# Generic Side-channel attacks on CCA-secure lattice-based PKE and KEM schemes

Prasanna Ravi[1,2], Sujoy Sinha Roy[3], Anupam Chattopadhyay[2], and Shivam Bhasin[1]

[1] Temasek Laboratories, Nanyang Technological University, Singapore
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3] School of Computer Science, University of Birmingham, United Kingdom

prasanna.ravi@ntu.edu.sg s.sinharoy@cs.bham.ac.uk sbhasin@ntu.edu.sg anupam@ntu.edu.sg

**Abstract** In this article, we demonstrate practical side-channel assisted chosen-ciphertext attacks (CCA) over multiple CCA-secure lattice-based public-key encryption schemes (PKE) and key-encapsulation mechanisms (KEM). Most lattice-based PKE/KEMs suffer from the problem of decryption failures and some of these schemes use forward error correction codes to reduce the failure probability. These error correcting codes, when used within public-key cryptographic schemes, involve computations with secret components and hence might leak sensitive side-channel information. In this work, we identify a side-channel vulnerability in constant-time error correcting codes, which help the attacker distinguish between faulty and valid codewords through the EM/power side-channel information. We exploit the vulnerability to demonstrate a practical chosen-ciphertext attacks on the CCA-secure Round5 algorithm which uses timing attack resistant error correcting code.
We further identify a generic side-channel vulnerability within the CCA transformation steps used in multiple CCA-secure lattice-based PKE/KEM schemes. Exploiting the vulnerability, we demonstrate a practical chosen-ciphertext attack which can be performed on multiple CCA-secure lattice-based PKE/KEM schemes.
We perform experimental validation of our attacks using Electromagnetic measurements observed over optimized implementations of multiple NIST candidates taken from the *pqm4* library, a benchmarking framework for post quantum cryptographic implementations for the ARM Cortex-M4 microcontroller. We thus establish that (1) lattice-based schemes that use error correcting codes, no matter constant-time or not, are vulnerable to power/EM side-channel attacks and (2) the notion that CCA-secure schemes are as insecure as their CPA-secure versions unless suitably masked against side-channel attacks.

## 1 Introduction

Shor's algorithm running on a powerful quantum computer can break widely used public-key algorithms, namely the RSA and Elliptic Curve cryptosystems, in polynomial time. Post-quantum public-key cryptography is a branch which focuses on designing public-key algorithms and protocols that cannot be broken using very powerful quantum computers. In 2017, NIST initiated a project to standardize post-quantum public-key algorithms.

In the first round of NIST's standardization project, there were 69 candidates based on mathematical problems that cannot be solved by present-day classical as well as quantum computers. After intense scrutiny by the the cryptography community, the NIST selected 17 Public-key Encryption (PKE) & Key Encapsulation schemes (KEM) and 9 Digital Signature schemes (DS) for the second round of the standardization project. While the main selection criterion for the first round had been theoretical security and uniqueness of the schemes, the second round will also consider implementation aspects such as performance on both hardware and software platforms as well as resistance to side-channel and fault attacks.

Twelve out of the 26 candidates in the second round are based on computationally infeasible problems from the lattice theory. The majority of these lattice-based candidates are based on some variants of the Learning With Errors (LWE) or Learning With Rounding (LWR) problems. The hardness of the LWE/LWR problems is based on the fact that when properly chosen small errors are added to a system of linear equations, then it becomes computationally infeasible for both present-day classical and quantum computers to solve the system of equations. Naturally, cryptosystems based on LWE/LWR are noisy due to the presence of small errors (which make them secure). Several research works have investigated efficiency aspects of these algorithms and have resulted in improved performance on high-end computers [20] as well as resource-constrained microcontrollers [6, 17, 21, 22, 23]. Interestingly, side-channel security of these lattice-based cryptographic schemes have received only limited attention in [7, 24]. In this paper we analyze side-channel security of several lattice-based candidates that have been selected for the second round of NIST's standardization project.

Although, LWE/LWR-based cryptographic constructions are generally more efficient compared to our present-day elliptic curve or RSA cryptosystems, one major disadvantage is that most of them have non-zero decryption failure rate due to the presence of noise in the ciphertext. For security against a Chosen Ciphertext Attack (CCA), where an attacker can adaptively craft ciphertexts, it is essential for a cryptographic scheme to have negligible decryption failure rate, e.g. in the order of $2^{-\lambda}$ for $\lambda$-bit security. While some schemes [1, 5] tune their parameter sets to achieve negligible decryption failure rates, others [2, 18] use forward error correcting codes. The advantages of using error correcting code is that the LWE/LWR-related parameters (ciphertext modulus and standard deviation of error distribution) can be reduced to improve computation performance and communication bandwidth. This is possible because the error correcting mechanism can compensate the increased noise which comes from the use of smaller ciphertext modulus. For example, LAC is the only scheme which has byte-level ciphertext modulus.

While the application of error correcting code can be lucrative from a performance point of view, it can open new doors for implementation attacks and sometimes over-complicate or even deteriorate performance. D'Anvers *et al.* in [9] demonstrated practical chosen-ciphertext attacks against CCA-secure KEMs LAC [18] and RAMSTAKE [30] by exploiting timing leakage from non-constant-time error correcting codes. Such an attack can be prevented by using constant-time error correcting code. In [31], a constant-time BCH error correcting code was implemented and it was found that the computation time of LAC increases by 40% thus negating the theoretical advantages of using error correcting. Such a performance degradation is not always the case as can be observed in Round5 which uses a constant-time but lightweight error correcting code called 'XEf'. With the help of XEf, Round5 achieves better performance and bandwidth compared to its predecessor Round2 which was a candidate in the first round of NIST's standardization project. The timing attack by D'Anvers *et al.* cannot be used to break the constant-time implementation of Round5.

**Contributions:** In this paper we investigate side-channel security of several second round lattice-based key exchange schemes in the CCA setting.

1. We demonstrate successful *Electromagnetic Emanation-based* (EM-based) side-channel attacks against the Round5 scheme which uses constant-time error correcting code that are inherently resistant against *timing* attacks. We identify a vulnerability within the ECC's decoding procedure which leak EM side-channel information about validity of the codeword. We show that this vulnerability can be exploited to perform chosen-ciphertext attacks that lead to retreival of the long term secret key of CCA-secure Round5 algorithm.
2. We also identify a very similar side-channel vulnerability in the operations within the CCA transformation used in lattice-based KEMs. Exploiting the vulnerability in a similar manner, we craft chosen-ciphertexts and introduce differential side-channel behavior during the execution of CCA transform steps in the decapsulation operation. This generic attack leads to retrieval of long term secret keys from CCA secure KEMs.
3. We perform experimental validation of our attacks on the implementations of the aforementioned NIST candidates obtained from the *pqm4*[3] public library, a testing and benchmarking framework for post quantum cryptographic schemes on the ARM Cortex-M4 microcontroller [16].
4. We thus establish/strengthen the notion that 1) lattice-based key exchange schemes using constant-time error correcting are not secure in the present of power side-channel attacker and 2) that CCA-secure schemes are as insecure as their CPA-secure versions unless suitably masked against possible side-channel attacks.

**Organization of the Paper:** This paper is organized as follows. Section 2 covers the necessary background by introducing the required concepts, Section 3 presents our side-channel-assisted chosen-ciphertext attack on the CCA-secure

Round5 algorithm exploiting side-channel vulnerability in its constant time error correcting code. Section 4 presents our chosen-ciphertext attack exploiting a very similar vulnerability in the CCA transformation utilized by multiple lattice-based PKE/KEMs. Section 5 discusses potential countermeasures against our attacks followed by Section 7 which presents our conclusions.

## 2 Lattice Preliminaries

In this section, we briefly review some background on the Learning With Errors (LWE) and the Learning With Rounding (LWR) problem. Further, we briefly explain the LPR Encryption scheme [19] based on which multiple lattice-based NIST candidates have been developed. We also touch upon the issue of decryption failures and usage of error correction codes in lattice-based schemes followed by the well-known FO transformation [13] used by several lattice-based PKE/KEMs for security in the CCA security model.

### 2.1 Notation

We denote the ring of integers modulo a prime $q$ as $\mathbb{Z}_q$. The polynomial ring $\mathbb{Z}_q(X)/\phi(X)$ is denoted as $R_q$ where $\phi(X)$ is its reduction polynomial. Let the module of dimension $k \times \ell$ be the ring of matrices of the size $k \times \ell$ with each element in $R_q$. Polynomials in $R_q$ are denoted in bold lower case letters and the $i^{th}$ coefficient of a polynomial $\mathbf{a} \in R_q$ is denoted as $\mathbf{a}[i]$. Similarly, matrices/vectors in $\mathbb{Z}_q^{k \times l}$ are denoted in bold upper case letters and their individual elements are represented similar to the coefficients of polynomials in ring $R_q$. Multiplication of two polynomials $\mathbf{a}$ and $\mathbf{b}$ is denoted as $\mathbf{c} = \mathbf{a} \times \mathbf{b}$. Byte arrays of length $n$ are denoted as $\mathcal{B}^n$.

An element $x \in \mathbb{Z}_q$ rounded to a lower modulus $p$ with $p < q$ is denoted as $\lfloor x \rceil_{q \to p}$. The same rounding operation can be applied coefficient-wise or element wise to larger elements such as polynomials in rings $R_q$, modules in rings $R_q^{k \times \ell}$ or matrices/vectors in $\mathbb{Z}_q^{k \times \ell}$.

### 2.2 LWE problem

The Learning With Errors (LWE) problem, introduced by Regev in 2009 [27] is one of the most well known average-case hard problems on which several lattice-based NIST candidates are built on. Solving the LWE problem on random lattices in the average case is at-least as hard as solving the related Bounded Distance Decoding (BDD) problem on the same lattices in the worst case. There are two versions of the LWE problem - Search LWE and Decisional LWE. The search variant of the LWE problem requires the attacker to solve for a secret $\mathbf{S} \in \mathbb{Z}_q^{\ell \times n}$ given polynomially many LWE instances of the form

$$\Big(\mathbf{A}, \mathbf{T} = \mathbf{A} \times \mathbf{S} + \mathbf{E}\Big) \in (\mathbb{Z}_q^{k \times \ell} \times \mathbb{Z}_q^{k \times n}),$$

with $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{k\times\ell})$ and $\mathbf{E} \leftarrow \mathcal{D}_\sigma(\mathbb{Z}_q^{k\times n})$ where $\mathcal{U}$ represents uniform distribution and $\mathcal{D}_\sigma$ represents a discrete error distribution with standard deviation $\sigma$. The decisional variant of this problem requires the attacker to distinguish similarly structured ordered pairs of LWE instances $(\mathbf{A}, \mathbf{T}) \in (\mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^{k\times n})$ from uniformly random pairs $\mathcal{U}(\mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^{k\times n})$.

## 2.3 LWR problem

While errors in the LWE problem are sampled from an error distribution $\mathcal{D}_\sigma(\mathbb{Z}_q^{k\times n})$ and explicitly added to the linear system of equations $(\mathbf{A} \times \mathbf{S}) \in \mathbb{Z}_q^{k\times n}$, Banerjee *et al.* [3] showed that it is possible to implicitly add errors to the linear system $\mathbf{A} \times \mathbf{S}$ by scaling down the coefficients from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ with $p < q$ as follows:

$$\left(\mathbf{A}, \mathbf{T} = \left\lfloor \frac{p}{q}(\mathbf{A} \times \mathbf{S}) \right\rceil\right) \in (\mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_p^{k\times n})$$

Such ordered pairs are referred to as the Learning With Rounding (LWR) instances which are also associated with their corresponding search and decisional problems that are at least as hard as the search and decisional problems over LWE instances [19].

## 2.4 Structured variants of LWE/LWR problem

Cryptographic schemes built upon the aforementioned standard version of the LWE and LWR problems involve costly matrix-vector arithmetic and impractical key-sizes as they both scale quadratically with the dimension of the lattice. Thus, all LWE/LWR based NIST candidate PKE/KEMs, except for FRODO (based on the standard LWE problem) are based on algebraically structured variants of the LWE and LWR problems known as Ring/Module-LWE (RLWE/MLWE) and Ring/Module-LWR (RLWR/MLWR) problems respectively. The ring variant of the LWE/LWR problem (RLWE/RLWR) [19] deals with computation over polynomials in polynomial rings $R_q$ with $\mathbf{s} \leftarrow \chi_{\mathbf{s}}(R_q)$ and $\mathbf{e} \leftarrow \chi_{\mathbf{e}}(R_q)$ such that the corresponding RLWE instance is $(\mathbf{a}, \mathbf{t} = \mathbf{a} \times \mathbf{s} + \mathbf{e}) \in (R_q \times R_q)$ and RLWR instance is $(\mathbf{a}, \mathbf{t} = \lfloor \mathbf{a} \times \mathbf{s} \rceil_{q\to p}) \in (R_q \times R_p)$.

The module variant deals with computations over vectors/matrices of polynomials in $R_q^{k_1\times k_2}$ with $(k_1, k_2) > 1$. In this paradigm, the public parameter $\mathbf{a} \leftarrow \mathcal{U}(R_q^{k_1\times k_2})$ and the secret $\mathbf{s}$ and error $\mathbf{e}$ are sampled from $\chi_{\mathbf{s}}(R_q^{k_2})$ and $\chi_{\mathbf{e}}(R_q^{k_1})$ respectively. Thus, the corresponding MLWE instance is $(\mathbf{a}, \mathbf{t} = \mathbf{a} \times \mathbf{s} + \mathbf{e}) \in (R_q^{k_1\times k_2}, R_q^{k_1})$ and MLWR instance is $(\mathbf{a}, \mathbf{t} = \lfloor \mathbf{a} \times \mathbf{s} \rceil_{q\to p}) \in (R_q^{k_1\times k_2}, R_p^{k_1})$. Moreover, it can be easily seen that the special case of $k = 1$ in the Module-LWE/LWR problem is nothing but the corresponding Ring-LWE/LWR problem. Both the ring and module variants provide asymptotic improvements in both computational times and key-size requirements, compared to their corresponding unstructured variants.

### 2.5 A Generic Framework for LWE/LWR based PKE schemes

Most of the lattice-based schemes based on the LWE/LWR problem, contain in their core a public-key encryption scheme secure in the chosen-plaintext (CPA-secure) model. Moreover, it is important to note that PKE core within these schemes is based on a general paradigm/framework proposed by Lyubashevskey, Peikert and Regev in 2010 [19], better known as the "LPR Encryption scheme". We describe the LPR encryption scheme in detail as it captures the basic essence of multiple lattice-based PKE and KEMs. But, we generalize its description so that specific parameter choices such as the structure of the underlying ring ($R_q$, $R_q^{k\times k}$, $\mathbb{Z}_q^{n\times n}$), choice to use error correcting codes to reduce decryption failures ($\mathcal{R}_{choice} = 0/1$) and relative sizes of the rounding moduli $(p, q)$ can be used to describe specific schemes using this generic framework.

**2.5.1 LPR Encryption Scheme [19]:** We define $\mathsf{gen}$ to be an extendable output function that takes as input a seed $\rho$ and outputs a matrix $\mathbf{A} \in R_q^{k\times k}$. We also define $\mathsf{Ecc\_Enc}_{\mathcal{R}}$ and $\mathsf{Ecc\_Dec}_{\mathcal{R}}$ to denote the encoding and decoding functions corresponding to the error correcting code $\mathcal{R}$. Moreover, additional message encoding (resp. decoding) functions $\mathsf{enc}$ (resp. $\mathsf{dec}$) are used to convert binary messages ($\in \mathbb{Z}_2^n$) into elements in the underlying ring and vice versa. The generalized version of the LPR encryption scheme is described in Alg. 1. Please note this is only a high level description with the lower level technical details ignored for generalization.

From Alg.1, we can see that specific parameter choices can help describe different schemes based on the same framework. For instance, choice of the parameteric tuple ($R_q = \mathbb{Z}_q^n, p = q, k > 1$) describes the encryption scheme of FRODOKEM which is based on the standard LWE problem, while the choice of tuple ($R_q \neq \mathbb{Z}_q^n, p = q, k = 1$) describes the encryption scheme of NewHope or LAC, which is based on the Ring-LWE problem. While the encryption scheme of Round5 based on the Ring-LWR problem can be described using the tuple ($R_q \neq \mathbb{Z}_q^n, p \neq q, k = 1$), schemes such as Saber and Kyber can be described using the tuple ($R_q \neq \mathbb{Z}_q^n, p \neq q, k > 1$).

#### Security and Correctness of LPR Encryption Scheme

We briefly explain the basic principles underlying the security and correctness of the LPR encryption scheme (Refer Alg.1). We explain the scheme using LWE instances for simplicity, we note that the same also applies to schemes using LWR instances or combinations of both. Firstly, the key-generation procedure generates the public parameter $\mathbf{a}$ and further samples the secret and error vectors $\mathbf{s}$ and $\mathbf{e}$ respectively. The LWE instance $(\mathbf{a}, \mathbf{t})$ (Line 6 in $\mathsf{PKE.KeyGen}$) is generated and output as part of the long term public-key while the secret $\mathbf{s}$ used to generate the LWE instance is considered as part of the long term secret key. The indistinguishability of the public key (LWE instance) from a random element in the same space comes from the hardness of solving the decisional LWE problem while the the computational hardness of retrieving the secret key $\mathbf{s}$ from the public key $(\mathbf{a}, \mathbf{t})$ comes from the hardness of solving the search variant of the LWE problem.

The encryption procedure takes the public key $pk$ and the message $m$ to be encrypted as input and generates its corresponding ciphertext. The ciphertext consists of two LWE instances whose secret $(\mathbf{s}')$ and error components $(\mathbf{e}', \mathbf{e}'')$ are internally generated within the encryption procedure. The first LWE instance $\mathbf{u}$

**Algorithm 1:** Generic framework of LWE/LWR based PKE schemes

```
Procedure PKE.KeyGen()
    $publicseed \leftarrow \mathcal{U}(\mathcal{B}^{32})$
    $\mathbf{a} = \mathsf{gen}(publicseed) \in R_q^{k \times k}$
    $\mathbf{s} \leftarrow \chi_{\mathbf{s}}(R_q^k)$
    $\mathbf{e} \leftarrow \chi_{\mathbf{e}}(R_q^k)$
    $\mathbf{t} = \lfloor \mathbf{a}^T \times \mathbf{s} + \mathbf{e} \rceil_{q \to p} \in R_p^k$
    return $(pk = \mathsf{EncodePK}(\mathbf{t}, publicseed), sk = \mathsf{EncodeSK}(\mathbf{s}))$
------------------------------------------------------------
Procedure PKE.Encrypt($pk, m \in \mathcal{B}^{32}, r \in \mathcal{B}^{32}$)
    $(\mathbf{t}, publicseed) = \mathsf{DecodePK}(pk)$
    $\mathbf{a} \leftarrow \mathsf{gen}(publicseed)$
    $\mathbf{s}' \leftarrow \chi_{\mathbf{s}'}(R_q^k)$
    $\mathbf{e}' \leftarrow \chi_{\mathbf{e}'}(R_q^k)$
    $\mathbf{e}'' \leftarrow \chi_{\mathbf{e}''}(R_q)$
    $\mathbf{t}_r = \lfloor \mathbf{t} \rceil_{q \to p} \in R_p^k$
    $\mathbf{u} = \lfloor \mathbf{a}^T \times \mathbf{s}' + \mathbf{e}' \rceil_{q \to p} \in R_p^k$
    if $\mathcal{R}_{choice} = 1$ then
        $c = \mathsf{Ecc\_Enc}_{\mathcal{R}}(m)$
    else
        $c = m$
    $\mathbf{v} = \lfloor \mathbf{t}_r^T \mathbf{s}' + \mathbf{e}'' + \mathbf{enc}(c) \rceil_{q \to t} \in R_t$
    return $ct = \mathsf{EncodeCT}(\mathbf{u}, \mathbf{v})$
------------------------------------------------------------
Procedure PKE.Decrypt($ct, sk$)
    $\mathbf{u}, \mathbf{v} = \mathsf{DecodeCT}(ct)$
    $\mathbf{s} = \mathsf{DecodeSK}(sk)$
    $\mathbf{u}' = \lfloor \mathbf{u} \rceil_{p \to q}$
    $\mathbf{v}' = \lfloor \mathbf{v} \rceil_{t \to q}$
    $\mathbf{r} = (\mathbf{v}' - (\mathbf{u}')^T \mathbf{s}) \in R_q$
    $c' = \mathbf{dec}(\mathbf{r})$
    if $\mathcal{R}_{choice} = 1$ then
        $m' = \mathsf{Ecc\_Dec}_{\mathcal{R}}(c')$
    else
        $m' = c'$
    return $m'$
```

is generated using the public parameter $\mathbf{a}$ (Line 8 in $\mathsf{PKE.Encrypt}$) while another LWE instance is generated using the component $\mathbf{t}$ in the public key as its public parameter. For simplicity, we will assume that $\mathcal{R}_{choice} = 0$ and that we do not use error correcting codes (i.e) $c = m$. Thus, the message $c$ to be encrypted is first converted into an element in the underlying ring using the $\mathsf{enc}$ function before being added to the second LWE instance to generate $\mathbf{v}$ (Line 13 in $\mathsf{PKE.Encrypt}$). Essentially, the encoded message is added to $\mathbf{v}$ in order to hide it within the pseudo-random LWE instance. Both $\mathbf{u}$ and $\mathbf{v}$ are output as part of the ciphertext $ct$.

Further, the decryption procedure extracts $\mathbf{u}$ and $\mathbf{v}$ from the ciphertext $ct$ and essentially computes $\mathbf{r} = \mathbf{v}' - \mathbf{u}' \times \mathbf{s}$ where $\mathbf{s}$ is the long term secret key and $\mathbf{v}'$ and $\mathbf{u}'$ are rounded versions of $\mathbf{v}$ and $\mathbf{u}$ respectively. Let the errors introduced due to rounding of a component $\mathbf{x}$ be represented as additional error components $\mathbf{e_x}$ and hence $\mathbf{v}'$ can be represented as $\mathbf{v}' = \mathbf{v} + \mathbf{e_x}$ and the same applies to $\mathbf{u}'$. The computed element $\mathbf{r}$ is approximately equal to the encoded form of the message $\mathbf{m}$ with high probability and this can be clearly seen from the computation below:

$$\begin{aligned}
\mathbf{r} &= \mathbf{v}' - \mathbf{u}' \times \mathbf{s} \\
&= \mathbf{v} + \mathbf{e_v} - (\mathbf{u} + \mathbf{e_u})\mathbf{s} \\
&= \mathbf{ts}' + \mathbf{e}'' + \mathsf{enc}(c) + \mathbf{e_v} - (\mathbf{as}' + \mathbf{e}' + \mathbf{e_u})\mathbf{s} \\
&= \mathbf{ass}' + \mathbf{es}' + \mathbf{e}'' + \mathsf{enc}(c) + \mathbf{e_v} - (\mathbf{as}' + \mathbf{e}' + \mathbf{e_u})\mathbf{s} \\
&= \mathsf{enc}(c) + \mathbf{es}' + \mathbf{e}'' + \mathbf{e_v} - \mathbf{e}'\mathbf{s} - \mathbf{e_u}\mathbf{s} \\
&= \mathsf{enc}(c) + \hat{\mathbf{e}}
\end{aligned}$$

Thus, as long as the total error $\hat{\mathbf{e}}$ is less than a certain threshold $q_t$ (based on the parameters of the scheme), the computed component $\mathbf{r}$ can be correctly decoded back to the message $c$.

### 2.6 Error Correcting Codes in LWE/LWR-based schemes

Almost all semantically secure lattice-based PKE scheme based on the LWE/LWR problem are associated with a certain decryption failure probability. Thus, KEMs built upon such PKE schemes may encounter a failure event when the two involved parties fail to establish a shared secret key. Designers strive to achieve a non-negligible failure probability if not zero, for PKE schemes since it comes with a number of obvious advantages. Firstly, zero or negligible failure probability reduces the attacker's chances of triggering decryption failures and performing cryptanalysis using such observed decryption failures, similar to the attacks demonstrated in several previous works [4, 10, 12]. Secondly, having a negligible failure probability also satisfies a formal requirement for PKE schemes to achieve security against adaptive chosen-ciphertext attacks in the CCA-secure model [8].

But on the contrary, having negligible decryption failures is also associated with some disadvantages. Sometimes, reducing decryption failures could come at the cost of reducing the size of the error $\mathbf{e}$ of the LWE instance which reduces security against attacks trying to solve the underlying lattice problem. This could impose a severe penalty on performance since the lost security is compensated by increasing the dimension of the underlying lattice. Thus, one can always observe a fine trade-off between decryption failure probability and performance of LWE/LWR-based PKE/KEMs.

In order to strike a balance, certain LWE/LWR based PKE schemes opted for usage of error correcting codes (ECC) to artificially reduce decryption failures by correcting the errors in the decrypted message [2, 18]. The message $m$ is first converted into a codeword $c$ (Line 10 in $\mathsf{PKE.Encrypt}$ of Alg.1) which is used by the encryption scheme as the modified message. Further during decryption, the decrypted codeword $c'$ is decoded into $m'$ (Line 9 in $\mathsf{PKE.Decrypt}$) as the output of the decryption operation. If the number of bit errors in $c'$ are less than the maximum error correcting capability of the ECC, then $m' = m$ and hence will result in correct decryption.

Thus, usage of ECC enable schemes to achieve negligible decryption failure probability required for CCA security without compromising on competitive parameter settings. Thus, LWE/LWR-based lattice PKE/KEMs can be broadly divided into two classes based on the usage of ECC. Among all such current NIST candidates for PKE/KEMs, LAC and Round5 utilize ECC while the majority schemes rely on other conventional parametric changes to achieve the desired decryption failure probability.

While LAC relies on the BCH [15] as its ECC, Round5 utilizes a constant-time linear parity code XEf [2] as its ECC to tackle decryption failures. But, usage of error correcting codes also comes with its own set of disadvantages such as complexity of implementation and possible susceptibility to side-channel attacks. D'Anvers *et al.* [9] demonstrated chosen-ciphertext attacks against two CCA-secure KEMs LAC and RAMSTAKE[1] through exploitation of timing side-channel information from the decoding procedure execution of its ECC. Their main idea was to exploit the difference in execution times of the decoding procedure of ECC between valid and non-valid/faulty codewords.

This differential behaviour of the decoding procedure helps the attacker distinguish between valid and faulty decrypted codewords for chosen ciphertexts, which subsequently leads to direct retrieval of the secret key. Though their attack only target non-constant time ECC, it has definitely necessitated discussions about hardening implementation of ECC procedures against possible side-channel attacks. Moreover, this has also raised questions about existence of other side-channel vulnerabilities in implementation of ECCs. As a first step towards answering this question, we identify side-channel vulnerabilities in constant-time implementations of error correcting codes and exploit them to perform chosen-ciphertext attacks against CCA-secure lattice-based PKE/KEMs.

[1] RAMSTAKE is a Mersenne prime based KEM which was not selected for the second round of the NIST standardization process

### 2.7 CCA transformation

The LPR encryption scheme is provably secure in the CPA (Indistinguishability under Chosen-Plaintext Attack) secure model. A PKE scheme is said to be secure in the CPA model when the scheme is secure against an attacker who can obtain ciphertexts for arbitrarily crafted plaintexts of the attacker's choice. But, the LPR encryption scheme does not provide security against attackers who can decrypt arbitrary ciphertexts adaptively chosen by the attacker. This security model which assumes a more powerful attacker is known as the CCA (Indistinguishability under Adaptive Chosen-Ciphertext Attack) security model.

A lattice-based PKE scheme secure in the CPA model can be converted into a CCA-secure PKE/KEM using the well known post-quantum variant of the Fujisaki-Okamoto transformation [13]. This transformation is used by multiple lattice-based NIST candidates to achieve CCA security. It utilizes a pair of hash functions $\mathcal{H}$ and $\mathcal{G}$ and operates over the top of the encryption and decryption schemes respectively as shown in Alg.2. Upon decapsulation failures, some schemes simply return a failure symbol $\perp$, while certain schemes such as Kyber [5] and Saber [11] use a different variant of the FO transformation, which returns a pseduo random value as the shared secret key, that is generated by simply hashing the ciphertext with the secret key.

In theory, the FO transformation helps protect KEMs against chosen-ciphertext attacks since the validity of ciphertexts are checked through the re-encryption procedure during decapsulation. Thus in theory, the attacker only sees decapsulation failures for any handcrafted (invalid) ciphertext. Moreover, the decryption procedure is encapsulated within the decapsulation procedure and hence the attacker cannot directly observe the output of the decryption module. This provides strong theoretical security guarantees against chosen-ciphertext attacks

**Algorithm 2:** FO-transformation for CCA-secure KEM

```
1 Procedure KEM.Encaps($pk$)
2     $m \leftarrow \mathcal{U}(\mathcal{B}^{32})$
3     $r = \mathcal{G}(m, pk)$
4     $ct$ = PKE.Encrypt($pk, m, r$)
5     $K = \mathcal{H}(r)$
6     return $ct, K$
7 ------------------------------------------------
1 Procedure KEM.Decaps($sk, pk, ct$)
2     $m'$ = PKE.Decrypt($sk, ct$)
3     $r' = \mathcal{G}(m', pk)$
4     $ct'$ = PKE.Encrypt($pk, m', r'$)
5     if $ct' = ct$ then
6         return $K = \mathcal{H}(r')$
7     else
8         return $K = \perp$;
```

which were previously possible over CPA-secure PKE/KEMs. But, in this work we show that side-channel information from certain operations within the FO transformation leak information about the decrypted message, which can be used to mount chosen-ciphertext attack against CCA secure schemes using the FO transformation.

### 2.8 Test Vector Leakage Assessment (TVLA)

The Test Vector Leakage Assessment (TVLA) [14] is a popular conformance-based methodology in side-channel analysis which has been widely used by both academia and the industry to perform side-channel evaluation of embedded cryptographic implementations. TVLA uses the well known Welsh's $t$-test over two sets of measurements and tries to identify the differentiating features in them. By testing for a null hypothesis such that the mean of two sets is identical, a *PASS/FAIL* decision is taken. Let the two sets of measurements be denoted as $t_r$ and $t_f$. The formulation of TVLA is as follows:

$$TVLA = \frac{\mu_r - \mu_f}{\sqrt{\frac{\sigma_r^2}{m_r} + \frac{\sigma_f^2}{m_f}}} \ , \tag{1}$$

where $\mu_r$, $\sigma_r$ and $m_r$ are mean, standard deviation and cardinality of the trace set $t_r$ and $\mu_f$, $\sigma_f$ and $m_f$ are mean, standard deviation and cardinality of the trace set $t_f$. The null hypothesis is accepted only if the TVLA value stays in the range $[-4.5, 4.5]$ with a confidence of 99.9999% [14]. A rejected null hypothesis implies that the two trace/data sets are different and that they might leak some side-channel information and hence is considered to *FAIL* the test. While TVLA is mainly used as a metric for side-channel evaluation, it has been also been used as a tool for feature selection in multiple cryptanalytic efforts [26]. In this work, we use TVLA as a tool for performing binary classification of side-channel measurements into two classes.

## 3 Side-Channel analysis of Error Correcting Codes

In this section, we demonstrate our side-channel assisted chosen-ciphertext attack on CCA-secure lattice-based schemes through exploitation of EM side-channel information from execution of constant-time error correcting codes. In particular, we demonstrate an attack on the Round5 CCA-secure KEM which utilizes constant-time error correcting procedures that are resistant to timing attacks.

### 3.1 Prior works

D'Anvers *et al.* [9] demonstrated successful timing attacks on two KEMs, namely LAC and RAMSTAKE, which use non-constant-time error correction algorithms. They showed that the difference in the execution times, which can also be observed for the whole decapsulation operation, can be used to distinguish validity of decrypted codewords for chosen ciphertexts with very high probability. This timing information subsequently facilitated their attack over the aforementioned two CCA-secure KEMs. However, their timing attack cannot be used to extract secret from schemes which use constant-time error correction. Round5 is one such lattice-based key exchange scheme that is resistant against the timing attacks due to the application of constant-time error correction algorithm.

### 3.2 Side-channel vulnerability of Error Correcting Codes

We attempt to analyze the power/EM side-channel behaviour of constant-time decoding procedures of error correcting codes which are inherently resistant to timing attacks. As a target algorithm for our evaluation, we choose the constant-time XEf error correction code which has been used in the Round5 scheme [2]. XEf is an f-bit forward error correcting block code which can always correct atleast f errors in any given codeword. In the following part of this sub-subsection, we describe the functioning of XEf code and then highlight its differential behavior.

**3.2.1 XEf error correction code:** Let the $k$-bit message to be encoded be $m = (m_{k-1}, m_{k-2}, \ldots, m_0)$ and its corresponding binary polynomial be defined as $mp = m_{k-1}x^{k-1} + \ldots + m_1 x + m_0$. XEf is a linear parity code and works with $2f$ registers ($r_i$ for $i \in \{0, 2f-1\}$) each of size $l_i$ which are computed as follows:

$$r_i = mp \mod (x^{l_i} - 1)$$

The codeword $c = \mathsf{Ecc\_Enc}_{\mathcal{R}}(m)$ consists of the message $m$ appended with the register set $r$ (i.e) $c = (m|r)$, which amounts to a total of $\mu$ bits where $\mu = k + \sum_{i=0}^{2f-1} l_i$. This codeword $c$ is used by the encryption procedure as the modified message to be encrypted. The decryption module firstly decrypts the ciphertext $ct$ to retrieve the codeword $(m'|r'')$ which is subsequently passed to the $\mathsf{Ecc\_Dec}_{\mathcal{R}}$ procedure. The decoding procedure computes the register set $r'$ for the received message $m'$. Further, $r''$ is compared with the computed $r'$ and those bits $j$ in the received message $m'$ are flipped based on the following condition:

$$\sum_{i=1}^{2f} \left( \left( r''_{(i, j \bmod l_i)} - r'_{(i, j \bmod l_i)} \right) \mod 2 \right) \geq (f+1)$$

One of the main advantages of this approach, as claimed by the authors is that it can be implemented devoid of any look-up tables and also in constant-time, thus resistant to timing attacks. But, we show in the following discussion that the constant time implementation of XEf still leaks side-channel information about validity of the codewords during its decoding procedure.

**3.2.2 Side-channel vulnerability of XEf:** We analyze the decoding procedure of XEf in the optimized implementation of Round5 taken from the open-source *pqm4* library [16], a benchmarking and testing framework for PQC schemes on the ARM Cortex-M4 family of microcontrollers. We ported the optimized implementation of Round5 (R5ND_1KEM_5d as a representative variant in particular) to the STM32F4DISCOVERY board (DUT) housing the STM32F407, ARM Cortex-M4 microcontroller. The implementation (compiled with `-O3 -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16`) was running at a clock frequency of 24 MHz. We used 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).

The XEf decoding procedure involves a decision-making operation to decide the position of the bits to be flipped in the decrypted message. One of the steps in this decision-making operation involves a majority logic which takes as input a modified form of the register set $r$ as input, which we denote as $\hat{r} = \{\hat{r}_i\}$ for $i \in \{0, 2f-1\}$. Moreover, this majority logic operation is implemented using bitwise operations such as bitwise and (&), bitwise xor ($\wedge$) and bitwise not ($\sim$) operations over $r''$.

We observe over multiple trials that for a correct codeword with zero erroneous bits, all the inputs to the majority logic are always zero. On the other hand, for a faulty code word with one or more erroneous bits, at least one of the input registers $\hat{r}_i$ are non-zero. Since the input to the majority logic differs based on the validity of the codeword, the corresponding computations also differ based on the validity of the codeword.

This differential behavior can be identified through the power/EM side-channel. In particular, we consider the all zero codeword ($c = 0$) as the valid codeword. An all zero codeword ($c = 0$) is valid and corresponds to encoding an all zero message ($m = 0$), since XEf is a linear parity code. For faulty codewords, we consider a single bit error in $c = 0$, where one of the bits of $c$ is 1. For simplicity, we denote the valid codeword case to be $\mathsf{HW}(c) = 0$ and the faulty case to be $\mathsf{HW}(c) = 1$.

Please refer Fig.3 in the appendix for the screenshots from the Openocd-Open On-Chip debugger tool [25] showing values of the internal general purpose registers during operation of the targeted majority logic for different cases. We can clearly see all the internal registers are zero when $\mathsf{HW}(c) = 0$, while more than one registers are non-zero when $\mathsf{HW}(c) = 1$.

There are two possible types of faulty codewords depending on the location of the erroneous bit in the codeword $c = (m|r)$. The erroneous bit could either be present in the message portion ($m$) or the register set portion ($r$) of the codeword. Comparing Fig.3(b) and 3(c), we can clearly see that the number of non-zero registers during the targeted majority logic operation is higher when the erroneous bit is in the message $m$ (i.e) $\mathsf{HW}(m) = 1$, as opposed to the case when the erroneous bit is in the register set portion (i.e) $\mathsf{HW}(\mathsf{r}) = 1$. Thus, we only consider faulty codewords with the erroneous bit in $m$ for maximum distinguishability. Henceforth, faulty codewords only mean single bit errors in $m$ and not in $r$.

We perform EM measurements of the majority logic in order to distinguish this differential behaviour through the EM side-channel. EM Measurements were observed from the same DUT using a near-field probe and are processed using a Lecroy 610Zi oscilloscope at a sampling rate of 500MSam/sec. We utilize the standard TVLA approach based on the Welch's $t$-test to differentiate the EM measurements. It is important to note that we consider absolute $t$-test values since their signs does not matter for our attack. We collect two sets of 25 measurements corresponding to repeated executions of the decoding operation for valid codewords ($\mathsf{HW}(c) = 0$). Let the two sets of measurements be denoted as $t_{v'}$ and $t_{v''}$. Refer Fig.1(a) for the $t$-test results between $t_{v'}$ and $t_{v''}$. Since both are valid codewords and all the operating registers are zero, the operations within the majority logic corresponding to the two sets of measurements are the same. Hence, we do not observe any peaks (greater than the pass/fail threshold 4.5) in the TVLA plot.

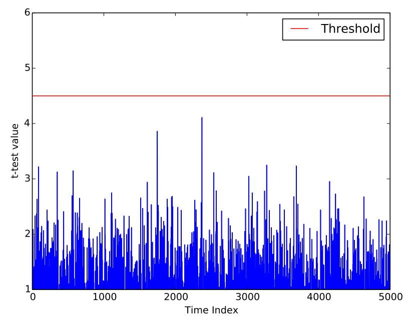
We collect another set of 25 measurements corresponding to repeated executions for the faulty case when $\mathsf{HW}(c) = 1$, which is denoted as $t_f$. Refer Fig.1(b) for the $t$-test results between $t_{v'}$ and $t_f$. Since the data being operated on are very different in the two cases, we are able to observe clear peaks (greater than the pass/fail threshold 4.5) indicating that the two sets of measurements are different.

So, an attacker collects a set of reference traces corresponding to $\mathsf{HW}(c) = 0$, which we denote as $t_{ref}$. During the attack, the attacker can then collect equal number of EM traces for those computations which he wants to classify. We denote this trace set as $t_{attack}$. He can then simply compute the TVLA between $t_{ref}$ and $t_{attack}$ and based on the number of peaks observed in the $t$-test plot, can decide upon the case to which the $t_{attack}$ traces belong to. Thus, we have shown that the validity of the codeword (even in the presence of a single bit error) can be easily distinguished using only a few EM measurements.

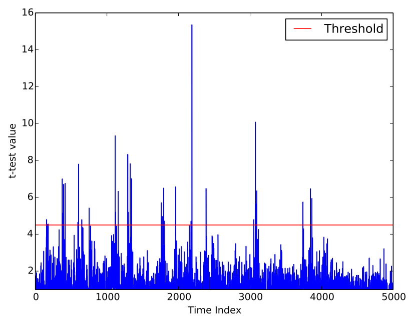
### 3.3 Side-channel Attack on Round5

In this section, we show that the identified side-channel vulnerability of XEf's decoding procedure can be exploited to perform a practical chosen-ciphertext attack on the CCA-secure Round5 PKE/KEM.

**3.3.1 Adversary Model:** We assume that the attacker has complete physical access to the device performing the decapsulation operation. The attacker must be capable of triggering the decapsulation operation arbitrarily many number of times in order to decapsulate ciphertexts of the attacker's choice. The attacker does not require knowledge of the output of the decapsulation operation. We assume that the attacker is able to passively observe the behaviour of the device during the decapsulation operation through side-channels such asEM.



(a)



(b)

Figure1: TVLA results based on Welch's $t$-test between (a) decoding of two valid codewords (b) decoding of valid and faulty codeword

**Algorithm 3:** Round5 CPA.PKE scheme

```
1 Procedure Round5.Decrypt($ct, sk$)
2     $\mathbf{u}, \mathbf{v} = \text{DecodeCT}(ct)$
3     $\mathbf{s} = \text{DecodeSK}(sk)$
4     $\mathbf{w} = \text{Truncate}_{\mu}(\mathbf{u} \times \mathbf{s} \bmod (x^{n+1} - 1))$
5     $\mathbf{c}'' = \text{Round}_{p \to 2}\left(\frac{p}{t}\mathbf{v} - \mathbf{w}, \frac{1}{2}\right)$
6     $\mathbf{m}'' = \text{Ecc\_Dec}(\mathbf{c}'')$
```

#### 3.3.2 Description of Decryption in Round5:

Refer Alg.3 for the decryption operation of the Round5 PKE scheme[2]. Since our attack only deals with the decryption procedure, we do not describe its corresponding encryption and key generation procedures, for brevity. The reader is referred to [2] for complete details of the PKE scheme of Round5.

Round5 is based on the RLWR problem and hence both $\mathbf{u}$ and $\mathbf{v}$ are polynomials with degree $n-1$ and $\mu-1$ respectively with $n > \mu$[3]. Furthermore, $\mathbf{s}$ is a ternary polynomial with degree $n-1$ with coefficients in $\{-1, 0, 1\}$. The decryption procedure firstly computes the product $\mathbf{u} \times \mathbf{s}$ (Line 4 of Round5.Decrypt in Alg.3) in the polynomial ring defined by the modular polynomial $(x^{n+1} - 1)$. Further, the product is truncated to only the $\mu$ lower order coefficients using the Truncate function. The truncated result is subtracted from a rounded version of

[2] The description of the decryption procedure in Alg.3 ignores a lot of technical details for simplification. For complete details of the same, the reader is referred to [2]

[3] Since the encryption scheme of Round5 also follows the paradigm of the LPR encryption scheme, components $\mathbf{u}$ and $\mathbf{v}$ are directly taken from the description of the LPR encryption scheme in Alg.1

$\mathbf{v}$ (Line 5 in $\mathsf{Round5.Decrypt}$) which approximately yields the codeword in the underlying ring. Further, it is converted into the codeword $\mathbf{c}'' \in \mathbb{Z}_2^{\mu}$ using the $\mathsf{Round}_{p\to 2}$. This codeword is decoded to the final message $m''$ using the $\mathsf{Ecc\_Dec}$ decoding procedure of XEf.

**3.3.3 Attack Methodology:** We briefly state the summary of our attack methodology before diving into its technical details. Our attack recovers the secret polynomial one coefficient at a time. The main idea is to craft chosen ciphertexts for the decryption procedure of the KEM such that the validity of the resulting codewords solely depend on the targeted secret coefficient. We trigger the decapsulation operation with these chosen ciphertexts and observe the corresponding EM side-channel of the majority logic within the decoding procedure of XEf. Since the validity of the codeword can be detected from the EM side-channel as shown in Sec.3.2.2, this leads to a straightforward recovery of the targeted secret coefficient. This procedure can be repeated to recover all the coefficients of the secret key polynomial.

For our attack, we create chosen ciphertexts such that both $\mathbf{u}$ and $\mathbf{v}$ are polynomials with a single non-zero coefficient. For now, let us choose $\mathbf{u} = k_{\mathbf{u}} \times x^i$ and $\mathbf{v} = k_{\mathbf{v}} \times x^j$. Thus, the product $\mathbf{u} \times \mathbf{s}$ in the ring defined by the modular polynomial $\phi(x) = x^{n+1} - 1$ will be nothing but a cyclic rotation of $\mathbf{s}$ by $i$ coefficients (considering the degree of $\mathbf{s}$ is $n$ with $\mathbf{s}[n] = 0$) and scaled by the constant $k_{\mathbf{u}}$. Due to truncation (Line 4 in $\mathsf{Round5.Decrypt}$), only the lower order $\mu$ coefficients of the product are considered. Thus, the final result involves computations with $\mu$ coefficients of $\mathbf{s}$. If $i \leq 1$, the exposed secret coefficients are $\mathbf{s}[1-i], \ldots, \mathbf{s}[\mu - i]$ If $i \geq 2$, the exposed secret coefficients are the last $(i-1)$ coefficients (i.e) $(\mathbf{s}[n-i-1], \mathbf{s}[n-i-2], \ldots, \mathbf{s}[n-1], \mathbf{s}[n] = 0)$ and the first $(\mu - i + 1)$ coefficients (i.e) $(\mathbf{s}[0], \mathbf{s}[1], \ldots, \mathbf{s}[\mu - i])$. Since $\mathbf{s}[n] = 0$, effectively $\mu - 1$ coefficients of $\mathbf{s}$ are exposed when $i \geq 2$ is chosen.

The values of $k_{\mathbf{u}}$ and $k_{\mathbf{v}}$ are together chosen for our attack based on certain criteria, which are derived from the following observations. Since $\mathbf{v}[k] = 0 \,\forall\, k \neq j$ and $k \in \{0, \mu - 1\}$, only the bit $c''[j]$ depends on both $k_{\mathbf{v}}$ and $k_{\mathbf{u}}$, while all the other bits of $c''$ depend on $k_{\mathbf{u}}$, but not on $k_{\mathbf{v}}$. Thus, $k_{\mathbf{u}}$ should be chosen such that $c''[k] = 0 \,\forall\, k \neq j$ and $k \in \{0, \mu - 1\}$. The values for $k_{\mathbf{v}}$ are chosen for $\mathbf{v}[j]$ such that, given $(k_{\mathbf{u}}, k_{\mathbf{v}})$ the bit $\mathbf{c}''[j]$ solely depends on the value of the corresponding secret coefficient. If $i \leq 1$, the targeted secret coefficient is $\mathbf{s}[1 + j - i]$. But if $i \geq 2$, then for $(j < i - 1)$, the targeted secret coefficient is $\mathbf{s}[n - i - 1 + j]$, while for $j \geq i$, the targeted secret coefficient is $\mathbf{s}[j - i]$.

If $\mathbf{c}''[j] = 1$ and $\mathbf{c}''[i] = 0 \,\forall\, i \neq j$ and $i \in \{0, \mu - 1\}$, then $\mathbf{c}''$ $(\mathsf{HW}(\mathbf{c}'') = 1)$ becomes a faulty codeword while $\mathbf{c}'' = 0$ $(\mathsf{HW}(\mathbf{c}'') = 0)$ is a valid codeword. Thus, if a specific choice of $k_{\mathbf{u}}$ and $k_{\mathbf{v}}$ yields $\mathbf{c}''[j] = 1$ for a given secret coefficient, then it can be easily detected using the EM side-channel information as shown in Sec.3.2.2.

Please refer Tab.1 for the choices of $(k_{\mathbf{u}}, k_{\mathbf{v}})$ which we used for our attack on the R5ND_1KEM_5d variant of the Round5 CCA-secure KEM. Given $(k_{\mathbf{u}}, k_{\mathbf{v}})$, the validity of the codeword purely depends on the corresponding secret coefficient.

Since the validity of the codeword can be detected through the EM side-channel, the attacker can use the side-channel information along with information in Tab.1 to directly retrieve the corresponding secret coefficient.

The attacker can similarly change the position $j$ of the non-zero coefficient of $\mathbf{v}$ over positions 0 to $k-1$ ($k$ is the length of the message $m$ in $c$) and retrieve the $k$ corresponding coefficients of $\mathbf{s}$ ($\mathbf{s}[n-i-1], \mathbf{s}[n-i], \ldots, \mathbf{s}[n-1], \mathbf{s}[n] = 0, \mathbf{s}[0], \mathbf{s}[1], \ldots, \mathbf{s}[k-i-2]$). Though $\mu - 1$ coefficients of $\mathbf{s}$ are exposed, we only recover $k$ of them, since we only consider errors in the message portion ($m$) of the codeword for maximum side-channel distinguishability.

Further, the attacker can increment the position of the non-zero coefficient of $\mathbf{u}$ by $k$ (i.e) $k+i$ to attack the next set of $k$ coefficients of $\mathbf{s}$. Thus, the attacker can retrieve the whole secret in $\lceil \frac{n}{k} \rceil$ such iterations. Since the targeted decryption procedure is used in the CCA-secure PKE and KEM of Round5, our side-channel assisted chosen-ciphertext attack breaks both the CCA-secure PKE and CCA-secure KEM.

Table1: Values for $(\mathbf{u}[i], \mathbf{v}[j]) = (k_{\mathbf{u}}, k_{\mathbf{v}})$ chosen for our attack on the R5ND_1KEM_5d variant of Round5 CCA-secure KEM. **V** and **F** refer to valid ($\mathsf{HW}(c'') = 0$) and faulty codewords ($\mathsf{HW}(c'') = 1$) respectively.

| **Secret Coeff.** | $\mathsf{HW}(c'') = 0/\mathsf{HW}(c'') = 1$ | |
|---|---|---|
| | (21,3) | (12,1) |
| -1 | **F** | **F** |
| 0 | **F** | **V** |
| 1 | **V** | **V** |

### 3.4 Experimental Results

We implemented our attack on the optimized implementation of Round5, taken from the *pqm4* library [16], a post-quantum cryptographic suite for the ARM Cortex-M4 microcontrollers. In particular, we ported the R5ND_1KEM_5d variant of Round5 to the STM32F4DISCOVERY board (DUT) housing the STM32F407, ARM Cortex-M4 microcontroller. The attacker requires to trigger the decapsulation operation using two chosen ciphertexts to retrieve one coefficient of $\mathbf{s}$. For each ciphertext choice, we collect about 25 traces for repeated executions, thus totalling to about 50 executions for retrieval of every coefficient of the secret key $\mathbf{s}$. Thus, for full key recovery on the R5ND_1KEM_5d variant, the attacker requires to collect traces for about $50 \cdot 490$ executions. We are able to perform full key recovery with a 100% success rate and we are also able to replicate the attack multiple times over, thus demonstrating the importance of providing side-channel protection to implementation of ECCs.

## 4 Side-Channel Analysis of the FO transform

In section 3, we showed that the execution of XEf ECC's decoding procedure leaks side-channel information about the validity of the decrypted codeword. The vulnerability existed due to the differential behaviour of the ECC's decoding procedure based on the validity of the codeword, which was detectable through the power/EM side-channel.

We found that a very similar vulnerability also exists in the underlying FO transform which governs multiple CCA-secure lattice-based KEM schemes. In this section, we explain about the identified vulnerability and demonstrate a chosen ciphertext-attack against one such lattice-based KEM scheme, that uses the FO transformation for CCA security.

### 4.1 Side-Channel vulnerability in the FO transform

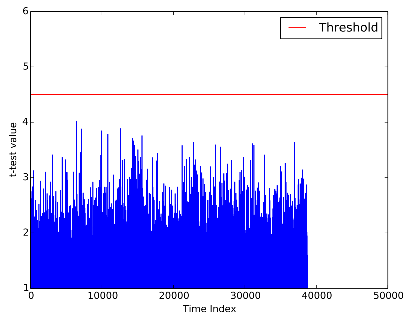
We refer to the KEM.Decaps procedure of Alg.2 for the generic framework of the decapsulation procedure underlying multiple lattice-based KEM schemes. It can be seen that operations from Line 2 in the KEM.Decaps procedure operate over the decrypted message $m'$. If any of these operations exhibit any differential behaviour based on the value of the decrypted message $m'$ that is observable through the power/EM side-channel, then we will be able to extend our attack to multiple lattice-based KEM schemes that utilize the FO-transformation for CCA conversion.

Similar to the identified vulnerability in the XEf ECC, we would like to distinguish between the two cases (1) $\mathsf{HW}(m') = 0$ and (2) $\mathsf{HW}(m') = 1$ with high probability through the power/EM side-channel. We target the hash operation $\mathcal{G}$ in particular (Line 2 of KEM.Decaps in Alg.2), which operates directly over the decrypted message $m'$, immediately after the PKE.Decrypt procedure. We attempt to exploit the diffusion property of hash functions which alters statistically half the number of bits of the output for a single bit change in the input. Thus, we observe side-channel traces corresponding to the end of the hash computation so that the altered bit is diffused enough to be easily distinguishable from the traces.

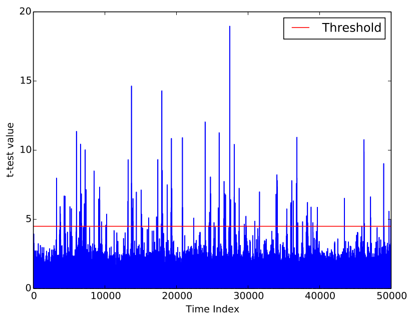
We utilize the same DUT and the experimental setup as described in Sec.3. Our main objective is to distinguish between the two cases $\mathsf{HW}(m') = 0$ and $\mathsf{HW}(m')$ $= 1$ with very high probability. We first collect two sets of 25 measurements for replicated runs corresponding to two different ciphertexts, but both of which yield the same decrypted message $m' = 0$. Let the two sets of measurements be denoted as $t_{v'}$ and $t_{v''}$. Refer to Fig.2(a) for the TVLA results between $t_{v'}$ and $t_{v''}$. We can see that there are no values greater than the threshold of 4.5, thus determining that both computations are the same with $m' = 0$.

We further collect a new set of 25 measurements for replicated experiments corresponding to $\mathsf{HW}(m') = 1$, which we denote as $t_f$. Refer Fig.2(b) for the TVLA results between $t_{v'}$ and $t_f$. We can clearly see many number of peaks (greater than the pass/fail threshold 4.5), which indicates that there is a considerable difference between the two sets of measurements at those peaks. Thus, an attacker can collect a set of reference traces corresponding to $m' = 0$, which we denote as $t_{ref}$. Further, the attacker can collect a set of traces which he wants to classify, which we denote as $t_{attack}$. He can then assign the class to which $t_{attack}$ belongs to based on the TVLA results between $t_{ref}$ and $t_{attack}$. Thus, we can see that the EM/power side-channel information from the hash function $\mathcal{G}$ utilized in the FO transformation clearly helps to distinguish between the values of the decrypted message output (i.e) $\mathsf{HW}(m') = 0$ and $\mathsf{HW}(m') = 1$. This observation thus violates a very fundamental theoretical assumption about CCA-secure schemes that the attacker sees nothing except the output of the decapsulation operation.

It is important to note that the above identified side-channel vulnerability is present in the FO transformation and thus applies to multiple lattice-based KEM schemes utilizing the same to achieve CCA security.



(a)



(b)

Figure2: $t$-test results for traces corresponding to $\mathcal{G}(m', pk)$ between (a) $\mathsf{HW}(m'_1) = 0$ and $\mathsf{HW}(m'_2) = 0$ (b) $\mathsf{HW}(m'_1) = 0$ and $\mathsf{HW}(m'_2) = 1$

## 4.2 Attack methodology

In this section, we will show that the identified side-channel vulnerability can be exploited to perform chosen-ciphertext attacks on multiple lattice-based KEM schemes based on the FO transformation, which do not utilize error correcting codes within decryption. Our adversary model is already described in Sec.3.3 and our attack methodology also follows very closely to the one used to demonstrate our attack over the Round5 KEM scheme.

As a proof of concept, we demonstrate our attack on module lattice-based KEM schemes which utilize the FO transformation to achieve security in the CCA model. We remark that the attack is generic and is applicable to other lattice-based schemes as well. This is because of the fact that a module lattice can be transformed into a standard or an ideal lattice.

**Algorithm 4:** Decryption operation in module lattice-based KEM

```
1 Procedure KEM.Decrypt$(ct, sk)$
2     $\mathbf{u}, \mathbf{v} = \mathsf{DecodeCT}(ct)$
3     $\mathbf{s} = \mathsf{DecodeSK}(sk)$
4     $\mathbf{w} = \mathbf{u} \times \mathbf{s}$
5     $\mathbf{m}' = \mathsf{Compress}(\mathbf{v} - \mathbf{w}, 1)$
```

The high-level description of the decryption procedure in Alg. 4 closely resembles the decryption procedure of Round5. The vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{s}$ are in $R_q^k$ consisting of $k$ polynomials each of degree $n-1$ in the polynomial ring $R_q$. We will demonstrate our attack to retrieve the $i^{th}$ polynomial element $\mathbf{s}_i$ of the secret vector $\mathbf{s}$. The same steps can be repeated to retrieve the other polynomials of $\mathbf{s}$, one at a time.

Let the $i^{th}$ polynomials of $\mathbf{u}$ and $\mathbf{v}$ be denoted as $\mathbf{u}_i$ and $\mathbf{v}_i$ respectively. To retrieve the $j^{th}$ coefficient of $\mathbf{s}_i$, we set $\mathbf{u}_i[0] = k_{\mathbf{u}}$ and $\mathbf{v}_i[j] = k_{\mathbf{v}}$ respectively, while all the other coefficients of $\mathbf{u}_i$ and $\mathbf{v}_i$ are set to zero. Furthermore, all the other polynomials in the $\mathbf{u}$ and $\mathbf{v}$ vectors are set to zero. The values for $k_{\mathbf{u}}$ and $k_{\mathbf{v}}$ are chosen together such that only the $j^{th}$ bit of $m'$ is determined by the corresponding secret coefficient $\mathbf{s}_i[j]$ and all other bits of $m'$ become 0.

The attacker collects a set of EM measurements $t_{attack}$ of the targeted hash function $\mathcal{G}$ for the different crafted ciphertexts. The attacker already has a set of reference traces $(t_{ref})$ corresponding to $\mathsf{HW}(m') = 0$. He simply computes the TVLA between $t_{attack}$ and $t_{ref}$ to determine the class of $t_{attack}$ based on the results of the $t$-test. He can then correlate the results of the side-channel analysis to retrieve the targeted secret coefficient $\mathbf{s}_i[j]$. The position $j$ of the non-zero coefficient of $\mathbf{v}_i$ can be varied to similarly retrieve all the coefficients of $\mathbf{s}_i$. Furthermore, the same attack can be repeated over the other polynomials of $\mathbf{u}$ and $\mathbf{v}$ to retrieve the entire secret module $\mathbf{s}$.

Table2: Values for $(\mathbf{u}_i[0], \mathbf{v}_i[j]) = (k_{\mathbf{u}}, k_{\mathbf{v}})$ chosen for our attack on the KYBER512 variant of the CCA-secure Kyber KEM scheme. While **V** refers to the case of $\mathsf{HW}(m') = 0$, **F** refers to the case of $\mathsf{HW}(m') = 1$.

| $\mathbf{s_i[j]}$ | $\mathsf{HW}(m') = 0/\mathsf{HW}(m') = 1$ | | | | |
|---|---|---|---|---|---|
| | $(210, 209)$ | $(210, 2705)$ | $(101, 644)$ | $(100, 2626)$ | $(415, 1041)$ |
| -2 | **F** | **V** | **F** | **V** | **F** |
| -1 | **V** | **V** | **F** | **V** | **F** |
| 0 | **V** | **V** | **V** | **V** | **F** |
| 1 | **V** | **V** | **V** | **F** | **V** |
| 2 | **V** | **F** | **V** | **F** | **V** |

### 4.3 Experimental Results

As an example, in Table 2 we list the values of $(k_{\mathbf{u}}, k_{\mathbf{v}})$ for the KYBER512 variant of the CCA-secure Kyber KEM scheme. Similar values can be computed for other lattice-based KEM schemes to perform our attack.

To evaluate our attack, we used the optimized implementation of KYBER512 available in the *pqm4* library [16]. The implementation was run on the ARM Cortex-R4 based STM32M407 microcontroller hosted on the STM32MF4DISCOVERY board. Recovery of one coefficient of $\mathbf{s}$ requires the attacker to trigger the decryption oracle with five choices of ciphertexts, as shown in Tab.2. For each choice, we collect about 25 traces corresponding to repeated executions, thus amounting to 125 executions for recovery of every coefficient. We were able to retrieve the secret coefficients with a 100% success rate and were able to successfully repeat the attack multiple times. Since a secret vector in KYBER512 consists of two polynomials of degree 255, the attacker performs total $2 \cdot 125 \cdot 256$ queries to recovery the entire secret for KYBER512. Similarly for LightSaber (which has similar security as KYBER512), the number of traces will be $2 \cdot 175 \cdot 256$ as each secret coefficient can have 7 possible values in the range -3 to 3. Similar numbers can be computed for other lattice-based schemes such as NewHope, LAC, Frodo, Round5 etc. The number of traces will change linearly with the width of the secret distribution, module dimension and polynomial degree. Naturally, schemes with binary secrets will be the easiest targets.

## 5 Countermeasures

In this section, we briefly talk about the countermeasures separately for both our attacks. Firstly, our attack targeting the constant-time error correcting codes can be protected through incorporation of side-channel protection measures for the implementation of the error correcting code procedures. Additionally, given the earlier attack of D'Anvers *et al.* [9] targeting the same error correcting procedures, it is very important to start considering side-channel protected implementation of error correcting codes.

Secondly, our attack exploiting the vulnerability of the FO transformation requires a more generic countermeasure, provided the generic nature of our attack. Masking the decryption/decapsulation operation can definitely protect against our attack exploiting vulnerabilities in the FO transform. There are several works on masking lattice-based primitives [28, 29] and Oder *et al.* [22] describe masking techniques for protecting the FO transformation steps in the CCA setting. Of course, masking countermeasures will slowdown performances significantly. Hence, efficient masking strategies for CCA-secure LWE/LWR based schemes is an interesting research problem that warrants immediate attention by the cryptographic research community.

Masking might also help protect the schemes using error correcting codes. However, efficient masking schemes for the error correcting codes used in lattice-based cryptography are yet to be invented. It will be an interesting but a challenging research topic to design masking schemes for complicated strong error correcting codes such as BCH.

## 6 Conclusion

In this work, we demonstrated side-channel assisted chosen-ciphertext attacks against IND-CCA secure lattice-based PKE/KEM schemes. We identified vulnerabilities within the constant-time error correcting codes used in the Round5 KEM scheme which help the attacker distinguish between faulty and valid codewords through the EM side-channel information. We showed how the vulnerabilites can be exploited to perform chosen-ciphertext attacks against the CCA-secure Round5 algorithm. Our attack thus shows that the application of error correcting codes in lattice-based cryptography weakens the side-channel security of a lattice-based PKE/KEM scheme.

We also identified a generic vulnerability in the CCA transformation steps used in multiple lattice-based PKE/KEM schemes. We showed that this vulnerability can be exploited in a similar manner and we demonstrated practical chosen-ciphertext attacks on CCA-secure lattice-based PKE/KEMs. We performed practical validation of our attacks through EM measurements over the CCA transform steps from the open source *pqm4* library, a benchmarking framework for PQC schemes on the ARM Cortex-M4 microcontroller.

The attacks proposed in this paper show that CCA-secure schemes need to be protected against side-channel attacks thus reaffirming the need for appropriate countermeasure such as masking.

## References

1. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092 (2015), `https://eprint.iacr.org/2015/1092`
2. Baan, H., Bhattacharya, S., Fluhrer, S., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Zhang, Z.: Round5: Compact and fast post-quantum public-key encryption. Cryptology ePrint Archive, Report 2019/090 (2019), `https://eprint.iacr.org/2019/090`
3. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 719–737. Springer (2012)
4. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of newhope. In: Cryptographers' Track at the RSA Conference. pp. 272–292. Springer (2019)
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. Cryptology ePrint Archive, Report 2017/634 (2017), `https://eprint.iacr.org/2017/634`
6. Botros, L., Kannwischer, M.J., Schwabe, P.: Memory-efficient high-speed implementation of kyber on cortex-m4. In: Progress in Cryptology - AFRICACRYPT

2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings. pp. 209–228 (2019), https://doi.org/10.1007/978-3-030-23696-0_11

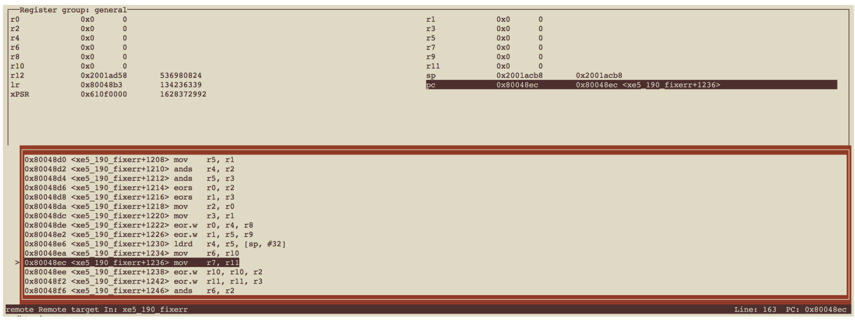
7. Bruinderink, L.G., Pessl, P.: Differential fault attacks on deterministic lattice signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems 2018(3) (2018), https://eprint.iacr.org/2018/355.pdf
8. Cachin, C.: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings, vol. 3027. Springer Science & Business Media (2004)
9. D'Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum secure schemes. IACR Cryptology ePrint Archive 2019, 292 (2019)
10. D'Anvers, J.P., Vercauteren, F., Verbauwhede, I.: On the impact of decryption failures on the security of LWE/LWR based schemes. IACR Cryptology ePrint Archive 2018, 1089 (2018)
11. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: International Conference on Cryptology in Africa. pp. 282–305. Springer (2018)
12. Fluhrer, S.R.: Cryptanalysis of ring-LWE based key exchange with key share reuse. IACR Cryptology ePrint Archive 2016, 85 (2016)
13. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999)
14. Gilbert Goodwill, B.J., Jaffe, J., Rohatgi, P., et al.: A testing methodology for side-channel resistance validation. In: NIST non-invasive attack testing workshop. vol. 7, pp. 115–136 (2011)
15. Huffman, W.C., Pless, V.: Fundamentals of error-correcting codes. Cambridge university press (2010)
16. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4
17. Karmakar, A., Mera, J.M.B., Roy, S.S., Verbauwhede, I.: Saber on ARM CCA-secure module lattice-based key encapsulation on ARM. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(3), 243–266 (2018), https://doi.org/10.13154/tches.v2018.i3.243-266
18. Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B., Wang, K., Liu, Z., Yang, H.: LAC: Practical ring-LWE based public-key encryption with byte-level modulus. IACR Cryptology ePrint Archive 2018, 1009 (2018)
19. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–23. Springer (2010)
20. NIST: Post quantum cryptography - round 2 submissions. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions/ (2019)
21. Oder, T., Pöppelmann, T., Güneysu, T.: Beyond ECDSA and RSA: Lattice-based digital signatures on constrained devices. In: Proceedings of the 51st Annual Design Automation Conference. pp. 1–6. ACM (2014)
22. Oder, T., Schneider, T., Pöppelmann, T., Güneysu, T.: Practical CCA2-secure and masked ring-LWE implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems 2018(1), 142–174 (2018)
23. Pöppelmann, T., Oder, T., Güneysu, T.: Speed records for ideal lattice-based cryptography on avr. IACR Cryptology ePrint Archive 2015, 382 (2015)

24. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 513–533. Springer International Publishing, Cham (2017)
25. Rath, D.: Openocd-open on-chip debugger (2005)
26. Ravi, P., Jungk, B., Jap, D., Najm, Z., Bhasin, S.: Feature selection methods for non-profiled side-channel attacks on ecc. In: 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP). pp. 1–5. IEEE (2018)
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34 (2009)
28. Reparaz, O., de Clercq, R., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Additively homomorphic ring-LWE masking. In: Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. pp. 233–244 (2016), `https://doi.org/10.1007/978-3-319-29360-8_15`
29. Reparaz, O., Roy, S.S., Vercauteren, F., Verbauwhede, I.: A masked ring-LWE implementation. In: Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. pp. 683–702 (2015), `https://doi.org/10.1007/978-3-662-48324-4_34`
30. Szepieniec, A.: Ramstake. Tech. rep., Technical report, National Institute of Standards and Technology (2017)
31. Walters, M., Roy, S.S.: Constant-time BCH error-correcting code. IACR Cryptology ePrint Archive 2019, 155 (2019)

## A Visualization of Activity in Registers

Please refer Fig.3 for the screenshots from the Openocd- Open On-Chip debugger tool [25] showing values of the internal general purpose registers during operation of the targeted majority logic within the decoding procedure of XEf error correcting code of the Round5 PKE scheme. It is important to note that the values of the registers for the different cases are captured at the same point of time in the program (Refer to the program counter in the register set (highlighted in brown) in each figure).

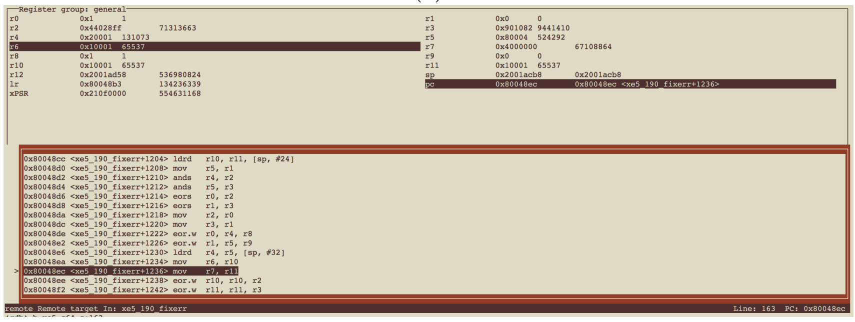
While Fig.3(a) corresponds to the decoding of a valid codeword (i.e) $c = 0$, Fig.3(b) corresponds to the decoding of a faulty codeword, with the single erronous bit is present in the message portion $m$ of the codeword $c$ (i.e) $c \neq 0$ with $HW(m) = 1$ and $HW(r) = 0$. Finally, Fig.3(c) also corresponds to the decoding of a faulty codeword, but with erronous bit present in the register portion $r$ of the codeword $c$ (i.e) $c \neq 0$ with $HW(m) = 0$ and $HW(r) = 1$. We can see that all the operating registers are zeros in case of the valid codeword (a), but is not the case with the faulty codewords ((b) and (c)). Moreover, on comparing Fig.3(b) and Fig.3(c), we can also see that there are more number of non-zero registers when the message $m$ is faulty, as opposed to when the register $r$ is faulty. Thus, there is more observable activity in the side-channel traces when the message is faulty, compared to the registers. Thus, we only consider the former case for our attack. Since there is a significant difference between the data being operated on, based on the validity of the codeword, we are able to easily distinguish the same using the power/EM side-channel information.



```
┌Register group: general──────────────────────────────────────────────────────────────────────┐
r0             0x0      0                                   r1             0x0      0
r2             0x0      0                                   r3             0x0      0
r4             0x0      0                                   r5             0x0      0
r6             0x0      0                                   r7             0x0      0
r8             0x0      0                                   r9             0x0      0
r10            0x0      0                                   r11            0x0      0
r12            0x2001ad58       536980824                   sp             0x2001acb8       0x2001acb8
lr             0x80048b3        134236339                   pc             0x80048ec        0x80048ec <xe5_190_fixerr+1236>
xPSR           0x610f0000       1628372992

   0x80048d0 <xe5_190_fixerr+1208> mov    r5, r1
   0x80048d2 <xe5_190_fixerr+1210> ands   r4, r2
   0x80048d4 <xe5_190_fixerr+1212> ands   r5, r3
   0x80048d6 <xe5_190_fixerr+1214> eors   r0, r2
   0x80048d8 <xe5_190_fixerr+1216> eors   r1, r3
   0x80048da <xe5_190_fixerr+1218> mov    r2, r0
   0x80048dc <xe5_190_fixerr+1220> mov    r3, r1
   0x80048de <xe5_190_fixerr+1222> eor.w  r0, r4, r8
   0x80048e2 <xe5_190_fixerr+1226> eor.w  r1, r5, r9
   0x80048e6 <xe5_190_fixerr+1230> ldrd   r4, r5, [sp, #32]
   0x80048ea <xe5_190_fixerr+1234> mov    r6, r10
  >0x80048ec <xe5_190_fixerr+1236> mov    r7, r11
   0x80048ee <xe5_190_fixerr+1238> eor.w  r10, r10, r2
   0x80048f2 <xe5_190_fixerr+1242> eor.w  r11, r11, r3
   0x80048f6 <xe5_190_fixerr+1246> ands   r6, r2
remote Remote target In: xe5_190_fixerr                                               Line: 163  PC: 0x80048ec
```

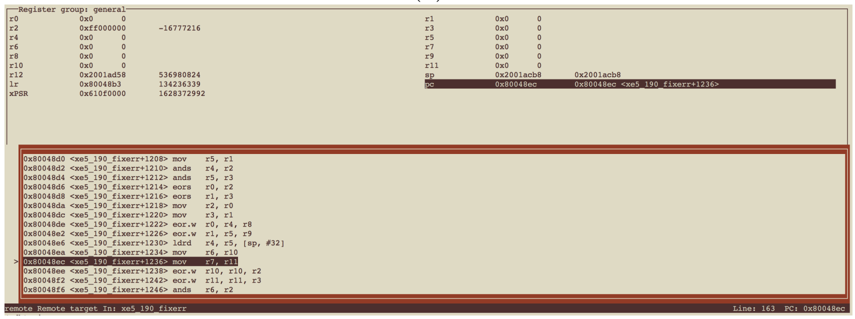
(a)



```
┌Register group: general──────────────────────────────────────────────────────────────────────┐
r0             0x1      1                                   r1             0x0      0
r2             0x44028ff        71313663                    r3             0x901082 9441410
r4             0x20001  131073                              r5             0x80004  524292
r6             0x10001  65537                               r7             0x4000000        67108864
r8             0x1      1                                   r9             0x0      0
r10            0x10001  65537                               r11            0x10001  65537
r12            0x2001ad58       536980824                   sp             0x2001acb8       0x2001acb8
lr             0x80048b3        134236339                   pc             0x80048ec        0x80048ec <xe5_190_fixerr+1236>
xPSR           0x210f0000       554631168

   0x80048cc <xe5_190_fixerr+1204> ldrd   r10, r11, [sp, #24]
   0x80048d0 <xe5_190_fixerr+1208> mov    r5, r1
   0x80048d2 <xe5_190_fixerr+1210> ands   r4, r2
   0x80048d4 <xe5_190_fixerr+1212> ands   r5, r3
   0x80048d6 <xe5_190_fixerr+1214> eors   r0, r2
   0x80048d8 <xe5_190_fixerr+1216> eors   r1, r3
   0x80048da <xe5_190_fixerr+1218> mov    r2, r0
   0x80048dc <xe5_190_fixerr+1220> mov    r3, r1
   0x80048de <xe5_190_fixerr+1222> eor.w  r0, r4, r8
   0x80048e2 <xe5_190_fixerr+1226> eor.w  r1, r5, r9
   0x80048e6 <xe5_190_fixerr+1230> ldrd   r4, r5, [sp, #32]
   0x80048ea <xe5_190_fixerr+1234> mov    r6, r10
  >0x80048ec <xe5_190_fixerr+1236> mov    r7, r11
   0x80048ee <xe5_190_fixerr+1238> eor.w  r10, r10, r2
   0x80048f2 <xe5_190_fixerr+1242> eor.w  r11, r11, r3
remote Remote target In: xe5_190_fixerr                                               Line: 163  PC: 0x80048ec
```

(b)



```
┌Register group: general──────────────────────────────────────────────────────────────────────┐
r0             0x0      0                                   r1             0x0      0
r2             0xff000000       -16777216                   r3             0x0      0
r4             0x0      0                                   r5             0x0      0
r6             0x0      0                                   r7             0x0      0
r8             0x0      0                                   r9             0x0      0
r10            0x0      0                                   r11            0x0      0
r12            0x2001ad58       536980824                   sp             0x2001acb8       0x2001acb8
lr             0x80048b3        134236339                   pc             0x80048ec        0x80048ec <xe5_190_fixerr+1236>
xPSR           0x610f0000       1628372992

   0x80048d0 <xe5_190_fixerr+1208> mov    r5, r1
   0x80048d2 <xe5_190_fixerr+1210> ands   r4, r2
   0x80048d4 <xe5_190_fixerr+1212> ands   r5, r3
   0x80048d6 <xe5_190_fixerr+1214> eors   r0, r2
   0x80048d8 <xe5_190_fixerr+1216> eors   r1, r3
   0x80048da <xe5_190_fixerr+1218> mov    r2, r0
   0x80048dc <xe5_190_fixerr+1220> mov    r3, r1
   0x80048de <xe5_190_fixerr+1222> eor.w  r0, r4, r8
   0x80048e2 <xe5_190_fixerr+1226> eor.w  r1, r5, r9
   0x80048e6 <xe5_190_fixerr+1230> ldrd   r4, r5, [sp, #32]
   0x80048ea <xe5_190_fixerr+1234> mov    r6, r10
  >0x80048ec <xe5_190_fixerr+1236> mov    r7, r11
   0x80048ee <xe5_190_fixerr+1238> eor.w  r10, r10, r2
   0x80048f2 <xe5_190_fixerr+1242> eor.w  r11, r11, r3
   0x80048f6 <xe5_190_fixerr+1246> ands   r6, r2
remote Remote target In: xe5_190_fixerr                                               Line: 163  PC: 0x80048ec
```

(c)

Figure3: Visualization of internal registers during the targeted majority logic (a) Decoding of valid codeword $c = 0$ (b) Decoding of faulty codeword $c \neq 0$ with $HW(m) = 1$ and $HW(r) = 0$ (c) Decoding of faulty codeword $c \neq 0$ with $HW(m) = 0$ and $HW(r) = 1$