How to attack a galaxy: from Star Wars to Star Trek

Luis Hernández-Álvarez*
Institute for Physical and Information Technologies (ITEFI)
Spanish National Research Council (CSIC)
Madrid, Spain
luis.hernandez@csic.es
0000-0003-2637-8901

Miguel Ángel González de la Torre
Institute for Physical and Information Technologies (ITEFI)
Spanish National Research Council (CSIC)
Madrid, Spain
ma.gonzalez@csic.es
0000-0001-8398-1884

Eva Iglesias Hernández
Institute for Physical and Information Technologies (ITEFI)
Spanish National Research Council (CSIC)
Madrid, Spain
eva.iglesias@csic.es
0009-0005-2824-7405

Luis Hernández Encinas
Institute for Physical and Information Technologies (ITEFI)
Spanish National Research Council (CSIC)
Madrid, Spain
luis.h.encinas@csic.es
0000-0001-6980-2683

***Abstract*—Recently, the National Institute of Standards and Technology set CRYSTALS–Kyber as post-quantum public key encryption/key encapsulation mechanism standard, and CRYSTALS–Dilithium as post–quantum digital signature standard. These post quantum cryptosystems are also recommended for national security systems. As a result, it is important to identify and analyze the weaknesses and potential information leakage points, so that they can be resolved. In this work, we study the newest side channel attacks based on artificial intelligence models against Kyber and Dilithium, focusing on the specific function attacked. We also examine the artificial intelligence algorithms employed in these attacks and their configurations, discussing which parameters and setting are suitable, and identifying different tools that might be useful.**

***Index Terms*—Aritificial Intelligence, Kyber, Dilithium, Multi Layer Perceptron, Post Quantum Cryptography**

I. Introduction

Nowadays, public–key cryptosystems are based on difficult or intractable mathematical problems that can not be solve in polynomial time by conventional computers. However, if it was possible to employ large–scale quantum computers, it would be feasible to solve these mathematical problems in polynomial time with Shor's algorithm [1]. Certainly, it is thought that the development of quantum computers is currently limited by engineering challenges, and they will become a reality in the near future. In that case, the integrity and confidentiality of current digital communications would be compromised and no longer secure. Motivated by this risk, Post–Quantum Cryptography (PQC) has emerged as one of the most promising and important scientific areas. The goal of PQC is to elaborate new cryptographic systems that are resistant to both classical and quantum computers, while being suitable to existing communications protocols.

In 2016, the National Institute of Standards and Technology (NIST) initiated a process to establish quantum–resistant (i.e. post-quantum cryptography, PQC) public–key cryptographic algorithms. This call has two cathegories, Public Key Encryption (PKE)/Key Encapsulation Mechanism (KEM) and digital signatures. The objective was to evaluate the initiatives and to select as PQC standard at least two algorithms, based on different mathematical primitives. The NIST PQC call is currently in its fourth round and, in July 2022, CRYSTALS–Kyber [2] was standardized in the PKE/KEM cathegory, while CRYSTALS–Dilithium [3] was standardized for digital signatures [4] (Kyber and Dilithium are fictitious minerals from the Star Wars and Stark Trek universes).

In parallel to the development of PQC algorithms, the area of Artificial Intelligence (AI) has gained a lot of interest in the scientific community. AI can be defined as the science and engineering of making intelligent machines [5], and its algorithms can be categorized into two subareas: Machine Learning (ML) and Deep Learning (DL). Recently, new applications of AI have been explored, including, among others, Natural Language Processing (NLP), medical procedures, biometric user authentication and cybersecurity [6]. In the field of PQC, AI has been studied to create and explode weaknesses in the proposed algorithms, usually enabling Side Channel Attacks (SCAs). In this sense, Neural Networks (NNs) have been shown to be a particular useful AI model against PQC systems.

The objective of this work is to collect and study recent AI–based SCAs on the PQC standards CRYSTALS–Kyber and CRYSTALS–Dilithium. In this way, we provide an analysis of current AI–based SCA against PQC algorithms, studying their limitations and the vulnerability exploited. Having this goal in mind, we include a description of the CRYSTALS–Kyber and CRYSTALS–Dilithium algorithms, emphasizing the specific functions and processes that may by attacked by SCAs.

The rest of the paper is organized as follows: Section II contains a brief explanation of SCAs and NNs. In Section III and Section IV we present the algorithms, targets and attacks of CRYSTALS–Kyber and CRYSTALS–Dilithium, respectively. Section V includes an analysis of the AI tools employed in current SCAs, and the conclusions are presented in Section VI.

979-8-3503-2759-5/23/$31.00 ©2023 IEEE
DOI 10.1109/CSCE60160.2023.00381

II. Background

A. Side Channel Attacks

SCAs were introduced by Kocher [7] and refer to a procedure in which sensitive data related to a cryptosystem can be obtained by measuring physical specificities of cryptographic devices [8]. This means that, instead of exploiting the weaknesses of the algorithm implemented or potential errors in the software, SCAs benefit from information leakages in the physical implementation of a computing system. An example is the power consumption attack [7], in which it is assumed that power consumption is associated with an intermediate value of the cryptographic algorithm operation. Usually, in this type of attack the Hamming Weight (HW), that is, the number of binary symbols different from zero, is employed. Other measurements used in attacks are, for example, acoustic events [9], or electromagnetic radiation [10].

Depending on the objective of the SCA, it can be categorized as Key Recovery or Message Recovery [11]. The goal of the former is to extract the long term secret key directly (targeting operations such as polynomial multiplication) or indirectly (with side-channel data from the decrypted message for chosen–ciphertexts). On the contrary, the latter focuses on the message encoding operation to retrieve the message.

B. Neural Networks

AI algorithms aim to mimic the way humans learn. Both ML and DL are subareas of AI and, actually, DL is a part of ML [12]. However, they are differentiated due to the specific algorithms employed in DL: Neural Networks. NNs (or Artificial NNs) are inspired by the human brain and how the neurons work and signal to one another. As a result, NNs are composed of nodes (neurons) organized in different layers: an input layer, where the data from which we want to learn is introduced; a variable number of hidden layers, in charge of executing the learning process; and a output layer, which produces the result [13]. Each neuron of each layer is connected with all neurons of the next layer, and each connection is defined by a weight and a threshold. If the output of a node is greater than the threshold, its connection is activated and it sends information to the next layer.

In order to learn, NNs need training data from which features and patterns are automatically extracted. This learning procedure allows the NN to optimize its parameters, so that they are able to generalize the results and produce and accurate output on new, previously unseen data, called testing set [12]. There exist techniques that help the model to generalize (i.e. cross–validation, dropout, batch normalization techniques) and to avoid being excessively adjusted to the training data, which is called overfitting [14]. Equally significant is taking into account the type of problem to be solve; it can be 1) a binary problem, in which the range of the final outcome is formed by two mutually exclusive results; 2) a multi–class problem, where several mutually exclusive values are possible results, and 3) a multi–label problem, meaning that the potential results are not mutually exclusive. Multi Later Perceptrons (MLPs) are a type of NN commonly used to develop profiling attacks, that is, to simulate the activity of the attacked device and extract sensitive information [15].

III. CRISTALS-Kyber

Kyber is a lattice-based post-quantum key encapsulation mechanism (KEM) selected by NIST as PQC standard in the public key encryption category (PKE/KEM) in 2022.

The security of Kyber is based in a variant of the *Learning With Errors* (LWE) problem, called *Ring Learning With Errors* (R-LWE). We denote as $R = \mathbb{Z}[x]/(x^n + 1)$ and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. R_q defines the lattice R_q^k used in Kyber, where k takes values 2, 3 or 4 depending on the three parameters sets presented in the Kyber submission. The other parameters involved are $q = 3329$ and $n = 256$.

A. Algorithm

Kyber submission consists in two public key cryptosystems Kyber.PKE and Kyber.KEM. Kyber.PKE is defined as the three algorithms key generation, encrypt and decrypt (see Tables I, II), then these algorithms are used to define Kyber.KEM as key generation, encapsulation and decapsulation (see Table III). The transformation used is called FO$^{\not\perp}$ transformation.

Most of the attacks target the functions used in Kyber.PKE, but the attacks are against Kyber.KEM security.

Kyber.PKE uses several functions, see [16], here are presented the following functions: GenMatrix, Encode (E), Decode (D), Compress (Co) and Decompress (Dc). Also the hash and other auxiliar functions used in both Kyber.PKE and Kyber.KEM are denoted as H, G and KDF.

TABLE I
Kyber.PKE.KeyGen ($\mathcal{G}$) and Kyber.PKE.Encrypt ($\mathcal{E}$)

Kyber.PKE.KeyGen(), $\mathcal{G}()$	**Kyber.PKE.Encrypt**(pk, m, r)**,** $\mathcal{E}(pk, m, r)$
Sample $\rho \| \sigma$	$\hat{t} := D_{12}(B)$
GenMatrix$(\rho) = A$	$\rho \leftarrow pk$
Sample noise $_\sigma \rightarrow s, e$	GenMatrix$(\rho) = A$
$\hat{s} := NTT(s)$	Sample noise $_r \rightarrow s', e_1, e_2$
$\hat{e} := NTT(e)$	$\hat{s'} := NTT(s')$
$\hat{t} := A \circ \hat{s} + \hat{e}$	$u := NTT^{-1}(A^T \circ \hat{r}) + e_1$
$B = (\mathrm{E}_{12}(\hat{t}\ mod^+ q) \| \rho)$	$v := NTT^{-1}(t^T \circ r) + e_2 + \mathrm{Dc}_q(\mathrm{D}_1(m), 1)$
$sk := \mathrm{E}_{12}(\hat{s}\ mod^+ q)$	$c_1 := \mathrm{E}_{d_u}(\mathrm{Co}_q(u, d_u))$
return $(pk := (seed_A, B), sk)$	$c_2 := \mathrm{E}_{d_v}(\mathrm{Co}_q(v, d_v))$
	return $c = (c_1 \| c_2)$

TABLE II
Kyber.PKE.Decrypt ($\mathcal{D}$)

Kyber.PKE.Dec(sk, c)**,** $\mathcal{D}(sk, c)$
$u := \mathrm{Dc}_q\ (\mathrm{D}_{d_u}(c), d_u)$
$v := \mathrm{Dc}_q\ (\mathrm{D}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
$\hat{s} := \mathrm{D}_{12}(sk)$
$m := \mathrm{E}_1(\mathrm{Co}_q(v - NTT^{-1}(\hat{s}^T \circ NTT(u)), 1))$
return m

The following scheme shows how Kyber.KEM works, the algorithms Kyber.PKE.KeyGen, Kyber.PKE.Encrypt and Kyber.PKE.Decrypt are denoted as $\mathcal{G}$, $\mathcal{E}$ and $\mathcal{D}$ respectively.

Is important to highlight the re-encryption algorithm in the decapsulation (Table III) of Kyber.KEM as a common target of SCA. The re-encryption consists in the process of

TABLE III
KYBER.KEM ALGORITHMS

Kyber.KEM.KeyGen()	**Kyber.KEM.Encaps**(pk)	**Kyber.PKE.Decaps**(sk, c)
$z \leftarrow B^{32}$	$m' \leftarrow B^{32}$	$h := sk + 24 \cdot k \cdot n/8 + 32$
$(pk,\ sk') \leftarrow \mathcal{G}$	$m \leftarrow H(m')$	$z := sk + 24 \cdot k \cdot n/8 + 64$
$sk := (sk' \| pk \| H(pk) \| z)$	$(\bar{K}, r) \leftarrow\ (m \| H(pk))$	$\tilde{m} \leftarrow \mathcal{D}(sk, c)$
return: $(pk,\ sk)$	$c \leftarrow \mathcal{E}(pk, m, r)$	$(\bar{K}', r') \leftarrow G(\tilde{m} \| h)$
	$K \leftarrow KDF(\bar{K} \| H(c))$	$c' \leftarrow \mathcal{E}(pk, \tilde{m}, r')$
	return $(c,\ K)$	**if** $c = c'$, **return** $K = KDF(\bar{K}' \| H(c))$
		else return $K = KDF(z \| H(c))$

encypting again a decrypted message and checking if the ciphertext obtained is the same as the one received as input. Inside the re-encryption part of the algorithm the functions that are the primary target are the Encode and Decode functions. In this work we analyze three SCA (see Table IV) against Kyber.KEM, that use a side channel leakage to train an AI model. The attacks in [17] and [18], target both the re-encryption part of the decapsulation algorithm. In the case of [19] the target is the encapsulation, although still attacks the Encode function used in Kyber.PKE.Encrypt (see Table I).

B. Determiner Leakage Attack

In Kyber.PKE.Encrypt of Table I the encode function is applied to the message m. This function encodes each bit of a message m, m_i, into a coefficient of a polynomial $x \in R_q$, x_i. The operation performed by the encode function is the bitwise–and operation: $x_i = m_i \ \& \ \frac{q+1}{2}$. This enconde function is the target of the attack presented in [17], since the masking used in this function meets the definition of a determiner.

In [19] a determiner is defined as "an intermediate value that is defined according to a sensitive bit value, and the difference between the Hamming weights of the elements of the determiner domain is greater than or equal to 2. The cardinal number of the determiner domain is 2". The domain of $\mathbf{mask}$ contains two vales: $\mathbf{mask} = 0$xFFFF if the message bit $m_i = 1$, and $\mathbf{mask} = 0$x0000 if $m_i = 0$. Consequently, $\mathbf{mask}$ can take two possible values, whose Hamming Weight (HW) is $16 \geq 2$.

The side channel attack presented in [19] uses a single power consumption trace. The first step of the attack is to select, for each m_i, a points of interest (PoIs). These are the points (for each i) where the $\mathbf{mask}$ value is calculated, stored and loaded. These PoIs are then classified using clustering algorithms. The PoIs are classified then in two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, then the mean of each of these groups is calculated $E(\mathbb{G}_1)$ and $E(\mathbb{G}_2)$. Then based in the difference of the means, one set contains the PoIs of the $m_i = 0$ and the other the PoIs such that $m_i = 1$. Recovering the message and with public information of the Kyber.KEM, any shared key can be recovered. This strategy is known as determiner leakage attack [11]. This attack employs k–means, an unsupervised ML algorithm, to classify power consumption traces measured at the message encoding operation in the encapsulation phase, and determine if a trace represents a $\mathbf{mask} = 0$ or $\mathbf{mask} = 1$ case (determiner leakage attack). By doing this, the authors are able to retrieve the original message $\mathbf{m}$ with a 100% of success rate and, with that, the secret shared key. From the engineering perspective, the strengths of this attack are based on the fact that only one trace is required to recover the whole message. This simplifies the attack, as it is only necessary to measure the power consumption once. Additionally, it is mentioned that the k–means algorithm is trained with 500 traces, which should be considered as a low quantity. However, k–means is one of the simplest ML models, which means that the attack might be simplified with more sophisticated algorithms.

Other study that also exploits the determiner leakage is presented in [17]. In this case, the authors propose an attack that is successful up to the 5^{th}–order masked implementation of CRYSTALS–Kyber. In this case the target is not the encode function used in the encapsualtion, but the re–encryption procedure of the decapsulation. Higher order masking is a measure against side channel leakage. A w-order masking consists into splitting a sensitive variable x into $w+1$ shares, and operate separately on the shares. In this case, the attack is based on MLPs that learn from power consumption traces. Specifically, these MLPs are able to identify the shares for each message bit in a 1^{st}–order implementation by analyzing the gamma parameter of their first layer. With this information, the authors augment the MLPs to be successful against higher–order implementations of CRYSTALS–Kyber by transfer learning (they call it recursive learning). This means that, to a attack a w–order masked implementation, the MLP is initialized with the weights of the $(w-1)$–order masked implementation MLP. Additionally, the authors specify that this methodology works better with the two first bits of each byte, and propose a cyclic rotation method to enhance the overall results. To obtain all the bits correctly is used a property of the LWE-based cryptosystem, this is that the bits of the message can be rotated manipulating the ciphertext. Let consider a polynomial $p \in R_q$ and $f_t(x) = x^t \in R_q$, then $pt = p \cdot x^t$ is a polynomial with the same coefficients of p. Since R_q is an anti-cyclic polynomial ring, we have that for $i \leq t$ $pt_i = -p_{n-t+i}$ and for $t \leq i \leq n-1$ $pt_i = p_{i-t}$. This property is exploited in several attacks against Kyber and other LWE-based cryptosystems.

The average bit recovery success against the implementation with 5^{th}–order masking, with only one trace is of 97.99%, which implies a message recovery success of $(97.99\%)^{256} = 0.56\%$. This results are much better for a higher number of traces, being for 5 traces the best result presented with a 87.07% of success rate in message recovery. A limitation of this attack is the large size of the training and testing sets that uses; since are required 30000 training traces and 2500 testing traces, combined with the original message (MLPs are supervised). Finally, it is not specified if dropout layers or any cross–validation process was executed, which are typical tools to avoid overfitting.

C. Plaintext Checking Oracle

The attack presented in [18] consists in the construction of a *Plaintext-Checking* (PC) oracle, using side channel leakage of information. the target of the attack is the re-encryption part of the decapsulation algorithm. A previous binary PC

TABLE IV
AI–BASED SCAS AGAINST CRYSTALS–KYBER.

Reference	Target	traces	Algorithm	Results
[19]	message encoding operation (encapsulation)	Power consumption	k–means	Message recovery (100%)
[17]	re–encryption step of the decapsulation	Power consumption	MLP	Message Recovery
[18]	re–encryption step of the decapsulation	Electromagnetic emanations	Welch's t–test and k–means	Secret key recovery
[20]	decapsulation algorithm	Power consumption	MLP	Message Recovery (100%)

oracle attack, published by Ravi et al. in [21] succeed in recovering the key of Kyber512 one coefficient at a time. The mayor improvement in the parallel PC oracle approach is to optimize the attack in the number or traces needed and the overall time that it takes. The target of the PC oracle is uased to recover the shared secret bites. To set up the PC oracle Rajendran et. al. make use of the fact that the re-encryption procedure is deterministic and depends solely on the message (shared secret) m. Knowing this information one can design ciphertexts of specifics m and study the side channel leakage.

Any ciphertext of Kyber has the form $c = (u, v) \in R_q \times R_q$, so an attacker can choose k_u, $K_v \in \mathbb{Z}_q$ such that $u = k_u x^0$ and $v = k_v x^i$. Let s denote the secret key, then the decrypted message coefficients are given as $m_0 = \mathrm{Dc}(v - u \cdot s[0])\mathrm{and}m_i = \mathrm{Dc}(-u \cdot s_i)\mathrm{for}i \in \{1, n-1\}$. The bit m_i of the decrypted message depends on the corresponding secret bit of the key. The attacker can choose (k_u, k_v) such that $m_i = 0$ for $i \geq P$. For $P = 1$, if the attacker has access to a binary PC oracle through side channels then the bit s_0 of the key can be recover. The viability of PC oracle attacks is estimated, among other factors, on the number of queries to the oracle. This is optimized using Binary Decision Threes (BDT). In [18] is introduced how to construct an optimal BDT, i.e. one of minimum queries. The idea of using a BDT consists on asking the oracle with an initial parameter t, defined by the three, then, depending on the answer of the oracle (0 or 1), the BDT indicates a new t or the value of s_i.

The attack presented in [18] chooses the ciphertext as $u = 208x^0$ and $v = 208 \cdot t \cdot \sum_{i=0}^{P-1} x^i$. The parameters of the attack are P, called parallelization factor, the number of bits to discover of s, and t, the parameter of the BDT. The average number of queries for a full recovery is $Q = \lceil \frac{2^8}{P} \rfloor \cdot k \cdot Q_{set}$, where Q_{set} is the average number of queries required to recover P coefficients using the BDT.

The side channel attack used to setup the PC oracle in [18] expands a previous work that used a binary leakage of the voltage during the re-encryption procedure of Kyber. The attack consists in two phases a pre-processing phase and classification phase. In the multi-target version of this oracle construction these phases are expanded from the binary version to target P bits of the message. In the pre-processing phase the adversary collects the measurements of the re-encryption for all 2^P values of $m \in \{0, 2^P - 1\}$. In the following classification phase, given an attack trace tr the adversary compares it with the set of traces obtained before. The correct class is selected after $2^P - 1$ classifications.

The best case scenario of the attack presented in [18] recovers the full key in just 72 queries and the experimentally verified case of $P = 10$ the number of queries is ≈ 232. For these results is consider an attacker with access to a clone device. Without using the cloning device the number of total queries $Q_t = Q + Q_{temp}$, where Q_{temp} denote the queries during the pre–processing phase. In this setup the experimental attack with the lowest number of queries is for the parameters $t = 5$ and $P = 4$, reaching 613 queries. However, it is proposed in [18] (though not experimentally), an attack with parameter values $t = 1$ and $P = 6$ that reaches the number of queries of 437.

D. Multi–bit Error Injection Attack

In [20] is presented a expansion of the multi–bit error injection attack against SABER introduced in [22]. This attack uses the bit flipping property [11] of LWE/R-LWE algorithms. Let x be the encoded message polynomial and $c = (u, v)$ the corresponding ciphertext. Following the scheme in I it can be expressed $v_i = x_i + v_i'$ for $i \in \{0, 255\}$. The coefficients of x can only be $\frac{q+1}{2}$ or 0, depending on the value of the corresponding bit of m. During decryption II there are no operations between the coefficients of x, hence any bit m_i can be flipped in the ciphertext simply subtracting $\frac{q+1}{2}$ from the corresponding coefficient v_i.

The work in [20] presents an attack against a hardware implementation of Kyber768. Attacking hardware implementations is more difficult than software implementations and the attacks need to be adapted, in this case the multi–bit error injection attack. Let consider a message $m = (m_i)_{i \in \{0,31\}}$ and the corresponding ciphertext c. A standard multi–bit error injection attack consists on choosing an error $e \in \{0, 255\}$ and finding a modified ciphertext c_e, such that the decrypted message $m_e = (m_0 \oplus e, \ldots, m_{31} \oplus e)$. The bits of e with value 1 flip the corresponding bits of m_i for all $i \in 0, 32$. After computing the traces of the decapsulation of c_e for all $e \in \{0, 255\}$, the segments where the Decode function is applied to each message byte $m_i \in \{0, 31\}$ are all extracted. Finally these trace segments are given as input to the MLP to train.

This attack against a hardware implementation requires some changes. It is introduced a tecnique called slicing. The error e is introduced into every fourth byte. In this case, an error is injected every four bits, hence, to recover the hole message, 256×4 traces are needed. The authors show that with 5 repetitions of each trace, that is, a total of $256 \times 4 \times 5$ traces, the message success rate is 100%. The attack is carried out with MLPs that analyzed the power consumption traces

under the described conditions acquired at the execution of the decapitation procedure of the decapsulation. It should be considered that the initial training set was composed of 200000 traces, which were expanded via the cut–and–join technique for a final set of 6.4M traces.

IV. CRYSTALS–Dilithium

Dilithium, a lattice-based post-quantum digital signature algorithm, is one of the third round finalists of the NIST standardization project. The security of Dilithium is based on the hardness assumption of Module Short Integer Solution (MSIS) and Module Learning with Errors (MLWE) problems. As in the previous section, we denote as R and R_q the polynomial rings where these problems are defined, taking $n = 256$ and $q = 2^{23} - 2^{13} + 1$ as particular values for this case. The elements of these rings are identified with coefficient vectors in $\mathbb{Z}^N$ and $\mathbb{Z}_q^N$, respectively.

A. Algorithm

The scheme design of Dilithium is based on the "Fiat-Shamir with aborts" approach [23], consisting on a three tuple of algorithms: key generation, signing (Table V) and verification.

TABLE V
Dilithium Signing Procedure

Signature Generation
$\mathbf{A} \in R_q^{k \times l} := ExpandA(\rho)$
$\mu \in \{0,1\}^{384} := CRH(tr \| \| M)$
$\kappa := 0, (\mathbf{z}, \mathbf{h}) := \perp$
while $(\mathbf{z}, \mathbf{h}) := \perp$ **do**
$\quad \mathbf{y} \in \tilde{S}_{\gamma_1}^l := ExpandMask(\rho', \kappa)$
$\quad \mathbf{w} := \mathbf{Ay}$
$\quad \mathbf{w_1} := HighBits_q(\mathbf{w}, 2\gamma_2)$
$\quad c \in B_\tau := H(\mu \| \| \mathbf{w_1})$
$\quad \mathbf{z} := \mathbf{y} + c\mathbf{s_1}$
$\quad r_0 := LowBits_q(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)$
$\quad$ **if** $\| \| \mathbf{z} \| \|_\infty \geq \gamma_1 - \beta$ and $\| \| \mathbf{r_0} \| \|_\infty \geq \gamma_2 - \beta$ **then**
$\quad\quad (\mathbf{z}, \mathbf{h}) := \perp$
$\quad$ **else**
$\quad\quad \mathbf{h} := MakeHint_q(-c\mathbf{t_0}, \mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0}, 2\gamma_2)$
$\quad\quad$ **if** $\| \| c\mathbf{t_0} \| \| \geq \gamma_2$ or the # of 1's in $\mathbf{h}$ is greater than $\mathbf{w}$ **then**
$\quad\quad\quad (\mathbf{z}, \mathbf{h}) := \perp$
$\quad\quad$ **end if**
$\quad$ **end if**
$\quad \kappa := \kappa + l$
end while
return $\alpha = (\mathbf{z}, \mathbf{h}, c)$

To efficiently compute polynomial multiplication, one of the most expensive operations, the NTT-based multiplication is adopted. Considering r as the 512-th root of unity modulo q, the domain R_q is isomorphic, by the Chinese remainder theorem, to the product $\prod_i \mathbb{Z}_q[x]/(x - r^i)$. We have then that each of these rings $\mathbb{Z}_q[x]/(x - r^i) \cong \mathbb{Z}_q$. Thus, multiplication on R_q can be easily done by pointwise multiplication.

The attacks described on the subsequent sections aim for signature forgery by recovering the secret key vectors. On the first case they recover both $\mathbf{s}_1$ and $\mathbf{s}_2$ [24], while the second only needs $\mathbf{s}_1$ to perform universal forgery of signatures [25].

B. Number Theoretic Transform Attack

The target of the attack presented in [24] is the step computing NTT representation (Table VII) of secret vectors $\mathbf{s}_1$, $\mathbf{s}_2$ [3]. In each loop of it, the initial value of len is divided by 2. Thus, we can divide the NTT operation into eight stages with $len = 2^{8-m}$ at the m-th stage. First we consider $\mathbf{s}_1$. Polynomials $s_{1,i}$, $0 \leq i < L$, are the input values $p[N]$ of Table VII and, at the first stage ($m = 1$), $p[j]$ is the secret coefficient $s_{1,i,j}$ for each $0 \leq j < N$. Therefore, at the first substage, $p[j + len]$ is the coefficient $s_{1,i,j+len}$ with $0 \leq j < 128$. As $zetas$ are known and precomputed values, the difference in power consumption when the intermediate value $t = Mont\_r(zeta * p[j + len])$ is calculated, stored, and loaded leaks information about $s_{1,i,j+len}$. Then, we can recover $s_{1,i,j}$ for $128 \leq j \leq 255$ targeting this value at the first substage of the first stage.

By repeatedly targeting t at the first substages of stages $2 \leq m \leq 8$, we can recover $s_{1,i,j+len}$ for $0 < j < len = 2^{8-m}$ (having fixed the values recovered on the previous stage). Finally, coefficient $s_{1,i,0}$ is not used as input of $Mont\_r$, but it affects the computation at the last stage $m = 8$ of $p[j]$ with $j = 0$ for all $0 \leq i < L$. Thus, it is possible to recover it by summing the leakage from line $p[j] = p[j] + t$ at this point.

In the case of targeting the signature procedure, this method can be used to recover $\mathbf{s}_2$ too, as this algorithm computes NTT representation for this vector as well. Key generation only computes NTT representation for $\mathbf{s}_1$. In this case, $\mathbf{s}_1$ is recovered targeting NTT and $\mathbf{s}_2$ is obtained by attacking sampling, addition, rounding and packing operations [24].

When applying masking to the signing procedure, the advantage of this attack is lost. That motivates targeting key generation [26]. When the masking scheme uses as modulus a power of two the NTT multiplication is not available. In that case, the target operation is sparse multiplication.

The studies available in [26] and [24] elaborate single power consumption trace profiling attacks that take advantage of the NTT leakage. Both employed MLPs to profile the device under attack, and are able to retrieve the secret key and forge signatures. On one hand, in [26] the authors achieve 100% of random secret key recovery using 2000 and 8000 traces in the training and testing sets, respectively. These results are maintained when the masked version of Dilithium is attacked. In this case, the target is the polynomial multiplication, but the number of traces in the training set increases to 9000, which represents the complexity increase when countermeasures are applied. On the other hand, the secret key vectors s_1 and s_2 are obtained in [24] with a 100% and 92.91% of recovery success rate, respectively. To solve the masked version, the attacks targets the key generation procedure, specifically the sampling, addition, rounding and, packing operations. With this methodology, the recovery success rate increases to 98%, and the size of the training and testing sets are equal in the non–protected and protected versions.

TABLE VI
AI–BASED SCAS AGAINST CRYSTALS–DILITHIUM.

Reference	Target	Traces	Algorithm	Results
[26]	NTT of signing procedure	Power consumption	MLP	long term key recovery (100%)
[24]	NTT of signing procedure	Power consumption	MLP	long term key recovery
[25]	bit–unpacking function of signing process	Power consumption	MLP	secret key recovery

TABLE VII
DILITHIUM NTT SCHEME

Procedure NTT (p[N])

$k = 1$
for $(len = 128; len > 0; len >>= 1)$
  for $(start = 0; start < N; start = j + len)$
  $zeta = zetas[k]$
  $k := k + 1$
  for $(j = start; j < start + len; ++j)$
  $t =$ Mont_r$(zeta * p[j + len])$
  $p[j + len] = p[j] + 2 * Q - t$
  $p[j] = p[j] + t$

$zetas$: precomputed table $Mont\_r$: Montgomery reduction, Q: prime, N: dimension

C. Bit–Unpacking Function Attack

There is another vulnerable leaking point on the signature algorithm found when generating the vector $\mathbf{y}$ (see Table V line 5) [25]. This polynomial vector is gathered from a bit string obtained expanding an initial randomness seed ρ'. This is performed by a function that views the bit-string as a byte-string and unpacks it into l polynomials, where each one is unpacked separately [25]. This function iterates $N/4$ times for each polynomial, as it computes four coefficients, $(i)^{th}$, $(i+1)^{th}$, $(i+2)^{th}$, $(i+3)^{th}$, on each i-th iteration.

The power traces of its execution leak information about the generated coefficients, concretely whether they are zero or not. Marzoygui et al., [25] developed a profiling attack exploiting this bit–unpacking function leakage in the signing process. They constructed four different MLPs, one for each i^{th}, $(i+1)^{th}$, $(i+2)^{th}$, and $(i+3)^{th}$ coefficients of the polynomial. The results provided, in terms of accuracy, precision, recall and specifity, show the success of their proposal, as their values are all above 99.90%.

This knowledge can be used to recover the secret key $\mathbf{s_1}$ in four steps. We denote with $y_{i,j}$ the j^{th} coefficient of the i^{th} polynomial of vector $\mathbf{y}$, for $i \leq l$ and $j \leq 256$. First we need to filter the predictions of the trained classifiers to avoid false-positives (i.e. predicting that a coefficient $y_{i,j}$ is zero when it is not), as this would affect the system of equations we are to solve. Thus, conditions are defined for the $\mathbf{z}$ values (Table V line 9). As $|(cs_1)_{i,j}| \leq \tau \cdot \eta = \beta$, assuming $y_{ij} = 0$ means $|z_{i,j}| = |y_{i,j} + (cs_1)_{i,j}| \leq \beta$ as well. Then, we can dismiss the possibility that a coefficient $y_{i,j}$ is zero if the corresponding $|z_{i,j}| \geq \beta$. Also, each coefficient $(cs_1)_{i,j}$ can be approximated by a normal distribution with variance $\sigma^2 = \frac{(2\cdot\eta)^2 - 1}{12\cdot\tau}$. Then, if $|z_{i,j}| > 2 \cdot \sigma$, we can suppose $y_{i,j} \neq 0$. This reduces the number of false positives as the machine-learning classifier is only invoked to predict whether $y_{i,j}$ is zero if $|z_{i,j}| \leq 2 \cdot \sigma$.

Considering we traced M signatures, we have $\mathbf{z}^m = \mathbf{y}^m + c^m\mathbf{s}_1$ for each m-th signature. Using the filter previously described, we obtain a list of L triples (m, i, j) such that $y^m_{i,j} = 0$ is predicted. Factoring in the erroneous predictions, we have the following set of equations $z^m_{i,j} = (c^m s_1)_{i,j} + e$, where $e = 0$ when the classifier correctly predicted $y^m_{i,j} = 0$ and $e \neq 0$ otherwise. Thus, the goal is to obtain $\mathbf{s}_1$ from $\mathbf{z} = \mathbf{C}\mathbf{s_1} + \mathbf{e}$, where $\mathbf{C} \in \mathbb{Z}^{|L| \times n}$ stems from the challenge polynomials c^m, $\mathbf{z}$ enclose the signature coefficients $z^m_{i,j}$ and $\mathbf{e}$ is a vector of error coefficients. Notice that the problem can be separated into l independent equation systems, one for each polynomial in $\mathbf{s}_1$, as to obtain $c\mathbf{s}_1$ we multiply c (a single polynomial with exactly τ non-zero coefficients) independently with each one of the l polynomials in the vector $\mathbf{s}_1 \in S^l_\eta$. From now we denote as $\mathbf{s}$ the polynomial $(\mathbf{s}_1)_i$ currently solving.

Assuming that $y_{i,j} = 0$ is correctly predicted means that e is zero for most of the equations. From this is possible to obtain a first key candidate $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ close to the correct secret key $\mathbf{s_1}$. Since $||c\mathbf{s} + \mathbf{e}||_\infty < q$, there are no modular reductions involved. then, we can view the problem of solving the system of linear equations $\mathbf{z} = \mathbf{C}\mathbf{s} + \mathbf{e}$ as a LWE problem without modular reduction and obtain a solution $\hat{\mathbf{s}}_1 \in \mathbb{R}^n$ applying the least-squares method [ref approach]. This method computes $\hat{\mathbf{s}}_1$ as the vector minimizing the squared euclidean norm $||\mathbf{C}\hat{\mathbf{s}} - \mathbf{z}||_2^2$ and can be easily computed using the formula $\hat{\mathbf{s}} = (\mathbf{C}^T\mathbf{C})^{-1} \cdot \mathbf{C} \cdot \mathbf{z}$. This converges to a correct solution: given enough equations, $\lfloor \hat{\mathbf{s}}_i \rceil = \mathbf{s}_i$ for all $i \in \{1, \ldots, n\}$. Then, to obtain the secret key from this solution candidate they observe that the following should hold for each coefficient in $\hat{\mathbf{s}}$: (1) Either rounding up or down should yield the correct solution, $\lfloor \hat{s}_j \rfloor = (s)_j$ or $\lceil \hat{s}_j \rceil = (s)_j$, (2) for coefficients j such that $\hat{\mathbf{s}}_j$ is close to an integer, the coefficient candidate should be correct.

As for most of the equations in the system $e = 0$, we are looking for the polynomial that maximizes the number of fulfilled equations. Therefore they formulate an Integer Lineal Program where the information about the solution candidate $\hat{\mathbf{s}}$ is included as constraints. Finally, they run solvers for this program until they obtain a solution that satisfies at least $(1 - e) \cdot N$ equations, where e is the assumed false-positive rate. This solutions should match the i-th secret key polynomial from $\mathbf{s}_1$. Running this method for each one of the l polynomials produce a final secret key candidate $\mathbf{s}_1$.

V. AI ANALYSIS

In this section we provide a deeper analysis of the AI models, configurations, and metrics employed in the previ-

TABLE VIII
ATTACK ALGORITHMS SPECIFICATIONS.

Reference	Algorithm	Train/Test set	Cross–validation/Overfitting	Act. func.	Optimizer	Batch size	Epochs	Loss func.
[17]	MLP (binary)	30K/2.5K	Batch–Normalization	ReLU/Softmax	Nadam	1024	100	Binary cross–entropy
[20]	MLP (multi–class)	200K/5.12K	Batch–Normalization	ReLU/Softmax	Nadam	1024	100	Categorical cross–entropy
[26]	MLP (multi–class)	2K/8K	Batch–Normalization	ReLU/Softmax	adam	32	100	Categorical cross–entropy
[24]	MLP (multi–class)	50K/10K	Batch–Normalization/Grid–search	ReLU/Softmax	Nadam	32	500	Categorical cross–entropy
[25]	MLP (multi–class)	-	Dropout	-	Hyperband	-	-	-

ously described works. Table VIII summarizes the principal characteristics of the AI algorithm of each attack, including: the size of the training and testing sets, how overfitting was prevented and which cross–validation procedure was executed, the activation function and optimizer used, the size of the batches, the number of epochs, and the loss function used. We did not include [19] and [18] in Table VIII because they utilized a simple k–means clustering algorithm and do not specify any of the parameters of interest. As a comment, it would be useful to study different classification algorithms, such as Support Vector Machines (SVM) or Random Forest Classifiers, in similar experiences to evaluate a possible enhancement in the results.

The first remarkable aspect is that all studies are based on the construction of a MLP. Despite it is a well–known and useful model, the study of SCAs against CRYSTALS–Kyber and CRYSTALS–Dilithium should be extended by analyzing the performance of other algorithms, such as CNNs or RNNs. For example, the use of CNNs has been explored against Frodo and NewHope in [27].

It can be easily seen that most of the studies employ batch–normalization layers to deal with overfitting. It is a procedure that provides regularization in the learning process, making the training faster and more stable. However, it is not enough to ensure that the model generalizes and, therefore, in [24] the authors also apply a 10–fold grid search process to optimize the parameters. Alternatively, [25] is the only article that uses dropout layers. To prove that the proposed algorithm does not overfitt, it is usual to include graphics that represent the training and testing accuracy and loss. Nevertheless, among the studied works, only [26] and [24] provides these graphics. We find this aspect crucial to show the reliability of the models.

Regarding the activation functions, all works used the Rectified Linear Unit (ReLU) [28] between hidden layers and the Softmax before the output layer. The selection of the Softmax activation function is logical for this application since the built MLPs are not excessively complex, the number of outputs is reduced, and the proposed problems are binary or multi–class. However, it would be interesting to study if the results can be improved using the Exponential Linear Unit (ELU) [29] or the Scaled Exponential Linear Unit (SELU) [30] instead of ReLU. It has been shown that the former enhances the outcomes in classification problems, while the latter learns faster and does not have the vanishing gradient problem [28].

Similarly, all settings employed the Adam (Adaptive Moment Estimation) or the Nadam (Nesterov–Adam) optimizers. Their principal characteristics are very similar, as both update the learning rate in each cycle and store an exponentially decaying average of past square gradients. These features lead Adam and Nadam to be fast methods that converge rapidly, but that are computationally expensive. Their difference lays in the use of the Nesterov momentum by Nadam, instead of the classical momentum used by Adam. This means that Nadam does not miss any local minima when optimizing, but it may slow the optimization. There is no doubt these are suitable optimizers to construct SCAs MLPs. However, both should be tested in order to select the best one for each specific application, as their performance may vary depending on the preprocessing steps, data utilized or model architecture.

The selection of the loss functions is adequate; the works that solve a binary problem (i.e. bit prediction) use the binary cross–entropy, whereas the multi–class classification problems apply the categorical cross–entropy.

Lastly, it should be noticed that only the use of dropout and the hyperband optimization technique is specified in [25]. Hyperband is a hyperparameter optimization method based on the early–stopping algorithm (optimize the parameters until the model starts to overfitt) and that adaptively allocates a pre-defined resource, e.g., iterations, data samples or number of features, to randomly sampled configurations [31]. Despite we find its use appropriate, it would be interesting to know the specific loss and activation functions employed. Interestingly enough, while the rest of studies provide their outcomes in terms of success rate, this work is the only one that uses accuracy, precision, recall and specificity. These metrics are very useful to analyze how a AI model performs and verify if it generalizes well and, in case it does not, understand if it fails in false negative or false positives. From the security point of view, this is a critical aspect, being a false positive the worst case scenario.

VI. CONCLUSION

The recent technological advances suggest that the possibility of developing a quantum computer able to execute Shor's algorithm is an upcoming reality. For this reason, the popularity of PQC has notably increased the last few years, and the NIST has selected two new schemes have been standardized: CRYSTALS–Kyber for PKE and KEM, and CRYSTALS–Dilithium for digital signatures, among others.

However, similarly to classical cryptography, this algorithms present certain limitations that may be exploited with SCAs to recover senstive information. Among this attacks, those based on AI models are becoming more important due to their potential. In this work, we collected the newest and most

important AI–based SCAs against the recently standardized cryptosystems, and analyze their reliability and limitations. Among them, we found that more emphasis to avoid overfitting should be made, and the analysis of more complex proposals that require less training samples might be beneficial. In future works, we plan to study the performance of different AI models, such as SVMs, to reduce the number of traces needed to conduct the training.

ACKNOWLEDGMENT

This work was supported in part by the R&D&I project P2QProMeTe, Grant PID2020-112586RBI00 funded by MCIN/AEI/10.13039/501100011033; in part by ORACLE Project, with reference PCI2020-120691-2, funded by MCIN/AEI/10.13039/501100011033, and European Union "NextGenerationEU/PRTR", and in part by the EU Horizon 2020 research and innovation programme, project SPIRS (Grant Agreement No. 952622). L.H.A. and E.I.H. would like to thank CSIC Projects CASDiM and EFiDiP, respectively, for their support.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, 1997, https://doi.org/10.1137/S0097539795293172.

[2] R. M. Avanzi, J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation," 2017, https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf.

[3] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevesky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation," 2020, https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[4] G. Alagic, D. Cooper, Q. Dang, T. Dang, J. M. Kelsey, J. Lichtinger, Y.-K. Liu, C. A. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, and D. Apon, "Status report on the third round of the NIST post-quantum cryptography standardization process," 2022, https://doi.org/10.6028/NIST.IR.8413.

[5] J. McCarthy, "What is artificial intelligence?" 2004. [Online]. Available: http://35.238.111.86:8080/jspui/bitstream/123456789/274/1/McCarthy_John_What%20is%20artificial%20intelligence.pdf

[6] L. Hernández-Álvarez, L. González-Manzano, J. de Fuentes, and L. Hernández Encinas, "Biometrics and artificial intelligence: Attacks and challenges," in *Breakthroughs in Digital Biometrics and Forensics*. Springer, 2022, pp. 213–240, http://doi.org/10.1007/978-3-031-10706-1_10.

[7] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. 1996 Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conferenc*. Springer, 1996, pp. 104–113, https://doi.org/10.1007/3-540-68697-5_9.

[8] F. Standaert, *Introduction to Side-Channel Attacks*. Springer, 2010, http://doi.org/10.1007/978-0-387-71829-3_2.

[9] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *J Cryptol*, vol. 30, pp. 392–443, 2017, https://doi.org/10.1007/s00145-015-9224-2.

[10] J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001*. Springer, 2001, pp. 200–210, https://doi.org/10.1007/3-540-45418-7_17.

[11] 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, http://doi.org/10.1109/TIFS.2021.3139268.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, https://www.springer.com/gp/book/9780387310732.

[13] G. Hinton, *Deep Belief Nets*. Springer US, 2010, pp. 267–269, http://doi.org/10.1007/978-0-387-30164-8_208.

[14] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, 2019, http://doi.org/10.1088/1742-6596/1168/2/022022.

[15] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 1–36, 2019, http://doi.org/10.13154/tches.v2020.i1.1-36.

[16] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber," Online publication, 2020, https://pq-crystals.org/.

[17] E. Dubrova, K. Ngo, and J. Gärtner, "Breaking a fifth-order masked implementation of CRYSTALS-Kyber by copy-paste," Cryptology ePrint Archive, Paper 2022/1713, 2022, https://eprint.iacr.org/2022/1713.

[18] G. Rajendran, P. Ravi, J.-P. D'Anvers, S. Bhasin, and A. Chattopadhyay, "Pushing the limits of generic side-channel attacks on LWE-based KEMs - parallel PC oracle attacks on Kyber KEM and beyond," Cryptology ePrint Archive, Paper 2022/931, 2022, https://eprint.iacr.org/2022/931.

[19] 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, https://doi.org/10.1109/ACCESS.2020.3029521.

[20] Y. Ji, R. Wang, K. Ngo, E. Dubrova, and L. Backlund, "A side-channel attack on a hardware implementation of CRYSTALS-Kyber," Cryptology ePrint Archive, Paper 2022/1452, 2022, https://eprint.iacr.org/2022/1452.

[21] P. Ravi, S. Sinha 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*, vol. 2020, no. 3, pp. 307–335, 2020, http://doi.org/10.13154/tches.v2020.i3.307-335.

[22] R. Wang, K. Ngo, and E. Dubrova, "Making biased dl models work: Message and key recovery attacks on saber using amplitude-modulated em emanations," Cryptology ePrint Archive, Paper 2022/852, 2022, https://eprint.iacr.org/2022/852. [Online]. Available: https://eprint.iacr.org/2022/852

[23] V. Lyubashevsky, "Fiat-shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 598–616, https://doi.org/10.1007/978-3-642-10366-7_35.

[24] J. Han, T. Lee, J. Kwon, J. Lee, I.-J. Kim, J. Cho, D.-G. Han, and B.-Y. Sim, "Single-trace attack on NIST round 3 candidate Dilithium using machine learning-based profiling," *IEEE Access*, vol. 9, pp. 166 283–166 292, 2021, https://doi.org/10.1109/ACCESS.2021.3135600.

[25] S. Marzougui, V. Ulitzsch, M. Tibouchi, and J.-P. Seifert, "Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all," Cryptology ePrint Archive, Paper 2022/106, 2022, https://eprint.iacr.org/2022/106.

[26] I.-J. Kim, T. Lee, J. Han, B.-Y. Sim, and D.-G. Han, "Novel single-trace ML profiling attacks on NIST 3 round candidate Dilithium," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1383, 2020, https://www.semanticscholar.org/paper/Novel-Single-Trace-ML-Profiling-Attacks-on-NIST-3-Kim-Lee/650d74589c7f5e4fadc2168d8b78b6ad0c715cde.

[27] F. Aydin, P. Kashyap, S. Potluri, P. Franzon, and A. Aysu, "Deeparsca: Breaking parallel architectures of lattice cryptography via learning based side-channel attacks," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer, 2020, pp. 262–280, https://doi.org/10.1007/978-3-030-60939-9_18.

[28] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, "A review of activation function for artificial neural network," in *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2020, pp. 281–286, https://doi.org/10.1109/SAMI48414.2020.9108717.

[29] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *ArXiv*, vol. abs/1511.07289, 2016.

[30] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *ArXiv*, vol. abs/1706.02515, 2017.

[31] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Efficient hyperparameter optimization and infinitely many armed bandits," *ArXiv*, vol. abs/1603.06560, 2016.