Number "Not Used" Once - Practical fault attack on *pqm4* implementations of NIST candidates

Abstract In this paper, we demonstrate practical fault attacks over a number of lattice based schemes, in particular NewHope, Kyber, Frodo, Dilithium which are based on the hardness of the Learning with Errors (LWE) problem. One of the common traits of all the considered LWE schemes is the use of nonces as domain separators to sample the secret components of the LWE instance. We show that simple faults targeting the usage of nonce can result in a nonce-reuse scenario which allows key recovery and message recovery attacks. To the best of our knowledge, we propose the first practical fault attack on lattice-based Key encapsulation schemes secure in the CCA model. We perform experimental validation of our attack using Electromagnetic fault injection on reference implementations of the aforementioned schemes taken from the *pqm4* library, a benchmarking and testing framework for post quantum cryptographic implementations for the ARM Cortex-M4. We use the instruction skip fault model, which is very practical and popular in microcontroller based implementations. Our attack requires to inject a very few number of faults (numbering less than 10 for recommended parameter sets) and can be repeated with a 100% accuracy with our Electromagnetic fault injection setup.

1 Introduction

Ever since the discovery of the Shor's algorithm [26], there has always been an imminent danger of the possibility of large scale quantum computers threatening our existing public key infrastructure. The cryptographic community has long felt the need to replace the existing public key cryptosystems with quantum resistant alternatives, which is also justifiable given that research in the quantum computing field has grown by leaps and bounds [24].

NIST recently initiated the process for standardization of post quantum cryptographic alternatives for public key encryption (PKE), Key Exchange (KEX) and digital signatures (DS) [21]. Among the 64 submissions which still remain in the competition, lattice-based cryptography fields the largest contingent in terms of the number of submissions. This is due to the fact that they provide a very good balance of a number of attributes like key sizes, ciphertext sizes, computational performance which are on par with existing public key cryptographic primitives based on RSA and ECC along with providing post-quantum security.

The assessment of each candidate is being done based on multiple aspects such as classical security, post-quantum security, performance on a wide-range

of devices (from Desktop PCs to resource constrained 8-bit microcontrollers) among other other parameters. Another crucial aspect that is being looked at is the implementation security of post quantum cryptographic alternatives against active and passive physical attacks. In this regard, there have been a number of works that have reported physical attacks over lattice-based schemes through exploitation of a number of side channels like power/EM side [15, 23], cache timing [10] and induced faults [11].

In this work, we try to analyze the fault vulnerabilities of multiple lattice-based schemes which base their security on the Learning with Errors (LWE) problem. One of the crucial components with respect to implementation of all the LWE based schemes is the error sampling procedure. We analyzed the implementations of multiple lattice-based schemes such as NewHope [3], Kyber [6], Frodo [9] Key Encapsulation (KEM) schemes & Dilithium digital signature (DS) scheme [19] and observed common traits with respect to the usage of fixed nonces as simple domain separators in the error sampling procedure. The simplistic use of nonce to generate the secret components in the scheme raises questions concerning security against potential fault attacks. Though nonce-reuse based attacks such as the ones reported on ECDSA are well known in literature [12], surprisingly, none of the specification documents of the any of the schemes discuss about possible issues due to misuse of the nonces. Thus, we extend the applicability of nonce-reuse based fault attacks to cryptographic schemes based on the LWE problem. *In this work, we mainly focus on targeting the simplistic use of nonces through fault injection to create weak faulty LWE instances, resulting in key recovery and message recovery attacks in multiple lattice-based schemes.*

The contribution of this work are as follows:

- We extend the applicability of nonce-misuse based attacks to lattice-based LWE schemes mainly targeting nonces used as domain separators during generation of LWE instances.
- We analyze four lattice-based LWE schemes such as NewHope, Kyber, Frodo and Dilithium and demonstrate how nonce-misuse in these schemes could result in key recovery (long term key) and message recovery (session key) attacks. To the best of our knowledge, we perform the first fault analysis of lattice-based KEM schemes while all prior works focussed on fault attacks on lattice-based digital signatures [11, 14].
- We propose a novel fault assisted Man-In-The-Middle (MITM) attack to perform message recovery in the considered KEM schemes secure in the Chosen Ciphertext Attacker (CCA) model. We fault the encapsulation procedure to perform successful message recovery, which is counter-intuitive given the fact that a re-encapsulation is done at the decapsulator's side in a Chosen Cipehertext secure KEM scheme which might detect tampering due to fault injection.
- We validated the vulnerabilities using electromagnatic fault injection on the ARM Cortex-M4 microcontroller. We performed practical fault attacks over reference implementations of the aforementioned NIST candidates taken from

the *pqm4*[3] public library, a testing and benchmarking framework for post quantum cryptographic schemes on the ARM Cortex-M4 microcontroller.

The rest of the paper is organized as follows. Section 2 provides a brief description of various lattice-based LWE cryptosystems. The identified fault vulnerabilities and the associated key recovery and message recovery attacks are described in Sec.3 with practical experimental results using our EMFI setup covered in Sec.4. Possible countermeasures against the proposed fault attacks are discussed in section 5 with final conclusions drawn in section 6.

2 Background on Lattice Based Cryptography

2.1 Lattice Preliminaries

This section provides a brief background on the Learning With Errors problem and the LPR encryption scheme, which is the first Ring-LWE based PKE scheme [20] that has been the foundation of a number of efficient lattice-based PKE and KEM schemes including the schemes considered in this work. This section also further touches upon known insecure instantiations of the LWE problem and prior work done with respect to fault analysis on lattice-based cryptographic schemes.

We denote the polynomial ring $\mathbb{Z}_q[X]/(X^n+1)$ as R_q for $q \in \mathbb{Z}^+$. Polynomials in R_q are denoted using bold lower case letters while matrices ($\mathbb{Z}_q^{k\times l}$) and vectors($\mathbb{Z}_q^l$) are denoted using bold upper case letters. Multiplication of two polynomials $\mathbf{a}$ and $\mathbf{b}$ is denoted as $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ while point wise multiplication of two entities is denoted as $\mathbf{c} = \mathbf{a} * \mathbf{b}$. We use the notation $\mathcal{B}^n$ to denote an array of n bytes.

2.2 The Learning with Errors Problem

The Learning With Errors (LWE) problem, introduced by Regev in 2006 [25] is a versatile average case problem related to worst case hard lattice problems like the Shortest Vector Problem (SVP) and the Bounded Distance Decoding Problem (BDD) on related lattices. The general LWE problem can be briefly defined as follows: Given a small secret $\mathbf{S} \in \mathbb{Z}_q^n$, an LWE distribution consists of ordered pairs $(\mathbf{A}, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where $\mathbf{A} \in \mathbf{Z}_q^n$ is public and $t = (\mathbf{A} \times \mathbf{S} + e) \in \mathbb{Z}_q$ where $e \leftarrow \mathcal{D}_\sigma$. Given polynomially many pairs ($\mathbf{A}$, t), the search LWE problem requires one to find a solution for $\mathbf{S}$ and the decision LWE problem requires one to distinguish structured ordered pairs $(\mathbf{A}, t)$ from random ones in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Its more structured variants like the Ring-LWE problem [20] and the Module-LWE problem [18] that compute over polynomial rings R_q possess greatly reduced key-sizes and computational time. All the four schemes considered in this paper except for FRODO, base their hardness on these structured variants of the LWE problem. There is another variant of the LWE problem, called the Learning With

[3] Available on `https://github.com/mupq/pqm4`

Rounding (LWR) problem wherein deterministic noise is generated by rounding every coefficient of the product $(\mathbf{A} \times \mathbf{S})$ to a lower modulus and their subsequent expansion back to the higher modulus. It is also interesting to note that this rounding/modulus switching technique has been used in all the considered LWE schemes in the paper, as an effort to reduce the key or ciphertext sizes.

We classify the LWE instances in general into two types, based on the purpose for which they are utilized in the considered schemes.

1. An LWE instance can serve as the public key of a KEM/DS scheme. Such an LWE instance is denoted as LWE_{PK}.
2. An LWE instance which is indistinguishable from random, can help hide a suitably encoded message in a KEM scheme. Ciphertexts are formed by adding an encoded message with an LWE instance, thus obscuring the message. Such an LWE instance is denoted as LWE_{OBS}.

We will henceforth utilize this terminology for LWE instances throughout this paper.

2.3 LPR Encryption scheme [20]

The three LWE based KEM schemes (i.e) NewHope, Frodo and Kyber contain in their core, the LPR public key encryption scheme which is based on the hardness of the Ring-LWE problem. The key generation procedure of all the three KEM schemes (including Dilithium) and the encryption procedure of all the three KEM schemes follow the same framework as that of the LPR encryption scheme. It would be sufficient to describe the LPR encryption scheme, as it captures the essence of our attack analysis in all the considered schemes. The LPR encryption scheme can be briefly described as follows:

- $\mathsf{KeyGen}(\mathbf{a})$: Generate random polynomials $\mathbf{a} \leftarrow R_q$ and $\mathbf{e}, \mathbf{s} \leftarrow \mathcal{D}_\sigma^n$. The public-key polynomial is calculated as follows: $\mathbf{p} = \mathbf{e} + \mathbf{a} \times \mathbf{s} \in R_q$ with $\mathbf{s}$ being the private key.
- $\mathsf{Encrypt}(\mathbf{a},\mathbf{p},\mathbf{m})$: Three polynomials $\acute{\mathbf{s}}$, $\acute{\mathbf{e}}$, $\acute{\mathbf{e}} \in R_q$ are sampled from $\mathcal{D}_\sigma^n$. The message m $(m_0, m_1, m_2, \ldots, m_{n-1})$ to be encrypted is encoded coefficient-wise into a polynomial as follows: The bit m_i is encoded to coefficient $q/2$ if it is 1, else it is encoded to 0. Ciphertext $\mathbf{c}_1$ is calculated as $\mathbf{a} \times \acute{\mathbf{s}} + \acute{\mathbf{e}}$ while ciphertext $\mathbf{c}_2$ is formed by embedding the message into an LWE instance as $\mathbf{c}_2 = \mathbf{p} \times \acute{\mathbf{s}} + \acute{\mathbf{e}} + \mathbf{m}$.
- $\mathsf{Decrypt}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{r}_2)$: $\hat{\mathbf{m}} \in R_q$ is computed as $\mathbf{c}_2 - \mathbf{c}_1 \times \mathbf{s}$. The decoder further retrieves the message m one bit at a time as $m = \mathcal{D}(\acute{\mathbf{m}})$ such that $m_i = 1$ if the corresponding coefficient is on $[q/4, 3q/4]$ else $m_i = 0$. This encoding procedure can withstand a coefficient wise error in $\hat{\mathbf{m}}$ upto $q/4$.

Our attack works by targeting appropriate variants of the aforementined KeyGen and $\mathsf{Encrypt}$ procedures instantiated within the considered KEM (NewHope, Frodo and Kyber) and signature schemes (Dilithium).

2.4 Insecure Instantiations of the LWE problem

The error component plays a major role in ensuring the hardness of the LWE instance. An LWE instance without the error component is nothing but a system of well defined modular linear equations, which can be solved by gaussian elimination. There are indeed certain other trivially solvable instantiations of the LWE problem. If the error component only has values in a fixed interval $[z + \frac{1}{2}, z - \frac{1}{2}]$, then one can just "round away" the non-integral part and subtract z to remove the error from every sample [22]. From a given set of n LWE instances, if k of the n error components add up to zero, then one can simply add the corresponding samples to cancel the error and obtain an error free sample. It is also possible to solve an LWE instance in roughly n^d time and space using n^d samples if the error in the samples lie in a known set of size d [5]. For a very small d, this yields a very practical attack. Our attack works by realizing such an insecure instance of LWE through faults to mount key recovery and message recovery attacks in the considered schemes.

2.5 Error Sampling procedure

Almost all of the earlier lattice-based schemes like BLISS [13], LPR Encryption scheme [20] resorted to using complex discrete gaussian samplers for their error sampling procedures [17]. But, gaussian samplers turned out to be inefficient and very difficult to be implemented in constant time. Adding to that, implementations of discrete Gaussian samplers came under heavy scrutiny owing to a number of side channel attacks [10, 15]. Moreover, Alkim *et al.* [4] reported that high precision gaussian sampling is an overkill for encryption schemes and that it was only required for schemes that require zero-knowledge proofs. Subsequently, all the newer lattice-based proposals (including the schemes considered in this paper) resorted to sampling from simpler and more secure noise distributions like the centered Binomial distribution (CBD) [3, 6]. All the schemes considered in this paper expand a given small seed using XOF's such as SHAKE256, CSHAKE (Extendable Output Functions) from the SHA3 family of cryptographic primitives to random outputs of desired length, which is further processed to generate samples from simpler distributions like the CBD or uniform distribution. Alternatively, coefficient-wise modulus switching is also used to generate deterministic error in schemes based on the hardness of the LWR problem.

2.6 Prior work on fault analysis of lattice-based cryptographic schemes

Bindel *et al.* [8] proposed the first fault analysis of a number of lattice based signature schemes like the GLP [16], BLISS [13] and Ring-Tesla [2] schemes that follow the Fiat-Shamir framework. They identified a number of fault vulnerabilities in multiple operations across the various key generation, signing and verification procedures of lattice-based digital signature schemes. Later, Espitau *et al.* [14]

reported a generic and a stronger fault attack based on loop abort faults on both the Fiat-Shamir type and Hash-and-Sign type signature schemes. It worked by converting the signature component into a solvable closest vector problem (CVP) instance when the error polynomial is limited to low degrees using loop-abort faults. The first differential style fault attacks on Dilithium and qTESLA signatures that are potential NIST standards, have been reported by Bruinderink and Pessl [11] that mainly targets the deterministic nature of the signing procedure by inducing random faults that can be injected during a large section of the execution time. They utilize clock glitches to realize their faults on an ARM Cortex-M4 based microcontroller. To the best of our knowledge, while all related prior works have only reported fault attacks on lattice-based signature schemes, our work is the first practical fault attack on lattice-based KEM schemes.

3 Fault attacks on LWE Schemes

3.1 General Attack Idea

The core idea of our attack is to create a nonce-reuse scenario using faults to generate a trivially solvable LWE instance. One of the common traits of all the considered schemes in this paper is the error sampling procedure. We explain the intuition of our attack using the NewHope KEM scheme as an example. The same vulnerability also applies to all the other considered schemes in this paper. Refer to the key generation procedure of the NewHope KEM scheme in Alg.2. We observe that both the secret ($\mathbf{s}$) and error ($\mathbf{e}$) components of the LWE instance $\hat{\mathbf{b}}$ (in the NTT domain) are generated using the seeds which differ only based on a single nonce value (Line 8,10). These seeds are further input to the Sample function which generate the required polynomials through use of XOF functions from the SHA3 family. The most important observation being the use of almost similar seeds that differ only based on the nonce value, to generate the secret and error components of the LWE instance.

For example, assume a Ring-LWE instance as created in the LPR encryption scheme as follows:

$$\mathbf{t} = \mathbf{a} \times \mathbf{s} + \mathbf{e} \in \mathbf{R}_q$$

The above equation can be alternatively seen as a modular linear system of equations with n equations and $2n$ unknowns. Assume that the attacker injects faults to create a nonce-reuse scenario where both $\mathbf{s}$ and $\mathbf{e}$ are generated using the same seed. The corresponding faulty LWE instance generated is

$$\mathbf{t} = \mathbf{a} \times \mathbf{s} + \mathbf{s} \in \mathbf{R}_q$$

The faulty LWE instance is a modular well defined linear system of equations with n equations and n unknowns ($\because n$ coefficients for each polynomial) which can be trivially solved using Gaussian elimination. This principle also applies to all the versions of the LWE problem such as the general-LWE, Ring-LWE and

the Module-LWE problem. Thus, the modus operandi of our attack is to ensure that the nonces used for generation of both the secret and error are the same through injection of appropriate faults.

3.2 Key Recovery Attacks

In the following discussion, we will show how the identified vulnerabilities with respect to nonce reuse can result in key recovery attacks in the considered schemes. By key recovery attacks, we refer to the recovery of the long term secret key.

3.2.1 Attacking NewHope and Frodo

Refer to Alg.2 for the key generation procedure of the NewHope KEM scheme. The LWE instance $\hat{\mathbf{b}}$ formed as the public key in the key generation procedure is of type LWE_{PK}. Its corresponding secret and error components are created using the same seed, but with different nonces 0 and 1 respectively (Line 8 and 10 of KeyGen procedure of Alg.2). If faults can be injected to realize a nonce-reuse scenario, the secret key $\mathbf{s}$ can be trivially recovered from the public key, as per the analysis shown in Sec.3.1. It is important to note that attack on the key generation procedure is applicable to both the CPA (Chosen Plaintext Attacker) and CCA (Chosen Ciphertext Attacker) secure model. The same attack can be very similarly adapted to the key generation procedure of the FRODO KEM scheme secure in both the CPA and CCA secure models. For brevity, we have not included the key generation algorithm of the FRODO scheme and hence please refer to [9] for the exact key generation procedure of the FRODO KEM scheme.

3.2.2 Attacking Dilithium

We also examine the applicability of our key recovery attack on schemes operating over modules (matrices/vectors of polynomials in ring) such as Dilithium and Kyber. This section explains our key recovery attack on the Dilithium signature scheme which targets the key generation procedure (Refer Alg.1).

We can identify the LWE instance $\mathbf{t}$ as type LWE_{PK} (Line 14 in the KeyGen procedure of Alg.1). The multiple polynomials in its corresponding secret and error components are created using very similar seeds which only differ by a deterministically incrementing nonce value (Line 5-12 in the KeyGen procedure of Alg.1). If multiple faults can be injected to realize nonce-reuse during generation of both the secret and error components, $\mathbf{t}$ reduces to a set of well defined linear equations ($k \times n$ unknowns since $k > l$).

There is a subtle but considerable difference with respect to publicly revealed LWE instances in the Dilithium scheme. The public key reveals only $\mathbf{t}_1$, the d higher order bits of $\mathbf{t}$, while $\mathbf{t}_0$ (the lower order component) is part of the secret key. Even on ensuring nonce-reuse, we would not be able to trivially solve for the secret $\mathbf{s}$ from the faulty public key. But, note that the security analysis of `DILITHIUM` is done with the assumption that the whole of $\mathbf{t}$ is declared as

the public key. In addition to this, some information about $\mathbf{t}_0$ is leaked with every published signature and thus the whole of $\mathbf{t}$ can be reconstructed by just observing several signatures generated using the same secret key [1]. Thus it is reasonable to assume that successful faults injected in the key generation procedure results in a key recovery attack over the Dilithium signature scheme.

3.2.3 Attacking Kyber

Refer to Alg.3 for the key generation procedure of Kyber KEM scheme. We identify the LWE instance $\mathbf{t}$ to be of type LWE_{PK}(Line 19 of KeyGen procedure of Alg.3). Its corresponding secret and error components are generated in a similar fashion as that of the key generation procedure of the Dilithium signature scheme (Line 10-17 of KeyGen procedure of Alg.3). But, similar to the compression technique used in Dilithium, we can also see that the LWE instance $\mathbf{t}$ is not directly published as the public key. A coefficient-wise modulus switching procedure is performed over the LWE instance $\mathbf{t}$, which is subsequently revealed as the public key *pk* (Line 20 of KeyGen procedure of Alg.3).

This procedure adds a certain deterministic noise to the LWE instance similar to the Module-Learning-with-Rounding (Module-LWR) problem. But, the authors do not consider this as an added layer of security but simply as a technique to reduce the output size, due to the absence of a Ring/Module variant of a hardness reduction for LWR. The authors also state that they "believe" the compression technique adds some security, but this has not been quantified. Due to this compression technique, successful faults injected does not trivially result in key recovery, Our attack directly targets LWE hardness bringing down the security to LWR hardness. The attack on the residual LWR instance is out of scope of this work.

3.2.4 Applicability of Key Recovery Attacks

Among many aspects that are considered for evaluation in the standardization process, NIST also expects KEM schemes in particular, to provide *perfect forward secrecy*. This requires the KEM schemes to often perform key generation over frequent intervals to generate fresh public-private key pairs. It is often claimed that key generation is performed in certain secure locations thus removing the threat from possible physical attacks, especially side-channel/fault attacks. But, considering the use case of an IoT network housing a mesh of low power constrained devices, frequent communication of an end device with a server for a fresh public-private key pair would have a heavy toll on power-consumption of remote devices. Power is a very critical resource and thus communication of keys using power hungry RF transceiver modules would be much more expensive than performing key generation directly on the device. Thus, it is reasonable to assume that the key generation procedure will be performed over end-devices, thus leaving it prone to possible fault attacks.

3.3 Message Recovery Attacks

In this section, we will show how the identified vulnerabilities with respect to nonce reuse can result in message recovery attacks in the considered schemes. We specifically target the encryption procedures within the larger encapsulation procedures and our attack applies to the KEM schemes secure in both the CPA and CCA model. By message recovery, we refer to the recovery of the short term session key that is exchanged at the end of the encapsulation-decapsulation procedure.

Our message recovery attack directly applies to the considered CPA secure KEM schemes, similar to the analysis described for our key recovery attacks albeit involving some additional analysis to recover the message. But, intuition tells us that faulting the encapsulation procedure cannot be done in CCA secure version of KEM schemes. This is due to the employed Fujisaki-Okamoto (FO) transformation which performs a re-encapsulation during the decapsulation procedure to check for the validity of ciphertexts. This technique effectively thwarts use of chosen/faulted ciphertext attacks. But, we show that an MITM (Man-In-The-Middle) attacker can still perform valid message recovery attacks over CCA secure KEM schemes when faulting the encapsulation procedure.

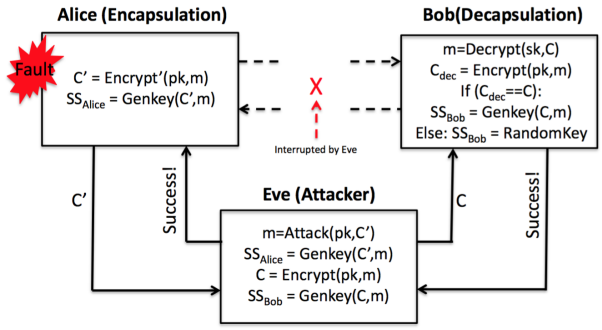


Figure1: Fault assisted MITM attack on CCA Secure KEM scheme

Refer to Fig.1 for a pictorial description of our proposed fault assisted MITM attack on a CCA secure KEM scheme employing the FO transformation. Alice performs the encapsulation operation, Bob performs decapsulation with Eve being the MITM attacker. We have abstracted away the internal details of both the encapsulation and decapsulation procedures and have represented each of these procedures as composition of the functions Encrypt,Decrypt and GenKey. The Encrypt and Decrypt functions represent the encryption and decryption procedures underlying the considered KEM schemes. The GenKey function which is used to calculate the shared session key is a publicly known fixed transformation, varies

according to the KEM scheme. The faulted encryption procedure is denoted as $\mathsf{Encrypt}'$.

Lets assume that the attacker Eve performs a targeted fault into the $\mathsf{Encrypt}$ function of the encapsulation procedure to evoke a faulty ciphertext C' for an internally generated message m. The shared key ss_{Alice} is computed as GenKey(C',m). Lets assume that Eve receives the faulty ciphertext C' from Alice and recovers the message m using our analysis (Attack(C',pk)), whose details will be furnished later in this section. Further, Eve calculates Alice's shared secret key ss_{Alice} similar to Alice, now that Eve has the knowledge of both the faulted ciphertext C' and the message m. Having recovered the message, Eve now performs the correct encapsulation procedure with the recovered message m to generate the correct ciphertext C and the corresponding shared secret key which we denote as ss_{Bob}. The correct ciphertext is shared with Bob. Bob successfully decapsulates the ciphertext to generate the shared key ss_{Bob}.

Though the keys shared by both Bob and Alice and different, Eve has the knowledge of both the shared secret keys (session keys) ss_{Alice} and ss_{Bob} through which she can decrypt all communication transcripts between Alice and Bob during that session. To the best of our knowledge, we propose the first fault attack methodology to perform message recovery in CCA secure lattice-based KEM schemes. In the following discussion, we will show how nonce-reuse can be performed over encryption procedures to recover messages in the individual KEM schemes.

3.3.1 Attacking NewHope and Frodo

Refer to Alg.2 for the encryption procedure of the NewHope KEM scheme. We identify $\hat{\mathbf{u}}$ as a compound LWE instance of type LWE_{OBS} which is used to hide the encoded message $\mathbf{v}$ (Line 11 of Encrypt procedure of Alg.2). Its corresponding secret $\acute{\mathbf{s}}$ ($\hat{\mathbf{t}}$ in the NTT form) is also used to create an LWE instance $\hat{\mathbf{u}}$ (Line 9). We can also see that the secret and error components of this LWE instance $\hat{\mathbf{u}}$ share the same input sampling seed while only differing by one byte on the nonce value (Line 5 and 6). It is important to note that $\hat{\mathbf{u}}$ is encoded as part of the ciphertext, but is not tampered with (Not compressed). Thus, the attacker can directly access the LWE instance $\hat{\mathbf{u}}$, as part of the ciphertext.

On ensuring the same nonce for both $\acute{\mathbf{s}}$ and $\acute{\mathbf{e}}$ through faults, the resulting faulty LWE instance $\hat{\mathbf{u}}$ can be easily compromised, revealing $\hat{\mathbf{t}}$. Subsequently, we can calculate and retrieve message μ as follows:

$$\mathbf{m} = \mathsf{Decompress}(\mathbf{h}) - \mathsf{NTT}^{-1}(\hat{\mathbf{b}} * \hat{\mathbf{t}}) \ (\because \hat{\mathbf{t}} = \mathsf{NTT}(\acute{\mathbf{s}}))$$
$$\mu = \mathcal{D}(\mathbf{m})$$

where $\mathcal{D}$ denotes the corresponding message decoder used in the scheme.

Upon recovery of the message, an attacker can use our aforementioned attack methodology in an MITM setting to mount a successful message recovery attack on both the CCA and CPA secure versions of the NewHope KEM scheme. The same attack can be very similarly adapted to perform successful message recovery in both the CPA and CCA secure versions of the FRODO KEM scheme as well.

3.3.2 Attacking Kyber

Refer to Alg.3 for the encryption procedure of the Kyber KEM scheme. We identify a compound LWE instance $\mathbf{u}$ of type LWE_{OBS} is used to hide the encoded version of the message m (Line 15 of Encrypt procedure of Alg.3). Its corresponding secret $\mathbf{r}$ is also used to create another LWE instance $\mathbf{u}$ which is exposed as part of the ciphertext (Line 14 of Encrypt procedure of Alg.3). So, $\mathbf{u}$ can be appropriately faulted to enforce nonce-reuse. But, $\mathbf{u}$ is compressed using the modulus-switching technique (LWR) before it is revealed as part of the ciphertext. We thus reduce $\mathbf{u}$ to an LWR instance, whose security has not been analysed for the given parameters.

4 Experimental Validation

In this section, we perform an experimental validation of all our proposed attacks on a real device. We start by introducing our experimental setup, providing details of our device under target, implementation details and our attack setup. Since our attack requires to inject targeted faults, we further demonstrate our analysis of the various implementations to identify our target operation and ensure successful faults with very high repeatability.

4.1 Experimental Setup

For our experiments, we target the reference implementations of the considered schemes taken from the *pqm4* library, a benchmarking and testing framework for PQC schemes on the ARM Cortex-M4 family of microcontrollers. We ported the reference implementations to the STM32F4DISCOVERY board (DUT) housing the STM32F407, ARM Cortex-M4 microcontroller. All our implementations (compiled with `-O3 -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16`) were running at a clock frequency of 24 MHz. We use the ST-LINK/v2.1 add-on board for USART communication with our DUT. We used the OpenOCD framework for flash configuration and on-chip hardware debugging with the aid of the GNU debugger for ARM (arm-none-eabi-gdb). We use Electromagnetic Fault injection (EMFI) to inject faults into our device.

Refer Fig.2 for our EMFI setup. The EMFI setup injects electromagnetic pulses with high voltage with low rise time (<4ns) in order to disturb the target operation. A controller software running on the laptop control both the EM pulse generator and the DUT and synchronizes their operation through serial communication. The EM pulse generator is directly triggered by an external trigger signal from the DUT. The EM pulse injector, which is a customized

[1] Our attack removes the hardness guarantees of the generated hard instance from the Module-LWE problem, while the Module-LWR problem remains to be solved.

[2] Attack works under the assumption that the attacker is able to reconstruct the whole of the generated instance $\mathbf{t}$ (Refer Alg.1)

[3] Available on `https://github.com/mupq/pqm4`

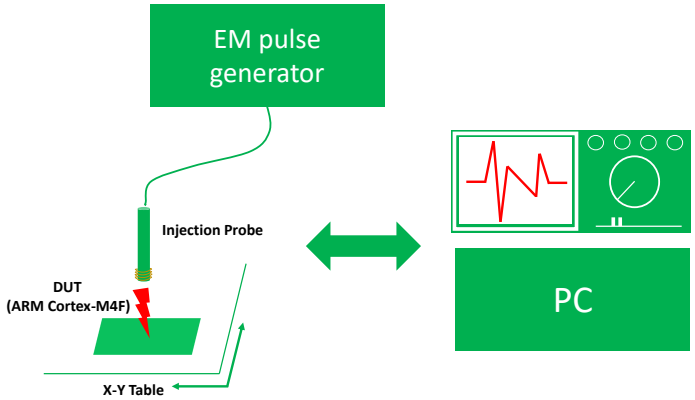
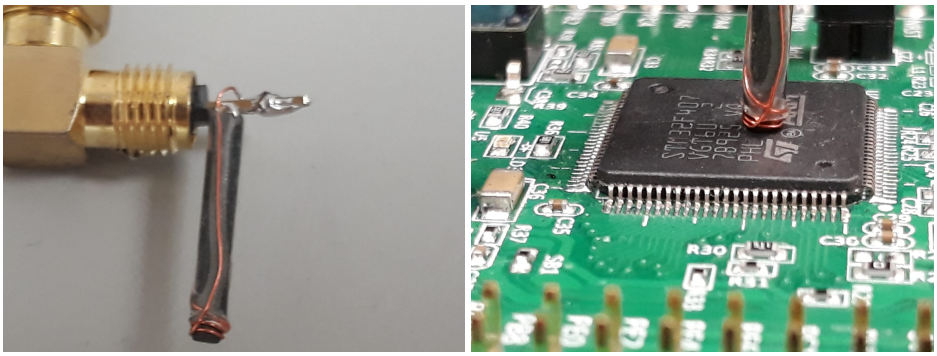


Figure2: Experimental setup for the fault injection

hand-made EM probe designed as a simple loop antenna. Refer to Fig.3 for the EM probe used for our experiments.



(a) (b)

Figure3: (a) Hand-made probe used for our EMFI setup (b) Probe placed over the DUT

4.2 Implementation of EMFI attack

We first analyzed the operations within our target implementations that utilized the nonce value. The nonce value in implementations of all the considered schemes are used as inputs to Extendable Output Function (SHAKE256 for Kyber, NewHope, Dilithium and CSHAKE for FRODO) wherein they are simply

stored to a given location in an array (say A). The array A determines the value of the sampled components. Different polynomials within the schemes are generated simply by changing the nonce value at the same index (memory location) while all the other elements of the array are fixed. The nonce updates in the array A were realized as store instructions (STR instruction for ARM) to the memory. We attempt to skip all these store instructions and by doing so, we ensure that a single random value is used as a nonce to generate both the secret and error components, thus creating a nonce-reuse scenario.

```
ldr     r3,[r5,#28]
stmia   r4!,{r0,r1,r2,r3}
strb.w  r7,[r6,#-132]!
movs    r1,#1
mov     r0,r6
```

(a) Target operation in NewHope

```
movs    r1,#1
add     r0,sp,#52
strb.w  r9,[r6,#32]
movs    r2,#33
movs    r3,#0
```

(b) Target operation in Kyber

```
lsrs    r2,r7,#8
ldr     r3,[pc,#264]
strb.w  r2,[sp,#7]
movw    r2,#4097
mov     r1,sp
```

(c) Target operation in Frodo

```
movs    r1,#128
ldr     r0,[pc,#208]
strb.w  r7,[sp,#44]
add     r1,sp,#12
add     r0,sp,#48
```

(d) Target operation in Dilithium

Figure4: Code snippet from reference implementations of the considered schemes. Our target store operation is highlighted in red.

Hence, we rely on the instruction skip fault model that has been widely studied and practically demonstrated on a range of devices (AVR and ARM microcontrollers) with high repeatability to satisfy our attack requirement [7, 27]. It has been realized over different architectures through multiple fault injection methodologies like laser shots [7, 27], clock [11] glitches and EM injection in addition to serving as a basis for multiple cryptanalytic efforts. We scanned the entire top layer of the chip and could identify a precise location (close to the center of the chip near the ARM logo), where we could achieve *a 100% repeatability in skipping the same store instruction*, thus preventing update of the nonce. Since we are only skipping the update of the nonce, it is important to note that our attack works irrespective of the value of the nonce.

Refer to Fig.4 for the assembly code snippets of the compiled reference implementations of the considered schemes. The faulted store operation in each implementation is highlighted in red. Refer Tab.1 for the fault complexity of our

Table1: Fault Complexity (Number of Instruction Skip Faults) of our attack on the various recommended parameter sets of the considered schemes

Attack Objective	**Fault Complexity**			
	NEWHOPE		FRODO	
	NEWHOPE512	NEWHOPE1024	Frodo-640	Frodo-976
Key Recovery	1	1	1	1
Message Recovery	1	1	1	1

	KYBER			DILITHIUM		
	KYBER512[1]	KYBER768[1]	KYBER1024[1]	Weak[2]	Med.[2]	Rec.[2]	High[2]
Key Recovery	4	6	8	5	7	9	11
Message Recovery	4	6	8	-	-	-	-

nonce misuse attack when applied to the recommended parameter sets of all the schemes. Only a single fault is required in the case of NewHope and Frodo since it is enough to skip the update of the nonce for only the error component and not the secret component. But, in the case of Module-LWE schemes like Kyber and Dilithium, it is required to skip the update of all the nonces used for generation of polynomials. Thus, the number of faults amount to $(k+l)$ when the dimension of the public constant A in these schemes is assumed to $R_q^{k\times l}$.

5 Countermeasures

We have shown that the use of nonces in the reference implementations of all the aforementioned schemes can be easily targeted through fault attacks. The main reason however is due to use of seeds for generation of secret and error components which only vary by one or two bytes due to the nonce. The value of the nonce used primarily decide the difference between the secret and error components. Thus, it becomes important to perform a sanity check on the value of the nonce, which can possibly mitigate the attack. There are a lot of known vulnerable instances of the LWE problem (Refer Sec.2.1 for some of them) and we have realized one of these instances using faults ($\mathbf{s} = \mathbf{e}$). Thus, performing simple checks on the secret and error components of the LWE instances for known trivial weaknesses could also be a potential countermeasure against our attack.

6 Conclusion

In this work, we present practical fault attacks on a number of potential NIST candidates for post quantum cryptography, mainly targeting schemes based on the LWE problem such as NewHope, Kyber, Dilithium, Frodo. We exploit the use of nonces in the sampling procedure in each of these schemes to demonstrate key recovery and message recovery attacks. While long term keys are directly recovered by faulting the key generation procedure, message recovery (session keys) is demonstrated through a novel fault assisted MITM attack on the encryption

procedure on the CCA secure KEM schemes. We perform a practical validation of all our attacks on an ARM Cortex-M4F microcontroller running reference implementations taken from the *pqm4* library.

References

1. Suppressed for blind review
2. Akleylek, S., Bindel, N., Buchmann, J., Krämer, J., Marson, G.A.: An efficient lattice-based signature scheme with provably secure instantiation. In: International Conference on Cryptology in Africa. pp. 44–60. Springer (2016)
3. Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Pöppelmann, T., Schwabe, P., Stebila, D.: Algorithm specifcations and supporting documentation (2017)
4. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-Quantum Key Exchange-A New Hope. In: USENIX Security Symposium. pp. 327–343 (2016)
5. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: International Colloquium on Automata, Languages, and Programming. pp. 403–415. Springer (2011)
6. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber algorithm specifications and supporting documentation (2017)
7. Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on. pp. 105–114. IEEE (2011)
8. Bindel, N., Buchmann, J., Krämer, J.: Lattice-based signature schemes and their sensitivity to fault attacks. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on. pp. 63–77. IEEE (2016)
9. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. Tech. rep., National Institute of Standards and Technology (2017), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`
10. Bruinderink, L.G., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 323–345. Springer (2016)
11. Bruinderink, L.G., Pessl, P.: Differential Fault Attacks on Deterministic Lattice Signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems 2018(3) (2018), `https://eprint.iacr.org/2018/355.pdf`
12. Bushing, S., Sven, M.: Console hacking 2010: Ps3 epic fail. In: Talk at 27th Chaos Communication Congress (2010)
13. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Advances in Cryptology–CRYPTO 2013, pp. 40–56. Springer (2013)
14. Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Loop-abort faults on lattice-based Fiat-Shamir and hash-and-sign signatures. In: International Conference on Selected Areas in Cryptography. pp. 140–158. Springer (2016)
15. Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Side-channel attacks on bliss lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1857–1874. ACM (2017)

16. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: A signature scheme for embedded systems. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 530–547. Springer (2012)
17. Howe, J., Khalid, A., Rafferty, C., Regazzoni, F., O'Neill, M.: On practical discrete gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers (2016)
18. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography 75(3), 565–599 (2015)
19. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS-Dilithium. Tech. rep., National Institute of Standards and Technology (2017), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`
20. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM 60(6), 43 (2013)
21. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. `https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf` (2016)
22. Peikert, C.: How (not) to instantiate ring-lwe. In: International Conference on Security and Cryptography for Networks. pp. 411–430. Springer (2016)
23. Pessl, P.: Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In: Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17. pp. 153–170. Springer (2016)
24. Preskill, J.: Reliable quantum computers. In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. vol. 454, pp. 385–410. The Royal Society (1998)
25. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34 (2009)
26. Shor, P.W.: Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In: International Algorithmic Number Theory Symposium. pp. 289–289. Springer (1994)
27. Trichina, E., Korkikyan, R.: Multi fault laser attacks on protected crt-rsa. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on. pp. 75–86. IEEE (2010)

A Appendix

Algorithm 1: Dilithium Signature scheme

Procedure Dilithium.KeyGen()
- $\rho, \rho' \leftarrow \{0,1\}^{256}$
- $K \leftarrow \{0,1\}^{256}$
- $N = 0$
- **for** i *from* 0 *to* $\ell - 1$ **do**
  - $\mathbf{s}_1[i] = \mathsf{Sample}(\mathsf{PRF}(\rho', N))$
  - $N := N + 1$
- **end**
- **for** i *from* 0 *to* $k - 1$ **do**
  - $\mathbf{s}_2[i] = \mathsf{Sample}(\mathsf{PRF}(\rho', N))$
  - $N := N + 1$
- **end**
- $\mathbf{a} \sim R_q^{k \times \ell} = \mathsf{ExpandA}(\rho)$
- $\mathbf{t} = \mathbf{a} \times \mathbf{s}_1 + \mathbf{s}_2$
- $\mathbf{t}_1 = \mathsf{Power2Round}_q(\mathbf{t}, d)$
- $tr \in \{0,1\}^{384} = \mathsf{CRH}(\rho || \mathbf{t}_1)$
- **return** $pk = (\rho, \mathbf{t}_1), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

Algorithm 2: NewHope KEM scheme

Procedure NewHope.KeyGen()
- $seed \leftarrow \{0, \ldots, 255\}^{32}$
- $z \leftarrow \mathsf{SHAKE256}(64, seed)$
- $publicseed \leftarrow z[0:31]$
- $noiseseed \leftarrow z[32:63]$
- $\hat{\mathbf{a}} \leftarrow \mathsf{GenA}(publicseed)$
- $\mathbf{s} \leftarrow \mathsf{PolyBitRev}(\mathsf{Sample}(noiseseed, 0))$
- $\hat{\mathbf{s}} = \mathsf{NTT}(\mathbf{s})$
- $\mathbf{e} \leftarrow \mathsf{PolyBitRev}(\mathsf{Sample}(noiseseed, 1))$
- $\hat{\mathbf{e}} = \mathsf{NTT}(\mathbf{e})$
- $\hat{\mathbf{b}} = \hat{\mathbf{a}} * \hat{\mathbf{s}} + \hat{\mathbf{e}}$
- **return** $(pk = \mathsf{EncodePK}(\hat{\mathbf{b}}, publicseed), sk = \mathsf{EncodePolynomial}(\mathbf{s}))$

Procedure NewHope.Encrypt($pk \in \mathcal{B}^{7 \cdot n/4+32}, \mu \in \mathcal{B}^{32}, coin \in \mathcal{B}^{32}$)
- $\hat{\mathbf{b}}, publicseed \leftarrow \mathsf{DecodePk}(pk)$
- $\hat{\mathbf{a}} \leftarrow \mathsf{GenA}(publicseed)$
- $\acute{\mathbf{s}} \leftarrow \mathsf{PolyBitRev}(Sample(coin, 0))$
- $\acute{\mathbf{e}} \leftarrow \mathsf{PolyBitRev}(Sample(coin, 1))$
- $\acute{\acute{\mathbf{e}}} \leftarrow \mathsf{Sample}(coin, 2)$
- $\hat{\mathbf{t}} = \mathsf{NTT}(\acute{\mathbf{s}})$
- $\hat{\mathbf{u}} = \hat{\mathbf{a}} * \hat{\mathbf{t}} + \mathsf{NTT}(\acute{\mathbf{e}})$
- $\mathbf{v} = \mathsf{Encode}(\mu)$
- $\acute{\mathbf{v}} = \mathsf{NTT}^{-1}(\hat{\mathbf{b}} * \hat{\mathbf{t}}) + \acute{\acute{\mathbf{e}}} + \mathbf{v}$
- $\mathbf{h} = \mathsf{Compress}(\acute{\mathbf{v}})$
- **return** $c = \mathsf{EncodeC}(\hat{\mathbf{u}}, \mathbf{h})$

Algorithm 3: Kyber KEM scheme

Procedure Kyber.KeyGen()
- $d \leftarrow \{0,1\}^{256}$
- $(\rho, \sigma) := G(d)$
- $N := 0$
- **for** $i = 0; i = k-1; i{+}{+}$ **do**
  - **for** $j = 0; j = k-1; j{+}{+}$ **do**
    - $\mathbf{a}[i][j] \leftarrow \mathsf{Parse}(\mathsf{XOF}(\rho||j||i))$
  - **end**
- **end**
- **for** $i = 0; i = k-1; i{+}{+}$ **do**
  - $\mathbf{s}[i] \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(\sigma, N))$
  - $N := N + 1$
- **end**
- **for** $i = 0; i = k-1; i{+}{+}$ **do**
  - $\mathbf{e}[i] \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(\sigma, N))$
  - $N := N + 1$
- **end**
- $\hat{\mathbf{s}} \leftarrow \mathsf{NTT}(\mathbf{s})$
- $\mathbf{t} = \mathsf{NTT}^{-1}(\hat{\mathbf{a}} * \hat{\mathbf{s}}) + \mathbf{e}$
- $pk := (\mathsf{Encode}_{d_t}(\mathsf{Compress}_q(\mathbf{t}, d_t))||\rho)$
- $sk := \mathsf{Encode}_{13}(\hat{\mathbf{s}}\ mod^+ q)$
- **return** (pk, sk)

Procedure NewHope.Encrypt($pk \in \mathcal{B}^{d_t \cdot k \cdot n/8+32}$, $m \in \mathcal{B}^{32}$, $r \in \mathcal{B}^{32}$)
- ⋮

- $N = 0$
- **for** $i = 0; i = k-1; i{+}{+}$ **do**
  - $\mathbf{r}[i] \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$
  - $N := N + 1$
- **end**
- **for** $i = 0; i = k-1; i{+}{+}$ **do**
  - $\mathbf{e}_1[i] \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$
  - $N := N + 1$
- **end**
- $\mathbf{e}_2 \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$
- $\hat{\mathbf{r}} = \mathsf{NTT}(\mathbf{r})$
- $\mathbf{u} = \mathsf{NTT}^{-1}(\hat{a}^T * \hat{\mathbf{r}}) + \mathbf{e}_1$
- $\mathbf{v} = \mathsf{NTT}^{-1}(\hat{t}^T * \hat{\mathbf{r}}) + \mathbf{e}_2 + \mathsf{Decode}_1(\mathsf{Decompose}_q(m, 1))$
- $\mathbf{c}_1 = \mathsf{Encode}_{d_u}(\mathsf{Compress}_q(\mathbf{u}, d_u))$
- $\mathbf{c}_2 = \mathsf{Encode}_{d_v}(\mathsf{Compress}_q(\mathbf{v}, d_v))$
- **return** $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$