On Exploiting Message Leakage in (few) NIST PQC Candidates for Practical Message Recovery Attacks

Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, Anupam Chattopadhyay

***Abstract*—In this work, we propose generic and practical side-channel attacks for message recovery in post-quantum lattice-based public key encryption (PKE) and key encapsulation mechanisms (KEM). The targeted schemes are based on the well known Learning With Errors (LWE) and Learning With Rounding (LWR) problem and include three finalists and six semi-finalist candidates of the ongoing NIST's standardization process for post-quantum cryptography. Notably, we propose to exploit inherent *ciphertext malleability* properties of LWE/LWR-based PKEs as a powerful tool for side-channel assisted message recovery attacks. The use of ciphertext malleability widens the scope of previous attacks with the ability to target multiple operations for message recovery. Moreover, our attacks are adaptable to different implementation variants and are also applicable to implementations protected with concrete *shuffling* and *masking* side-channel countermeasures. Our work mainly highlights the presence of inherent algorithmic properties in LWE/LWR-based schemes that can aid side-channel attacks for message recovery, thereby stressing on the need for strong side-channel countermeasures against message recovery for LWE/LWR-based schemes.**

I. Introduction

The impending threat of large scale quantum computers to conventional RSA and ECC-based public-key cryptographic schemes has prompted significant interest in the cryptographic community towards developing alternate public-key cryptographic schemes which are resistant to attacks from quantum computers, better known as *Post-Quantum Cryptography* (PQC) [1]. This, NIST in 2017 initiated a global standardization process for PQC-based Public Key Encryption (PKE), Key Establishment Mechanisms (KEM) and Digital Signature (DS) schemes. This process is currently in its third and final round with seven main candidates and eight alternate candidates [2]. NIST and the PQC research community anticipate that a subset of PQC candidates will be standardized around 2024 and soon reach wide scale adoption.

While the initial rounds considered theoretical post-quantum security and implementation performance on hardware and software platforms as key selection criteria, NIST has made it clear that resistance against side-channel and fault attacks will also be considered as a key criteria for standardization. NIST states that it "*encourages additional research regarding side-channel analysis*" of the finalist candidates and that it "*hopes to collect more information about the costs of implementing these algorithms in a way that provides resistance to such attacks*" [2]. This is very relevant for adoption of PQC in embedded devices, which are typically deployed in scenarios where an attacker has direct physical access to the device.

In this respect, we focus on Side-channel Analysis (SCA) of lattice-based PKE/KEMs built upon hardness of the well known *Learning With Errors* (LWE) and *Learning With Rounding* (LWR) problem. They form the majority in the standardization process with *six* (6) out of *seventeen* (17) candidates in the *semi-final* round and *three* (3) out of the *nine* (9) candidates in the *final* round. Several works have demonstrated the efficacy of implementing LWE/LWR-based schemes on constrained embedded devices such as 8-bit/32-bit microcontrollers and FPGAs [3] and thus SCA of their embedded implementations gained considerable traction with several works on practical attacks as well as protected implementations [4], [5], [6].

While a majority of side-channel attacks have focussed on recovery of the long term secret key used for decryption, much lesser attention has been devoted to side-channel security of the secret message, whose knowledge leads to recovery of the session key. The known message recovery attacks [7], [8], [9] on LWE/LWR-based schemes exploit preventable implementation level vulnerabilities that leak side-channel information about the message. These attacks are very specific to the target implementation and thus can be easily thwarted using appropriate implementation level changes. We however observe that LWE/LWR-based schemes inherently involve unique bitwise manipulation of the message which could pave way for a wider class of side-channel vulnerabilities leading to message recovery.

In this paper, we propose generic message recovery attacks for LWE/LWR-based PKE/KEMs which exploit inherent algorithmic properties to assist side-channel based message recovery. The generic nature of our message recovery attacks can be attributed to the utilization of *ciphertext malleability* properties of LWE/LWR-based schemes as an effective tool in a side-channel setting. Our message recovery attacks operate in two phases i.e. pre-processing and exploitation phase and the overall attack flow is illustrated in Fig. 1.

P. Ravi and A. Chattopadhyay are with Temasek Laboratories and School of Computer Science and EngineeringNanyang Technological University, Singapore (e-mail: prasanna.ravi@ntu.edu.sg; anupam@ntu.edu.sg).

S. Bhasin is with Temasek Laboratories, Nanyang Technological University, Singapore (e-mail: sbhasin@ntu.edu.sg).

S.S. Roy is with Graz University of Technology, Graz, Austria (e-mail: sujoy.sinharoy@iaik.tugraz.at).

Manuscript received May ??, ????; revised May ??, 2021.

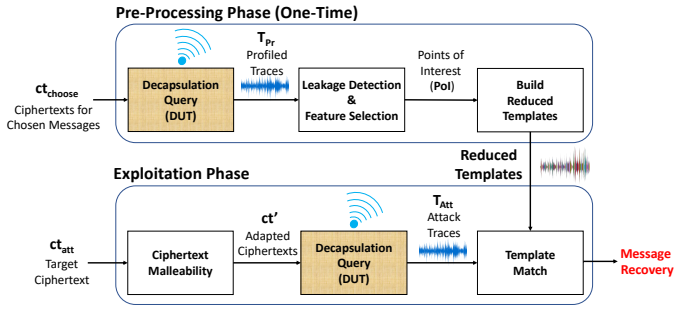


Figure 1: Attack flow for our message recovery attacks targeting the IND-CCA secure decapsulation procedure

Contributions:

1) (a) We demonstrate a novel message recovery attack targeting the *message decoding* operation within the decryption procedure of LWE/LWR-based PKE/KEMs, exploiting *incremental storage* of the decrypted message in memory (**Sec. IV**).
   (b) We perform experimental validation of our message recovery attack using the Electromagnetic Emanation (EM) side-channel from optimized implementations of PQC schemes from the *pqm4* public library [10], a testing and benchmarking framework for PQC schemes on the ARM Cortex-M4 microcontroller (**Sec. V**).
2) (a) To the best of our knowledge, our work demonstrates the first utilization of the inherent *ciphertext malleability* property of LWE/LWR-based schemes as a tool for side-channel analysis (**Sec. VI**).
   (b) We exploit ciphertext malleability to propose generic message recovery attacks targeting storage of the decrypted message in memory, irrespective of the bit width as compared to only bit-wise in our original attack (**Sec. VII**).
   (c) Our attacks are applicable to *six* LWE/LWR-based PKE/KEMs from the NIST standardization process - Kyber, Saber (main finalists), Frodo (alternative finalist) and semi-finalist candidates such as NewHope, Round5 and LAC.
3) We also exploit the *ciphertext malleability* property to break well known side-channel countermeasures such as (**Sec. VIII**):
   (a) Shuffling countermeasure for the message encoding operation proposed by Amiet *et al.* [8] in PQCrypto'20.
   (b) Adaptation of the PQCrypto'20 countermeasure to the *message decoding* operation.
   (c) Masking countermeasures such as that of Oder *et al.* [4] of CHES'18.
4) As a secondary contribution, we also propose improvements to the chosen ciphertext based key recovery attacks proposed by Xu et al. [10] which rely upon complete message recovery. While the original attack proposed on Kyber512 required 8 decrypted messages for full key recovery, we improve the requirement to 6 queries and also propose non-trivial extensions of the same attack to LWE/LWR-based schemes such as NewHope (Supplementary material, Sec. V).

Our work mainly highlights the presence of inherent algorithmic properties in LWE/LWR-based schemes that can aid side-channel attacks for message recovery, which can also lead to easy compromise of the long-term secret key. It therefore reiterates the need for strong and concrete evaluation of side-channel countermeasures against message recovery in LWE/LWR-based schemes.

II. Lattice Preliminaries

A. Notation

We denote the ring of integers modulo a prime q as $\mathbb{Z}_q$. We denote the the space of all byte arrays of length n bytes as $\mathcal{B}^n$. The i^{th} byte of $\mathsf{m} \in \mathcal{B}^n$ is denoted as $\mathsf{m}[i]$, while the j^{th} bit of $\mathbf{m}$ is denoted as $\mathbf{m}_j$ and the k^{th} bit of $\mathsf{m}[i]$ as $\mathsf{m}[i]_k$. The polynomial ring $\mathbb{Z}_q(x)/\phi(x)$ is denoted as R_q where $\phi(x)$ is its reduction polynomial. Polynomials in R_q are shown in bold lower case letters and the i^{th} coefficient of $\mathbf{a} \in R_q$ is referred to as $\mathbf{a}[i]$. Matrices/vectors in $\mathbb{Z}_q^{k\times l}$ are shown in bold upper case letters. Multiplication of two polynomials $\mathbf{a}$ and $\mathbf{b}$ in R_q is denoted as $\mathbf{c} = \mathbf{a} \times \mathbf{b}$. An element $\mathbf{x} \in R_q$ sampled from the distribution χ with standard deviation σ is denoted as $\mathbf{x} \leftarrow \chi_\sigma(R_q)$. While $\mathcal{U}$ refers to a uniform distribution, $\mathcal{D}$ refers to a Gaussian distribution.

B. Learning With Errors/Rounding Problem (LWE/LWR)

The security of several lattice-based PKE/KEMs is governed by the well-known Learning With Errors (LWE) problem [11]. A standard LWE instance is denoted as a tuple $(\mathbf{A}, \mathbf{T}) \in (\mathbb{Z}_q^{k\times\ell} \times \mathbb{Z}_q^{k\times n})$ where $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{k\times\ell})$ is a public constant and $\mathbf{T} = \mathbf{A}\times\mathbf{S}+\mathbf{E}$ where $\mathbf{S} \in \mathcal{D}_\sigma(\mathbb{Z}_q^{\ell\times n})$ is the secret and $\mathbf{E} \in \mathcal{D}_\sigma(\mathbb{Z}_q^{k\times n})$ is the error. Learning With Rounding (LWR) is a slight variant of the LWE problem where the error component $\mathbf{E}$ is implicitly generated by rounding the elements of the product $(\mathbf{A} \times \mathbf{S})$ to a lower modulus p [12].

Frodo (NIST finalist candidate) is based on the standard LWE problem while most other schemes are based on structured variants of the LWE/LWR problem known as the Ring-LWE/Ring-LWR (RLWE/RLWR) [13] and Module-LWE/Module-LWR (MLWE/MLWR) problem [14]. These variants involve computation over polynomials in polynomial rings such as $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ or $R_q = \mathbb{Z}_q[x]/(x^n - x - 1)$. Schemes such as NewHope, LAC and Round5 are based on the RLWE/RLWR problem which involve computation over polynomials in R_q, while schemes such as Kyber and Saber are based on the MLWE/MLWR problem which involve computation over small matrices and vectors of polynomials in polynomial rings R_q^k for $k > 1$ referred to as *modules*. Concrete details of the aforementioned schemes can be found in their respective specification documents available in [15].

C. A Generic Framework for LWE/LWR based PKE/KEMs

Most LWE/LWR-based PKE/KEMs are built upon a generalized paradigm for public key encryption schemes proposed by Lyubashevskey, Peikert and Regev [13] in 2010,

Algorithm 1: LPR Encryption Scheme [13]

Procedure PKE.KeyGen()
  $\mathbf{a} \in R_q$
  $\mathbf{s}, \mathbf{e} \leftarrow \chi_\sigma(R_q) \in R_q$
  $\mathbf{t} = \mathbf{a} \times \mathbf{s} + \mathbf{e} \in R_q$
  return $\mathsf{pk} = (\mathbf{a}, \mathbf{t}), \mathsf{sk} = (\mathbf{s})$

Procedure PKE.Encrypt(pk, $\mathsf{m} \in \mathcal{B}^{32}, \mathsf{r} \in \mathcal{B}^{32}$)
  $\mathbf{s}', \mathbf{e}', \mathbf{e}'' \leftarrow \chi_\sigma(R_q)$
  $\mathbf{u} = \mathbf{a} \times \mathbf{s}' + \mathbf{e}'$
  $\mathbf{v}' = \mathbf{t} \times \mathbf{s}' + \mathbf{e}''$
  $\mathbf{x} = \mathsf{Encode}(\mathsf{m})$
  $\mathbf{v} = \mathbf{v}' + \mathbf{x}$
  return $\mathsf{ct} = (\mathbf{u}, \mathbf{v})$

Procedure PKE.Decrypt(ct, sk)
  $\mathbf{x}' = (\mathbf{v} - \mathbf{u} \times \mathbf{s}) \in R_q$
  $\mathsf{m}' = \mathsf{Decode}(\mathbf{x}')$
  return m'

now well-known as the *LPR Encryption scheme*. We provide a high level description of the LPR encryption scheme based on the RLWE problem in Alg. 1, while the same can be adapted to both the standard and module variants of the LWE/LWR problem. We define the procedure Encode which encodes a byte array in $\mathcal{B}^n$ into a corresponding polynomial in the ring R_q and the corresponding inverse procedure Decode which maps a polynomial in R_q to a byte array in $\mathcal{B}^n$.

1) Security in the Chosen-Ciphertext Model: The LPR PKE is provably secure in the *Indistinguishability under Adaptive Chosen-Plaintext Attack* (IND-CPA) security model. However, an adversary with access to the decrypted message for chosen ciphertexts can recover the long term secret key. Thus, most LWE/LWR-based schemes employ the Fujisaki-Okamoto (FO) transform [16] to achieve security in the *Indistinguishability under Adaptive Chosen-Ciphertext Attack* (IND-CCA) model. The FO transform forms a wrapper around the encryption and decryption procedures using several instantiations of one-way hash functions resulting in IND-CCA secure encapsulation (KEM.Encaps) and decapsulation (KEM.Decaps) procedures respectively, as shown in Alg.2. The FO transform performs a re-encryption of the decrypted message (line 4 in KEM.Decaps). The resulting ciphertext $\mathbf{ct'}$ is then compared with with the received ciphertext $\mathbf{ct}$ (line 5). For an invalid ciphertext, this comparison step will always fail and thus an adversary does not get any information about the decrypted message for maliciously chosen ciphertexts, thereby defeating chosen ciphertext attacks.

D. Tools for Feature Selection in Side-Channel Analysis

1) Test Vector Leakage Assessment (TVLA) [17]: TVLA is a popular conformance-based evaluation methodology widely used by both academia and the industry to perform side-channel evaluation of cryptographic implementations. It involves computation of the well known univariate Welch's t-test over two sets of side-channel measurements to identify differentiating features in them. The TVLA formulation over two sets of measurements $\mathcal{T}_r$ and $\mathcal{T}_f$ is

Algorithm 2: IND-CCA secure LWE/LWR-based KEM

Procedure KEM.Encaps(pk)
  $\rho \leftarrow \mathcal{U}(\mathcal{B}^{32})$
  $\mathsf{m} = \mathcal{H}(\rho)$
  $r = \mathcal{G}(m, \mathsf{pk})$
  $\mathsf{ct} = \mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{m}, r)$
  $K = \mathcal{H}(r, \mathsf{ct})$
  return (ct, K)

Procedure KEM.Decaps(sk, pk, ct)
  $\mathsf{m}' = \mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$
  $r' = \mathsf{PRF}(\mathsf{m}', \mathsf{pk})$
  $\mathsf{ct}' = \mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{m}', r')$
  if $\mathsf{ct}' = \mathsf{ct}$ **then**
    return $K = \mathsf{KDF}(r' \| \mathsf{ct}')$
  end
  else
    return $K = \mathsf{KDF}(z \| \mathsf{ct}')$ // $z \in \mathcal{B}^{32}$ is a random secret
  end

given by:

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

where μ_r, σ_r and m_r (resp. μ_f, σ_f and m_f) are univariate mean, standard deviation and cardinality of the trace set $\mathcal{T}_r$ (resp. $\mathcal{T}_f$). The null hypothesis (two means are equal) is rejected with a confidence of 99.9999% when the absolute value of the t-test score is greater than 4.5 [17]. A rejected hypothesis implies that the two sets are noticeably different and hence could leak some side-channel information.

2) Normalized Inter-Class Variance (NICV) [18]: While TVLA can be used to differentiate between two classes, NICV is a more generic metric which can simultaneously differentiate between two or more classes. Let the variable X have n possible classes and let $\mathsf{C}(X)$ denote the class of a given value of X. If the observed leakage of X is denoted as $\mathcal{T}$, then NICV can be calculated as follows:

$$\mathsf{NICV} = \frac{\sigma^2(\mu(\mathcal{T}|\mathsf{C}(X))}{\sigma^2(\mathcal{T})} \ , \tag{2}$$

where $\mu(x)$ and $\sigma(x)$ refer to the univariate mean and standard deviation of x. It is a univariate ANOVA (ANalysis Of VAriance) F-test, as a ratio between the variance of means of leakage conditioned upon the class and the total leakage variance. There is no definitive threshold for NICV. Thus, higher the value of NICV at a given point, more significant is the difference in leakage between each class. In this work, we utilize TVLA and NICV as tools for *feature selection* from side-channel traces.

III. Prior Works and Motivation

LWE/LWR-based PKE/KEMs have been targeted by side-channel attacks for two reasons: (1) Key Recovery (2) Message Recovery. However, most of works on SCA of these schemes have focussed on key recovery attacks targeting the long term secret key, while message recovery attacks leading to session key recovery are much less studied.

A. Key Recovery Attacks

Key recovery attacks on LWE/LWR-based PKE/KEMs can be broadly split into the following two classes.

1) Direct Key Recovery: These attacks work by directly targeting the polynomial/matrix-vector multiplication that manipulates the long term secret key in the decryption procedure. Several attacks have targeted different implementation variants such as the schoolbook multiplier [19], Number Theoretic Transform (NTT) [20] and the product-scanning based multiplier [21].

2) Message Recovery leading to Key Recovery: The second class of key-recovery attacks work by obtaining side-channel information about the decrypted message for *chosen-ciphertexts*, which leads to key recovery in LWE/LWR-based PKE/KEMs. D'Anvers *et al.* [22] reported the first such attack on two post-quantum KEMs LAC and RAMSTAKE, by utilizing timing side-channel information from the decryption procedure. It was used to extract binary information about the decrypted message for chosen-ciphertexts which led to full key recovery. Subsequently, Ravi *et al.* [5] used the EM side-channel and generalized the attack to constant-time implementation of several LWE/LWR-based KEMs. Both these attacks extracted binary information about the decrypted message, leading to key recovery in a few thousand queries.

More recently, Xu *et al.* [23] showed that an attacker with complete knowledge of the decrypted message for chosen ciphertexts can perform full key recovery only using 8 decryption queries for Kyber (Kyber512) and the same attack can be extended to other LWE/LWR-based schemes as well. This work highlights the need to protect the message in LWE/LWR-based schemes since any SCA vulnerability that leaks the complete message easily leads to recovery of the long term secret in a handful of decryption queries. This motivates us to analyze the presence of SCA leakage of the message in LWE/LWR-based PKE/KEMs.

B. Message Recovery Attacks

However, side-channel attacks targeting complete message recovery is much less studied and the *message encoding* operation within the encryption procedure is the only operation that has been analyzed in the context of message recovery. In this respect, Amiet *et al.* [8] in PQCrypto 2020 proposed the first single trace template style attack on NewHope KEM targeting the message encoding function. Their attack exploits leakage from a sensitive *determiner* variable which leaks information about single bits of the message. We refer to it as the Determiner-Leakage vulnerability throughout this work. Subsequently, Sim *et al.* [7] generalized the attack to target several LWE/LWR-based PKE/KEMs. In a very recently posted paper, Ngo *et al.* [9] exploited Determiner-Leakage to perform single trace message recovery on a masked implementation of Saber[2]. In the following, we briefly analyze the source of Determiner-Leakage at the micro-architectural level.

[2]The pre-print of our work [24] was posted prior to the posting of [9] by Ngo *et al.* . In fact, the authors also appropriately cite the pre-print of our work.

C. Analyzing Determiner-Leakage Vulnerability:

Referring to the encryption procedure PKE.Encrypt in Alg.1, the function Encode maps a message $\mathbf{m}$ with n bits into a corresponding polynomial $\mathbf{x} \in R_q$. Each message bit m_i for $i \in [0, n-1]$ is encoded into a corresponding coefficient $\mathbf{x}[i]$ such that $\mathbf{x}[i] = C \cdot \mathsf{m}_i$ where C is the center of the operating integer ring $\mathbb{Z}_q$.

Schemes such as NewHope and Kyber compute $\mathbf{x}[i] = \mathsf{mask}\ \&\ C$ where & is a *bitwise-and* operation and the mask takes two values $\mathsf{mask} = 0\text{xFFFF}$ if $\mathsf{m}_i = 1$ and $\mathsf{mask} = 0\text{x}0000$ otherwise. Though an efficient technique to encode, side-channel leakage (referred to as *determiner* leakage in [7]) from the *bitwise-and* operation easily leaks the value of mask (0x0000 or 0xFFFF) which reveals $\mathbf{m}_i$. This vulnerability has already been exploited for attacks on embedded ECC implementations [25], thus this leaky operation could have been avoided with a little more care. However, schemes such as Saber and Frodo avoid the use of the leaky mask and compute the product $\mathbf{x}[i] = C \cdot \mathsf{m}_i$ using a simple arithmetic shift operation since C is a power of 2. In these schemes, leakage mainly arises due to storage of the encoded coefficients $\mathbf{x}[i]$ (C or 0) of the message polynomial in memory. It is well known that the storage operation leaks the Hamming Weight (HW) of the stored value [26] (i.e) storage of X leaks $\mathsf{HW}(X)$. Thus, an attacker who can distinguish between $\mathsf{HW}(C)$ and $\mathsf{HW}(0) = 0$ can recover m_i and subsequently the complete message.

D. Looking Beyond Determiner-Leakage Vulnerability

We observe that the Determiner-Leakage vulnerability lies at the implementation level and can thus be easily circumvented using implemenation level changes. Analysis of the implementation of schemes such as Round5 and LAC shows that a mere alternate implementation choice for the Encode function seems to unintentionally eliminate the vulnerability. Instead of implementing Encode as a standalone function, it is combined with the subsequent polynomial addition operation (line 6) in an *interleaved* manner. The coefficient of the encoded message polynomial $\mathbf{x}[i] = k_d \cdot \mathsf{m}_i$ is computed and immediately added to the ciphertext coefficient $\mathbf{v}[i]$ and then stored to memory. This eliminates direct storage of $\mathbf{x}[i]$ thereby eliminating Determiner-Leakage. Alternatively, vectorized computation of multiple coefficients can be used as a fix to protect against the reported message recovery attacks targeting the Determiner-Leakage vulnerability. Thus, Determiner-Leakage can be avoided using fixes at the implementation level.

In this work, we however show that LWE/LWR-based schemes have inherent algorithmic properties which can be exploited in a side-channel setting to perform message recovery in a generic manner, adaptable to different implementation variants. Our attacks target a more fundamental operation (i.e.) storage of the decrypted message in memory. This operation cannot be easily avoided as the computed message is typically too long to be retained in registers (i.e) 256 bits or more and hence has to be moved to memory.

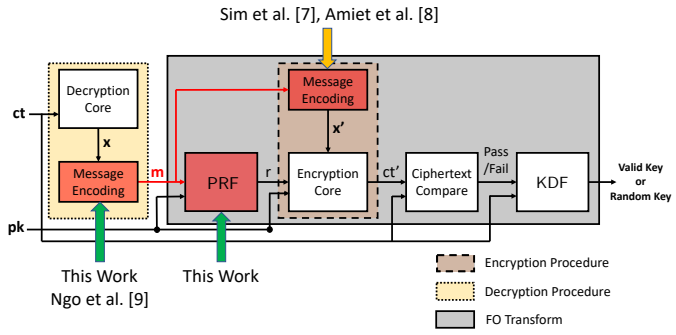


Figure 2: Illustration of our message recovery attacks targeting the IND-CCA secure decapsulation procedure (KEM.Decaps() in Alg.2) of LWE/LWR-based PKE/KEMs

This is especially true for embedded RISC based devices which typically contain very few working registers. We also propose novel attack techniques to break well known side-channel countermeasures such as shuffling and masking. Fig. 2 illustrates the targeted operations to perform message recovery attack within the IND-CCA secure decapsulation procedure.

IV. Side-Channel Analysis of the Message Decoding Operation

We observe that the secret message in LWE/LWR-based PKE/KEMs is manipulated in a very unique manner compared to conventional PKE/KEMs based on RSA and ECC. The decryption procedure computes the message polynomial $\mathbf{x}' \in R_q$ from the ciphertext ct (line 2 in PKE.Decrypt of Alg.1). Subsequently, a decoding procedure denoted as Decode (line 3) is used to iteratively map each coefficient $\mathbf{x}'[i]$ for $i \in [0, n-1]$ into a corresponding message bit m'_i, thereby computing the message one bit at a time. This type of *bitwise manipulation* at the algorithmic level is not observed in RSA/ECC-based schemes where the message is typically computed as a whole. In the following, we show that this behaviour gives rise to an exploitable side-channel vulnerability within the message decoding function, leading to full message recovery.

A. SCA Vulnerability of Message Decoding Operation:

We use Kyber based on the MLWE problem, which is one finalist of the NIST PQC competition, as a representative scheme for illustration, while our analysis applies in the same manner to schemes such as Saber, NewHope, Round5 and LAC unless otherwise specified. Refer to Fig. 3 for the C code snippet of the message decoding function Decode used in Kyber KEM [27] ($\mathsf{m} \in \mathcal{B}^{32}$). Please note that the same implementation is used in the NIST submission as well as within several indepdendently developed PQC libraries such as *pqm4* [10] and LibOQS [28].

1) Vulnerability Analysis: The Decode function takes as input $\mathbf{x} \in R_q$ ($n = 256$ coefficients) and outputs the message $\mathsf{m} \in \mathcal{B}^{32}$. The message bytes are first initialized to zero (line 9 in Fig. 3). Every coefficient $\mathbf{x}[k]$ for $k \in [0, 256]$ with $k = (8 * i + j)$ is iteratively decoded to bit t (line x)

```
1  void Decode(unsigned char *m, poly *x)
2  {
3    uint16_t t;
4    int i, j;
5    poly_csubq(x);
6    for (i = 0; i < 32; i++)
7    {
8      /* init byte m[i] to zero */
9      m[i] = 0;
10     for (j = 0; j < 8; j++)
11     {
12       k = 8*i+j;
13       t = (x->coeffs[k] ≪ 1) + Q/2;
14       /* Calculate Message Bit */
15       t = (t/Q) & 1;
16       /* Bit Update in Memory */
17       m[i] |= t ≪ j;
18     }
19   }
20 }
```

Figure 3: C code snippet of message decoding operation in Kyber KEM

which is then updated in $\mathsf{m}[i]_j$ (i.e) bit j of byte $\mathsf{m}[i]$ in memory (line 17). Thus, every byte $\mathsf{m}[i]$ for $i \in [0, 31]$ is incrementally updated in memory one bit at a time in 8 iterations of the *for* loop running over variable j.

```
1  /* t = (x->coeffs[n]<<1)+Q/2; in r6 */
2  LDRSH.W r6, [r4, #2]
3  LSLS  r6, r6, #1
4  ADD.W r6, r6, #1664 ; 0x680
5  /* t = (t/Q) & 1; in r6 */
6  SMULL ip, r7, r1, r6
7  ADD r7, r6
8  ASRS  r6, r6, #31
9  RSB r6, r6, r7, asr #11
10 AND.W r6, r6, #1
11 /* m[i] |= t << j; in r3 */
12 ORR.W r3, r3, r6, lsl #1
13 /* Store updated m[i] in memory */
14 STRB r3, [r2, #0]
```

Figure 4: Assembly code snippet of a single iteration of the message decoding function in Kyber KEM

We analyze the compiled assembly code to better understand the effect of bitwise manipulation at the micro-architectural level on our target platform (i.e) 32-bit ARM Cortex-M4. We compiled our implementations using the `arm-none-eabi-gcc` compiler with the highest compiler optimization level `-O3`. Refer to Fig. 4 for the compiled assembly code for the body of the innermost loop running over variable j in the C code of Fig. 3. We denote the intermediate value of message byte $\mathsf{m}[i]$ at the end of the j^{th} iteration as $\mathsf{m}[i, j]$ with $j \in [0, 7]$. We consider the update of bit $\mathsf{m}[i]_j$ to the intermediate byte $\mathsf{m}[i, j-1]$ in memory for illustration. Register r3 contains the current value (i.e) $\mathsf{m}[i, j-1]$ and the decoded bit t is computed in register r6 (line 10). Then, r6 is left shifted by j positions (in our case, $j = 1$) and subsequently *bitwise-or'red* with

r3 to compute the updated message byte $\mathsf{m}[i,j]$ (line 12). The result in r3 is then stored to memory using the STRB instruction (line 14). The same set of operations is repeated 8 times for every message byte $\mathsf{m}[i]$ with $i \in [0, 31]$.

2) Attack Methodology: Since the power/EM side-channel leaks the Hamming weight (HW) of the stored value, side-channel information from the STRB instruction of every iteration leaks (roughly) the HW of the corresponding intermediate value $\mathsf{m}[i,j]$. Recovery of $\mathsf{HW}(\mathsf{m}[i,j])$ for all $j \in [0,7]$ can be used to trivially recover $\mathsf{m}[i]$ in the following manner. Since $\mathsf{m}[i]$ starts with a value of zero, HW of the first store $\mathsf{m}[i,0]$ is nothing but the first bit $\mathsf{m}[i]_0$. Subsequently, the other bits $\mathsf{m}[i]_j$ for $j \in [1,7]$ can be retrieved using the following rule:

$$\mathsf{m}[i]_j = \begin{cases} 0, & \text{if } \mathsf{HW}(\mathsf{m}[i,j]) = \mathsf{HW}(\mathsf{m}[i,j-1]) \\ 1, & \text{if } \mathsf{HW}(\mathsf{m}[i,j]) = \mathsf{HW}(\mathsf{m}[i,j-1]) + 1 \end{cases} \quad (3)$$

The same procedure can be applied to the other message bytes for full message recovery. Thus, we observe that bitwise computation of the decrypted message leads to an incremental update of message in memory and we refer to this as the Incremental-Storage vulnerability throughout the paper. Thus, an attacker with a perfect HW classifier can recover the full message in a single trace. The same vulnerability/behaviour also exists in the compiled code at all optimization levels (-O0 to -O3) of Kyber KEM and we also observe a very similar behaviour in implementations of *four* other schemes - NewHope, Round5, Saber and LAC whose analysis is provided in Sec. II of the supplementary material.

V. Single Trace Message Recovery Attack

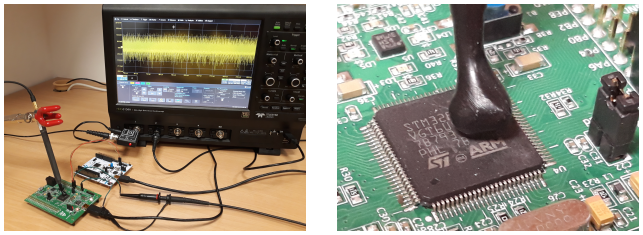
We now demonstrate efficient attack techniques to perform practical single trace message recovery attacks targeting the Incremental-Storage vulnerability in LWE/LWR-based PKE/KEMs.

A. Adversary Model

Given a ciphertext ct, the attacker's main motive is to recover the hidden message m. The ciphertext corresponds to a valid PKE/KEM instance between the target device (DUT) and another legitimate device. With the recovered message and the corresponding ciphertext, an attacker can recover the corresponding shared secret/session key as shown in Alg.2. We assume the following attacker capabilities:

- Physical access to DUT performing decapsulation for power/EM measurement.
- Ability to request the DUT to decrypt arbitrary number of chosen ciphertexts.
- No knowledge of secret key of the DUT or any innate knowledge of the underlying implementation such as the source or compiled executable.

While recent works have shown that remote power measurement on embedded devices is possible [29],our experiments assume physical access and uses the setup described in the following.



(a) (b)

Figure 5: Experimental Setup for SCA (a) SCA Setup (b) Zoomed-in view of EM-probe over the DUT

B. Experimental Setup

The DUT is the STM32F407VG microcontroller housed on the STM32F4DISCOVERY evaluation board. The implementations of the targeted schemes are taken from the public *pqm4* library [10], a benchmarking and testing framework for PQC schemes on the 32-bit ARM Cortex-M4 microcontroller, which is a NIST recommended optimization target for embedded software implementations. All our target implementations are clocked at 24 MHz. We use the EM side-channel for our experiments and side-channel measurements/traces were observed using a Langer RF-U 5-2 near-field probe placed on top of the chip and are then collected using a Lecroy 610Zi oscilloscope at a sampling rate of 1.25 GSam/sec, amplified 30dB with a pre-amplifier. Refer Fig. 5 for our EM-based SCA setup used for our experiments. *We omit results of measurements at 100MSam/sec, which also reported successful attacks. Attack success at these low sampling rate makes our work compatible with low-cost platforms like Chipwhisperer.*

For efficient attacks, measured traces are desired to have a high Signal to Noise Ratio (SNR). Some common techniques to boost SNR involve employing high precision EM probes, hardware analog filters, averaging of repeated measurements, advanced digital filtering, trace re-synchronization to remove jitter, averaging etc. The choice of noise reduction technique is completely platform dependent. For all our experiments, we emulate SNR boosting by averaging of side-channel information from repeated experiments.

C. Leakage Detection

We first validate the presence of side-channel leakage due to the Incremental-Storage vulnerability. We adopt the TVLA metric to perform leakage detection and focus on detecting leakage from the first byte $\mathsf{m}[0]$. We construct two sets of ciphertexts, denoted as CT_0 and CT_1. Both sets contain ciphertexts of random messages except that their first message byte is fixed to 0 ($\mathsf{m}[0] = 0$) and 1 ($\mathsf{m}[0] = 1$) respectively. If $\mathsf{m}[0] = 0$, then $\mathsf{HW}(\mathsf{m}[0,j]) = 0 \; \forall \, j \in [0,7]$, else $\mathsf{HW}(\mathsf{m}[0,j]) = 1 \; \forall \, j \in [0,7]$ if $\mathsf{m}[0] = 1$. This persistent 1 bit difference in the Hamming weight of all eight intermediate updates (i.e) $\mathsf{HW}(\mathsf{m}[0,j]) \forall j \in [0,7]$ should be detectable through the EM side-channel.

We collect two sets of $\ell = 500$ EM side-channel traces corresponding to decapsulation of ciphertexts in sets CT_0

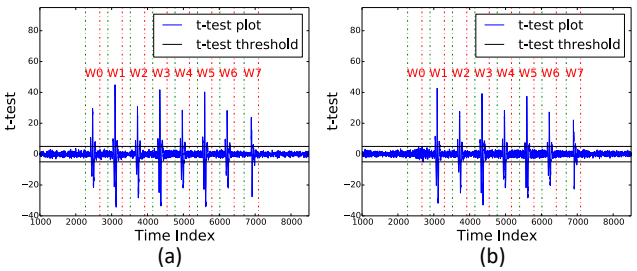


Figure 6: TVLA results for Kyber targeting message byte $m[0]$ (a) $m[0] = 0$ and $m[0] = 1$ (b) $m[0] = 0$ and $m[0] = 2$

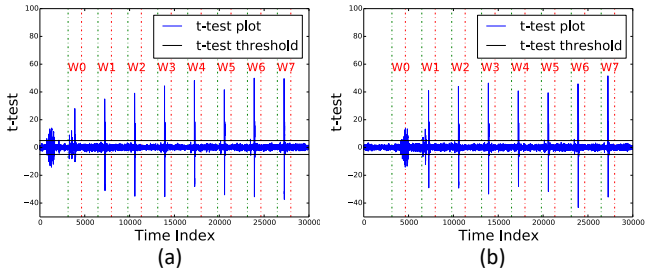


Figure 7: TVLA results for NewHope for message byte $m[0]$ (a) $m[0] = 0$ and $m[0] = 1$ (b) $m[0] = 0$ and $m[0] = 2$

and CT_1 denoted as $\mathcal{T}_0$ and $\mathcal{T}_1$ respectively. We normalize each trace and compute the Welch's t-test to identify the differentiating features between the trace sets. Refer to Fig. 6(a) for the t-test plot for Kyber which shows eight distinct peaks (greater than the pass-fail threshold ± 4.5) that correspond to the storage of $\mathsf{m}[0, j]$ for $j \in [0, 7]$. We repeated the same experiments between $m[0] = 0$ and $m[0] = 2$ and observed 7 distinct peaks since $\mathsf{HW}(m[0, 0]) = 0$ for both sets (Fig. 6(b)). For validation, we also repeat the same experiments on NewHope which also showed the same behaviour (Refer to Fig. 7(a) and Fig. 7(b)), thus confirming our hypothesis of side-channel leakage due to the Incremental-Storage vulnerability. We also refer the reader to Sec. I of supplementary material which similarly demonstrates leakage in Kyber using a low-cost setup with low sampling rate of 100 MSam/sec. This leakage detection test also helps us precisely identify the narrow time window W_j of every intermediate byte update $\mathsf{m}[i, j]$ for $j \in [0, 7]$ as shown in Fig. 6 and Fig. 7.

D. Two Phase Message Recovery Attack

Our message recovery attack works in two phases - (1) *Pre-Processing* phase and (2) *Exploitation* phase. The attack technique must not be confused with popular profiled attacks which needs complete access to a clone device used for profiling. *In our attack, the pre-processing is done over public information without any knowledge of secret information. Thus, the attacker can directly perform the pre-processing on DUT without a need of clone device. The attack technique also applies in a generic manner to all schemes that exhibit* Incremental-Storage *of the decrypted message.*

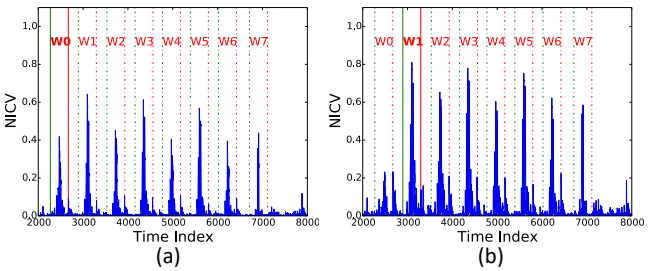


Figure 8: NICV plot for iterations (a) $j = 0$ and (b) $j = 1$ for Kyber. The corresponding time windows $W0$ and $W1$ have been highlighted in bold.

1) Pre-Processing *Phase:* It involves building side-channel templates for different values of the decrypted message. It is only a one-time process for a given target device since the same templates can be used for multiple attacks. Moroever, the chosen ciphertexts used for profiling can correspond to different public-private key pairs $(\mathsf{pk}, \mathsf{sk})$ used by the target device since templates are only built for the message.

We individually profile each message byte and profiling byte $\mathsf{m}[i]$ requires to build HW templates independently for all of its eight intermediate updates (i.e) update of $\mathsf{m}[i, j]$ for $j \in [0, 7]$. The first update $m[i, 0]$ only has two possible HWs (0 and 1). The number of possible HWs increases by one with every iteration, with 9 possible HWs (0 to 8) in the last iteration $j = 7$. Thus, $(j+2)$ HWs (i.e) (0 to $j+1$) are possible for the update of $\mathsf{m}[i, j]$. We focus on building templates for the update of $\mathsf{m}[i, j]$ in memory.

For each class $k \in [0, j + 1]$, we construct a valid ciphertext set $\mathsf{CT}^k_{(i,j)}$ containing ℓ ciphertexts of random messages which satisfy the condition: $\mathsf{HW}(\mathsf{m}[i, j]) = k$. The corresponding side-channel traces are denoted as $\mathcal{T}^k_{(i,j)}$. We use the NICV metric to select those features in $\mathcal{T}^k_{(i,j)}$ for $k \in [0, j+1]$ that distinguishes the corresponding HW class. We compute NICV over $\mathcal{T}^k_{(i,j)}$ for $k \in [0, j + 1]$ and select those features within the corresponding window W_j whose NICV value is above a certain threshold $Th_{(i,j)}$ as our set of Points of Interest (PoI) denoted as $\mathcal{P}_{(i,j)}$. The threshold $Th_{(i,j)}$ for each $\mathsf{m}[i, j]$ is a parameter of the experimental setup and is empirically determined. Please refer to Fig. 8(a)-(b) for the NICV plot for iterations $j = 0$ and $j = 1$ of byte $\mathsf{m}[0]$ in Kyber where we can identify clear NICV peaks within their respective time windows W_0 and W_1. When profiling iteration j, we also observe NICV peaks in time windows of other iterations $(\mathsf{W}_k \neq \mathsf{W}_j)$ but they can be ignored.

We now use the selected features $\mathcal{P}_{(i,j)}$ to build a reduced trace set $\mathcal{RT}^k_{(i,j)}$ from $\mathcal{T}^k_{(i,j)}$ and the mean of each reduced trace set $\mathcal{RT}^k_{(i,j)}$ denoted as $\mathsf{rt}^k_{(i,j)}$ serves as the reduced template for $\mathsf{HW}(\mathsf{m}[i, j]) = k$. Building similar templates for all $k \in [0, j+1]$ completes the profiling of the update of $\mathsf{m}[i, j]$. Similarly, the other iterations $j \in [0, 7]$ of $\mathsf{m}[i]$ can

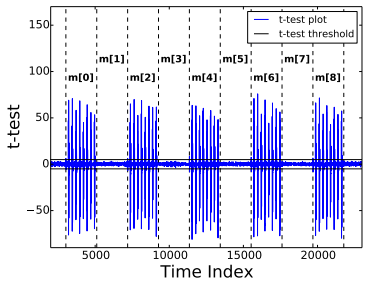


Figure 9: TVLA results for Kyber showing simultaneous leakage from multiple message bytes (i.e.) $m[j]$ for $j \in \{0, 2, \ldots, r-1\}$ (r denotes total number of message bytes)

be profiled in the same manner resulting in a full template set for message byte $\mathsf{m}[i]$. For practical measurements, we collected 500 traces each for every 256 possible values of the message byte $\mathsf{m}[0]$ to build templates for the message byte $\mathsf{m}[0]$ that amounts to about $128k$ traces. Thus, building similar templates for all message bytes ($r = 32$ bytes) would require $4.096m$ traces.

However, we observe that all message bytes are processed independently and we can utilize the same trace set ($128k$ traces) to simultaneously build templates for multiple message bytes. For instance, we can profile bytes at even indices (i.e.) $m[j]$ for $j \in \{0, 2, \ldots, r-1\}$ using a single trace set and the remaining bytes with another trace set in the following manner. We create ciphertexts with message bytes at even indices (i.e) $m[j]$ for $j \in \{0, 2, \ldots, r-1\}$ fixed to the value k with $k \in [0, 256]$ while the other bytes are random. Refer to Fig.9 for the t-test plot between two sets of ciphertexts whose message bytes at even indices have a value of 0 and 1 respectively, while the other bytes are random.

The t-test clearly shows simultaneous leakage from multiple message bytes and thus a single trace set with $128k$ traces can be used to build templates for all these message bytes. The same can be done for the other half of message bytes (odd indices). Thus, all message bytes can be profiled only using $(128 \times 2) = 256k$ traces. Note that the exact number of traces required for profiling is an empirical parameter of the experimental setup. We merely establish that multiple message bytes can be processed simultaneously, thereby reducing the number of traces required in the pre-processing phase.

2) Exploitation *Phase:* The attacker now matches the obtained HW templates with the trace tr obtained from decapsulation of target ciphertext ct to perform message recovery. A given byte $\mathsf{m}[i]$ can be recovered using the HWs of all of its intermediate updates (i.e) $\mathsf{HW}(\mathsf{m}[i,j])\ \forall\ j \in [0, 7]$.

To recover $\mathsf{HW}(\mathsf{m}[i,j])$, we build a reduced trace $\mathsf{tr}'_{(i,j)}$ corresponding to the PoI set $\mathcal{P}_{(i,j)}$. We then compute the sum-of-squared difference Γ_k between tr' and each reduced template $\mathsf{rt}^k_{(i,j)}$ for $k \in [0, j+1]$ as follows:

$$\Gamma^k = (\mathsf{tr}' - \mathsf{rt}^k_{(i,j)})^T \cdot (\mathsf{tr}' - \mathsf{rt}^k_{(i,j)})$$

Algorithm 3: SCA-Assisted Message Recovery Attack

Procedure *Pre-Processing*()
  for $i = 0$ *to* $r-1$ **do**
    for $j = 0$ *to* 8 **do**
      /* Trace Acquisition */
      for $k = 0$ *to* $j+1$ **do**
        $\mathcal{T}_{(i,j)} \Longleftarrow \mathsf{Decaps}(\mathsf{CT}_{(i,j)})$;
      end
      /* NICV-based Feature Selection */
      $\mathcal{P}_{(i,j)} = \mathsf{NICV\text{-}Select}(\mathcal{T}^0_{(i,j)}, \mathcal{T}^1_{(i,j)}, \ldots, \mathcal{T}^{(j+1)}_{(i,j)})$;
      /* Build Reduced HW templates */
      for $k = 0$ *to* j **do**
        $\mathcal{RT}^k_{(i,j)} = \mathcal{T}^k_{(i,j)}(\mathcal{P}_{(i,j)})$;
        $rt^k_{(i,j)} = \mathsf{Mean}(\mathcal{RT}^k_{(i,j)})$;
      end
    end
  end
Procedure *Attack*$(rt^*_{(i,j)}, \mathcal{P}_{(i,j)}$ *for* $i \in [0, r-1]$ *and* $j \in [0, 7])$
  for $i = 0$ *to* $r-1$ **do**
    for $j = 0$ *to* 8 **do**
      /* Build Reduced trace */
      $tr' = tr(\mathcal{P}_{(i,j)})$;
      /* LSQ-Test with Reduced HW templates */
      for $k = 0$ *to* $j+1$ **do**
        $\Gamma[k] = \mathsf{LSQ\text{-}Test}(tr', rt^k_{(i,j)})$;
      end
      /* Class Assignment based on LSQ-test */
      $k_* = \mathsf{argmin}(\Gamma)$;
      $\mathsf{HW}(\mathsf{m}_{(i,j)}) = k_*$;
    end
    /* Recover m[i] using HW progression */
    $\mathsf{m}[i] = \mathsf{Recover}(\mathsf{HW}(\mathsf{m}_{(i,0)}), \mathsf{HW}(\mathsf{m}_{(i,1)}), \ldots, \mathsf{HW}(\mathsf{m}_{(i,7)}))$;
  end

We then assign $\mathsf{HW}(\mathsf{m}[i,j]) = k$ based on the smallest value of Γ^k (i.e) reduced template with the least distance from the reduced attack trace. We can similarly recover $\mathsf{HW}(\mathsf{m}[i,j])\ \forall\ j \in [0, 7]$ leading to recovery of $\mathsf{m}[i]$ and similarly the full message.

Confidence in HW *Classification*: The SNR available in the side-channel measurements heavily impacts the success rate of Hamming weight classification. We devise a techique to label a given HW classification of $\mathsf{HW}(\mathsf{m}[i,j])$ as confident or doubtful in the following manner. We sort the classes in increasing order of Γ^k for $k \in [0, j+1]$ and let the corresponding ordered set of HW classes be denoted as $\mathcal{W} = \{\mathsf{HW}_k\}$ with $k \in [0, j+1]$. We label the classification as confident only if $\Gamma_{\mathsf{HW}_2} \geq (\mathcal{C}_{(i,j)} \cdot \Gamma_{\mathcal{HW}_1})$ else the classification is labelled doubtful. The value of $\mathcal{C}_{(i,j)}$ for each iteration is a parameter of the experimental setup and is empirically determined. Thus, all the updates whose Hamming weight class is labelled as doubtful will need to be brute-forced for message recovery.

We summarize our *pre-processing* and *attack* methodology in the form of an algorithm in Alg.3 where the function $\mathsf{NICV\text{-}Select}()$ refers to NICV-based feature selection. $\mathsf{LSQ\text{-}Test}()$ refers to the least sum-of-squared difference computation and $\mathsf{Recover}()$ refers to retrieval of byte $\mathsf{m}[i]$ from $\mathsf{HW}(\mathsf{m}[i,j])\ \forall\ j \in [0, 7]$, according to Eqn.3.

3) Experimental Results: We perform experimental validation of our attack on Kyber which serves as an exemplar for Module-LWE/LWR based schemes such as Saber. We additionally validate our attacks on NewHope which serves as an exemplar for Ring-LWE/LWR based schemes such as Round5 and LAC. Fig. 10(a)-(b) shows the evolution of success rate against SNR for message recovery

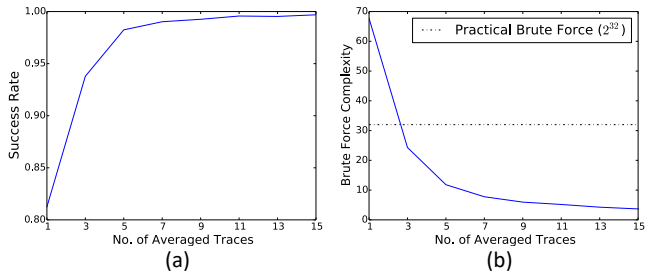


Figure 10: Success rate and Brute Force Complexity for full message recovery against SNR for Kyber

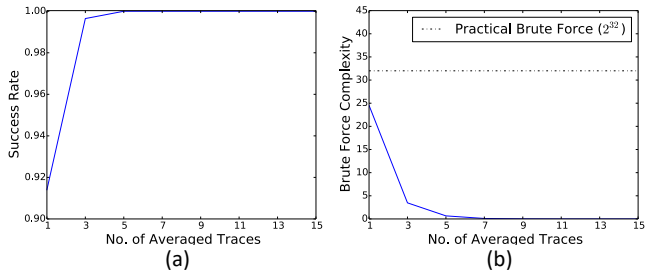


Figure 11: Success rate and Brute Force Complexity for full message recovery against SNR for NewHope

of a single message byte for Kyber (other bytes can be recovered independently in the same manner). Without SNR enhancement, the success rate stands at 81.25% with a brute-force complexity of 2^{67}, however the success rate quickly ramps to 98.24% with just 5 averaged traces and settles to about 99.5% with a brute-force complexity of 2^6. As stated earlier, there are a range of techniques which can be adopted to boost SNR. We emulate SNR boosting by averaging repeated measurements. Similarly, we also validate our attack on NewHope (Refer to Fig. 11(a)-(b)) and without SNR enhancement, the success rate stands at 91.25% with a 2^{24} brute-force however the success rate quickly goes to 100% with increase in SNR. An attacker with an optimized attack setup with high SNR can perform full message recovery in a single trace. Thus, a side-channel based HW classifier can be efficiently used to target the Incremental-Storage of decrypted message to perform full message recovery in LWE/LWR-based schemes.

E. Eliminating the Incremental-Storage *Vulnerability*

As shown in Sec. V, bitwise manipulation of the decrypted message manifests as an Incremental-Storage vulnerability at the implementation level leading to efficient message recovery. We attempt to propose an implementation fix to eliminate the Incremental-Storage vulnerability. From the C code snippet of the message decoding operation (Fig. 3), we observe that the message bit is directly updated within $\mathsf{m}[i]$ in memory, resulting in a store in each iteration (line 17). Instead, the message byte $\mathsf{m}[i]$ can simply be accumulated in a temporary variable temp over eight iterations (i.e) $temp \mid = t \ll j$ in line 17. Subsequently, the temp variable can be pushed to $\mathsf{m}[i]$ after the innermost for loop over variable j (i.e) once every eight iterations.

In the modified implementation, we observed that the message bits are now aggregated in *registers*, thereby eliminating the intermediate stores and only the fully updated message byte is stored in memory. We remark that this was not done by the compiler even upon compilation with the highest optimization level (O3). Though such Bytewise-Storage of the decrypted message seems to defeat our single trace attack, we show that inherent algorithmic properties of LWE/LWR-based PKEs can be exploited to still perform full message recovery.

VI. Ciphertext Malleability in LWE/LWR-based PKE/KEMs

In this section, we explain the ciphertext malleability property of LWE/LWR-based PKEs which can serve as a crucial tool for a side-channel attacker for message recovery. Given a ciphertext $\mathbf{ct}$ for an unknown message m, we can construct modified ciphertexts ($\mathbf{ct}'$), that decrypt to deterministic variants m' of the original message m. There are two ways to manipulate unknown messages of target ciphertexts - (1) Targeted flip of message bits and (2) Cyclic message rotation.

1) Targeted Flip of Message Bits: Referring to the encryption procedure PKE.Encrypt in Alg.1, the encoded message polynomial $\mathbf{x}$ is simply added to a pseudorandom LWE instance $\mathbf{v}'$, which is subsequently output as the ciphertext component $\mathbf{v}$ (line 6). Thus, the message polynomial is only additively hidden within the ciphertext $\mathbf{v}$ (i.e) $\mathbf{v}[i] = \mathbf{x}[i] + \mathbf{v}'[i]$ for $i \in [0, n-1]$. Moreover, $\mathbf{x}[i]$ can only take two values (i.e) $\mathbf{x}[i] = C$ (center of the integer ring $\mathbb{Z}_q$) if the corresponding bit $\mathsf{m}_i = 1$, else $\mathbf{x}[i] = 0$ otherwise. We also observe that the decryption procedure PKE.Decrypt extracts a noisy version of the message polynomial by simply subtracting the pseudorandom LWE instance $\mathbf{v}'$ from $\mathbf{v}$ (line 2). Thus in essence, there is no mixing/interaction between the different coefficients $\mathbf{x}[i]$ of the the encoded message polynomial $\mathbf{x}$. This type of *scalar* behaviour in handling the message within LWE/LWR-based PKE/KEMs is very different compared to classical RSA and ECC-based schemes and enables to target individual bits of the message.

Thus, a given bit m_i can be flipped ($1 \rightarrow 0$ or $0 \rightarrow 1$) by simply subtracting C from the corresponding ciphertext coefficient $\mathbf{v}[i]$. This property applies in a generic manner to all LWE/LWR-based PKEs. While schemes such as Kyber, Saber, Round5 and LAC encode a given bit into a single coefficient, NewHope and Frodo adopt redundancy in the encoding/decoding operation. We refer the reader to Sec. IV of supplementary material for adaptation of the targeted bit flip property to NewHope and Frodo.

Since all the message bits are handled independently, we can simultaneously flip any number of bits of the message m by subtracting C from the corresponding coefficients of $\mathbf{v}$. We can thus build ciphertexts $\mathbf{ct}'$ which decrypts to a modified message m' whose targeted bits are flipped compared to the original message m. We denote $\mathsf{m}'_i =$

$\mathsf{Flip}(m, i)$ whose i^{th} bit has been flipped compared to m and the corresponding ciphertext is denoted as $\mathbf{ct}'_i = \mathsf{Flip}(ct, i)$. This is very similar to the malleability property of Cipher block chaining (CBC) mode of operation for block ciphers that allows to selectively flip single bits of the decrypted plaintext [30]. We refer to this as the Bit-Flip property of LWE/LWR-based PKE/KEMs throughout this paper.

2) Cyclic Message Rotation: We refer the reader to Sec. III of the supplementary material for a detailed explanation of the cyclic message rotation property. We do not utilize this property to aid our message recovery attacks, but speculate its usage for future attacks.

VII. Generic Message Recovery Attacks for LWE/LWR-based PKE/KEMs

In this section, we demonstrate efficient exploitation of ciphertext malleability as a effective tool to perform generic message recovery attacks which can be adapted to different implementation variants of the LWE/LWR-based schemes for message recovery.

A. Exploiting Ciphertext Malleability for Message Recovery

Referring to the Bytewise-Storage type of implementation of the decoding procedure (Sec.V-E), we saw that the improved design eliminates incremental stores, thereby offering protection against the single trace attack. However, we observe that the decoding procedure still stores all message bytes $\mathsf{m}[i]$ for $i \in [0, r-1]$ in memory. Their side-channel leakage can be used to deduce their corresponding Hamming weights using the HW classifier (i.e.) $\mathsf{HW}(\mathsf{m}[i])$ for $i \in [0, m-1]$. We now use the Bit-Flip property to construct $\mathbf{ct}' = \mathsf{Flip}(\mathsf{ct}, 0))$ which decrypts to $\mathsf{m}' = \mathsf{Flip}(\mathsf{m}, 0))$ (i.e) flip bit m_0. We then query the target device to decrypt $\mathbf{ct}'$ and recover $\mathsf{HW}(\mathsf{m}'[0])$. Flipping m_0 will create a perturbation in $\mathsf{HW}(\mathsf{m}[0])$ and bit m_0 can be recovered as follows:

$$\mathsf{m}_0 = \begin{cases} 0, & \text{if } \mathsf{HW}(\mathsf{m}'[0]) = \mathsf{HW}(\mathsf{m}[0]) + 1 \\ 1, & \text{if } \mathsf{HW}(\mathsf{m}'[0]) = \mathsf{HW}(\mathsf{m}[0]) - 1 \end{cases} \tag{4}$$

In a similar manner, we can construct ciphertexts to separately flip other bits of $\mathsf{m}[0]$ and fully recovery $\mathsf{m}[0]$ one bit at a time. Since all message bytes are stored iteratively, it is possible to simultanouesly flip one bit in each byte $\mathsf{m}[i]$ for $i \in [0, r-1]$ and recover these r bits in a single decapsulation query/trace. Thus, complete message recovery is possible only using 8 adapted ciphertext queries and 1 original ciphertext query. While our attack on the $\mathsf{Incremental-Storage}$ implementation only required a single trace (provided enough SNR), our attack on the Bytewise-Storage requires 9 traces for full message recovery.

It is straightforward to see that the aforementioned attack methodology exploiting malleability can be used to target storage of the decrypted message in memory of any width (i.e) bytewise (8-bits), half-wordwise (16-bits) or wordwise (32-bits). In the presence of a side-channel HW classifier, full message recovery can be performed in $(w+1)$ traces where w is the storage width. This makes our attack applicable not only to the message decoding operation, but to any other operation that involves storage of the decrypted message in memory.

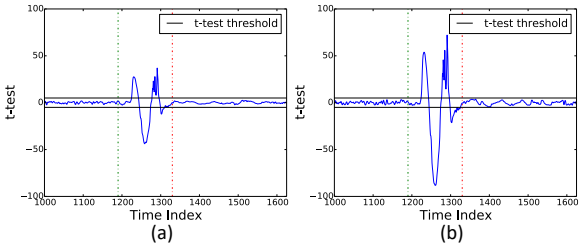


Figure 12: TVLA results for Kyber KEM (Kyber512) to distinguish $\mathsf{HW}(\mathsf{m}[0 \rightarrow 3])$ between HW classes (a) HW = 1 and HW = 2 and (b) HW = 1 and HW = 3

1) Targeting Other Operations: We analyze the IND-CCA secure decapsulation procedure to identify other operations that manipulate the decrypted message (procedure KEM.Decaps in Alg.2). We observe that the decrypted message is appended to the public key and passed to a Pseudo Random Function (PRF), immediately after decryption (line 3). The PRF is implemented in several schemes using the well known Keccak permutation and we identified an internal operation in the KeccakAbsorb function that copied the decrypted message from one memory location to another, 32-bits at a time.

We used the same t-test based leakage detection approach to confirm leakage from HW storage of the decrypted words and for illustration, we focus on leakage from the first word denoted as $\mathsf{m}[0 \rightarrow 3]$. Fig. 12(a) shows the t-test plot computed between two sets of $\ell = 500$ traces each for $\mathsf{HW}(\mathsf{m}[0 \rightarrow 3]) = 1$ and $\mathsf{HW}(\mathsf{m}[0 \rightarrow 3]) = 2$. The plot shows a single peak well beyond the t-test threshold. Similarly, the t-test plot between $\mathsf{HW}(\mathsf{m}[0 \rightarrow 3]) = 1$ and $\mathsf{HW}(\mathsf{m}[0 \rightarrow 3]) = 3$ in Fig. 12(b) shows a higher peak at the same time instance due to a larger difference in Hamming weight, concretely proving presence of HW leakage, which can be exploited in a similar manner.

In a nutshell, ciphertext malleability can be exploited to target any leakage related to storing of the message (bit-wise or otherwise). The earlier attacks of [7], [8], [9] targeting bitwise storage of the message can thus be considered as specific instances of our generic message recovery attacks.

VIII. Attacking Protected Implementations

Shuffling and masking are two well known countermeasures used to protect against side-channel analysis [31], [32]. In this section, we show that ciphertext malleability can yet again be used as an effective tool to break implementations of LWE/LWR-based PKE/KEMs protected with the aforementioned countermeasures.

Attack Assumption: We make an additional assumption along with the attacker capabilities stated in the adversary model in Sec. V-A. The *pre-processing* phase requires to build HW templates using decapsulation queries for known messages. However, both shuffling and masking countermeasures randomly modify the processed message,

thereby disabling the attacker from building HW templates. Hence, for attacks on protected implementation, we assume the presence of a clone device, in which the attacker can turn off or deactivate the countermeasure to build the required HW templates. This is a common assumption used often in profiled attacks [33].

A. Attacking the Shuffling Countermeasure

Shuffling the order of processing of message bits was proposed as a concrete countermeasure by Amiet *et al.* [8] in PQCrypto'20 as well as Sim *et al.* [7], to protect against single trace attacks targeting Determiner-Leakage in the message encoding operation [7], [8], which we denote as the Shuffled-Determiner-Leakage in this work.

1) Attacking PQCrypto'20 Countermeasure for Message Encoding: We target the message encoding operation of the re-encryption procedure in IND-CCA secure decapsulation (line 4 in KEM.Decaps of Alg.2). Shuffling does not remove the source of Determiner-Leakage but simply randomizes the order of the message bits. Thus, a side-channel attacker can recover all the individual bits m_i for $i \in [0, n-1]$ but without the correct ordering. Since shuffling does not modify the Hamming weight of m, we compute the Hamming weight of m (i.e) $\mathsf{HW}(\mathsf{m})$.

We use the Bit-Flip property to construct $\mathsf{ct}' = Flip(\mathsf{ct}, i)$ which decrypts to $\mathsf{m}' = \mathsf{Flip}(\mathsf{m}, i)$ (i.e) flip bit m_i. We query the target device to decapsulate ct' and similarly recover the modified Hamming weight $\mathsf{HW}(\mathsf{m}')$. The flipped bit m_i can then be recovered as follows:

$$\mathsf{m}_i = \begin{cases} 0, & \text{if } \mathsf{HW}(\mathsf{m}') = \mathsf{HW}(\mathsf{m}) + 1 \\ 1, & \text{if } \mathsf{HW}(\mathsf{m}') = \mathsf{HW}(\mathsf{m}) - 1 \end{cases} \tag{5}$$

In the same way, other bits of the message can be flipped to recover the full message one bit at a time. Assuming use of a single trace side-channel HW classifier, full message recovery can be done in $(n+1)$ traces (n adapted ciphertexts queries and 1 target ciphertext query).

2) Attacking PQCrypto'20 Countermeasure for Message Decoding: We now propose attacks to break shuffling countermeasure for the message decoding operation. We consider two variants - (1) Shuffled-Incremental-Storage and (2) Shuffled-Bytewise-Storage.

a) *Attacking* Shuffled-Incremental-Storage*:* To recall, our attack targeting Incremental-Storage works by recovering HW of the intermediate byte updates $\mathsf{HW}(\mathsf{m}[i,j])\ \forall j \in [0,7], i \in [0, r-1]$. Shuffling their order does not remove the source of leakage, thus their Hamming weights can be recovered using the HW classifier.

We hypothesize that a single bit flip will create an observable bias in the average Hamming weight of the intermediate byte updates observed across several executions. For the valid ciphertext ct, the average Hamming weight of the shuffled intermediate byte updates is denoted as hw_{avg}. Let $\mathcal{HW}_{avg}(\mathsf{m})$ be the set of hw_{avg} for ℓ such replicated executions. We repeat the same for the adapted ciphertext $\mathsf{ct}' = \mathsf{Flip}(\mathsf{ct}, i)$ which decrypts to $\mathsf{m}' = \mathsf{Flip}(\mathsf{m}, i)$ (i.e) flip bit m_i. We similarly obtain the set $\mathcal{HW}_{avg}(\mathsf{m}')$. If $\mathsf{m}_i = 1$, then $\mathsf{mean}(\mathcal{HW}_{avg}(\mathsf{m}))$ should be higher than that of $\mathsf{mean}(\mathcal{HW}_{avg}(\mathsf{m}'))$ since $\mathsf{m}'_i = 0$. We use the t-test score (denoted as $\mathcal{D}$) to distinguish the mean of two sets and recover m_i as follows:

$$\mathsf{m}_i = \begin{cases} 0, & \text{if } \mathcal{D} > +4.5 \\ 1, & \text{if } \mathcal{D} < -4.5 \end{cases} \tag{6}$$

Assuming the presence of a single trace HW classifier, we performed attack simulations over Kyber and NewHope ($n = 256$). We empirically observed that about $\ell = 1500$ queries are required to recover a single message bit with a 100% success rate. This amounts to a total of $1500 \times 256 + (1500) = 385.5k$ traces for full message recovery targeting Shuffled-Incremental-Storage. While shuffling significantly increases the attacker's effort for message recovery (compared to single trace for unprotected variant), it still does not offer concrete protection.

b) *Attacking* Shuffled-Bytewise-Storage*:* Even if the order of storage of the message bytes is randomized, their Hamming weights can still be recovered using the HW classifier. Subsequently, summing them up provides us the hamming weight of the complete message m. Thus, our attack methodology to break Shuffled-Determiner-Leakage (Sec.VIII-A1) can be directly used to also break Shuffled-Bytewise-Storage. Using the single-trace HW classifier, full message (n bits) can be recovered in $(n+1)$ side-channel traces. The same methodology can be used to target any operation which stores the decrypted message in memory (bytewise, wordwise or otherwise).

B. Attacking the Masking Countermeasure

We can also exploit ciphertext malleability to attack masked implementations of LWE/LWR-based PKE/KEMs. There have been several masking schemes proposed for LWE/LWR-based PKE/KEMs [4], [34] and all schemes process the decrypted message m in two or more boolean shares (depending on masking order) (i.e) $\mathsf{m} = \mathsf{m1} \oplus \mathsf{m2}$ where $\oplus$ is the *bitwise-xor* operation.

While masking is effective against DPA style attacks that work over multiple traces, they do not protect against single trace attacks since both shares can be attacked individually to recover the masked variable. Thus, the masking countermeasure is not effective against the single trace attacks targeting the Determiner-Leakage vulnerability [8], [7], [9] as well as our proposed single trace attacks targeting Incremental-Storage of the decrypted message (Sec. V). However, all the aforementioned attacks can be thwarted by performing Bytewise-Storage of the message shares (Masked-Bytewise-Storage) which removes leakage due to individual message bits in each share. However, we show that ciphertext malleability can yet again be exploited to perform message recovery even when performing Masked-Bytewise-Storage.

1) Attacking Masked-Bytewise-Storage*:* We illustrate recovery of the first message byte $\mathsf{m}[0]$. Since both the shares of byte $\mathsf{m}[0]$ are stored separately, we can use the HW classifier to recover Hamming weight of both shares

Table I: Trace requirement for full message recovery for different implementation variants of the storage of decrypted message in memory. The numbers are reported for message of length 256 bits and assuming a perfect single trace side-channel HW classifier

Vulnerability	No. of Traces
No Protection	
Determiner-Leakage [7], [8], [9]	1
Incremental-Storage [This work]	1
Bytewise-Storage [This work]	9
Wordwise-Storage [This work]	33
Shuffling Countermeasure	
Shuffled-Determiner-Leakage [This work]	257
Shuffled-Incremental-Storage [This work]	385, 500
Shuffled-Bytewise-Storage [This work]	257
Masking Countermeasure	
Masked-Determiner-Leakage [This work]	1
Masked-Incremental-Storage [This work]	1
Masked-Bytewise-Storage [This work]	1100

(i.e.) $\mathsf{HW}(\mathsf{m1}[0])$ and $\mathsf{HW}(\mathsf{m2}[0])$, which we denote as the ordered pair $(v1, v2)$. We observe that the set of possible values of $(v1, v2)$ for a given $\mathsf{m}[0] = k$ uniquely identifies $\mathsf{HW}(k)$ (property of *bitwise-xor* operation). Thus, ordered pairs $(v1, v2)$ recovered from several decapsulations of ct can be used to uniquely determine $\mathsf{HW}(\mathsf{m}[0])$. While a 1^{st} order attack only observes one of the random shares $(\mathsf{m1}[0]/\mathsf{m2}[0])$, observing both shares can be used to extract information about the original value, in this case $\mathsf{HW}(\mathsf{m}[0])$.

We repeat the same for the adapted ciphertext $\mathsf{ct}' = \mathsf{Flip}(\mathsf{ct}, 0)$ (flip m_0) to determine the modified Hamming weight $\mathsf{HW}(\mathsf{m}'[0])$. Perturbation in the Hamming weight can be used to recover the flipped bit m_0 shown as in Eqn.5. We performed attack simulations assuming a single trace HW classifier on both NewHope and Kyber and we approximately require 4.3 queries to recover a single bit m_0 and thus full message recovery takes approximately 1100 traces. Our attack can be trivially extended to 1) target any operation that stores the decrypted message with any storage width and 2) higher masking order albeit with appropriate change in trace complexity.

Thus, we have presented novel attack methodologies exploiting ciphertext malleability properties inherent in LWE/LWR-based schemes to break implementations protected with concrete shuffling and masking countermeasures. We summarize the trace requirement of our attacks over different variants of storage of the decrypted message in memory in Tab. I.

IX. Countermeasures

Our proposed attacks for message recovery clearly motivate the need for strong side-channel countermeasures to protect the sensitive decrypted message in LWE/LWR-based schemes. Based on the range of attacks presented earlier, we discuss few mitigation techniques here.

- *Random Jitter:* Introducing jitter adds horizontal noise and disturbs alignment of PoI across measurements. Thus, it increases the attack effort. However, a stronger adversary can adopt re-alignment techniques [35] to boost the SNR.
- *Combined Masking and Shuffling:* While individual shuffling and masking countermeasures were shown to be vulnerable, a combination of masking and shuffling would increase the trace requirement for the attack. However, a concrete analysis requires further investigation and out of scope of this work.
- *Key Refreshment Rate:* An ephemeral key setting will limit the attacker to only one trace. Combining this with jitter, shuffling and masking can make single trace attacks infeasible. Even in case the key is used for multiple runs, the refresh rate must be upper bounded by no. of runs in Tab.I.

X. Conclusion

This work demonstrates generic side-channel assisted message recovery attacks over LWE/LWR-based PKE/KEMs targeting storage of the decrypted message in memory, a fundamental and unavoidable operation in any embedded implementation. Notably, we exploit the inherent ciphertext malleability property of LWE/LWR-based schemes to propose novel attack techniques, adaptable to different implementation variants, including implementations protected with concrete side-channel countermeasures such as masking and shuffling. We propose the use of combined masking and shuffling as a concrete protection against message recovery attacks. Our proposed attacks therefore reiterate the need for concrete evaluation of side-channel countermeasures against message recovery in LWE/LWR-based PKE/KEMs.

References

[1] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 1–14.

[2] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the second round of the nist pqc standardization process," *NIST, Tech. Rep., July*, 2020.

[3] P. Ravi, J. Howe, A. Chattopadhyay, and S. Bhasin, "Lattice-based key-sharing schemes: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–39, 2021.

[4] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical CCA2-secure and masked ring-LWE implementation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 142–174, 2018.

[5] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 307–335, 2020.

[6] P. Ravi, R. Poussier, S. Bhasin, and A. Chattopadhyay, "On configurable sca countermeasures against single trace attacks for the ntt," *IACR Cryptology ePrint Archive*, vol. 2020, p. 1038.

[7] B.-Y. Sim, J. Kwon, J. Lee, I.-J. Kim, T.-H. Lee, J. Han, H. Yoon, J. Cho, and D.-G. Han, "Single-trace attacks on message encoding in lattice-based kems," *IEEE Access*, vol. 8, pp. 183 175–183 191, 2020.

[8] D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden, "Defeating newhope with a single trace," in *International Conference on Post-Quantum Cryptography*. Springer, 2020, pp. 189–205.

[9] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem," *IACR ePrint Archive*, 2021.

[10] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQM4: Post-quantum crypto library for the ARM Cortex-M4," https://github.com/mupq/pqm4.

[11] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.
[12] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 719–737.
[13] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.
[14] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.
[15] NIST, "Round 3 Submissions - NIST Post-Quantum Cryptography Project," https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions, 2020.
[16] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual International Cryptology Conference*. Springer, 1999, pp. 537–554.
[17] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.
[18] S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "Nicv: normalized inter-class variance for detection of side-channel leakage," in *2014 International Symposium on Electromagnetic Compatibility, Tokyo*. IEEE, 2014, pp. 310–313.
[19] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 81–88.
[20] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2019, pp. 130–149.
[21] W.-L. Huang, J.-P. Chen, and B.-Y. Yang, "Power analysis on ntru prime," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 123–151, 2020.
[22] J.-P. D'Anvers, M. Tiepelt, F. Vercauteren, and I. Verbauwhede, "Timing attacks on error correcting codes in post-quantum secure schemes." *IACR Cryptology ePrint Archive*, vol. 2019, p. 292, 2019.
[23] Z. Xu, O. Pemberton, S. S. Roy, and D. Oswald, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber," Cryptology ePrint Archive, Tech. Rep., 2020.
[24] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "On exploiting message leakage in (few) nist pqc candidates for practical message recovery and key recovery attacks," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1559, 2020.
[25] E. Nascimento and Ł. Chmielewski, "Applying horizontal clustering side-channel attacks on embedded ecc implementations," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2017, pp. 213–231.
[26] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.
[27] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber (version 2.0) - Algorithm Specifications And Supporting Documentation (April 1, 2019)," *Submission to the NIST post-quantum project*, 2020.
[28] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in *International Conference on Selected Areas in Cryptography*. Springer, 2016, pp. 14–37.
[29] C. O'Flynn and A. Dewar, "On-device power analysis across hardware security domains." *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 126–153, 2019.
[30] U. Maurer and B. ö. Tackmann, "On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 505–515.
[31] M. Rivain, E. Prouff, and J. Doget, "Higher-order masking and shuffling for software implementations of block ciphers," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2009, pp. 171–188.
[32] P. Pessl, "Analyzing the shuffling side-channel countermeasure for lattice-based signatures," in *International Conference on Cryptology in India*. Springer, 2016, pp. 153–170.
[33] L. Wu and S. Picek, "Remove some noise: On pre-processing of side-channel measurements with autoencoders," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 389–415, 2020.
[34] M. V. Beirendonck, J.-P. D'anvers, A. Karmakar, J. Balasch, and I. Verbauwhede, "A side-channel-resistant implementation of saber," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 2, pp. 1–26, 2021.
[35] S. Guilley, K. Khalfallah, V. Lomne, and J.-L. Danger, "Formal framework for the evaluation of waveform resynchronization algorithms," in *IFIP International Workshop on Information Security Theory and Practices*. Springer, 2011, pp. 100–115.

Prasanna Ravi is a Research Assistant at Temasek Labs, Nanyang Technological University, Singapore. He is currently pursuing PhD in the topic of SCA and FIA of lattice-based cryptography under the supervision of Dr. Anupam Chattopadhyay and Dr. Shivam Bhasin. Prasanna received joint best paper award at SPACE 2020 and received award for best PhD forum presentation at AsianHOST 2020.

Shivam Bhasin is a Senior Research Scientist and Programme Manager (Cryptographic Engineering) at Centre for Hardware Assurance, Temasek Laboratories, Nanyang Technological University Singapore. He received his PhD from Telecom Paristech in 2011, Master's from Mines Saint-Etienne, France in 2008. Before NTU, Shivam held position of Research Engineer in Institut Mines-Telecom, France. He was also a visiting researcher at UCL, Belgium (2011) and Kobe University (2013). His research interests include embedded security, trusted computing and secure designs. He has co-authored several publications at recognized journals and conferences. Some of his research now also forms a part of ISO/IEC 17825 standard.

Sujoy Sinha Roy is an assistant professor in Cryptographic Engineering at IAIK, the Graz University of Technology. He is interested in the implementation aspects of cryptography. His doctoral thesis on the implementation aspects of lattice-based cryptography was awarded the 'IBM Innovation Award 2018' that recognizes an outstanding doctoral thesis in informatics. He is a co-designer of 'Saber' which is a finalist key encapsulation mechanism (KEM) candidate in NIST's Post-Quantum Cryptography Standardization Project.

Anupam Chattopadhyay received his PhD from RWTH Aachen, Germany. Since 2014, Anupam was appointed as an Assistant Professor in SCSE, NTU, where he got promoted to Associate Professor with Tenure from 2019. His research interest is in computer architectures, security, design automation and emerging technologies. He is a senior member of ACM and IEEE. Anupam received Borcher's plaque from RWTH Aachen, Germany for outstanding doctoral dissertation in 2008, nomination for the best IP award in the ACM/IEEE DATE Conference 2016 and nomination for the best paper award in the International Conference on VLSI Design 2018, 2020.