair where errors can go undetected, a strong attacker can compute $\mathbf{S}$ and $\boldsymbol{\sigma}$ and thereby also $\mathbf{D}$ and thus find the wire pair where errors can go undetected and exploit this by injecting the two faults there. The fact that the Kyber and Falcon parameters do not meet the condition in Theorem Theorem 5 does not exclude that a two error-detecting vector pair exists. In Section 3.3 we have shown that

$$\binom{N(\log_2 N + 1)}{2}$$

elements must be nonzero. Choosing $\mathbf{a}$ and $\boldsymbol{\alpha}$ randomly and assuming each $D(\mathbf{S}, \boldsymbol{\sigma})_{ij}$ is random in $\mathbb{Z}_q$, the probability, that they are all nonzero is

$$\left(1 - \frac{1}{q}\right)^{\binom{N(\log_2 N+1)}{2}}.$$

Since $\mathbf{a}, \boldsymbol{\alpha} \in \mathbb{Z}_q^N$, there are q^N possible values for each, so the expected number of vectors satisfying all the non-zero conditions is

$$q^N \left(1 - \frac{1}{q}\right)^{\binom{N(\log_2 N+1)}{2}} \gg 1.$$

Hence, we conjecture that two error-detecting pairs of vectors exist. We leave proving/disproving the existence of such a pair and finding it for the Falcon and Kyber parameters as an open problem. We have shown that N additional multiplications are required for one checksum, while $2.5N$ are needed for two checksums. A complete NTT (used in Dilithium and Falcon) requires $\frac{N \log_2 N}{2}$ multiplications, and an incomplete NTT (used in Kyber) requires $\frac{N(\log_2 N - 1)}{2}$ since the last layer is omitted. The cost of applying our error detection methods, using one and two checksums (CS), relative to the cost of the NTT in terms of multiplications is given in Table 2. The numbers are computed as the additional number of multiplications required for the checksums and dividing it by the

Table 2: Relative cost of our error detection techniques in terms of multiplications.

Scheme	**1 CS**	**2 CS**
complete NTT	$\frac{2}{\log_2 N}$	$\frac{5}{\log_2 N}$
Incomplete NTT	$\frac{2}{\log_2 N-1}$	$\frac{5}{\log_2 N-1}$

Table 3: Relative cost of our error detection techniques in terms of multiplications for Kyber, Dilithium and Falcon.

Scheme	**1 CS**	**2 CS**
Dilithium	0.25	0.625
Falcon I	0.222	0.556
Falcon V	0.2	0.5
Kyber	0.286	0.714

number of multiplications needed for a complete or incomplete NTT. Now, setting N to the numbers presented in Section 2.2 results in the numbers presented in Table 3.

Our proposed technique using one checksum reduces the number of required multiplications compared to both [17] and [3] methods while still guaranteeing the detection of one error. This highlights the efficiency of the approach, especially for larger values of N. Note that unlike in the recomputation method, additional storage is needed for the coefficients in our method and in [3]. We need to store N coefficients for Dilithium and Falcon, whereas we only need $\frac{N}{2}$ coefficients for Kyber because the same coefficients can be used for each half-sized NTT performed for Kyber.

5 Conclusion

We have introduced a generalized error detection technique that improves the security of implementations using the NTT. Our approach is capable of detecting a single fault injected into a wire in the NTT network, requiring only N additional multiplications, significantly reducing the overhead compared to previous methods. We have further extended our technique to detect up to two faults with 100% reliability for Dilithium. This extended capability is achieved with an overhead of only $2.5N$ multiplications.

We leave finding an efficient algorithm that gives us $\mathbf{a}$ and $\boldsymbol{\alpha}$ that guarantee the finding of two faults as an open problem. Note that this technique can be extended to detect more than two errors by adding more checksums. However, this is not practical as the additional cost is close to the cost of recomputation.

References

1. Ahmadi, K., Aghapour, S., Kermani, M.M., Azarderakhsh, R.: Efficient Algorithm Level Error Detection for Number-Theoretic Transform used for Kyber Assessed on FPGAs and ARM (2024), https://arxiv.org/abs/2403.01215
2. Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Supporting documentation: Crystals-dilithium: Algorithm specifications and supporting documentation. NIST PQC (2020)
3. Bauer, S., Santis, F.D., Koleci, K., Aghaie, A.: A fault-resistant NTT by polynomial evaluation and interpolation. Cryptology ePrint Archive, Paper 2024/788 (2024), https://eprint.iacr.org/2024/788
4. Bisheh-Niasar, M., Azarderakhsh, R., Mozaffari-Kermani, M.: High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography. In: 2021 IEEE 28th symposium on computer arithmetic (ARITH). pp. 94–101 (2021)
5. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018)
6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Mathematics of computation 19(90), 297–301 (1965)
7. Derya, K., Mert, A.C., Öztürk, E., Savaş, E.: CoHA-NTT: A configurable hardware accelerator for NTT-based polynomial multiplication. Microprocessors and Microsystems 89, 104451 (2022)
8. Digital-Signature, N.M.L.B.: Mechanism standard. NIST Post-Quantum Cryptography Standardization Process; NIST: Gaithersburg, MD, USA (2024)
9. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z., et al.: Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NIST's post-quantum cryptography standardization process 36(5), 1–75 (2018)
10. Heinz, D., Pöppelmann, T.: Combined fault and DPA protection for lattice-based cryptography. IEEE Transactions on Computers 72(4), 1055–1066 (2022)
11. Jou, J.Y., Abraham, J.A.: Fault-tolerant FFT networks. IEEE Transactions on Computers 37(5), 548–561 (1988)
12. Key-Encapsulation, N.M.L.B.: Mechanism standard. NIST Post-Quantum Cryptography Standardization Process; NIST: Gaithersburg, MD, USA (2024)
13. Li, D., Pakala, A., Yang, K.: MeNTT: A compact and efficient processing-in-memory number theoretic transform (NTT) accelerator. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 30(5), 579–588 (2022)
14. Nejatollahi, H., Cammarota, R., Dutt, N.: Flexible NTT accelerators for RLWE lattice-based cryptography. In: 2019 IEEE 37th International Conference on Computer Design (ICCD). pp. 329–332 (2019)
15. Nguyen, T.H., Kieu-Do-Nguyen, B., Pham, C.K., Hoang, T.T.: High-speed NTT Accelerator for CRYSTAL-Kyber and CRYSTAL-Dilithium. IEEE Access (2024)
16. Ravi, P., Yang, B., Bhasin, S., Zhang, F., Chattopadhyay, A.: Fiddling the twiddle constants-fault injection analysis of the number theoretic transform. IACR Transactions on Cryptographic Hardware and Embedded Systems (2023)
17. Sarker, A., Canto, A.C., Kermani, M.M., Azarderakhsh, R.: Error detection architectures for hardware/software co-design approaches of number-theoretic transform. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 42(7), 2418–2422 (2022)

18. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41(2), 303–332 (1999)
19. Tao, D.L., Hartmann, C.R.: A novel concurrent error detection scheme for FFT networks. IEEE Transactions on Parallel and Distributed Systems 4(2), 198–221 (1993)
20. Wang, S.J., Jha, N.K.: Algorithm-based fault tolerance for FFT networks. IEEE Transactions on Computers 43(7), 849–854 (1994)
21. Zijlstra, T., Bigou, K., Tisserand, A.: FPGA implementation and comparison of protections against SCAs for RLWE. In: Progress in Cryptology–INDOCRYPT 2019. pp. 535–555 (2019)

A Input and Output Multipliers for Techniques

This appendix presents the input and output multipliers for the four schemes: Dilithium, Kyber, Falcon I, and Falcon V. The multipliers a and α represent the output multipliers, while b and β represent the input multipliers.

A.1 Dilithium

Table 4: Dilithium Multipliers

Multiplier	Values
b	$\{1\}^{256}$
a	7511306, 7268830, 5474525, 5631597, 117126, 2894273, 5415375, 6137690, 5412937, 879351, 7630876, 7950048, 2707004, 2223415, 7757689, 6794386, 4413478, 1434037, 6593365, 2371240, 3775900, 2069059, 3050174, 2348902, 885454, 950158, 391593, 800539, 7946762, 7724623, 5651610, 7657810, 2346552, 5905197, 5542002, 7885504, 2266250, 295216, 2670256, 3902893, 3120930, 2207058, 4168676, 7836275, 6401582, 3423687, 8256204, 7633393, 1059210, 2645505, 2868862, 5530426, 2218396, 6414667, 5248249, 5834839, 2638709, 8231049, 856716, 4909726, 317499, 2591320, 566630, 7259347, 6220539, 3522895, 5398834, 120765, 626895, 3309814, 6419699, 6297723, 1949702, 3647406, 993985, 5264941, 5720329, 317292, 6501330, 8088966, 2307577, 2816303, 5278157, 3404063, 5743422, 7610229, 6838252, 6520254, 7614068, 3264328, 94014, 3796289, 372831, 1903233, 7842375, 2906811, 5023664, 1466781, 3010429, 3141092, 8212262, 1157548, 2840242, 5561907, 907175, 1740496, 7577840, 3098394, 4536213, 3727033, 7910907, 910323, 6456321, 7599170, 1993767, 3785916, 5655461, 141639, 2230157, 3832335, 1244577, 8243668, 5047276, 8140557, 7668172, 6349172, 1865577, 458640, 7856305, 6449368, 1965773, 646773, 174388, 3267669, 71277, 7070368, 4482610, 6084788, 8173306, 2659484, 4529029, 6321178, 715775, 1858624, 7404622, 404038, 4587912, 3778732, 5216551, 737105, 6574449, 7407770, 2753038, 5474703, 7157397, 102683, 5173853, 5304516, 6848164, 3291281, 5408134, 472570, 6411712, 7942114, 4518656, 8220931, 5050617, 700877, 1794691, 1476693, 704716, 2571523, 4910882, 3036788, 5498642, 6007368, 225979, 1813615, 7997653, 2594616, 3050004, 7320960, 4667539, 6365243, 2017222, 1895246, 5005131, 7688050, 8194180, 2916111, 4792050, 2094406, 1055598, 7748315, 5723625, 7997446, 3405219, 7458229, 83896, 5676236, 2480106, 3066696, 1900278, 6096549, 2784519, 5446083, 5669440, 7255735, 681552, 58741, 4891258, 1913363, 478670, 4146269, 6107887, 5194015, 4412052, 5644689, 8019729, 6048695, 429441, 2772943, 2409748, 5968393, 657135, 2663335, 590322, 368183, 7514406, 7923352, 7364787, 7429491, 5966043, 5264771, 6245886, 4539045, 5943705, 1721580, 6880908, 3901467, 1520559, 557256, 6091530, 5607941, 364897, 684069, 7435594, 2902008, 2177255, 2899570, 5420672, 8197819, 2683348, 2840420, 1046115, 803639
$\boldsymbol{\beta}$	5791121, 3876506, 1812116, 6910599, 858961, 6913584, 1646428, 6089601, 1687322, 1692175, 8209040, 6038377, 3135400, 2658623, 5231154, 2825832, 7459415, 1536327, 673249, 8185212, 4466516, 4563857, 1324194, 1484330, 3250522, 6133580, 7011383, 3192114, 4672318, 852807, 734578, 1733239, 1530988, 2107944, 3476597, 4659735, 6931191, 3307246, 931936, 4318748, 3365542, 6899902, 5065856, 4622758, 1397997, 2951382, 754023, 1798099, 3195823, 3497845, 2416746, 6363325, 2797313, 7003747, 950818, 4537049, 3176956, 3716664, 3285715, 266803, 6849160, 863625, 725468, 2606776, 730552, 3686374, 878625, 4948166, 6650739, 7231932, 6848492, 820998, 2281956, 7314383, 58338, 283191, 587775, 5464612, 7304239, 2482172, 6313754, 5290807, 8091398, 2174999, 2840257, 7268978, 8085971, 3356631, 7419232, 1759623, 806177, 7326295, 5529423, 6049846, 6216333, 2683516, 4465040, 7884559, 3614972, 4621250, 3674006, 8096164, 4710879, 8243928, 959352, 2153760, 4194460, 2974857, 3417856, 7195563, 2452187, 6961339, 1097918, 4887793, 1147063, 6037604, 2152775, 6064359, 5454959, 7637950, 1036440, 3903066, 1624500, 2136616, 2488859, 3153127, 3709896, 5919480, 6466532, 4260647, 7762914, 8168859, 236105, 3298174, 447552, 3583058, 738963, 4792049, 6900868, 758349, 3272429, 3782168, 6597079, 2273128, 4607808, 1627153, 5103035, 7052454, 1405956, 3874393, 6113179, 4036651, 8149804, 4536498, 5262283, 456865, 7206170, 6825363, 8262546, 6872768, 6539836, 2760140, 7695741, 5619060, 3179045, 977927, 26228, 1912650, 4046403, 5715759, 5274874, 1804365, 695873, 4825361, 3914766, 1509198, 153827, 4966095, 1260046, 5380686, 4291999, 5707936, 7547877, 6232755, 1886681, 8174088, 7939977, 7936308, 2490571, 1780044, 5225834, 1798941, 6229697, 5730199, 2872745, 6764240, 1587611, 3918125, 2630225, 2908370, 739846, 1155391, 460911, 5297629, 2123787, 7670502, 1321504, 6147571, 6271373, 1671981, 2956561, 6354100, 185111, 151819, 4115853, 4922346, 1514357, 5212543, 3076389, 2485910, 1094922, 1153739, 7128445, 5878944, 8305779, 1056013, 3403448, 2243480, 402191, 1486893, 139458, 3216086, 7897803, 6272279, 138104, 4482737, 4426582, 8353746, 6751744, 3709583, 4836134, 6647156, 2451104, 1386815, 8279168, 6690625, 3490424, 7603346, 5219632, 2792155, 7179649, 6558019, 3076511, 6226001, 6144737, 1144017

Continued on next page

(Continued)

Multiplier	Values
$\boldsymbol{\alpha}$	$\{1\}^{128}$, 5240616, 6625160, 3870293, 451475, 4098101, 3509475, 4720287, 3204487, 2633339, 8261277, 5371451, 4802163, 1052454, 7985202, 7011421, 1789078, 2482935, 2162795, 2130324, 6911940, 8224373, 7975065, 7362675, 5318767, 256384, 2940530, 1953642, 2284237, 7680329, 4798738, 182096, 6420995, 6262728, 6853579, 1467056, 6679038, 3384943, 1364494, 1815539, 7173684, 381074, 5963412, 378030, 4343637, 1345703, 7882546, 7249321, 3605300, 2415392, 591713, 2396687, 7518040, 2221840, 8100078, 4900106, 5263091, 4483921, 6097599, 5483953, 5444106, 117789, 1529228, 6899376, 3873271, 4699322, 4450988, 1726840, 679011, 2219822, 4140793, 466656, 2203965, 6509925, 7787412, 603082, 2094235, 7492608, 5969231, 1342993, 5182765, 6680762, 3437557, 2969525, 2295436, 4479817, 4814112, 2440657, 2484460, 7892496, 237306, 530634, 816529, 6541158, 7820662, 5399117, 6330948, 3594342, 5863786, 3924672, 251287, 5757005, 7867614, 309896, 217175, 5619146, 2193097, 2578525, 7908807, 4382588, 6137892, 97233, 5691157, 6645351, 4427738, 6988557, 8131464, 431981, 1572591, 4052322, 508107, 748762, 3306458, 289281, 6956158, 7513153, 4539509, 3758126, 4320940

A.2 Kyber

Table 5: Kyber Multipliers

Multiplier	Values
b	$\{1\}^{128}$
a	777, 1317, 3320, 2467, 2492, 857, 2702, 6, 2870, 1105, 2433, 2589, 2739, 424, 2713, 1618, 2499, 1982, 225, 3045, 17, 3301, 1812, 2207, 472, 2128, 3204, 260, 2400, 3200, 459, 2102, 30, 1819, 3051, 3042, 2998, 1225, 1479, 1831, 2198, 1035, 312, 2126, 725, 2730, 1983, 1482, 2930, 9, 298, 2832, 2875, 36, 2231, 1321, 369, 2224, 1289, 373, 1042, 1305, 1550, 367, 2910, 1727, 1972, 2235, 2904, 1988, 1053, 2908, 1956, 1046, 3241, 402, 445, 2979, 3268, 347, 1795, 1294, 547, 2552, 1151, 2965, 2242, 1079, 1446, 1798, 2052, 279, 235, 226, 1458, 3247, 1175, 2818, 77, 877, 3017, 73, 1149, 2805, 1070, 1465, 3305, 3260, 232, 3052, 1295, 778, 1659, 564, 2853, 538, 688, 844, 2172, 407, 3271, 575, 2420, 785, 810, 3286, 1960, 2500

A.3 Falcon I

Table 6: Kyber Multipliers

Multiplier	Values
b	$\{1\}^{512}$

Continued on next page

(Continued)

Multiplier	Values
a	9731, 10896, 3629, 5768, 3702, 10752, 248, 5700, 894, 9075, 10343, 4800, 3736, 871, 1204, 3480, 6457, 5172, 1052, 7314, 10097, 7279, 836, 4745, 9877, 6410, 398, 7342, 7766, 6782, 6180, 11927, 2683, 3917, 8355, 292, 4446, 9568, 2659, 9230, 9949, 9749, 7319, 2216, 10032, 2960, 4485, 8998, 3331, 1247, 3997, 10227, 3656, 1473, 1039, 2347, 11136, 6413, 4135, 11082, 12276, 4709, 11349, 5174, 1338, 6381, 9557, 1752, 9842, 9356, 5253, 8858, 4562, 4643, 3078, 6293, 10951, 1048, 3042, 2500, 5139, 6229, 11523, 89, 3604, 520, 7321, 7150, 11840, 2406, 6780, 4399, 8453, 12240, 8969, 3360, 9846, 3866, 11229, 334, 266, 6548, 436, 4370, 11150, 7583, 5479, 25, 3109, 2808, 362, 8252, 1297, 4898, 6620, 7979, 1259, 10963, 5598, 11425, 4118, 9719, 6575, 10156, 11834, 1814, 10318, 7408, 9487, 3574, 9042, 1460, 9783, 11047, 8008, 1578, 10159, 1353, 4960, 7734, 303, 11821, 12175, 4413, 4983, 1439, 5183, 4518, 2723, 9038, 8559, 12001, 11286, 11302, 10582, 11585, 2034, 4579, 1031, 6169, 6774, 11430, 7864, 6385, 7147, 236, 704, 440, 7054, 7612, 5758, 572, 4756, 838, 1613, 10544, 11291, 6950, 5478, 4185, 9697, 7392, 2272, 3915, 8770, 12209, 10436, 2502, 10604, 9615, 3677, 3711, 10766, 11568, 3398, 2158, 8879, 10687, 6024, 10045, 1002, 9224, 8275, 4464, 8985, 5095, 2064, 3690, 1872, 5593, 2315, 6927, 3838, 9050, 4040, 9973, 8868, 1043, 11281, 1279, 4952, 8041, 3197, 10803, 414, 3684, 7917, 7302, 5368, 9701, 6023, 8313, 7977, 8892, 11156, 203, 1242, 10163, 4006, 7187, 3006, 616, 3257, 8552, 5295, 2163, 1155, 8409, 2140, 11540, 10048, 2588, 10907, 1160, 138, 3080, 9161, 12103, 11081, 1334, 9653, 2193, 701, 10101, 3832, 11086, 10078, 6946, 3689, 8984, 11625, 9235, 5054, 8235, 2078, 10999, 12038, 1085, 3349, 4264, 3928, 6218, 2540, 6873, 4939, 4324, 8557, 11827, 1438, 9044, 4200, 7289, 10962, 960, 11198, 3373, 2268, 8201, 3191, 8403, 5314, 9926, 6648, 10369, 8551, 10177, 7146, 3256, 7777, 3966, 3017, 11239, 2196, 6217, 1554, 3362, 10083, 8843, 673, 1475, 8530, 8564, 2626, 1637, 9739, 1805, 32, 3471, 8326, 9969, 4849, 2544, 8056, 6763, 5291, 950, 1697, 10628, 11403, 7485, 11669, 6483, 4629, 5187, 11801, 11537, 12005, 5094, 5856, 4377, 811, 5467, 6072, 11210, 7662, 10207, 656, 1659, 939, 955, 240, 3682, 3203, 9518, 7723, 7058, 10802, 7258, 7828, 66, 420, 11938, 4507, 7281, 10888, 2082, 10663, 4233, 1194, 2458, 10781, 3199, 8667, 2754, 4833, 1923, 10427, 407, 2085, 5666, 2522, 8123, 816, 6643, 1278, 10982, 4262, 5621, 7343, 10944, 3989, 11879, 9433, 9132, 12216, 6762, 4658, 1091, 7871, 11805, 5693, 11975, 11907, 1012, 8375, 2395, 8881, 3272, 1, 3788, 7842, 5461, 9835, 401, 5091, 4920, 11721, 8637, 12152, 718, 6012, 7102, 9741, 9199, 11193, 1290, 5948, 9163, 7598, 7679, 3383, 6988, 2885, 2399, 10489, 2684, 5860, 10903, 7067, 892, 7532, 12254, 1159, 8106, 5828, 1105, 9894, 11202, 10768, 8585, 2014, 8244, 10994, 8910, 3243, 7756, 9281, 2209, 10025, 4922, 2492, 2292, 3011, 9582, 2673, 7795, 11949, 3886, 8324, 9558, 314, 6061, 5459, 4475, 4899, 11843, 5831, 2364, 7496, 11405, 4962, 2144, 4927, 11189, 7069, 5784, 8761, 11037, 11370, 8505, 7441, 1898, 3166, 11347, 6541, 11993, 1489, 8539, 6473, 8612, 1345, 2510

A.4 Falcon V

Table 7: Dilithium Multipliers

Multiplier	Values
b	$\{1\}^{1024}$

Continued on next page

(Continued)

Multiplier	Values
a	5134, 848, 11089, 10938, 1011, 817, 2526, 9283, 782, 4219, 4751, 4902, 2193, 10753, 7706, 7898, 1171, 3639, 10713, 5439, 784, 11574, 7926, 12186, 3206, 4153, 4812, 7629, 696, 9045, 2514, 1858, 3783, 7541, 7222, 9668, 11459, 7212, 3677, 3807, 8190, 3655, 638, 5306, 6388, 235, 2885, 3353, 11680, 6854, 6079, 2426, 3500, 7777, 410, 257, 3888, 7642, 10472, 5288, 11453, 4349, 11933, 1838, 1912, 4377, 11043, 2822, 18, 8304, 6192, 3765, 4226, 2437, 3861, 8720, 7305, 4458, 3292, 8020, 9047, 8749, 11742, 4556, 7955, 8762, 5931, 11216, 10554, 758, 5264, 4577, 868, 12268, 4361, 12151, 6181, 9100, 11291, 10817, 5551, 8305, 158, 11465, 9595, 5815, 8785, 556, 8120, 9183, 9073, 2054, 7786, 10275, 321, 4321, 12222, 8887, 6048, 6501, 2053, 5318, 8165, 7372, 2348, 2222, 11053, 3465, 2471, 11883, 6716, 1438, 4119, 808, 305, 9401, 200, 3504, 417, 3936, 9027, 5682, 6292, 6939, 2220, 6545, 3447, 192, 10231, 11654, 3309, 11467, 1612, 2624, 2351, 1663, 600, 9333, 6039, 11273, 7588, 4972, 10760, 1306, 1464, 8269, 7051, 7911, 10815, 11337, 10774, 10680, 8923, 11610, 4844, 2006, 11589, 10511, 4242, 12180, 9547, 3882, 9440, 6655, 11862, 9356, 1038, 1814, 686, 6747, 5654, 9212, 7531, 8289, 1821, 10063, 3362, 7017, 9825, 3445, 10249, 5996, 4930, 7246, 9218, 6691, 7543, 2736, 12067, 3893, 1816, 9025, 9894, 6551, 12250, 9031, 3409, 6197, 8768, 2945, 10242, 372, 11116, 11828, 9532, 5820, 10618, 10328, 11597, 3667, 6131, 4312, 7679, 2651, 4059, 5621, 5483, 8308, 9906, 5631, 7658, 7748, 1820, 6429, 4275, 9685, 5127, 6917, 7651, 12006, 2761, 7896, 3755, 2036, 11315, 9806, 6702, 11153, 852, 5113, 5284, 4978, 442, 5487, 10773, 3948, 7498, 11245, 10142, 10828, 7108, 4628, 10709, 5876, 10410, 4217, 5394, 5168, 6024, 11967, 8264, 84, 6231, 6205, 3002, 2677, 2112, 9094, 1001, 7895, 4596, 5266, 1642, 9752, 5130, 6806, 11323, 8186, 9766, 6032, 2074, 11766, 10565, 10171, 11311, 5455, 1572, 209, 2758, 8920, 9037, 9223, 5665, 3777, 8595, 6748, 7881, 10688, 8986, 9030, 5477, 1953, 9397, 2347, 11577, 2867, 9412, 3068, 2716, 7144, 8755, 12115, 7971, 11029, 3537, 274, 3727, 3207, 10748, 4143, 10084, 4992, 3701, 3737, 7447, 6323, 8009, 6582, 7186, 2872, 5807, 790, 5080, 4984, 4730, 10831, 2343, 4034, 3587, 2591, 11731, 7015, 2592, 7484, 7939, 10529, 5082, 9502, 2502, 6382, 6126, 8700, 8467, 7722, 2127, 2306, 7799, 4562, 2026, 6601, 12243, 9939, 8285, 5249, 8322, 711, 384, 2779, 9481, 1566, 7864, 10200, 7341, 426, 10189, 10939, 8607, 5966, 5353, 6465, 6067, 8687, 6351, 4987, 507, 3282, 10497, 10710, 7071, 11987, 1733, 5261, 10459, 38, 5210, 8600, 8767, 7657, 4, 4680, 154, 10749, 10617, 10979, 4042, 10875, 8218, 4327, 4443, 195, 2496, 3884, 5496, 226, 3386, 8619, 2403, 759, 10059, 11929, 7172, 915, 11213, 2441, 1046, 1462, 10261, 550, 8929, 560, 11905, 11442, 6471, 6313, 1205, 5017, 10175, 11636, 6280, 12109, 4464, 2058, 5583, 8325, 6620, 10665, 9540, 8621, 3066, 5584, 6171, 10506, 6842, 994, 9909, 6989, 9352, 11132, 3600, 1076, 2196, 8599, 2239, 5195, 263, 4390, 10008, 5594, 7417, 7066, 4271, 2952, 10390, 9637, 10732, 6979, 4394, 6478, 9858, 10722, 78, 6673, 7409, 12153, 3559, 2254, 2330, 9574, 5146, 7436, 2217, 7668, 4597, 10048, 4829, 7119, 2691, 9935, 10011, 8706, 112, 4856, 5592, 12187, 1543, 2407, 5787, 7871, 5286, 1533, 2628, 1875, 9313, 7994, 5199, 4848, 6671, 2257, 7875, 12002, 7070, 10026, 3666, 10069, 11189, 8665, 1133, 2913, 5276, 2356, 11271, 5423, 1759, 6094, 6681, 9199, 3644, 2725, 1600, 5645, 3940, 6682, 10207, 7801, 156, 5985, 629, 2090, 7248, 11060, 5952, 5794, 823, 360, 11705, 3336, 11715, 2004, 10803, 11219, 9824, 1052, 11350, 5093, 336, 2206, 11506, 9862, 3646, 8879, 12039, 6769, 8381, 9769, 12070, 7822, 7938, 4047, 1390, 8223, 1286, 1648, 1516, 12111, 7585, 12261, 4608, 3498, 3665, 7055, 12227, 1806, 7004, 10532, 278, 5194, 1555, 1768, 8983, 11758, 7278, 5914, 3578, 6198, 5800, 6912, 6299, 3658, 1326, 2076, 11839, 4924, 2065, 4401, 10699, 2784, 9486, 11881, 11554, 3943, 7016, 3980, 2326, 22, 5664, 10239, 7703, 4466, 9959, 10138, 4543, 3798, 3565, 6139, 5883, 9763, 2763, 7183, 1736, 4326, 4781, 9673, 5250, 534, 9674, 8678, 8231, 9922, 1434, 7535, 7281, 7185, 11475, 6458, 9393, 5079, 5683, 4256, 5942, 4818, 8528, 8564, 7273, 2181, 8122, 1517, 9058, 8538, 11991, 8728, 1236, 4294, 150, 3510, 5121, 9549, 9197, 2853, 9398, 688, 9918, 2868, 10312, 6788, 3235, 3279, 1577, 4384, 5517, 3670, 8488, 6600, 3042, 3228, 3345, 9507, 12056, 10693, 6810, 954, 2094, 1700, 499, 10191, 6233, 2499, 4079, 942, 5459, 7135, 2513, 10623, 6999, 7669, 4370, 11264, 3171, 10153, 9588, 9263, 6060, 6034, 12181, 4001, 298, 6241, 7097, 6871, 8048, 1855, 6389, 1556, 7637, 5157, 1437, 2123, 1020, 4767, 8317, 1492, 6778, 11823, 7287, 6981, 7152, 11413, 1112, 5563, 2459, 950, 10229, 8510, 4369, 9504, 259, 4614, 5348, 7138, 2580, 7990, 5836, 10445, 4517, 4607, 6634, 2359, 3957, 6782, 6644, 8206, 9614, 4586, 7953, 6134, 8598, 668, 1937, 1647, 6445, 2733, 437, 1149, 11893, 2023, 9320, 3497, 6068, 8856, 3234, 15, 5714, 2371, 3240, 10449, 8372, 198, 9529, 4722, 5574, 3047, 5019, 7335, 6269, 2016, 8820, 2440, 5248, 8903, 2202, 10444, 3976, 4734, 3053, 6611, 5518, 11579, 10451, 11227, 2909, 403, 5610, 2825, 8383, 2718, 85, 8023, 1754, 676, 10259, 7421, 655, 3342, 1585, 1491, 928, 1450, 4354, 5214, 3996, 10801, 10959, 1505, 7293, 4677, 992, 6226, 2932, 11665, 10602, 9914, 9641, 10653, 798, 8956, 611, 2034, 12073, 8818, 5720, 10045, 5326, 5973, 6583, 3238, 8329, 11848, 8761, 12065, 2864, 11960, 11457, 8146, 10827, 5549, 382, 9794, 8800, 1212, 10043, 9917, 4893, 4100, 6947, 10212, 5764, 6217, 3378, 43, 7944, 11944, 1990, 4479, 10211, 3192, 3082, 4145, 11709, 3480, 6450, 2670, 800, 12107, 3960, 6714, 1448, 974, 3165, 6084, 114, 7904, 12286, 11397, 7688, 7001, 11507, 1711, 1049, 6334, 3503, 4310, 7709, 523, 3516, 3218, 4245, 8973, 7807, 4960, 3545, 8404, 9828, 8039, 8500, 6073, 3961, 12247, 9443, 1222, 7888, 10353, 10427, 332, 7916, 812, 6977, 1793, 4623, 8377, 12008, 11855, 4488, 8765, 9839, 6186, 5411, 585, 8912, 9380, 12030, 5877, 6959, 11627, 8610, 4075, 8458, 8588, 5053, 806, 2597, 5043, 4724, 8482, 10407, 9751, 3220, 11569, 4636, 7453, 8112, 9059, 79, 4339, 691, 11481, 6826, 1552, 8626, 11094, 4367, 4559, 1512, 10072, 7363, 7514, 8046, 11483, 2982, 9739, 11448, 11254, 1327, 1176, 11417, 7131