Deep learning enhanced side channel analysis on CRYSTALS-Kyber

Hoang, A.-T., Kennaway, M., Pham, T. D., Mai, T. S., Khalid, A., Rafferty, C., & O'Neill, M. (2024). Deep learning enhanced side channel analysis on CRYSTALS-Kyber. In *The 25th International Symposium on Quality Electronic Design (ISQED'24): Proceedings* (International Symposium on Quality Electronic Design (ISQED): Proceedings). Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.1109/ISQED60706.2024.10528674

Published in:
The 25th International Symposium on Quality Electronic Design (ISQED'24): Proceedings

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
Copyright 2024 IEEE.
This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Open Access
This research has been made openly available by Queen's academics and its Open Research team. We would love to hear how access to this research benefits you. – Share your feedback with us: http://go.qub.ac.uk/oa-feedback

Download date:09. Nov. 2026

Deep Learning Enhanced Side Chanel Analysis on CRYSTALS-Kyber

Anh Tuan Hoang, Mark Kennaway, Dung Tuan Pham, Thai Son Mai, Ayesha Khalid, Ciara Rafferty, Maire O'Neill
The Centre for Secure Information Technologies (CSIT)
Queen's University Belfast (QUB), UK
Email: {at.hoang, mkennaway01, tpham01, thaison.mai, a.khalid, c.m.rafferty}@qub.ac.uk; m.oneill@ecit.qub.ac.uk

***Abstract*—The combination of Deep-learning (DL) and Side-channel analysis (SCA) has been proven by several attacks targeting symmetric key cryptography implementations such as AES. This paper aims to demonstrate the effectiveness of DL in attacking a Post Quantum CRYSTALS-Kyber implementation to recover the private key. We propose a CNN model with additional ciphertext knowledge to attack each 12-bit coefficient of the polynomial vector representing the private key. The model assigns a label to each trace by combining the values of each coefficient from the private key and so the attacker does not require any knowledge about the implementation and little or no knowledge about the Kyber algorithm. The model needs only 50 traces to reveal the coefficients of the polynomial vector which represents the entire private key.**

***Index Terms*—Cryptanalysis, CRYSTALS-Kyber, Side Channel Analysis (SCA), Deep learning (DL), DL-SCA, Post Quantum Cryptography, IoT Security**

I. Introduction

In December 2016 the emerging prospect of a quantum enabled attack on contemporary cryptographic mechanisms prompted the National Institute of Standards and Technology (NIST) to begin the process to solicit, evaluate and standardise one or more Post Quantum Cryptography (PQC) schemes [1].

In July 2022, four candidates for standardisation were announced, and CRYSTALS-Kyber [2] was selected as the only Modular Lattice-based cryptography Key Exchange Mechanism (ML-KEM). CRYSTALS-Kyber is widely accepted to be mathematically secure from a quantum enabled attack to the level of hardness of a problem known as the Shortest Vector Problem. However, the *polynomial vector multiplications* and *Number Theoretic Transform (NTT)* have been shown to leak information such as power consumption and electromagnetic field emissions to the side-channel [3], [4], [5], [6], [7] [8].

Side-channel Analysis (SCA) [9] applies a divide and conquer approach to analyse the sensitive data such as the secret key by isolating independent parts. It breaks a large number of bits of sensitive data (e.g. 128-bit secret key or 512-coefficient private key polynomial vector) into smaller sub-bytes or single coefficients and analyses the correlation between the side-channel information and that sensitive data in execution. The same approach can be applied to each *polynomial vector multiplication* and/or *NTT* to find the correlation between each coefficient and side-channel information, making SCA a viable threat to any straightforward implementation of a ML-KEM.

The emergence of Deep Learning (DL) in recent years has become a disruptive technology, especially with applications to cyber security and its use as an enhancement to SCA has been a catalyst for research in the field of hardware security. In fact a DL-SCA attacker does not need knowledge of the implementation, since a DL model will be trained to independently learn patterns and predict key values. In this paper, we demonstrate that DL enhances SCA state-of-the-art in attacking a CRYSTALS-Kyber implementation to find the private key, as it is used during decryption. The DL-SCA successfully reveals coefficients of the private key with little or no detailed knowledge about the implementation of the decryption process.

II. Related Works

Aydin *et al* [10] proposed a CNN based attack, targeting two of the then NIST PQC candidates: FrodoKEM [11] and NewHope [12]. Bo-Yeon *et al* in [13] subsequently pioneered a DL-SCA by leveraging clustering recognition [14] [15] and pattern analysis [16], attacking three of the NIST PQC candidates: SABER [17], Kyber and FrodoKEM [11]. A multi-layer perceptron model (MLP) was used to recover the secret message from unprotected implementations, with attack points generated using the sum of squared pairwise t-differences (SOST) values of power traces.

Ngo *et al* [18] used incremental storage leakage shown by [4] and [5], to carry out a DL-SCA on a SABER implementation [19] which claimed SCA resistance through first-order masking and shuffling. Initially they performed single-trace message recovery through a MLP then by using mapping from error-correcting codes they further recovered the long-term secret key. This idea was later expanded into an Ensembled-MLP approach [20] before Bo-Yeon *et al* [21] built upon the fact that [20] omitted targeting vulnerabilities after the inverse NTT when they exploited leakages of Barrett Reduction in a successful DL-SCA, leveraging the side channel leakage study of Xu *et al* [22]. Separately, Kashyap *et al* [23] targeted NewHope and FrodoKEM with a CNN approach which used 1D time-series power measurement data which had been converted into into 2D images.

This research was supported in part by EPSRC Quantum Communications Hub grant EP/T001011/1.

Ngo *et al* [24] built upon [18] and [20] by attacking a higher-order masked SABER implementation [25], itself designed as a mitigation to the higher-order leakages exploited against [19]. Backlund *et al* subsequently brought these techniques to bear on a masked and shuffled software implementation of Kyber [26] before a new recursive Neural Network based DL method was introduced by Dubrova *et al* [27]. This was used to attack the re-encryption which occurs during the FO Transform of a masked Kyber implementation on the ARM Cortex M4. Table I summarises SCAs on Lattice-based PQC.

Ref	PQC	Target	Method
[10]	FrodoKEM	matrix/poly multiplication	CNN
[10]	NewHope	matrix/poly multiplication	CNN
[13]	Kyber	Message Encoding	MLP
[13]	SABER	Message Encoding	MLP
[13]	FrodoKEM	Message Encoding	MLP
[18]	SABER	POLY2MSG	MLP
[20]	SABER	Poly_A2A	Ensembled MLP
[21]	SABER	Barrett Reductions	MLP
[23]	FrodoKEM	matrix/poly multiplication	CNN
[23]	NewHope	matrix/poly multiplication	CNN
[24]	SABER	A2B_bitsliced_msg()	MLP
[26]	Kyber	FY Indexes	MLP
[26]	SABER	FY Indexes	MLP
[27]	Kyber	mask_poly_frommsg()	Recursive NN
[8]	Kyber	NTT/I-NTT	CPA

TABLE I
A SUMMARY OF SCAS ON LATTICE-BASED PQC.

Our contribution includes:

- A presentation of the first known CNN assisted SCA which targets CRYSTALS-Kyber.
- A presentation of a more practical and realistic SCA on a CRYSTALS-Kyber, no chosen ciphertexts, ciphertext manipulations or any other elaborate pre-conditions are required for a successful key recovery.
- A presentation of the first known SCA model which explicitly uses the secret key as the label. This labelling method allows less knowledge about both the CRYSTALS-Kyber algorithm and its implementation.
- A CNN model which allows a variable ciphertext input and allows recovery of each coefficient of the private key directly.

This paper is outlined as follows: Section III provides background to CRYSTALS-Kyber algorithms, a PQM4 implementation and SCA models targeting vulnerable areas. Section IV introduces CNNs for SCAs in general, then specifically when applied with ciphertext knowledge and our experimental environment. Section V contains results of our attack while Section VI discusses our results and makes conclusions, Section VII suggests future work.

III. PQC CRYSTALS-KYBER

Following a brief background introduction, three CRYSTALS-Kyber algorithms for Key Generation, Encryption and Decryption are described in this section. Particular attention is drawn to the decryption operation since it is performing calculations using the secret key. The implementation of decryption poses a risk of exposing secret key related data and as such it contains our attack point. If an adversary can somehow recover the secret key, this will allow them to perform encryption and decryption, hence gain access to the full cryptosystem.

The security of Kyber rests on the hardness of solving the module learning-with-errors (MLWE) problem which in turns rests on the Shortest Vector problem (SVP). Module lattices are a compromise between standard lattices and ideal lattices. Standard lattices are complex and resource heavy, while ideal lattices are more efficient but can lack security due to their structure [28]. MLWE and therefore Kyber works via vectors of polynomials which, although non intuitive, allow for a much better speed/ size/ security trade-off. These polynomial operations can be computed using the Number Theoretic Transform (NTT), allowing increased performance.

As shown in Fig. 1 below, k is the parameter which dictates one of three parameters sets known as Kyber512, Kyber768 and Kyber1024 respectively. This parameter k is introduced to restrict the dimensions of the public-key matrix A, but all elements of this matrix are on the ring $Z_q[x]/(x^n+1)$. The parameter n is always 256 and since the round 2 submission, the parameter q has been 3,329. Parameters $\eta 1$, $\eta 2$, control coefficient size while d_u and d_v control compression of the ciphertext values u and v respectively, with δ giving a probability of KEM failure.

In the case of Kyber512, since k=2, the public-key matrix A contains four elements and results in the creation of two polynomial vectors (t^0 and t^1), each containing 256 coefficients, denoted here as $a_0, a_1 ... a_{255}$.

$$t^0_{256} = a^0_0 x^0 + a^0_1 x^1 + a^0_2 x^2 + ... + a^0_{254} x^{254} + a^0_{255} x^{255}$$

$$t^1_{256} = a^1_0 x^0 + a^1_1 x^1 + a^1_2 x^2 + ... + a^1_{254} x^{254} + a^1_{255} x^{255}$$

	n	k	q	η_1	η_2	(d_u, d_v)	δ
KYBER512	256	2	3329	3	2	(10,4)	2^{-139}
KYBER768	256	3	3329	2	2	(10,4)	2^{-164}
KYBER1024	256	4	3329	2	2	(11,5)	2^{-174}

Fig. 1. The three Kyber parameter sets as defined in [2].

A. Kyber Public Key Encryption Algorithms

Key Generation - CPAPKE.KeyGen()

The CPAPKE.KeyGen key generation algorithm returns a pair of keys (*pk,sk*) where *pk* denotes public key and *sk* the secret key.

- The matrix $\hat{A} \in R_q^{k \times k}$ is generated directly in the **NTT** domain by uniform sampling.
- Two vectors $s \in R_q^k$ and $e \in R_q^k$ are generated through binomial distribution sampling.
- The polynomial $\hat{t} := \hat{A} \circ \hat{s} + \hat{e}$ is computed where $\hat{s} = NTT(s)$ and $\hat{e} = NTT(e)$.
- The key pair is generated with *sk* := (*NTT*(*s*)) and *pk* := (*pk*,ρ), where ρ is a random number.

Algorithm 4 KYBER.CPAPKE.KeyGen(): key generation

Output: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathrm{G}(d)$
$N := 0$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, j, i))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{s} \in R_q^k$ from B_{η_1}
  $\mathbf{s}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e} \in R_q^k$ from B_{η_1}
  $\mathbf{e}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
$\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$pk := (\mathsf{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q) \| \rho)$ ▷ $pk := \mathbf{As} + \mathbf{e}$
$sk := \mathsf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$ ▷ $sk := \mathbf{s}$
return (pk, sk)

Fig. 2. Algorithm for Key Generation [2]. It returns public key *pk* and private key *sk*.

Encryption - CPAPKE.Enc(*pk,m,r*)

The CPAPKE.Enc encryption algorithm takes a public key *pk* and a message $m \in M$ and produces a ciphertext *c*, no operations during encryption involve the secret key and as such it has no relevance to the target of our attack, however we did use it to encrypt random 32-byte messages.

Decryption - CPAPKE.Dec(*sk,c*)

The deterministic decryption algorithm CPAPKE.Dec takes a secret key *sk* and a ciphertext *c* as inputs, then outputs either a message $m \in M$,or an indication of rejection. This operation involves vector multiplication between the secret key *sk* and ciphertext *c*, each correspond to a polynomial of degree 255 with integer coefficients ranging from 0 - 3328.

- The ciphertext c is input along with secret key sk
- The first part of ciphertext c is decoded and decompressed into vector *u*
- The second part of ciphertext c is decoded and decompressed into vector *v*
- *sk* is de-serialised by $\hat{s} := Decode_{12}(sk)$
- The message *m* is recovered by $m := Compress_q(v - s^T u, 1)$

The Kyber decryption operation is called at the start of decapsulation with the input ciphertext always being multiplied with the secret key, entirely independently of the ciphertext validity. This is significant for any potential SCA since it enables the attacker to establish a decryption oracle and conduct adaptive Chosen Ciphertext Attacks (CCA).

Algorithm 6 KYBER.CPAPKE.Dec(sk, c): decryption

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

$\mathbf{u} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_u}(c), d_u)$
$v := \mathsf{Decompress}_q(\mathsf{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
$\hat{\mathbf{s}} := \mathsf{Decode}_{12}(sk)$
$m := \mathsf{Encode}_1(\mathsf{Compress}_q(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u})), 1))$ ▷ $m := \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1))$
return m

Fig. 3. Decryption algorithm returning message *m* from ciphertext *c* and private key *sk* [2].

B. CRYSTALS-Kyber Implementation on ARM M4

Decryption: The implementation of the decryption algorithm in PQM4 can be broken down into the nine sub operations outlined in Fig. 4. In particular, attention is drawn to the third sub operation, **poly_frombytes_mul**, which corresponds to line 4 of the decryption algorithm referred to in Fig. 3.

No.	Sub Operation	Description
1.	`poly_unpackdecompress (&mp, c, 0);`	The first part of the ciphertext *c* is unpacked and decompressed and the result stored in *mp*.
2.	`poly_ntt(&mp);`	The Number-Theoretic Transform (NTT) is performed on *mp* transforming the polynomial from the time domain to the frequency domain.
3.	`poly_frombytes_mul(&mp, sk);`	*sk* is de-serialised and then *mp* is multiplied with each de-serialised part of *sk*.
4.	`for(int i = 1; i < KYBER_K; i++) { poly_unpackdecompress(&bp, c, i); poly_ntt(&bp); poly_frombytes_mul(&bp, sk + i*KYBER_POLYBYTES); poly_add(&mp, &mp, &bp); }`	The `for` loop then iterates over the remaining part of the ciphertext *c* unpacking and decompressing it into *bp* where the NTT is performed upon it. Now the remaining part of *sk* is de-serialised and multiplied with *bp* with the result added to *mp*.
5.	`poly_invntt(&mp);`	The polynomial *mp* is then transformed back into the time domain using an inverse NTT function.
6.	`poly_decompress(v, c+KYBER_POLYVECCOMPRESSEDBYTES);`	The second part of the ciphertext *c* is de-serialised and decompressed into *v*.
7.	`poly_sub(&mp, v, &mp);`	the polynomial *mp* is subtracted from the polynomial *v* with the result stored *mp*.
8.	`poly_reduce(&mp);`	The coefficients in *mp* are reduced to a smaller range.
9.	`poly_tomsg(m, &mp);`	The polynomial *mp* is converted into the now decrypted message *m*.

Fig. 4. The sub operations of Decryption in our PQM4 Implementation.

Within the **poly_frombytes_mul** sub operation, the lowest level of functions involving parts of the secret key and ciphertext can be found and exploited. The Cortex M4 has 13 general purpose 32-bit registers, 12 of which are prepared for use within this sub operation, along with the link register. As part of decryption, the M4 prepares to perform the multiplication of 16-bit x 16-bit halfwords, storing the intermittent values in these 32-bit registers. The secret key and ciphertext coefficients are in fact each 12-bit values, and so **doublebasemul_asm** which simply comprises two **basemul** ($f_{basemul}$) functions, performs pair-pointwise multiplication of four 12-bit secret key inputs with four 12-bit ciphertext inputs, storing the results as r_0 and r_1 respectively. These functions are of interest as bits of the secret key are now placed upon the power lines of the microcontroller as it performs multiplicatively in conjunction with known aspects of the ciphertext.

Our attack targets these functions by extracting the intermittent ciphertext and secret key related values which are input to the *doublebasemul* function in Fig. 5 for inclusion as input data to our CNN.

1) The first values extracted are the four private key $\hat{s}$ coefficients being input to *doublebasemul* in Fig. 5.
2) The next values extracted are four ciphertext *NTT(u)* coefficients input to *doublebasemul* of Fig. 5.

The *doublebasemul* function receives four coefficients of the secret key polynomial vector $\hat{s}$ and four coefficients of ciphertext polynomial vector *NTT(u)*. It then produces the *basemul* results r_0 and r_1 in Fig. 5, which are the coefficients of polynomial vector m in Fig. 3.

We cross validated each of these values input to the **poly_frombytes_mul** sub operation on the Cortex M4, with

```
//basemul(r->coeffs + 4 * i, a->coeffs + 4 * i, b->coeffs + 4 * i, zetas[64 + i]);
smultt tmp, poly0, poly1
montgomery q, qinv, tmp, tmp2
smultb tmp2, tmp2, zeta
smlabb tmp2, poly0, poly1, tmp2
montgomery q, qinv, tmp2, tmp
// r[0] in upper half of tmp

smuadx tmp2, poly0, poly1
montgomery q, qinv, tmp2, tmp3
// r[1] in upper half of tmp3
pkhtb tmp, tmp3, tmp, asr#16
str tmp, [rptr], #4

neg zeta, zeta

//basemul(r->coeffs + 4 * i + 2, a->coeffs + 4 * i + 2, b->coeffs + 4 * i + 2, - zetas[64 + i]);
smultt tmp, poly2, poly3
montgomery q, qinv, tmp, tmp2
smultb tmp2, tmp2, zeta
smlabb tmp2, poly2, poly3, tmp2
montgomery q, qinv, tmp2, tmp
// r[0] in upper half of tmp

smuadx tmp2, poly2, poly3
montgomery q, qinv, tmp2, tmp3
// r[1] in upper half of tmp3
pkhtb tmp, tmp3, tmp, asr#16
str tmp, [rptr], #4
```

Fig. 5. Implementation of *doublebasemul* function in PQM4.

the corresponding inputs to this sub operation as part of the reference implementation running on a laptop computer, finding they match exactly. Likewise we also cross validated every result output from this **poly_frombytes_mul** sub operation across both platforms to find they also match exactly. Therefore we can say with confidence the reference implementation running on a laptop computer can be used to generate reliable input data for a DL-SCA and our attacker need not learn any detailed information about the implementation on M4.

C. SCA Models Targeting CRYSTALS-Kyber

The secrecy of the CRYSTALS-Kyber implementation is fully compromised if the private (secret) key *sk* in Fig. 2 is revealed in its entirety. We hypothesise that this can be achieved by attacking each 12-bit coefficient of the polyvector $\hat{s}$ in Fig. 3 during the decryption process. This section of the paper demonstrates the attack on the first coefficients of the first vector in $\hat{s}$.

The decryption function in Fig. 3 takes the secret key sk and the ciphertext c to retrieve the message m. During the execution of the decryption function, the ciphertext c is split into two even parts: u and v. The multiplication process between the ciphertext coefficient $\hat{u} = NTT(u)$ and secret key coefficient $\hat{s}^T$ consists of multiple $f_{basemul}$ functions. For example in the case of Kyber512, there are 256 $f_{basemul}$ functions which compute the pairwise multiplication between each 2-coefficient pair of $\hat{s}^T$ and $\hat{u}$. For the first coefficient pairs, we can denote the input values to the $f_{basemul}$ calculation to be $s = \{s_0, s_1\}$ and $u = \{u_0, u_1\}$. Next the point-wise multiplication on Zq is performed, this is denoted by f_{qmul} and for each $basemul$ occurs four times between s and u and one further time between $r0$ and $zeta$. Hence, each $basemul$ consists five f_{qmul} of $f_{qmul}(s_1, u_0)$, $f_{qmul}(r_0, zeta)$, $f_{qmul}(s_1, u_1)$, $f_{qmul}(s_0, u_0)$ and $f_{qmul}(s_0, u_1)$ to produce two outputs of r_0 and r_1.

Coefficient s_0 is exclusively involved in the $f_{qmul}(s_0, u_0)$ and $f_{qmul}(s_0, u_1)$ operations, while similarly coefficient s_1 is only related to the $f_{qmul}(s_1, u_0)$ and $f_{qmul}(s_1, u_1)$ operations. The computation of $f_{doublebasemul}$ is the same for every coefficient pair, simply comprising of two $f_{basemul}$ calculations. Therefore if we are able to retrieve s_0 from f_{qmul} we would be able to further attack the remaining f_{qmul} and $f_{doublebasemul}$ to retrieve all secret key coefficients.

Many existing works use the Hamming Weight (HW) and/or the Hamming Distance (HD) as a modelling system in their attack, such as Correlation Power Analysis [29]–[31], Differential Power Analysis [32] and Mutual Information Analysis [33]. HW returns the number of bits that equal 1 within a register, while HD is a measure of the number of bits changed in that register. A corresponding labelling function which returns the label for the machine learning classification task with input trace t can be denoted as $f_{label}(t)$. This can use Hamming Weight $f_{label}(t) = HW(f_{doublebasemul}(s, u))$ or Hamming Distance $f_{label}(t) = HD(f_{doublebasemul}(s, u))$ to generate sets of labels, however label sets derived in this way introduce bias [34] and can be considered unbalanced. Although not ideal for a machine learning classification task, this system of label set production can still be used within an ensembled approach [35]. Moreover, implementing a Hamming Distance oriented model requires much effort to locate and decode the register value for each calculation of Hamming Distance. To this end, our system involves labelling the trace directly by the input coefficient of the multiplication process as follows: $f_{label}(t) = s_0$. This is significantly different from other DL-SCA approaches, which use the output value of the function as the label (in this case f_{qmul}). In contrast, we use the input of the $f_{doublebasemul}$ function as the label, this is the coefficient of secret key $\hat{s}$.

Through this approach, the labels derived for each dataset are uniformly distributed across its range [0...3328], identical to the input distribution of s_0. We ensure a more balanced dataset which is much more ideal for a machine learning task. This labelling method further demonstrates our claim that attackers need not understand the low-level functionality of $f_{doublebasemul}$ in order to attack CRYSTALS-Kyber.

IV. Machine Learning based SCA

SCAs can be broadly classified into two categories: profiling attacks and non-profiling attacks. In a non-profiling attack, the attacker only has access to the physical leakage (e.g power consumption or electromagnetic emanations) measured on the target device. When performing a profiling attack, the attacker has access to a physical copy of the target device. Classical profiling attacks include Template Attacks [36], the Stochastic Model [37], and recently, Machine Learning-based (ML) attacks. With classical attacks, pre-processing procedures such as noise reduction, alignment, and dimension reduction must be performed before applying the attack. Additionally, certain attacks require Points of Interest (PoI) to be selected as part of this pre-processing procedure. Both pre-processing and PoI selection involve human engineering, which can significantly affect the results [38]. Since the process of feature selection is automatically performed as part of a ML attack, these might be considered advantageous over classical attacks. When power traces exhibit a high level of noise, ML models such

as the Support Vector Machine (SVM) have shown increased performance when compared to Template Attacks [39].

More recently, Deep Learning (DL) approaches such as Multi-layer Perceptron (MLP) and Convolutional Neural Network (CNN) have demonstrated their abilities in SCA. [40] showed that MLP performance is comparable with Template attacks. CNN approaches have shown increased robustness by mitigating effects of signal deformation such as jittering [41] and de-synchronisation [42].

A. Convolutional Neural Network (CNN)

CNNs are a DL architecture that incorporates several different types of layers for detecting features for classification. A typical CNN consists of Convolutional, Pooling, and Batch Normalisation layers. In the Convolutional layer, the filter (kernel) is a learnable parameter that slides across the input's height and width. With each sliding step (stride), the dot product between the input and the filter is calculated to extract the input's features and output a features map. This feature map is then fed into a Pooling layer to summarise the features present in each region of the feature map. The model then learns these summarised features from the Pooling layer instead of precisely positioned features generated by the Convolutional layer, which makes it more reliable and robust to variant positions. The Batch Normalisation layer normalises the distribution of the input by controlling their mean and variance. [43] hypothesise that Batch Normalisation layer reduces the internal co-variate shift in neural networks and enables the usage of higher learning rates to speed up training. The use of Batch Normalisation layer is optional but enabling Batch Normalisation layer can help the model generalise better and even enhance performance [44].

In the context of SCA, the traces are affected by distortions such as measurement noise and counter-measures. Therefore, using CNN is an appropriate approach since CNN is robust to distortion and several works have been conducted to prove the effectiveness of CNN. [45] showed that CNN can be used to recover the secret key in both unprotected and protected AES implementation. [41] proposed a CNN-based SCA on a protected AES with jitter-based hiding methods.

B. Convolutional Neural Network with Ciphertext Knowledge (CNNC)

We formalise the Post-quantum SCA problem as a Supervised Machine Learning problem. Given a set of data $S = \{(x_i, y_i)\}_{i=1}^{n}$ which contains n pairs of input and their corresponding label. Different from other CNN model with domain knowledge [34] [35], who can combine only one plaintext/ciphertext to the CNN model, we can choose to input the power traces $x_i = t_i$ or both the power traces and their multiple corresponding nc ciphertext coefficients: $x_i = \{t_i, u_i^j\}_{j=1}^{nc}$. A machine learning model that with parameter θ given by: $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{d_o}$ takes input dimension d and output dimension d_0. The output vector of this model is $\hat{y}$, which contains the probability for d_0 class label. After training f_θ on the data set S, the attack phase is conducted on the target device. The attack phase utilises the trained model f_θ to predict the output probability of m measured traces of the target device. A set of m traces is measured on the target device to generate an input set $\{x_1, x_2, ..., x_m\}$ for the prediction of model f_θ. In the prediction phase, f_θ outputs a set of probability vectors from the input traces and corresponding nc ciphertext coefficient: $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_m\}$. The final scores vector is then calculated by multiplying the maximum likelihood scores of m independent predicted outputs: $\hat{y}_1 \odot \hat{y}_2 \odot ... \odot \hat{y}_m$. The value of this scores vector is sorted for ranking from high to low likelihood. An ideal model would return the rank of the correct key as 1 and the closer the rank of the correct key, the more accurate the model is.

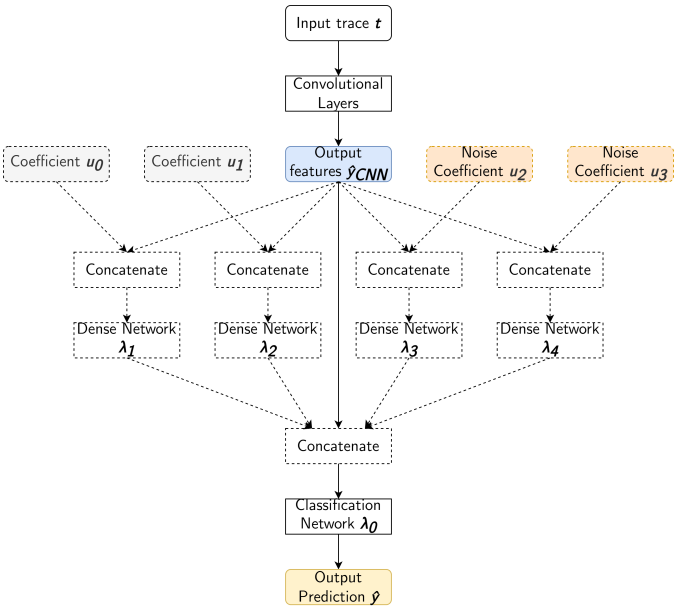

Fig. 6. Overview of our *Convolutional Neural Network with Ciphertext Knowledge (CNNC)* method. The component with a dotted line is an optional component.

We showed the overview of our *Convolutional Neural Network with Ciphertext Knowledge (CNNC)* method in Fig. 6. The architecture of model f_θ first contains Convolutional layers to produce a feature map y_{CNN}. During the multiplication process of $f_{doublebasemul}(s, u)$ between $s = \{s0, s1, s2, s3\}$ and $u = \{u_0, u_1, u_2, u_3\}$, only the first two ciphertext coefficients $\{u_0, u_1\}$ are involved in the computation of the target polyvector coefficient s_0 while the other two ciphertext coefficients are considered noise. Therefore, we denote $\{u_2, u_3\}$ as the noise coefficient in our attack. In the case where no ciphertext knowledge is utilised, a Classification network λ_0, which contains a series of dense layers, would directly process the produced feature map to output probability vectors. For other cases where ciphertext knowledge is utilised, nc ciphertext coefficients $\{u^j\}_{j=1}^{nc}$ are flattened then pairwise concatenation between each coefficient and the feature map y_{CNN} is performed. The output features set of the concatenation process then goes through nc dense networks λ_i to output nc features. These features are concatenated and

then processed by the Classification network λ_0 to output the probability of each class label.

C. Research Environment

Our environment for research involved setup of the ChipWhisperer [46] Lite Capture with CW308 UFO Baseboard hosting a CW308T-STM32F3 Cortex-M4 microcontroller [47]. The decryption oracle is established by implementing the Kyber512 decryption operations of the PQM4 library on the microcontroller. Our ciphertexts are generated by applying the deterministic CPAPKE.Enc to randomly generated 32-byte plaintext messages. We attack the **doublebasemul** function in assembly code by capturing traces pertaining to these operations only with default sampling rate of $4*7.37MHz$. We isolate the first coefficient of $\hat{s}$, and collect traces during the execution of the assembly code in Fig. 5. We collected 500k samples (traces) for training and 100k traces for testing, 600 points each. However the input for training is variable from 100k to 500k samples. We then label each trace according to the value of each coefficient for the private key underlining our previous assertion that the attacker needs no knowledge about the implementation and little knowledge about the Kyber algorithm.

V. Attack Results

A. Experiments Setup

Datasets To generate the datasets for our Supervised problem, we measured the power traces and their corresponding ciphertext coefficients. The label function is identity as described in section II.C. We then generate 3 datasets with different numbers of data points, from 200000(100K) data to 500000(500K) data points: S_{200K}, S_{300K} and S_{500K}.

Model architecture and hyper-parameters

As described in Fig. 6, the model architecture consists of Convolutional Layers, followed by Dense Networks based on the number of ciphertext coefficients and a Classification Network at the end. In our implementation, the number of Convolutional layers is 5 and the Dense Network contains 5 Dense Layer. The Classification Network has 5 Dense layers. The model is trained with Cross Entropy loss and RMSprop was chosen as the optimiser with $epsilon$ = 1e-7 and $momentum$ = 0.0. The learning rate of the training process was set at 1e-6. For stability, we test the trained model with 20 runs and average the results.

B. Attack Post-quantum Cryptography implementation with CNNC

To demonstrate our CNNC approach, we conduct the experiments with different numbers of ciphertext inputs using the model architecture described in Section V-A on S_{200K} dataset, which is the dataset with minimum data points. We summarised the results in Fig. 7. The 2 ciphertext coefficients model reached rank 12 after 125 traces while the model trained with no ciphertext reached rank 7 at the same trace. At the final trace of 200, the model trained with 4 ciphertext coefficients showed the best result with mean rank 5 achieved. However, no model could reach rank less than 1, which implies 200K data points might not be sufficient to conduct a successful attack.

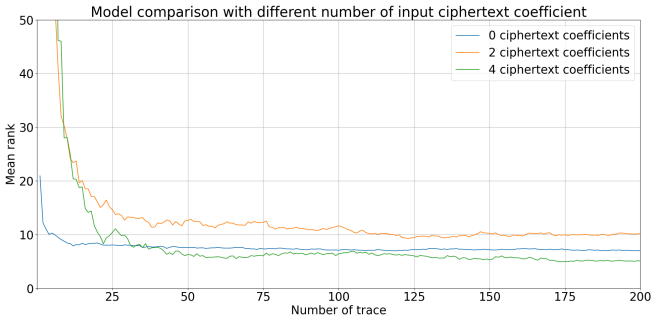

Fig. 7. Attack result with different number of cipher text input on 200K dataset

C. Ablation Study

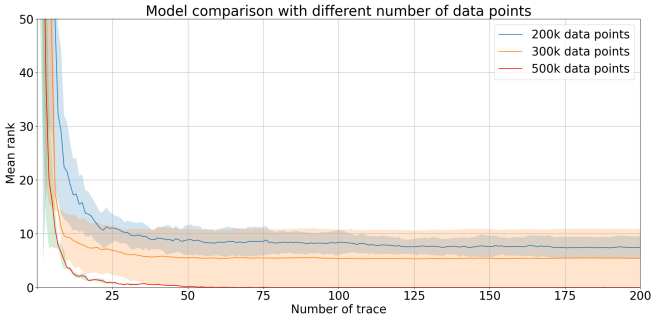

Fig. 8. Attack result with different number of data points. Each line shows the average result for the mean rank and the transparent area represent the standard deviation.

Motivated by the result from the previous section, we conduct experiments on two more datasets containing different numbers of data points: S_{300K} and S_{500K}. With each dataset, we continue to experiment on multiple models with different number of ciphertext coefficients: 0 coefficient, 2 coefficients and 4 coefficients. The mean rank of all models is averaged for each dataset and we summarise the results in Fig. 8. With 200K data points, our attack reaches a mean rank of 7 after 200 traces. In the case of the 300K dataset, the average rank reaches 7 after 25 traces and reaches rank 5 after 50 traces. For 500K data points, the mean rank is swiftly decreased and reaches 4 on trace 10. From 50 traces onward, the 500K dataset data reached a mean rank 0, implies a successful attack. Moreover, the 500K dataset showed better stability than dataset with fewer data points since the standard deviation of this dataset was noticeably smaller than the others.

We further explore the ability of our CNNC framework by analysing the effect caused by the number of ciphertext coefficients. We denote the experiment in the format of the number of coefficients and the number of data points, respectively. The results are summarised in Fig. 9. With no ciphertext

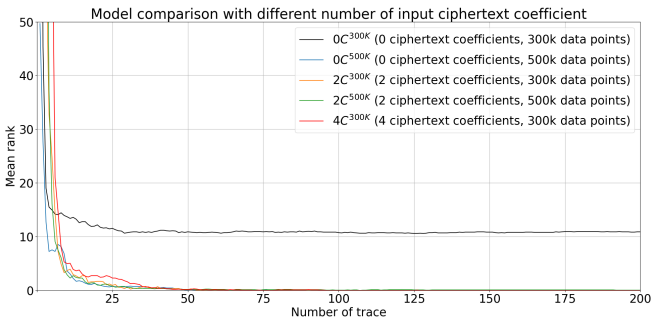

Fig. 9. Results of attacking with different numbers of ciphertext coefficient

coefficient, a model trained with 500K data points reaches rank 0 after 50 traces and conducts a successful attack. Using the same setting with no coefficient utilized, the model was trained with 300K data points can only reach rank 10 after 200 traces. In contrast, the models with 4 and 2 ciphertext coefficients reached rank 0 after 50 traces with the same number of data points at 300K. The performance of this model is comparable to a 2-coefficient model trained with more number of data points at 500K. This concludes that the ciphertext knowledge helps to reduce the required traces in profiling and attacking the target device. Compared to a model with no ciphertext knowledge, it is clear that two ciphertext coefficients (u_0 and u_1) help the model gain better performance. The result is unchanged if knowledge of two additional noise coefficients (u_2 and u_3) are provided to the model.

VI. Discussion

In terms of required profiling data, our experiment shows that the more profiling data we have, the better the training and attacking result achieved. When the amount of profiling data is as small as 200k, none of the CNN structures (CNN and CNNC) can be trained to attack the implementation with 200 traces. However, when the number of traces in profiling increases to 300k, the CNNC model can easily attack and find a coefficient of the private key using 50 traces. The CNN model, which requires less understanding of the Kyber algorithm, can recover a private key coefficient with 50 traces if the profiling data increases to 500k.

Knowledge of the Kyber algorithm helps to improve the attacking result. The ciphertext knowledge helps to reduce the required traces in profiling and attacking dataset. Without knowledge of the ciphertext, the attacker needs 500k data for training to be able to find a coefficient of the private key while this number of profiling data can be reduced to 300k for training if the knowledge of ciphertext is included in the model. Ciphertext knowledge can also help to reduce the number of traces required for the attacking process. Our experiments also show that including ciphertext coefficients, which are unused in the $f_{basemul}$ related to the targeted private key $\hat{s}$ coefficient ($\hat{s_0}$) to the input does not change the attack result. It implies that those noise factors can be detected and removed by the model and that the attackers need to know inputs to $f_{doublebasemul}$ function only but not the full operation of that function to reveal the coefficients of the private key.

Different from the other attack models, who label traces by result of operation (e.g. output of SBox or multiplication), our attack model directly uses the value of the key coefficient as the label. It reduces the required knowledge of the Kyber algorithm and implementation. In more detail, attackers do not need to understand the algorithm of $f_{basemul}$ and the implementation of $f_{doublebasemul}$ to attack Kyber on a M4 microcontroller.

VII. Conclusion and Future Work

Even though CRYSTALS-Kyber is selected for standardisation for lattice-based PQC with ability to change security level, SCA is still a big threat to Kyber implementation. This paper proposed a ML-based SCA called *Convolutional Neural Network with Ciphertext Knowledge (CNNC)* to practically attack a straightforward Kyber implementation on a M4 microcontroller. The model requires very little knowledge of the algorithm or implementation and is able to find a coefficient of the private key within 50 traces.

While considerable work has been done in the area of ML for SCA, there is still significant scope for improvement in current approaches. In future, we will evaluate effectiveness of the model structures in attacking CRYSTALS-Kyber implementations. Furthermore, we can explore the ensemble method using mixed number of ciphertext coefficients model to enhance the robustness and reduce the required number of data points for training.

References

[1] NIST, "Announcing request for nominations for public-key post-quantum cryptographic algorithms," 2016.

[2] R. Avanzi, L. D. Joppe Bos, T. L. Eike Kiltz, J. M. S. Vadim Lyubashevsky, G. S. Peter Schwabe, and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation v3.02." 2021.

[3] P. V. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEM schemes," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 948, 2019.

[4] P. V. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "Drop by drop you break the rock - exploiting generic vulnerabilities in lattice-based PKE/KEMs using EM-based Physical Attacks," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 549, 2020.

[5] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "On exploiting message leakage in (Few) NIST PQC candidates for practical message recovery attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 684–699, 2022.

[6] Y. Chang, Y. Yan, C. Zhu, and P. Guo, "Template attack of LWE/LWR-Based schemes with cyclic message rotation," *Entropy*, vol. 24, p. 1489, 10 2022.

[7] M. Azouaoui, O. Bronchain, C. Hoffmann, Y. Kuzovkova, T. Schneider, and F.-X. Standaert, "Systematic study of decryption and re-encryption leakage: The Case Of Kyber," in *Constructive Side-Channel Analysis and Secure Design: 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 236–256. [Online]. Available: https://doi.org/10.1007/978-3-030-99766-3_11

[8] H. Ma, S. Pan, Y. Gao, J. He, Y. Zhao, and Y. Jin, "Vulnerable pqc against side channel analysis - a case study on kyber," in *2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2022, pp. 1–6.

[9] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," N. I. Koblitz, Ed. Berlin: Springer, Sep. 1996, vol. 1109, pp. 104–13.

[10] F. Aydin, P. Kashyap, S. Potluri, P. Franzon, and A. Aysu, "Deepar-sca: Breaking parallel architectures of lattice cryptography via learning based side-channel attacks," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, A. Orailoglu, M. Jung, and M. Reichenbach, Eds. Cham: Springer International Publishing, 2020, pp. 262–280.

[11] E. Alkim, J. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila, "FrodoKEM: Practical quantum-secure key encapsulation from generic lattices." [Online]. Available: https://frodokem.org/

[12] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange - a new hope," Cryptology ePrint Archive, Report 2015/1092, 2015, http://eprint.iacr.org/2015/1092.

[13] 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.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[15] L. Rokach and O. Maimon, *Clustering Methods*. Boston, MA: Springer US, 2005, pp. 321–352. [Online]. Available: https://doi.org/10.1007/0-387-25465-X_15

[16] Y. Anzai, *Pattern recognition and machine learning*. Morgan Kaufmann, 1992.

[17] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "SABER: MLWR BASED KEM." [Online]. Available: https://www.esat.kuleuven.be/cosic/pqcrypto/saber/

[18] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked IND-CCA Secure Saber KEM," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 79, 2021.

[19] M. V. Beirendonck, J.-P. D'anvers, A. Karmakar, J. Balasch, and I. Verbauwhede, "A side-channel-resistant implementation of saber," *J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 2, apr 2021. [Online]. Available: https://doi.org/10.1145/3429983

[20] K. Ngo, E. Dubrova, and T. Johansson, "Breaking masked and shuffled CCA secure Saber KEM by power analysis," in *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, ser. ASHES '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 51–61. [Online]. Available: https://doi.org/10.1145/3474376.3487277

[21] B.-Y. Sim, A. Park, and D.-G. Han, "Chosen-ciphertext clustering attack on CRYSTALS-KYBER using the side-channel leakage of Barrett reduction," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 382–21 397, 2022.

[22] Z. Xu, O. Pemberton, S. S. Roy, D. Oswald, W. Yao, and Z. Zheng, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2163–2176, 2022.

[23] P. Kashyap, F. Aydin, S. Potluri, P. D. Franzon, and A. Aysu, "2deep: Enhancing side-channel attacks on lattice-based key-exchange via 2-d deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1217–1229, 2021.

[24] K. Ngo, R. Wang, E. Dubrova, and N. Paulsrud, "Side-channel attacks on lattice-based KEMs are not prevented by higher-order masking," *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 919, 2022.

[25] S. Kundu, J.-P. D'Anvers, M. Van Beirendonck, A. Karmakar, and I. Verbauwhede, "Higher-order masked saber," in *Security and Cryptography for Networks*, C. Galdi and S. Jarecki, Eds. Cham: Springer International Publishing, 2022, pp. 93–116.

[26] L. Backlund, K. Ngo, J. Gärtner, and E. Dubrova, "Secret key recovery attacks on masked and shuffled implementations of CRYSTALS-Kyber and Saber," Cryptology ePrint Archive, Paper 2022/1692, 2022, https://eprint.iacr.org/2022/1692. [Online]. Available: https://eprint.iacr.org/2022/1692

[27] E. Dubrova, K. Ngo, J. Gärtner, and R. Wang, "Breaking a fifth-order masked implementation of crystals-kyber by copy-paste," in *Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop*, ser. APKC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 10–20. [Online]. Available: https://doi.org/10.1145/3591866.3593072

[28] A. Khalid, S. McCarthy, M. O'Neill, and W. Liu, "Lattice-based cryptography for enhancingin A Quantum World: Are We Ready?" in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2019, pp. 194–199.

[29] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*. Springer, 2004, pp. 16–29.

[30] S. Yang, Y. Zhou, J. Liu, and D. Chen, "Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations," in *Information Security and Cryptology-ICISC 2011: 14th International Conference, Seoul, Korea, November 30-December 2, 2011. Revised Selected Papers 14*. Springer, 2012, pp. 169–185.

[31] Y. Yang, Z. Wang, J. Ye, J. Fan, S. Chen, H. Li, X. Li, and Y. Cao, "Chosen ciphertext correlation power analysis on Kyber," *Integration*, vol. 91, pp. 10–22, 2023.

[32] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*. Springer, 1999, pp. 388–397.

[33] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis: A generic side-channel distinguisher," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2008, pp. 426–442.

[34] A.-T. Hoang, N. Hanley, and M. O'Neill, "Plaintext: A missing feature for enhancing the power of deep learning in side-channel analysis? breaking multiple layers of side-channel countermeasures," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 49–85, 2020.

[35] A. T. Hoang, N. Hanley, A. Khalid, M. O'Neill *et al.*, "Stacked ensemble model for enhancing the DL based SCA," in *19th International Conference on Security and Cryptography SECRYPT*. SciTePress, 2022, pp. 59–68.

[36] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*. Springer, 2003, pp. 13–28.

[37] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert, "Univariate side channel attacks and leakage modeling," *Journal of Cryptographic Engineering*, vol. 1, pp. 123–144, 2011.

[38] S. Jin, S. Kim, H. Kim, and S. Hong, "Recent advances in deep learning-based side-channel analysis," *ETRI Journal*, vol. 42, no. 2, pp. 292–304, 2020.

[39] A. Heuser and M. Zohner, "Intelligent machine homicide: Breaking cryptographic devices using support vector machines," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 249–264.

[40] Z. Martinasek and L. Malina, "Comparison of profiling power analysis attacks using templates and multi-layer perceptron network," *Math. Methods Sci. Eng*, 2014.

[41] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Springer, 2017, pp. 45–68.

[42] P. Emmanuel, S. Remi, B. Ryad, C. Eleonora, and D. Cecile, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," *CoRR*, pp. 1–45, 2018.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[44] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *Advances in neural information processing systems*, vol. 31, 2018.

[45] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*. Springer, 2016, pp. 3–26.

[46] C. O'Flynn and Z. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," vol. 8622, 04 2014.

[47] NewAE Technology Inc., "ChipWhisperer Level 1 Starter Kit Product Datasheet." [Online]. Available: https://media.newae.com/datasheets/NAE-SCAPACK-L1_datasheet.pdf