PQC-SEP: Power Side-channel Evaluation Platform for Post-Quantum Cryptography Algorithms

Jungmin Park, N. Nalla Anandakumar, Dipayan Saha, Dhwani Mehta, Nitin Pundir, Fahim Rahman, Farimah Farahmandi and Mark M. Tehranipoor

Dept. of Electrical & Computer Engineering
University of Florida
Gainesville, FL, 32611, USA
{jungminpark, nnachimuthu, dsaha, dhwanimehta, nitin.pundir, fahim034, ffarahmandi, tehranipoor}@ufl.edu

Abstract. Research in post-quantum cryptography (PQC) aims to develop cryptographic algorithms that can withstand classical and quantum attacks. The recent advance in the PQC field has gradually switched from the theory to the implementation of cryptographic algorithms on hardware platforms. In addition, the PQC standardization process of the National Institute of Standards and Technology (NIST) is currently in its third round. It specifies ease of protection against side-channel analysis (SCA) as an essential selection criterion. Following this trend, in this paper, we evaluate side-channel leakages of existing PQC implementations using *PQC-SEP*, a completely automated side-channel evaluation platform at both pre-and post-silicon levels. It automatically estimates the amount of side-channel leakage in the power profile of a PQC design at early design stages, i.e., RTL, gate level, and physical layout level. It also efficiently validates side-channel leakages at the post-silicon level against artificial intelligence (AI) based SCA models and traditional SCA models. Further, we delineate challenges and approaches for future research directions.

Keywords: Post-quantum Cryptography, Lattice-based Cryptography, Side-channel Attack, AI-based Side-channel Attack.

1 Introduction

IBM has released a new 127-quantum bit (qubit) processor, named 'Eagle', in 2021, which is a step towards creating a 433-qubit chip called 'Osprey' next year, followed by an 1121-qubit chip called 'Condor' in 2023 [CDG21, Gam]. Other companies, including Google, Honeywell Quantum Solutions, and well-funded start-up companies, such as IonQ, have a similarly ambitious strategy to make a useful and error-corrected quantum computer [Bal21]. Due to the fast progress in the development of quantum computers, existing public key algorithms like RSA and Elliptic Curve Cryptography (ECC) are needed to be replaced since they could be broken by practical quantum computing [Sho97].

To address this need, the American National Institute of Standards and Technology (NIST) started the post-quantum cryptography (PQC) standardization process for key encapsulation mechanisms (KEM)/public key encryption (PKE) and digital signature schemes in 2016 [NISa]. The competition began with 69 proper submissions in December 2017. As of July 22, 2020, the competition entered the third round with seven finalist algorithms (4 KEM/PKE and 3 Signature). In contrast to traditional cryptography, PQC relies on different mathematical hard problems, which are believed to be secure against quantum attacks. There are three general hard problem families to build these schemes: code-based cryptography, lattice-based cryptography, and multivariate-based cryptography. Among the different schemes of PQC, lattice-based cryptography is one of the most promising approaches due to its simplicity, performance, and small key sizes. 5 out of 7 third-round finalist algorithms of the NIST processes are based on lattice-based schemes. The two remaining non-lattice schemes are not suited for all applications, so the new standard will probably include lattice-based schemes, one of the 3 KEM/PKE (CRYSTALS-KYBER, Saber, or NTRU), and one of the 2 Signature (FALCON or CRYSTALS-DILITHIUM) schemes. The components of lattice-based schemes are different compared to today's prevalent asymmetric cryptographic schemes. The recent advance in the PQC field has gradually shifted from the theory to the implementation of the cryptosystem, especially on the hardware platforms. During the standardization process, it is necessary to validate the candidates' implementation with secure countermeasures with regard to hardware vulnerability to side-channel attacks as well as mathematical cryptoanalysis.

Even though existing lattice-based KEM/PKE schemes are proven to be resistant to known mathematical cryptanalytic attacks, their hardware vulnerability has been studied recently. Especially, physical side-channel leakages, such as power and electromagnetic radiation, have been exploited to extract secret information [BSP$^+$22, PXJ$^+$18] and the primary threat to PQC implementations is caused by the physical side-channel leakages as well. In [HCY20], various side-channel techniques, such as vertical correlation power analysis, horizontal in-depth correlation power analysis (HICPA), online template attacks, and chosen-input simple power analysis, are exploited to recover the entire private key from NTRU-Prime, which targets generic polynomial multiplications. Xu *et al.* proposed power side-channel attacks on lattice-based Kyber to extract the whole secret key with less than 960 traces, targeting message-recovery decoding functions [XPSR$^+$21]. To prevent side-channel attacks, Beirendonck *et al.* proposed a side-channel resistant Saber implementation on an Arm Cortex-M4 (Arm CM4) core using a first-order masking method with a factor of 2.5x overhead [BDK$^+$21]. However, the first-order countermeasures can be broken by deep learning-based approaches in which masked values and random masks are trained to recover messages in a decoding step so that the secret key can be extracted in an IND-CCA secure Saber [NDGJ21a].

Based on many works of literature related to PQC side-channel attacks and countermeasures, there are challenging problems in this area as follows; i) Further study is necessary to discover more vulnerabilities against both traditional power/EM side-channel attacks and powerful AI-based side-channel attacks such that required countermeasures can be developed to satisfy both security standards and consumer constraints. ii) Existing side-channel evaluation methods are specific to a PQC implementation, so they are limited to various PQC HW or SW implementations. Therefore, a generic side-channel evaluation framework is required to verify the leakages of different PQC implementations efficiently. iii) Most existing side-channel leakage assessments have been performed in post-silicon validation. If the assessment result does not satisfy the security standard, the implementation should be modified at the expense of cost and time. Side-channel assessments at early design phases such as RTL or gate-level, can make a secure PQC implementation in an SoC or a standalone IP more efficient.

In this paper, for the first time to our knowledge, we attempt to exploit *PQC-SEP*, a completely automated side-channel evaluation platform to evaluate side-channel leakages of NIST KEM/PKE schemes at both pre- and post-silicon levels. PQC-SEP will provide the following distinctive capabilities to a chip designer and a security evaluator:

- It automatically estimates the amount of side-channel leakage in the power profile of a PQC design at early design stages, i.e., RTL, gate level, and physical layout level.
- It identifies which modules or primitive cells of a design are mainly leaking secret

information.

- The analysis is scalable to large complex system on chip (SoC) designs, which include a PQC module as an intellectual property (IP) block and master processors, hence accounting for the noise induced by other IP blocks in an SoC.

- It efficiently validates side-channel leakages at the post-silicon level against AI-based SCA models as well as traditional SCA models.

The rest of the paper is organized as follows. Section 2 provides the background of NIST Round 3 candidates, lattice-based KEM/encryption schemes and power/EM side-channel attacks on lattice-based PQC implementations. Section 3 describes the overview of power/EM side-channel attacks on PQC implementations. Section 4 describes PQC-SEP, a completely automated side-channel evaluation platform for the NIST PQC algorithm at both pre- and post-silicon levels. The experimental results of power side-channel leakages assessment at both pre- and post-silicon levels are given in Section 5. Section 6 discusses challenges and plan for secure PQC implementation against power side-channel attacks. Finally, Sections 7 present the conclusion. The detailed algorithm of Saber and Kyber is given in the Appendix A and B.

2 Preliminaries

2.1 Notation

We describe the ring of integers modulo q as $\mathbb{Z}_q$ and the polynomial ring as $R_q(X) = \mathbb{Z}_q[X]/(X^N + 1)$. A $l_1 \times l_2$ matrix over R_q is denoted as $R_q^{l_1 \times l_2}$. Matrices will be written in uppercase bold letters, vectors in lowercase bold, and polynomials in normal letters, e.g., $\mathbf{A}, \boldsymbol{s}$, and v, respectively. For a vector v (or matrix $\mathbf{A}$), we denote by v^T (or $\mathbf{A}^T$) its transpose. The number theoretic transform (NTT) of any element a represented as $\hat{a}$. We denote by $\circ$ the point multiplication.

We denote the sampling x according to a distribution $\mathcal{X}$ as $x \leftarrow \mathcal{X}$, e.g., $x \leftarrow \mathcal{U}$, where $\mathcal{U}$ is the uniform distribution. The binomial distribution with the parameter μ is denoted as β_μ. We can extend the sampling notation into a polynomial matrix sampled by a distribution such as $\mathbf{X} \leftarrow \mathcal{X}(R_q^{l_1 \times l_2})$, where the coefficient of $\mathbf{X}$ is sampled independently by the distribution $\mathcal{X}$.

Three hash functions, $\mathcal{F}, \mathcal{G}$, and $\mathcal{H}$, are instantiated with SHA3-256, SHA3-512, and SHA3-256, respectively. We denote the flooring operation and the rounding operation as $\lfloor \cdot \rfloor$ and $\lfloor \cdot \rceil$, respectively.

2.2 NIST Round 3 Candidates

In the third round of the PQC competition, the selected candidate algorithms are designated as either seven finalist algorithms (4 KEM/PKE and 3 Digital signatures) or eight alternate candidates (5 KEM/PKE and 3 Digital signatures). The finalists are the more likely schemes to be considered for standardization. At the same time, the alternates are schemes advanced into the third round with some, but the very low likelihood of being standardized [NISb]. The candidate algorithms represent multiple categories of cryptographic schemes by their underlying mathematical formulation; 1) code-based, 2) hash-based, 3) lattice-based, and 4) multivariate PKE-based cryptography. Table 1 shows the third-round candidates, their placement, and algorithm categories.

Table 1: NIST PQC competition round 3 candidates.

	Placement	Algorithm	Candidates
Finalist	KEM/PKE	Code-based	Classic McEliece [McE78]
		Lattice-based	CRYSTALS-Kyber [BDK+18]
			NTRU [HPS98]
			Saber [DKRV18]
	Digital Signatures	Lattice-based	CRYSTALS-Dilithium [DKL+18]
			FALCON [FHK+18]
Alternate Candidates	KEM/PKE	Code-based	BIKE [ABB+17]
		Lattice-based	HQC [AMBD+18]
			FrodoKEM [BCD+16]
			NTRU Prime [BCLvV17]
	Digital Signature	Hash-based	SPHINCS+ [BHK+19]
		Multivariate PKE	GeMSS [CFMR+17]

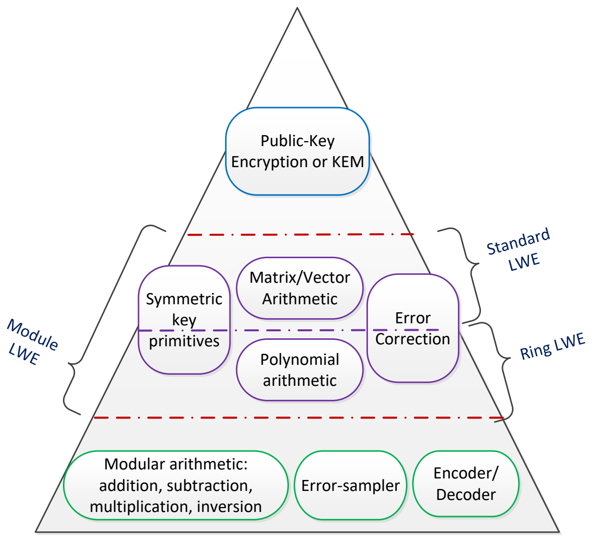


Figure 1: Lattice-based cryptographic implementation hierarchy.

2.3 Lattice-based KEM/Encryption Schemes

A lattice is the set of intersection points in n-dimensional space with a periodic structure (i.e., $L = \{\sum_{i=1}^{n} a_i \boldsymbol{b}_i : a_i \in \mathbb{Z}\}$, where $\boldsymbol{b}_i$ is a basis vector). Several complex mathematical problems are used to construct lattice-based schemes. The two most fundamental problems on lattices are the shortest vector problem (SVP) and the closest vector problem (CVP). Given a basis of a lattice L, SVP asks us to attempt to find the shortest non-zero vector of L. Given a basis of a lattice L and a target vector $\boldsymbol{t}$, CVP asks us to find an element of L closest to the target vector $\boldsymbol{t}$. The lattice-based schemes are based on the three problems such as standard learning with errors (SLWE) problem, ring learning with errors (RLWE) problem, and module learning with errors (MLWE) problem. The RLWE and MLWE are potentially reducible to SVP. The lattice-based cryptographic implementation hierarchy is shown in Figure 1. The SLWE, RLWE, and MLWE are built on the matrix/vector arithmetic, the arithmetic of polynomials, and the polynomial arithmetic and matric/vector arithmetic, respectively. Moreover, symmetric-key primitives (i.e., PRNG, hash functions) and error correction modules are required to build the schemes. The main components (bottom level components) used in lattice-based methods are modular arithmetic, polynomial multiplication, polynomial inversion, error sampler, encoder, and decoder.

In the finalists, 3 KEM/Encryption algorithms (CRYSTAL-Kyber [BDK+18], NTRU

[HPS98], and Saber [DKRV18]) and 2 digital signatures (CRYSTALS-Dilithium [DKL+18] and FALCON [FHK+18]) are based on lattice algorithm (see Table1). In this paper, we mainly focus on lattice-based KEM/PKE schemes such as Saber and CRYSTALS-Kyber. These schemes are fulfilled by the NIST security level 5 (i.e., very high-security level), which is considered as hard to break as an exhaustive key search attack on AES-256. A public-key encryption scheme (PKE) has private and public keys, where we can encrypt a message using the public key and decrypt using the private key. A key encapsulation mechanism (KEM) is a scheme with public and private keys, where we can use the public and private key pair to generate and securely exchange session keys. Specifically, Alice first generates the key pair, keeps the private key, and distributes only her public key. Bob can use Alice's public key to generate a ciphertext c and common secret key K. The ciphertext can now be sent to Alice. Alice uses her private key to decrypt the ciphertext c and generate the common secret key K. Next, we briefly discuss the Saber and CRYSTALS-Kyber.

2.3.1 SABER

It is the third structured lattice-based KEM whose security relies on the hardness of the module learning with rounding problem (MLWR), which is a variant of the LWE problem [DKRV18]. There are three versions of Saber: LightSaber (NIST security level 1: similar to AES-128), Saber (NIST security level 3: similar to AES-192), and FireSaber (NIST security level 5: similar to AES-256). Saber [DKRV18] uses the MLWR problem with both p and q power-of-two to construct an IND-CPA (indistinguishability under chosen-plaintext attack) secure PKE scheme. Following that, an IND-CCA (indistinguishability under chosen-ciphertext attack) secure Saber KEM scheme uses the post-quantum variant of the Fujisaki-Okamoto transformation [HHK17].

Saber PKE: The saber PKE scheme is also composed of three phases; key generation, encryption, and decryption. These operations are built on top of Saber PKE operations. In a *key generation*, the first generates a public matrix of polynomials $\mathbf{A}$ from randomly generating a seed ($seed_{\mathbf{A}}$) and a secret vector of polynomials $\boldsymbol{s}$ from a centered binomial distribution with parameter μ, i.e., $\mathbf{A} \leftarrow \mathcal{U}(R_q^{l\times l}), \boldsymbol{s} \leftarrow \beta_\mu(R_q^{l\times 1})$, where q is a power-of-two modulus. Then, it computes the vector $\boldsymbol{b}$ by scaling and rounding the product ($\boldsymbol{b} = \lfloor \mathbf{A}^T \boldsymbol{s} \rceil_p$, where p is a rounding modulus). At last, the public key consists of $\mathbf{A}$ matrix seed ($seed_{\mathbf{A}}$) and $\boldsymbol{b}$, while the secret key consists of the secret vector $\boldsymbol{s}$ ($pk := (seed_{\mathbf{A}}, \boldsymbol{b}), sk := (\boldsymbol{s})$). An *encryption* consists of generating a new secret $\boldsymbol{s}'$ and adding a 256-bit message m to the inner product v' between the public vector $\boldsymbol{b}$ and the new secret $\boldsymbol{s}'$, i.e., $c_m = (v' + h_1 - 2^{\epsilon_p - 1} m \mod p) \gg (\epsilon_p - \epsilon_T) \in R_T$, where $v' = \boldsymbol{b}^T(\boldsymbol{s}' \mod p) \in R_p$. The ciphertext c consists of the encrypted message c_m and the hidden secret $\boldsymbol{b}' = \lfloor \mathbf{A}\boldsymbol{s}' \rceil_p$, i.e., $c := (c_m, \boldsymbol{b}')$. The *decryption operation* takes the private key $\boldsymbol{s}$ and ciphertext c and produces a message m' such as $m' = ((v - 2^{\epsilon_p - \epsilon_T} c_m + h_2) \mod p) \gg (\epsilon_p - 1) \in R_2$, where $v = \boldsymbol{b}'^T(\boldsymbol{s} \mod p) \in R_p$, which is equal to the original message m with high probability.

Saber KEM: The saber KEM scheme consists of a key generation, an encapsulation, and a decapsulation phase. Additionally, it requires three hash functions that model random oracles: $\mathcal{F}, \mathcal{G}$, and $\mathcal{H}$, which are instantiated with SHA3-256, SHA3-512, and SHA3-256, respectively. The *KEM key generation* is similar operation as the PKE key generation to generate the public key ($seed_{\mathbf{A}}, \boldsymbol{b}$) and the secret vector $\boldsymbol{s}$ except for including a hashed public key, $pkh = \mathcal{F}(pk)$, i.e., $pk := (seed_{\mathbf{A}}, \boldsymbol{b}), sk := (\boldsymbol{s}, z, pkh)$, where $z = \mathcal{U}(\{0,1\}^{256})$. The *KEM Encapsulation* is constructed from Saber PKE encryption operation. It takes the public key and produces a common secret key K and ciphertext c. The *Saber KEM decapsulation* algorithm is based on the Saber-PKE encryption and decryption algorithms. It decrypts the ciphertext via Saber-PKE decryption with the private key $\boldsymbol{s}$ and generates the shared secret key K. The decrypted message is re-encrypted to check the integrity of the ciphertext. For further details, one may refer to [DKRV18].

2.3.2 CRYSTALS-KYBER

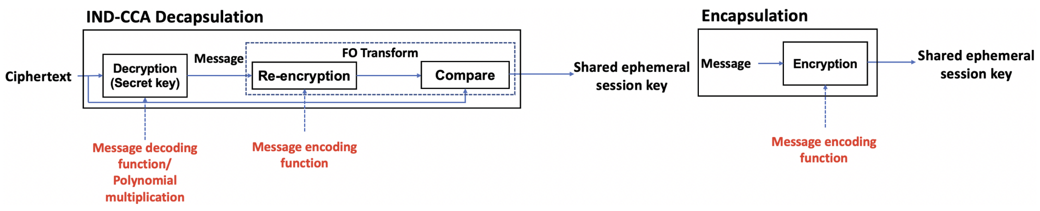
It is a lattice-based PQC whose security is based on the hardness of solving the LWE problem over module lattices (MLWE) [BDK+18]. It follows the conventional construction method to build an IND-CPA PKE scheme firstly and then turns it into an IND-CCA KEM through the tweaked Fujisaki-Okamoto transform. It involves finding a vector $\boldsymbol{s}$ when given a matrix $\boldsymbol{A}$ and a vector $t = \boldsymbol{As} + \boldsymbol{e}$ where $\boldsymbol{e}$ is a small (random masking) error vector which is used to hide the private key. In this scheme, vector-vector and matrix-vector multiplication can be optimized with the fast number-theoretic transform (NTT), which can reduce computational complexity from $O(n^2)$ to roughly $O(n \log n)$. Depending on the security level, Kyber comes in three versions such as Kyber-512 (security roughly equivalent to AES-128), Kyber-768 (security roughly equivalent to AES-192), and Kyber-1024 (security roughly equivalent to AES-256).

Kyber PKE: It is composed of three phases: key generation, encryption, and decryption. In the *key generation phase*, using a random seed, it generates the public matrix parameter $\boldsymbol{A}$ from a uniform distribution and secret vectors $\boldsymbol{s}$ and $\boldsymbol{e}$ sampled from a centered binomial distribution. Then, the LWE instance can be calculated as $\boldsymbol{\hat{t}} = \boldsymbol{\hat{A}} \circ \boldsymbol{\hat{s}} + \boldsymbol{\hat{e}}$ in the NTT domain. In the *encryption phase*, three vectors, $\boldsymbol{s'}$, $\boldsymbol{e1}$, and $\boldsymbol{e2}$, are sampled from uniformly distributed random numbers and centered binomial distributions. The message m to encrypted is first encoded to $m' = enc(m)$. Then, the ciphertext c_1 is calculated as $\boldsymbol{A^T s'} + \boldsymbol{e1}$ while the ciphertext c_2 is formed by embedding the message into an LWE instance as $c_2 = \boldsymbol{t^T s'} + \boldsymbol{e2} + \boldsymbol{m'}$. Then, both ($c1$, c_2) are rounded and published as the ciphertext $ct = (c_1, c_2)$. In the *decryption phase*, decompress (c_1, c_2) and calculate $r = (c_2 - c_1 \mathring{u} s)$, which when decoded as $dec(r)$ yields the message m.

Kyber KEM: It is constructed from Kyber-PKE operations and is consists of three phases; key generation, encapsulation, and decapsulation. First, Alice generates a matrix $\boldsymbol{A}$ and compute vector $\boldsymbol{t}$ by following similar operation as Kyber PKE key generation operation. Then, the seed used to generate a matrix $\boldsymbol{A}$ and the computed $\boldsymbol{t}$ encoded as public key is sent to Bob for encapsulation operation. Further, secret vector $\boldsymbol{s}$ encoded as private key is keeps for decapsulation operation later. In the encapsulation phase, Bob generates the ciphertext c by using Kyber-PKE encryption algorithm. He also computes the shared secret $\boldsymbol{K}$ by using the Alice's public key, message, and the hashed value of the ciphertext. In the decapsulation phase, Alice takes the ciphertext c and the private key $\boldsymbol{s}$ and then generate the message m' by using Kyber-PKE decryption algorithm. Then, she verifies whether it can be encrypted to the same ciphertext sent by Bob by using Kyber-PKE encryption algorithm. If ciphertexts match ($c = c'$), Alice computes the shared secret $\boldsymbol{K}$ by using the ciphertext c, the message, and her public key. Otherwise, she computes the shared secret $\boldsymbol{K}$ by using a random value and the ciphertext c. For more details of the algorithm, one may refer to [BDK+18].

3 Prevailing Side-channel Attacks

The existing lattice-based PKE and KEM schemes are resistant against known mathematical cryptanalytic attacks, such as chosen-plaintext attacks (CPA) and chosen-ciphertext attacks (CCA), respectively. However, they are vulnerable to physical attacks, such as power/EM side-channel attacks or fault injection attacks [RBRC20, XPSR+21, NDGJ21a, BDH+21, SKL+20, AAT+21, PPM17, PP19, KPP20, BFM+18]. It can be categorized into two kinds of side-channel attacks depending on indirect exploitation of side-channel leakages or direct exploitation of them to extract secret keys; 1) algorithmic-level attacks [RBRC20, XPSR+21, NDGJ21a, BDH+21, SKL+20], 2) implementation-level attacks [AAT+21, PPM17, PP19, KPP20, BFM+18]. Both can exploit traditional side-channel attacks, such as simple power/EM side-channel attacks (SPA), differential


IND-CCA Decapsulation
Ciphertext
Decryption (Secret key)
Message
FO Transform
Re-encryption
Compare
Shared ephemeral session key
Message decoding function/ Polynomial multiplication
Message encoding function
Encapsulation
Message
Encryption
Shared ephemeral session key
Message encoding function

(a) Targets of PQC KEM schemes.

	Target function	Attack	Goal
Decapsulation	Message decoding function	Algorithmic-level SCA with CCA	Long-term secret key
	Message encoding function	Algorithmic-level SCA	Shared ephemeral session key
	Polynomial multiplication	Implementation-level SCA	Long-term secret key
Encapsulation	Message encoding function	Algorithmic-level SCA	Shared ephemeral session key

(b) SCA attacks and goals.

Figure 2: Side-channel attacks on PQC KEM schemes.

power/EM side-channel attacks (DPA), correlation power/EM side-channel attacks (CPA), and AI-based side-channel attacks to extract the secret asset. An adversary can target both the decapsulation and encapsulation parties to extract the long-term secret key or the shared ephemeral session key as shown in Figure 2.

3.1 Algorithmic-level Attacks

Power/EM side-channel leakages can assist mathematical cryptanalyses, such as CCAs, even though lattice-based KEMs have indistinguishability under chosen-ciphertext attack (IND-CCA) security by using well known Fujisaki-Okamoto (FO) transform theoretically. In the FO transform scheme of the decapsulation part, an adversary does not have information of decrypted messages without physical attacks so that she/he cannot succeed CCAs. However, the message decoding function with invalidated ciphertexts in the decryption phase can be targeted by power/EM side-channel attacks to extract messages used to reveal the *long-term secret key* by a CCA. A single bit message converted from each coefficient of a polynomial in the message decoding function can be distinguishable by simple power analysis [ADSH18]. Ravi *et al.* recovered the secret key using EM side-channel assisted CCAs with templates to classify a single bit/byte message on NewHope KEM, Kyber KEM, Saber KEM, LAC KEM, and Round5 PKE implemented on an Arm Cortex-M4 core [RBRC20].

In addition, the message encoding function in the encapsulation phase can be targeted to reveal a randomly generated secret message. A *shared ephemeral session key* can be generated by the recovered message and public values. In the CRYSTALS-Kyber algorithm, a bit message is converted into 0x0000 or 0xFFFF by the message encoding function, i.e., when a bit message is 0, the coefficient of the polynomial is encoded into 0x0000, and when a bit message is 1, the coefficient of the polynomial is encoded into 0xFFFF. Each message can be observed through power/EM signatures since power consumption depends on the Hamming weight of the encoded coefficient. This attack scenario can use only a single trace since the target message is randomly generated every time. Sim *et al.* achieved a 100% single-trace attack to recover the message on CRYSTALS-Kyber and Saber SW implementations on an Arm Cortex-M4 core (ST32F3) [SKL+20].

Table 2: Side-channel Attacks on Lattice-based PQC.

Work	PQC	Implementations	Target	Leakage	Method	# Query	Success rate
[RBRC20]	NewHope, Kyber Saber, Round5 LAC	ARM CM4	Message Decoding	EM	SPA	490	100 %
[ACC+21]	Saber	ARM CM4	Polynomial multiplication	Power	TVLA	-	$\|t\| > 4.5$
[BDK+21]	Saber	ARM CM4	`A2A, Keccak-f, SecBitSlicedSampler`	Power	TVLA	10,000	$\|t\| > 4.5$
[SPH21]	KYBER	ARM CM4	Barrett reduction	Power	CPA	12	100%
[XPSR+21]	KYBER	ARM CM4	Inverse NTT	EM	SPA	960	100%
[SKB21]	KYBER	ARM CM4	Message encoding	Power	SPA	525	68.6%
[KdG21]	KYBER	ARM CM4	Polynomial multiplication	Power	CPA	200	100%
[PH16]	NTRU	XMEGA128D4	Modular addition	Power	SPA	1	100%
[HCY19]	NTRU Prime	ARM CM4	Polynomial multiplication	Power	HICPA	1	100%
[AR21]	NTRU	ARM CM4	Modular reduction (mod3())	EM	SPA	1	75%
[SKL+20]	KYBER	ARM CM4	Message encoding	Power	MLP	500	100%
[SKL+20]	Saber	ARM CM4	Message encoding	Power	MLP	10,000	100%
[SKL+20]	FrodoKEM	ARM CM4	Message encoding	Power	MLP	10,000	79%
[NDGJ21b]	Saber	ARM CM4	`POLY2MSG`	Power	MLP	100,000	97.4%
[NDJ21]	Saber	ARM CM4	`Poly_A2A`	Power	Ensembled MLP	7,800	100%
[AKP+20]	Frodo & NewHope	Xilinx Spartan-6	Polynomial Multiplication	Power	CNN	-	100%
[KAP+20]	Frodo	Xilinx Spartan-6	Polynomial Multiplication	Power	2-D CNN on images	2,200	100%
[KAP+20]	NewHope	Xilinx Spartan-6	Polynomial Multiplication	Power	2-D CNN on images	3,300	100%

3.2 Implementation-level Attacks

Adversaries can target some functions directly related to the secret key in lattice-based KEM/PKE algorithms such as polynomial/matrix-vector multiplication, error/secret sampling, or extendable output function using SHAKE to extract the secret key without the assistance of cryptanalytic methods [AAT+21, PPM17, PP19, KPP20, BFM+18]. For example, an input of the polynomial multiplication is the secret key in the lattice-based KEM/PKE algorithm. Power/EM signatures depend on the intermediate values of the polynomial multiplication. This side-channel information can be used to reveal the secret key for horizontal differential power analysis (DPA) attacks, template attacks, or AI-based side-channel attacks. Since a single trace includes several multiplications of a target subkey, single-trace key recovery is possible by exploiting sub-traces corresponding to target multiplications in a single trace. Primas *et al.* [PPM17] performed successful single-trace template attacks on number theoretic transform (NTT) in the ring lattice learning-with-error (LWE) decryption phase to extract the entire private key. Recently, Pessl and Primas [PP19] successfully demonstrated an advanced single-trace attack on NTT of CRYSTALS-Kyber software implementation running on an Arm Cortex-M4 microcontroller. Table 2 summarizes existing side-channel attacks on various PQC implementations with the target operations, leakage sources, the type of SCAs, and the required number of traces to satisfy a success rate.

3.3 AI-based side-channel Attacks

Researchers have recently become interested in an AI-based side-channel attack because of its potential to compromise secure hardware implementation. Few papers use AI-based methodologies to undertake side-channel attacks against software and/or hardware implementations of various PQC algorithms in the literature.

Using the deep learning model, Bo-Yeaon *et al.* [SKL+20] attempted side-channel attacks on Saber, Kyber, and FrodoKEM. In this study, point of interests (POIs) were first detected using a sum of squared pairwise t-differences (SOST) value of collected power traces, and then the message was recovered using a multi-layer perceptron model (MLP). One of the work's shortcomings is that the technique was evaluated on unprotected software implementations. Ngo *et al.* executed a message recovery attack on Saber using an MLP architecture. The authors successfully recovered both the session key and the long-term secret key from few traces by performing experiments on three separate devices [NDGJ21b]. In a separate study, Ngo *et al.* applied ensembled deep learning networks to attack the software version of CCA secure Saber KEM secured by first-order masking and shuffle. Aydin *et al.* [AKP+20] demonstrated the capability of the CNN by

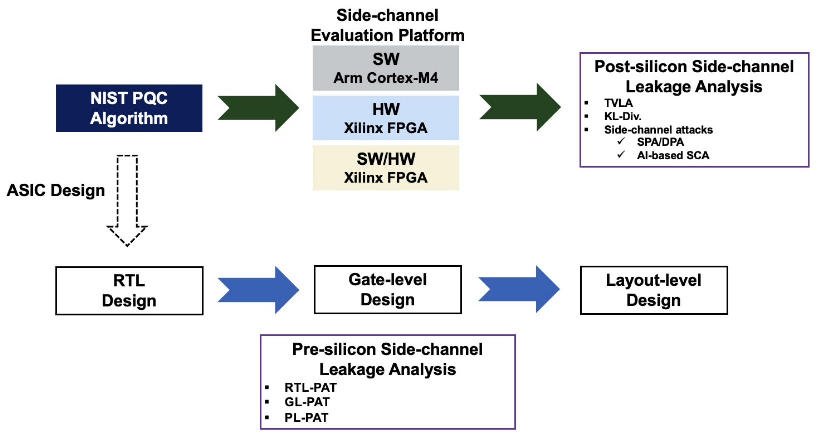


Figure 3: PQC side-channel evaluation platform.

attacking hardware implementations of the Frodo and NewHope protocols with a single trace, as well as demonstrating that traditional attacks such horizontal TA and DPA were outperformed by AI-based attacks by up to 25% and 900%, respectively. In a separate study, 1-D time-series power measurement data was converted into 2-D pictures, and DL techniques were performed to the SCA images [KAP+20]. The results of such attacks on the hardware implementation of NewHope and Frodo demonstrated their superiority over traditional tactics. Table 2 summarizes prevailing side-channel attacks on Lattice-based PQC protocols, including the target implementation, data set size, SCA method, and success rate.

4 PQC Side-channel Evaluation Platform (SEP)

In recent years, the area of PQC hardware security and trust has seen vastly increasing research activity. A large population of academic and industry researchers has been working on various aspects of securing a cryptographic module from power/EM side-channel analysis attacks. Most research in this area is still carried out in an ad-hoc fashion without the help of well-established metrics, test methods, and EDA tools. Although semiconductor companies are becoming increasingly aware of the requirement of automatic SCA resistance analysis and protection against it, a systematic framework to accomplish these goals is notably lacking in the industry. In this section, for the first time, we propose to develop a systematic framework for comprehensive side-channel evaluation of NIST PQC implementations during design phases (RTL, gate level, and layout level) that can be seamlessly integrated into the existing design flow as well as post-silicon validation shown in Figure 3.

4.1 Pre-silicon Side-channel Leakage Assessment

Power side-channel assessment of NIST PQC designs starts from early design phase and move coherently forward to different levels of design abstractions (RTL $\rightarrow$ gate level $\rightarrow$ layout level) using our power side-channel analysis tools (PAT).

4.1.1 RTL-PAT

At the RTL phase, vulnerable modules in the top-level design of the NIST PQC can be searched by estimating a statistical distance between two sets of switching activities in each module. For example, randomly generated input vectors given a secret key are used to make a set of switching activities. A statistical distance of two groups corresponding to two different secret keys is calculated based on KL divergence [SR51]. If the KL divergence of any module is larger than a pre-defined threshold value, the module is determined as a vulnerable module. The selected vulnerable modules can be replaced into secure enhanced modules at the early design phase. The secure RTL design will be evaluated repeatedly until passing the test, and then it will move forward to gate-level design abstraction.

4.1.2 GL-PAT

After logic synthesis, a gate-level netlist includes timing delay of gates and flip-flops so that switching activities of all nodes of gates and flip-flops will be simulated in fine-grained time scale, which causes infeasible analysis of the top-level design due to tremendous simulation time or a huge amount of memory requirements. To address this issue, gate-level assessments focus on only vulnerable modules selected at the previous design phase, and vulnerable gates and flip-flops in the vulnerable modules will be ranked. By continuously reducing target areas for the assessments according to design abstractions levels, the analysis can be manageable and efficient. The vulnerable cells will be replaced to secure primitives, such as masked logic gates, wave dynamic differential logic (WDDL) [TV04], or t-private logic circuits [ISW03]. The secure gate-level modules will be evaluated to verify side-channel robustness.

4.1.3 PL-PAT

After designing a power distribution network, synthesizing clock trees, placing standard cells, and routing wires, dynamic power estimation can be estimated accurately due to the physical information of each cell by which capacitance and resistance of all nodes in the target cells can be calculated. Dynamic power or current of vulnerable cells will be simulated using commercial power estimation tools such as Cadence Voltus [Cad21] to assess side-channel leakages. Even if secure cells or primitives in the previous design phase can mitigate side-channel leakages, it may not be enough to pass the layout-level side-channel leakage assessment due to broken power balance after placement and routing. In this case, inserting decaps, isolating the power network of vulnerable cells, or random frequency clock may be helpful to make data-dependent dynamic power balanced or randomized. Finally, a physically enhanced secure layout design will be evaluated. Figure 4 shows RTL-PAT, GL-PAT, and PL-PAT tools which can be incorporated in Cadence/Synopsys ASIC design flow. For simulation, randomly generated input vectors or specific input vectors by PSC-TG [ZPTF21, NHPT21] which can significantly reduce the required number of input vectors for accurate assessment are used. All switching activities are stored in VCD files during the evaluations. Results of the evaluation at each design phase will be described in terms of KL divergence or new AI-based metrics.

This pre-silicon side-channel leakage approach has some features as follows:

- **Various design-level analysis**: Early design estimation gives flexibility to modify the design and all design-level analysis can measure contribution of various countermeasures suitable to each design abstraction to mitigate side-channel leakages.
- **Scalability**: Due to significant performance improvement, the PAT tools can analyze an SoC design, including PQC IPs, small standalone crypto IPs, and microcontrollers, or large-scale designs such as PQC implementations which take thousands of clock cycles to finish.
- **Accuracy**: Powerful AI-based approaches as well as traditional statistical approaches can improve the accuracy of side-channel leakage estimation.

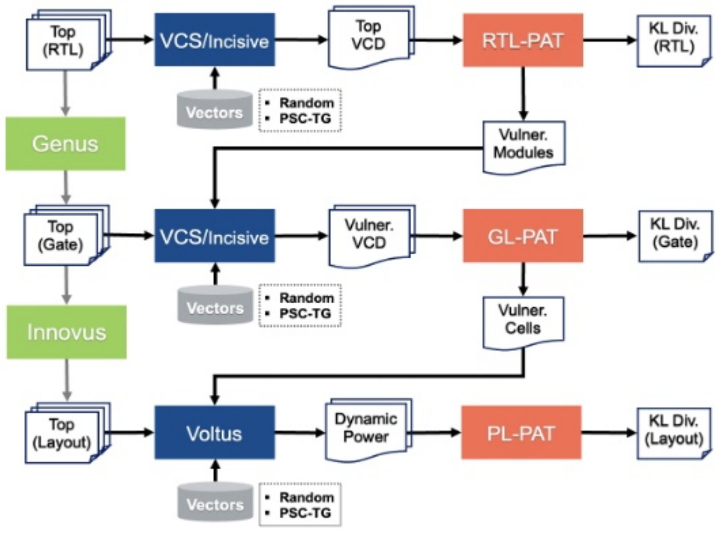


Figure 4: Pre-silicon side-channel leakage CAD tool with ASIC design flow; RTL-PAT, GL-PAT, and PL-PAT.

- **Efficiency**: By reducing target area according to the level of design abstraction and the number of input vectors based on PSC-TG method, efficient assessments are possible.
- **Generality**: This framework can be used to analyze different cryptographic algorithms without any customization to analyze for the leakage.

4.2 Post-silicon Side-channel Leakage Assessment

For measuring power or EM side-channel leakages of NIST PQC implementations, our side-channel evaluation FPGA platform in Figure 5 is used. All SW, HW, and SW/HW co-design of NIST PQC algorithm can be implemented on our customized Xilinx Kintex-7 FPGA/Xilinx Artix-7 FPGA board and collecting power traces, and statistical t-statistic or KL-divergence tests and SCA attacks are executed automatically in an assessment flow without additional tools. These assessment results are compared to pre-silicon side-channel leakages so that the accuracy of pre-silicon side-channel estimation can be analyzed, and it can be scrutinized how much contribution countermeasures at each level of design abstraction have to improve side-channel mitigation of the final implementation. Our SCA FPGA evaluation platform consists of three major parts as follows:

- **FPGA boards:** We can use our customized Xilinx Kintex-7 board or ChipWhisperer Artix-7 board to implement PQC designs. Since an Arm Cortex-M4 based SoC design is feasible on the customized FPGA board, we can evaluate SW, HW, and SW/HW PQC implementations using the board. The ChipWhisperer Artix-7 FPGA board [Incc] is generally used in academic and industry research areas. To compare with other research group works, we can use the ChipWhisperer board. However, it is not suitable for a large-scale SoC design due to the limited resource of the Xilinx Artix-7 FPGA.[1]

  The power analysis attack is mounted by measuring the voltage across the shunt resistor placed in between the wire connecting the power supply to the voltage pin of the target FPGA. Both FPGA boards also have an operational amplifier that will

[1]Xillinx Artix-7 XC7A100T has 103K logic cells, 4.8Mb BRAMs, and 240 DSP slices.

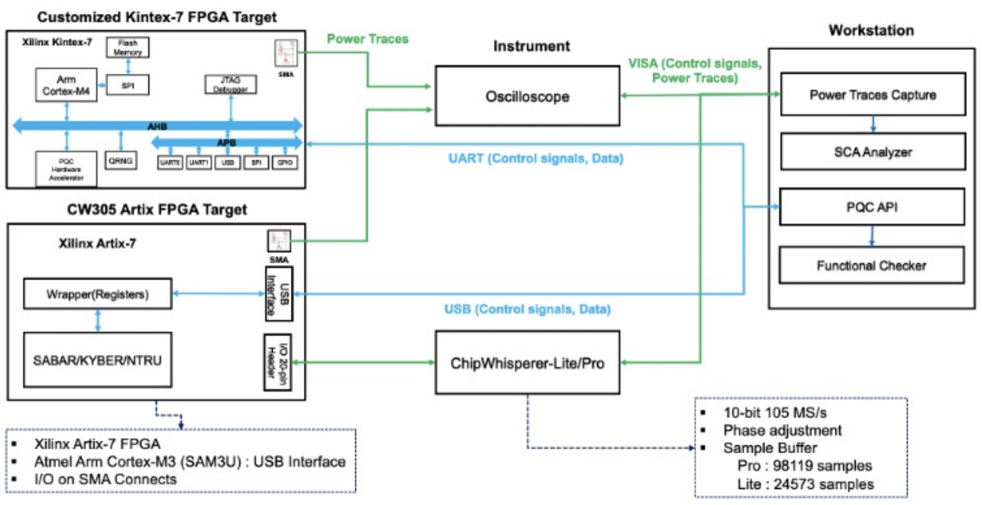


Figure 5: Side-channel evaluation FPGA platform.

amplify the value of the power signal so that it is readable by the capture board or the oscilloscope. Figure 6 shows both FPGA target boards. Specially, a quantum random number generator (QRNG) [PCL+19] is installed to generate high-entropy random numbers in the customized FPGA board.

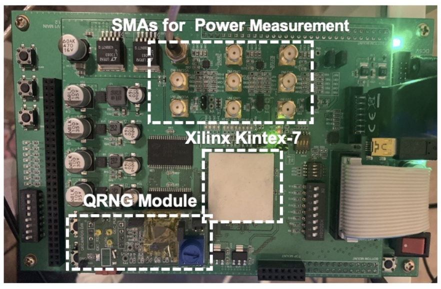
- **Measurement instruments:** For collecting power traces, we use either a Tektronix oscilloscope with 1GHz bandwidth and 5Gs/s sampling rate (MDO3102) or the NewAE ChipWhisperer-Pro/Lite [Incb, Inca] designed to conduct power side-channel analysis. The NewAE ChipWhisperer-Pro and Lite capture boards have a 10-bit analog-to-digital converter (ADC) with 105 MS/s but have different sizes of buffers to store 98119 samples and 24573 samples, respectively. Both can be connected via an SMA connector on the target board and a USB port to communicate with the workstation shown in Figure 5.

- **Workstation:** The workstation sends commands to control target boards such as start or stop operations and data such as plaintexts or random numbers and receives data such as ciphertexts or generated keys though a UART or USB channels. PQC APIs are developed to execute the PQC algorithm in an ARM Cortex-M4 core or an FPGA-based hardware design. A functional checker is installed to verify the functionality of the PQC implementations by comparing simulation results of the verified reference design in the workstation (see Figure 7). In addition, a power trace capture tool collects received data from the instruments, and a power side-channel analyzer performs power side-channel leakage assessment and side-channel attacks.

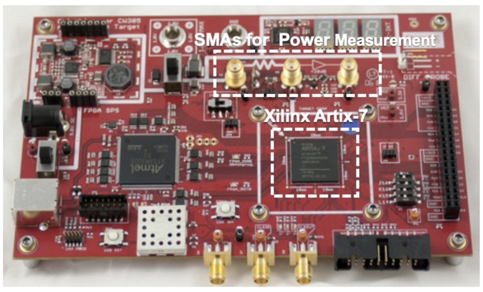
4.2.1 Leakage Assessment

To evaluate side-channel leakage of PQC implementations, generally used statistical methods such as test vector leakage assessment (TVLA) [CDG+13] or KL-divergence test are significantly exploited.

TVLA methodology: It is based on Welch's t-test which is used to test the hypothesis that two populations have equal means when two samples have unequal variance and unequal sample size. In the side-channel evaluation process, n power side-channel measurements are collected while the device under the test operates with an asset. The n measurements, $\boldsymbol{p}^i = [p_0^i, \ldots, p_{m-1}^i]$ for $i = 1, \ldots, n$, where m is the number of the sampling



(a) Customized FPGA board with a QRNG.



(b) NewAE ChipWhisperer CW305 FPGA board.

Figure 6: Target FPGA boards.

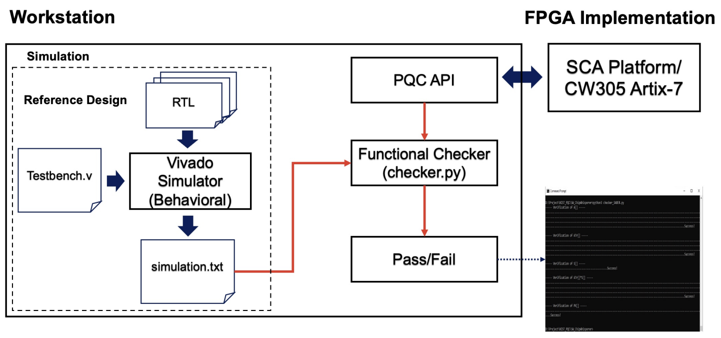


Figure 7: Functional verification of PQC implementations.

points, are classified into two sets by the determinant function D: $S_0 = \{\boldsymbol{p}^i | D = 0\}$ and $S_1 = \{\boldsymbol{p}^i | D = 1\}$. If the t-test statistic,

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{N_0} + \frac{\sigma_1^2}{N_1}}} \tag{1}$$

is out of the confidence interval, $|t| > C$, the null hypothesis, $H_o : \mu_0 = \mu_1$ is rejected, which means that two groups are distinguishable, and the implementation has a high probability of leaking information. Thus, it does not pass the leakage assessment test. In our experiment, the threshold value C is chosen as 4.5, which leads to a confidence of > 0.99999 to reject the null hypothesis.

KL divergence: Even though we can determine if the PQC implementation is vulnerable based on the TVLA test, the t-test statistic cannot estimate how much resistance or vulnerability the implement has against side-channel attacks. In order to quantify side-channel leakage, KL divergence between two sets can be exploited. Let $f_0(p)$ and $f_1(p)$ be the probability density functions of two measurement sets S_0 and S_1, respectively. KL divergence is defined as the following equation [SR51]:

$$D_{KL}(S_0||S_1) = \int f_0(p) \log \frac{f_0(p)}{f_1(p)} dp. \tag{2}$$

If S_0 and S_1 are the normal distribution with μ_0, σ_0^2 and μ_1, σ_1^2, respectively, then

$$D_{KL}(S_0||S_1) = \left\{(\mu_0 - \mu_1)^2 + \sigma_0^2 - \sigma_1^2\right\} / (2\sigma_1^2) + \ln(\sigma_1/\sigma_0). \tag{3}$$

Also, KL divergence is related to the number of traces N necessary to assert with a confidence of $(1-\alpha)$ that the two normal distributions S_0 and S_1 are different. The number of traces N is a significant contributor in quantifying a lower bound on the attack complexity. The smallest number of traces to satisfy that $\Pr\left[|\bar{S_0} - \bar{S_1} - (\mu_0 - \mu_1)| < \epsilon\right] = (1-\alpha)$ is

$$N \geq \frac{(\sigma_0 + \sigma_1)^2}{\epsilon^2(\mu_0 - \mu_1)^2} \cdot z^2_{1-\alpha/2}, \tag{4}$$

where the quantile $z_{1-\alpha/2}$ of the standard normal distribution has the property that $\Pr\left[Z \leq z_{1-\alpha/2}\right] = 1 - \alpha/2$. Comparing to Eq. (3), as KL divergence of two random variables increases, the number of traces N decreases.

4.3 AI-based SCA Attacks

Side-channel analysis has seen remarkable growth in the last five years because of the inclusion of machine learning and deep learning-based approaches for attacks and vulnerability assessment. In this section, we first discuss the road map for AI-based side-channel analysis. Later, we describe a framework for AI-based side-channel attacks using signal decomposition. Finally, we narrate the training scheme and model setup we used in the experiments for the framework.

4.3.1 Roadmap for AI-based Side-channel Analysis

The overview of an AI-based profiling side-channel attack is shown in Figure 8. The collected power trace is first passed through the *data pre-processing* step. In this stage, the collected data passes through one or many algorithms from a pool of data pre-processing techniques such as data augmentation, data standardization, noise reduction, etc. Data normalization through normalization or standardization is one of the most common practices for data pre-processing. Noise cancellation from data using signal processing techniques or auto-encoder neural networks can also be applied for pre-processing the power traces. In addition, data augmentation can increase the quality of an attack by introducing new data observations in the training set and reducing overfitting.

Feature engineering is the next step that extracts features from the pre-processed raw power traces. In several AI-based side-channel attacks, feature engineering is considered an optional step. In these cases, features are extracted from the neural network end-to-end. Additionally, signal decomposition techniques, such as empirical mode decomposition (EMD) [HSL+98], Hilbert vibration decomposition (HVD) [Fel06], variational mode decomposition (VMD) [DZ14], and other methods, can be applied. Such decomposition techniques depict the intrinsic features unseen in raw signals. Notably, EMD can be used as a denoising step to remove unnecessary information from the power traces. 2D transformation of 1-D data can be an effective way of adding new discriminative features. SCA images generated through 2-D transformation algorithms such as Gramian angular difference field (GADF), Gramian angular difference field (GADF), Markov transition field (MTF) [WO15], recurrence plots [EKR87], spectrogram, and scalogram can improve the quality of side-channel attack at the cost of computational complexity.

After feature engineering, *selecting an effective algorithm* is a crucial step. Deep neural networks such as convolution neural networks (CNNs) or multilayer perceptron (MLP) are used in most AI-based approaches. The performance of an attack largely depends on challenging tasks such as the model training and selection of appropriate hyperparameters. Automated hyperparameter tuning through Bayesian optimization or reinforcement learning (RL) is a viable way to adjust these hyperparameters properly. At the *attack* phase, the quality of attack is measured through different SCA evaluation metrics that differ from traditional machine learning evaluation metrics.

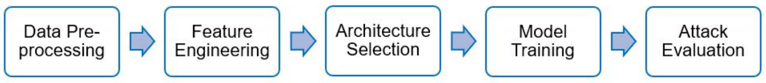


Figure 8: Major steps involved to perform an AI-based side-channel attack.

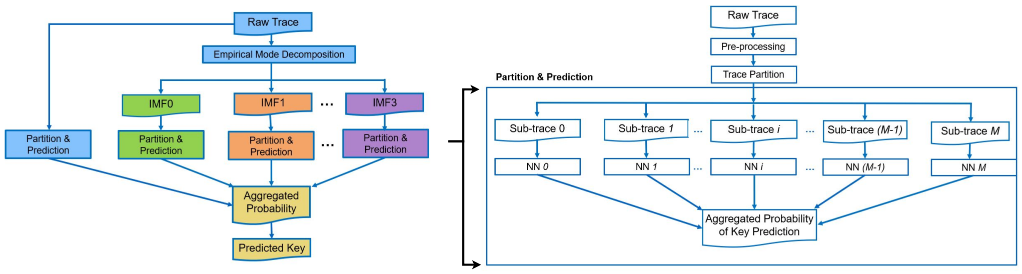


Figure 9: Overview of the generic framework of deep learning-based side-channel attack using signal decomposition.

4.3.2 DL-based framework using signal decomposition

In this work, we propose a generic framework of deep learning-assisted side-channel attack as shown in Figure 9. The framework includes signal decomposition technique, enabling of neural networks, and automated hyperparameter tuning. From the figure 9, it can be seen that the raw power signal and decomposed signals generated from raw traces are processed from a step named 'Partition and Prediction'. Figure 9 shows the partition and prediction approach of the methodology when the input is the raw trace. In this stage, at first, each trace is pre-processed and then partitioned into multiple sub-traces, and for each sub-trace, a separate neural network is trained at the profiling stage. The choice of these neural networks is decided through automated hyperparameter tuning. In the attack phase, key probabilities of each sub-trace of both raw and decomposed signals are summed up. Finally, the secret key is recovered from this aggregated key probability. As a signal decomposition technique, empirical mode decomposition is used. EMD [HSL$^+$98] is a recursive observation-based approach for decomposing a signal into numerous intrinsic mode functions (IMFs). EMD uses the signal's rhythmic activity at the local level to expose intrinsic properties, unlike Fourier Transforms and wavelet decomposition. Each IMF's envelopes function as a fundamental oscillatory mode, with an equal number of extrema and zero crossings. This work sets the maximum number of iterations per single sifting at 1000 while deconstructing a trace using EMD. The energy ratio and standard deviation threshold values are set at 0.2 as a termination condition in each IMF examination. The total power per EMD decomposition threshold value is set at 0.005.

Like other deep learning-based side-channel attack methodologies, the approach is performed in two stages; profiling and attack. Algorithm 1 and algorithm 2 outline the profiling and attack stage of the method, respectively. This method targets the matrix multiplication operation of Saber. As seen in Algorithm 1, in the profiling stage, post-silicon traces and corresponding key values are available to train neural network architecture. The profiling stage starts with decomposing the raw traces into $N+1$ distinct modes using EMD. The algorithm takes the raw trace and splits it into $M+1$ sub-traces in the next step. Each sub-trace contains an equal number of samples. Afterward, sub-traces of the same order are selected to construct a dataset, where the corresponding labels are the key values. Normalization is used to pre-process the data. Next, the entire dataset of the sub-traces is split into the train, validation, and test sets. Before training a neural network, the most suitable hyperparameters are selected through Bayesian optimization (BO)-based

Algorithm 1 Profiling Stage

```
1: **Input:** Raw ($t_r^i \in \mathcal{T}_r$, $i = 1, 2, 3, ..., n$), key ($k^i \in \mathcal{K}$, $i = 1, 2, 3, ..., n$)
2: **Output:** Trained $\mathcal{NN}$
3: Apply signal decomposition on raw trace $\mathcal{T}_r$ and derive $\mathcal{T}_r^j$, $j = 0, 1, 2, ..., N$
4: Take raw trace
5: Apply partition so that $t_r^i = \{t_{r,0}^i, t_{r,1}^i, ... \; ... \; ..., t_{r,M}^i\}$
6: for $j$=0:$M$ do
7:     Construct dataset, $\mathcal{D}_j = (\mathcal{T}_{r,j}, \mathcal{L})$
8:     Apply pre-processing on $\mathcal{T}_{r,j}$
9:     Split the dataset $\mathcal{D}_j$ into $\mathcal{D}_{j,train}$, $\mathcal{D}_{j,val}$, $\mathcal{D}_{j,test}$
10:    Select hyperparameters using automated hyperparameter tuning
11:    Train $\mathcal{NN}_j$ on $\mathcal{D}_j$
12: return Trained $\mathcal{NN}_j$
13: for $l$=0:$N$ do
14:    Take($\mathcal{T}_l$,$\mathcal{L}$)
15:    Repeat lines 5-12 on dataset ($\mathcal{T}_l$,$\mathcal{L}$)
16: end for
```

automated hyperparameter tuning [Nog14]. After choosing the hyperparameters, a separate neural network architecture is trained for each dataset of sub-trace coming from raw traces. Later, the same process is repeated for all orders of sub-traces. Consequently, $M + 1$ trained neural networks are obtained. Such steps of trace partition, dataset construction, automated hyperparameter tuning, and neural network training, described in Lines 5-12, are repeated for other decomposed traces. In this way, the profiling stage returns $(N + 2)(M + 1)$ neural networks altogether.

Algorithm 2 Attack Stage

```
1: **Input:** Raw trace $t_r^{test}$, Trained $\mathcal{NN}$
2: **Output:** Predicted key, $k_p$
3: for $j = 0 : M$ do
4:     Calculate key probability $p_{k,j}$ for sub-trace $t_{r,j}^{test}$
5: Calculate key probability $p_{r,k}$ for trace $t_r^{test}$
6: $p_{r,k} = \sum_{j=0}^{M} p_{k,j}$
7: for $l$=0:$N$ do
8:     determine $t_l^{test}$
9:     Repeat Lines 3-6 and calculate $p_{l,k}$
10: end for
11: Calculate aggregated key probability $S_k$ for key k
12: $S_k = p_{r,k} + \sum_{l=0}^{N} p_{l,k}$
13: Determine the predicted key, $k_p = \underset{k}{\operatorname{argmax}} S(k)$
```

The attack algorithm of the methodology is described in Algorithm 2. In the first step, the raw trace is partitioned into $M + 1$ number of sub-traces and pre-processed. Using the previously trained neural networks, the key probability for each sub-trace is calculated, described in Lines 3-4. The key probability for the whole raw trace is calculated by summing up the individual probabilities. Next, the decomposed signals are generated from the raw trace using the selected decomposition technique named EMD. The key probability for each sub-trace of each decomposed signal is measured following the steps described in Lines 7-10. Finally, the aggregated key probability is calculated, and the key is predicted from the probability.

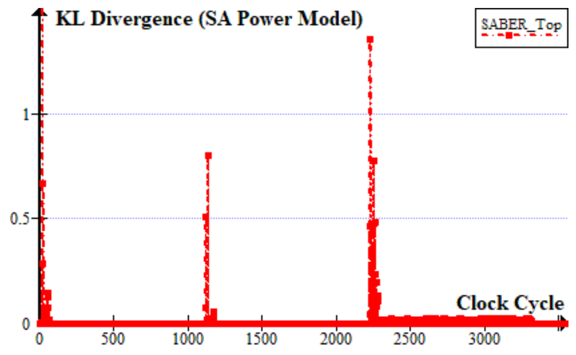


Figure 10: Total design KL analysis during Saber decryption.

4.3.3 Training Scheme

In this work, MLP is used as the network architecture. Each MLP architecture contains multiple fully connected layers with SeLU activation functions to solve non-linear complexity. Each hidden layer is followed by batch normalization to fasten the learning and work as a regularizer. Dropout is used between these hidden layers to prevent overfitting. The number of layers and number of neurons in each layer is decided through the implementation of Bayesian optimization (BO) based automated hyperparameter tuning [Nog14]. This automated process searches for the best hyperparameter by maximizing the specified objective function in a minimum number of iterations through exploration and exploitation approach. In order to find the best model architecture, we set the percentage of neurons at the initial layer and the percentage of neuron shrink in the subsequent layers as the hyperparameters. In this study, 10 steps of random exploration are performed with 100 iterations.

The full dataset is split into training, validation, and test sets for the experiments using 80%, 10%, and 20%, respectively. We use mini-batch optimization. RMSprop algorithm is used for optimization, and categorical cross-entropy is used as an objective function. We find the batch size and learning rate values through automated hyperparameter searching by defining a search space.

5 Experimental Results

5.1 Pre-silicon Side-channel Analysis Results

In this section, we present the efficacy of RTL-PAT to assess a Saber design [RB20] at the RTL, which can evaluate the individual modules of the Saber design for PSC leakage. Saber is a full co-processor hardware implementation of PQC based on learning with rounding (LWR). The regular submodules, i.e., `polynomial multiplier` (using quadratic-complexity schoolbook algorithm), `SHA3/SHAKE128`, `binomial sampler`, `Addpacks`, `AddRound`, `Verify`, `CMOV`, and `CopyWords`, are connected with the data memory (block memory) and program memory. Program memory controls the instruction execution, whereas data memory provides the needed data to the modules. In the RTL simulation, the random seed required by the Saber is provided through the SystemVerilog's internal *urandom* command. Saber uses the power of two for the polynomial multiplication, removing the need for modular reduction. In this implementation, the 256-bit polynomial multiplication is done in parallel. For more details on each sub-process of algorithm and the hardware implementation, refer to [RB20].

During encryption, a secret key is only used once during "Key Generation" from the random seed, making it very difficult for an attacker to extract the key from a single

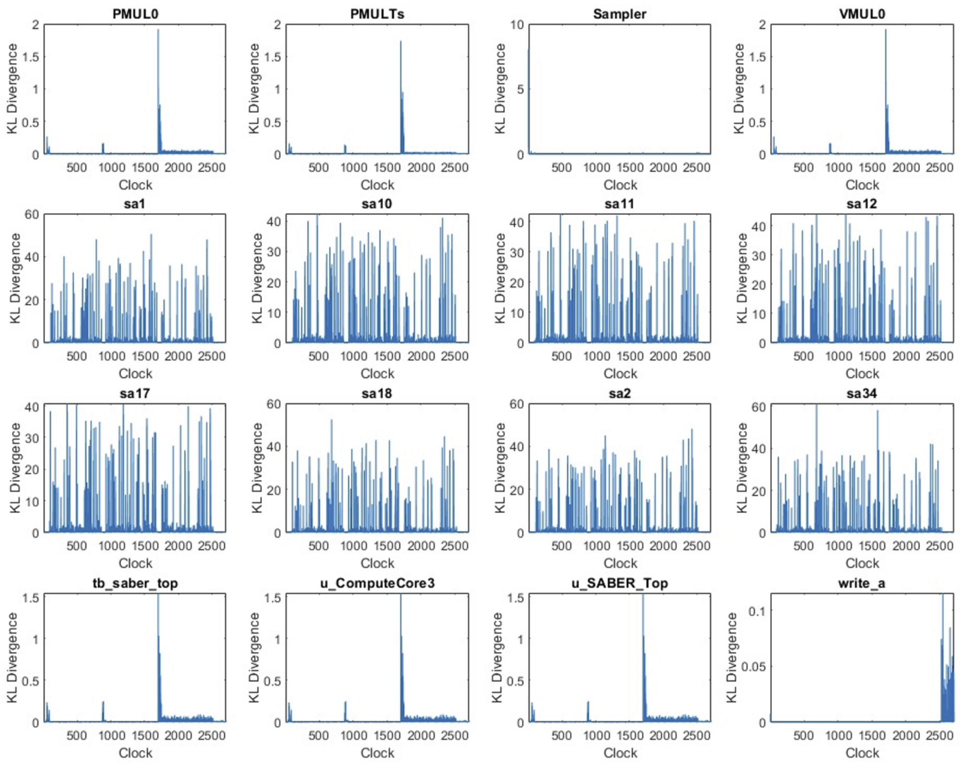
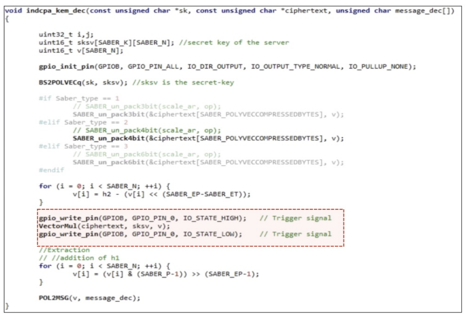


Figure 11: Modular KL analysis during Saber decryption.

power trace. Therefore, for power side-channel analysis in Saber, we target the decryption process of the implementation. We decrypt 1000 random ciphertexts for two different keys, and it takes more than 3K clock cycles to finish one decryption. Figure 10 shows the KL divergence analysis of the top module. It shows that the design is leaky at various places. The high peaks in the figure correspond to reading/writing data from/to the RAM. Figure 11 shows the modular KL divergence analysis results of the leaky modules. It is clear that modules related to multiplication, i.e., vector-multiplication (`VMUL`) and polynomial-256-multiplication (`PMUL`) consisting of schoolbook-based multipliers (`sa`), are the leakiest. Their KL divergence graphs are very similar to the total design leakage. Besides multiplication, the binomial sampler module also shows leakiness used to sample secret coefficients from a centered binomial distribution. `U_computeCore3` module is just a wrapper to handle the calling of different modules throughout the decryption. Based on RTL-PAT, we can identify vulnerable modules, such as `sa`, `VMUL`, and `PMUL`, in the Saber HW design. The lower-level PAT tools, i.e., GL-PAT and PL-PAT, will focus on the vulnerable modules activated by specific input vectors using PSC-TG [ZPTF21]. These lower-level analyses and other design's analyses are out of scope in this paper.

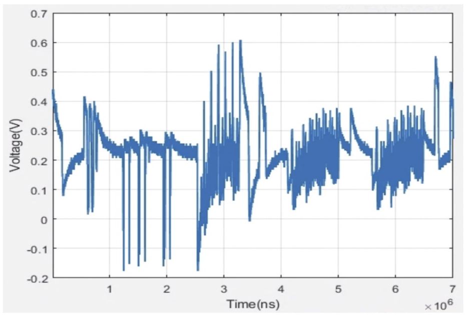
5.2 Post-silicon Side-channel Analysis Results

In this section, we describe the experimental platforms and implementations used for our experiments. We are currently evaluating both software and hardware implementations of target PQC KEM candidates such as Saber and KYBER.



```
void indcpa_kem_dec(const unsigned char *sk, const unsigned char *ciphertext, unsigned char message_dec[])
{

        uint32_t i,j;
        uint16_t sksv[SABER_K][SABER_N]; //secret key of the server
        uint16_t v[SABER_N];

        gpio_init_pin(GPIOB, GPIO_PIN_ALL, IO_DIR_OUTPUT, IO_OUTPUT_TYPE_NORMAL, IO_PULLUP_NONE);

        BS2POLVECq(sk, sksv); //sksv is the secret-key

        #if Saber_type == 1
                // SABER_un_pack3bit(scale_ar, op);
                SABER_un_pack3bit(&ciphertext[SABER_POLYVECCOMPRESSEDBYTES], v);
        #elif Saber_type == 2
                // SABER_un_pack4bit(scale_ar, op);
                SABER_un_pack4bit(&ciphertext[SABER_POLYVECCOMPRESSEDBYTES], v);
        #elif Saber_type == 3
                // SABER_un_pack6bit(scale_ar, op);
                SABER_un_pack6bit(&ciphertext[SABER_POLYVECCOMPRESSEDBYTES], v);
        #endif

        for (i = 0; i < SABER_N; ++i) {
                v[i] = h2 - (v[i] << (SABER_EP-SABER_ET));
        }

        gpio_write_pin(GPIOB, GPIO_PIN_0, IO_STATE_HIGH);   // Trigger signal
        VectorMul(ciphertext, sksv, v);
        gpio_write_pin(GPIOB, GPIO_PIN_0, IO_STATE_LOW);    // Trigger signal
        //Extraction
        // //addition of h1
        for (i = 0; i < SABER_N; ++i) {
                v[i] = (v[i] & (SABER_P-1)) >> (SABER_EP-1);
        }

        POL2MSG(v, message_dec);
}
```

(a) C code to generate trigger signals.



(b) Power trace of `VectorMul` function.

Figure 12: C code snippet for collecting power traces and a measured power trace.

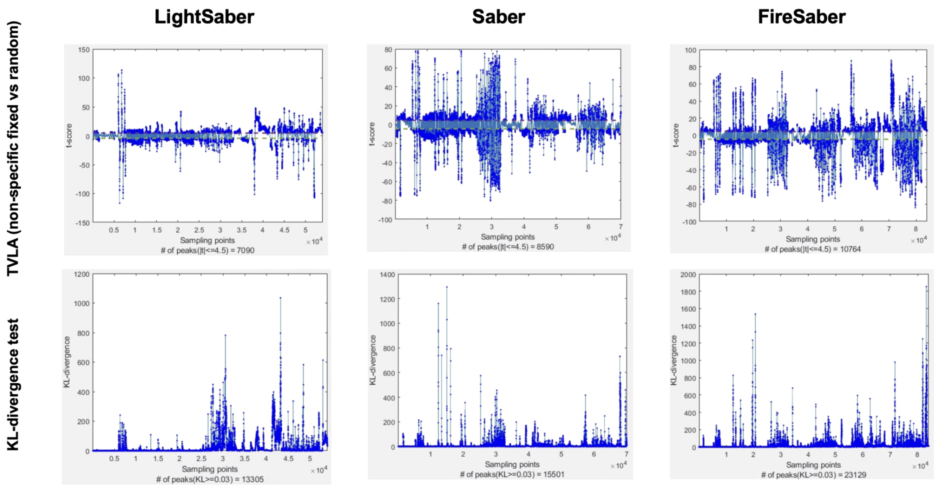


Figure 13: Side-channel leakage assessment of Saber SW implementation.

5.2.1 SW Implementations

We use our side-channel evaluation FPGA platform to access side-channel leakages of SW implementations on an ARM Cortex-M4 core as mentioned at Section 4.2. The SEGGER J-link EDU [SEG] and embedded studio are used for debugging/probing and programming the ARM-based PQC SW implementations. The system clock of the ARM core has 60MHz frequency and the sampling rate to collect power traces is set to 10 MS/s. Generating trigger signals at the start and the end of a target function, e.g., a vector multiplication during decapsulation, can help collect aligned power traces corresponding to the target operation. Figure 12 shows a C code to generate trigger signals and a collected power trace via a GPIO port at the start and the end time of a vector multiplication in a Saber KEM.Dec function.

We analyze three versions of a Saber SW implementation[2], i.e., `LightSaber, Saber`, and `FireSaber`, and three versions of a Kyber SW implementation in `pqm4` testbench [KRSS19], i.e., `Kyber512, Kyber768`, and `Kyber1024`. We focus on the vector multiplication, which is the most vulnerable operation based on the pre-silicon analysis at Section 5.1. The Saber and Kyber are implemented with 4-way Toom-Cook based multipliers and NTT based multipliers, respectively. Figure 13 and 14 show TVLA and KL divergence testing results

[2]The Saber SW implementation is open source and can be downloaded at https://github.com/KULeuven-COSIC/Saber.

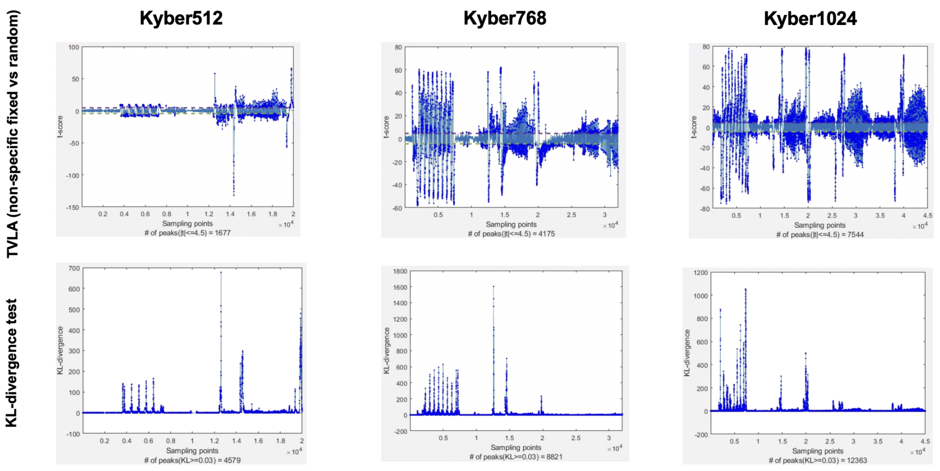


Figure 14: Side-channel leakage assessment of KYBER SW implementation.

Table 3: Post-silicon side-channel leakage assessment of Saber and Kyber SW implementations.

	Target	Time	TVLA (# of peaks ($\|t\| \leq 4.5$)/ # of samples)	KL (# of peaks ($D_{KL} \geq 0.3$)/ # of samples)	max(D_{KL})
LightSabe	`VectorMul`	5.4ms (324K cycles)	7090/54000 = 13.13%	13305/54000 = 24.64%	1035.3
Saber	(4-way Toom-	7ms (420K cycles)	8590/70000 = 12.27%	15501/70000 = 22.14%	1293.7
FireSaber	Cook)	8.4ms (504K cycles)	10764/84000 = 12.81%	23129/84000 = 27.53 %	1853.6
Kyber512	`Poly_frombyte_mul`	2ms (120K cycles)	1677/20000 = 8.38%	4579/20000 = 22.90 %	677.1
Kyber768	(NTT)	3.2ms (192K cycles)	4275/32000 = 13.35%	8821/32000 = 27.56%	1602.8
Kyber1024		4.5ms (270K cycles)	7544/45000 = 16.76%	12363/45000 = 27.47%	1053.9

of Saber SW and Kaber SW implementations, respectively. Based on these results, both SW implementations have many vulnerable leakage points during vector multiplications which occupy from 8% to 16% based on TVLA t-statistics (> 4.5 or < -4.5). Table 3 summarizes the target function, the processing time, and the results of TVLA and KL divergent tests of Saber and Kyber SW implementations.

5.2.2 HW Implementations

Xilinx Vivado 2020.2 is used for design synthesis, placement, and routing of PQC hardware implementations on Xilinx Kintex-7 or Artix-7 FPGA. To capture power traces from the target board, we used NewAE ChipWhisperer-Lite (CW-lite) capture board [Inca] that has a 10-bit analog-to-digital converter (ADC) with a 105 MS/s sampling rate and a high-performance oscilloscope with the maximum sampling rate of 5 GS/s, which is used to convert the analog power trace into discrete values. Then, these discrete values are stored in files on the connected computer, which are later used for TVLA and Kullback-Leibler (KL) divergence. A script on the host PC written in Python communicates with a control program running on the FPGAs via UART. The control program is responsible for communicating inputs and outputs with the target PQC IP cores.

CRYSTALS-KYBER: We implemented the CRYSTALS-Kyber512 KEM variant based on Xing *et al.* [XL21] on the Xilinx Virtex-7 FPGA platform. CRYSTALS-Kyber512 is based on a polynomial ring $R_q = Z_q[X]/(X^n + 1)$ of the dimension $n = 256$ and modulus $q = 3329$. Other public parameters for Kyber512 are $\eta = 2$ (as the n-th primitive root of unity in Z_q), k =2 (represents module dimension in MLWE) and $\eta = 2$ (binomial distribution with paramettter). The Kyber512-KEM is constructed from three top level Kyber-PKE block modules: key generation, encryption and decryption operations.

We briefly explained the Kyber-KEM algorithms in Section 2.3.2 and give the block

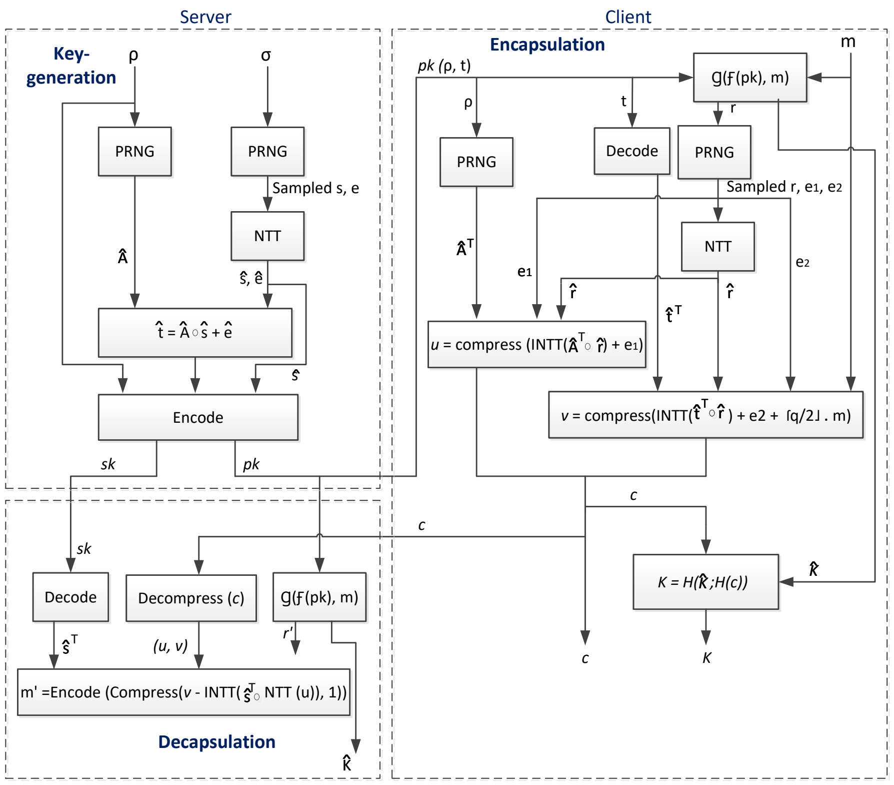

Server
Client
Key-generation
Encapsulation
Decapsulation
PRNG
NTT
Encode
Decode
Decompress (c)
g(f(pk), m)
$\hat{t} = \hat{A} \circ \hat{s} + \hat{e}$
u = compress (INTT($\hat{A}^T \circ \hat{r}$) + e1)
v = compress(INTT($\hat{t}^T \circ \hat{r}$) + e2 + ⌈q/2⌋ . m)
m' =Encode (Compress(v - INTT($\hat{s}^T \circ$ NTT (u)), 1))
$K = H(\hat{K};H(c))$
Sampled s, e
Sampled r, e1, e2
pk (ρ, t)
sk
pk
c
K

Figure 15: Block diagram of the Kyber-KEM.

diagram of the Kyber-KEM in Figure 15. The block diagram of the Kyber-KEM, which performs key generation, encapsulation, and decapsulation steps one by one (as explained in Section 2.3.2). In the key generation step, two random seed values (ρ, σ) are fed to the key-generation block to generate the sampled public matrix parameter $\hat{A}$ and secret vectors $\hat{s}$ and $\hat{e}$ then perform the computation $\hat{t} = \hat{A} \circ \hat{s} + \hat{e}$ in the NTT domain, and the results of encoded secret key is $\hat{s}$ and the public key pk (encoded t with ρ) are stored in the corresponding registers. In the encapsulation step, encoded public key pk, random message m and random seed value r are fed to the encryption block, first decode pk to generate the matrix A and $\hat{t}^T$, vectors r, e_1 and e_2 then perform the computation $u =$ INTT $(\hat{A}^T \circ \hat{r}) + e_1$ and $v =$ INTT $(\hat{t}^T \circ \hat{r}) + e_2 + Decompress(Decode(m))$ and the results are stored in the corresponding registers. Next, the u, v values are fed decompressed and decoded units to compute the cipertext $c = (c_1||c_2)$ and their results are stored in the corresponding registers. In the decapsulation step, the secret value $\hat{s}$, c_1 values are fed decryption block to calculate the message m^1. Then, the m^1, encoded public key pk, random seed value r' are fed again to the encryption block to compute the another cipertext c'. If ciphertexts match ($c = c'$), computes the shared secret K by using the ciphertext c, the value m, and public key. Otherwise, computes the shared secret K by using a random value and the ciphertext c (See Appendix B for more details).

For leakage detection, we rely on the widely used Welch t-test-based TVLA comparing fixed with random inputs measurements, and the resulting t-value is compared to a set threshold of ± 4.5, representing $\alpha = 0.0001$. Informally, if the threshold is exceeded, it is assumed to be possible to distinguish between fixed and random inputs, which confirms

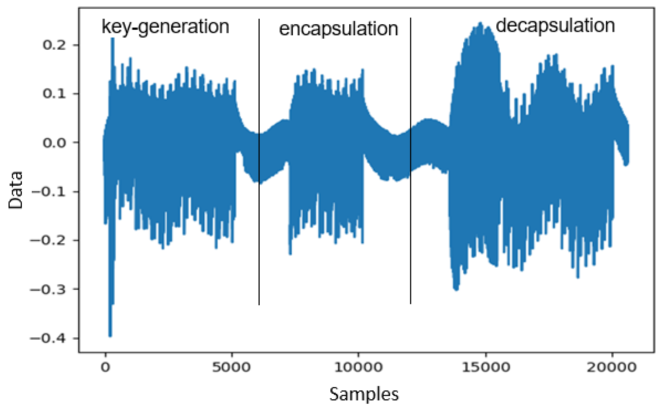


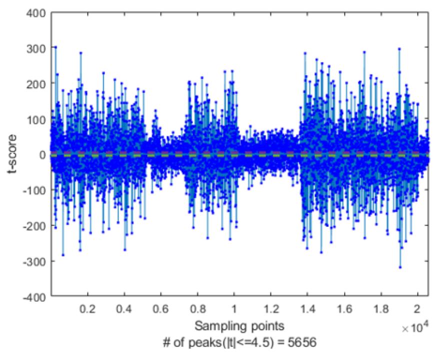
Figure 16: Power traces of hardware implementation of Kyber512 KEM algorithm collected on CW-Lite board.

the existence of exploitable leakage. For more details on TVLA, the readers are refer to [CDG+13]. Also, we use another vulnerability assessment such as KL-divergence to evaluate vulnerability of the targeted design. The KL-divergence is calculated based on power leakage distribution of the design. If KL divergence of any design is greater than pre-defined threshold value, the design is considered to be the vulnerable one [HPN+19]. Once Kyber512 KEM has been implemented on CW305 Artix-7 FPGA board, then we can start to capture the power traces based on feeding different input sets to the Kyber512 KEM core, which is target on the CW305 Artix-7 FPGA board. Our Python script is responsible for configuring bit file, arming CW-lite capture board, and capturing traces from the CW-lite board. The trigger signal is send to CW-lite board from the target FPGA enable trace capture when the CW-lite capture board is armed. The single power traces of the non-masked Kyber512 KEM hardware implementation is illustrated in Figure 16.

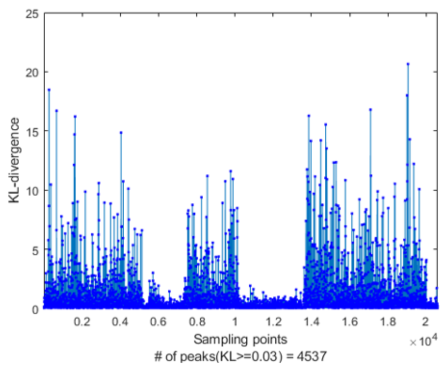
In our TVLA experiments, we use a non-specific fix-vs.-random test. We measure the power consumption of the Artix-7 FPGA board executing 10,000 capturing traces of the full Kyber512 KEM (i.e., including key generation, encapsulation, and decapsulation steps) on a fixed values (ρ, σ, m) for each execution, and another set of 10,000 capturing traces on random values (ρ, σ, m). The result of the t-test and KL-divergence for these sets are shown in Figure 17a and Figure 17b. As shown in fgure Figure 17a and Figure 17b, the results of the t-test and KL-divergence tests for full Kyber512 KEM show presence of lot of leakages in many locations on the traces such as 5656 leakage points in t-test analysis and 4537 vulnerability points in KL-divergence analysis.

Figure 18a and Figure 18b show the result of the t-test and KL-divergence test for only decapsulation step of Kyber512 KEM on the Artix-7 FPGA. In this experiment, we measure 10,000 capturing traces of the decapsulation step only on a fixed secret key (sk), fixed cybertext (c) for each execution at first set, and another set of 10,000 capturing traces on a fixed secret key (sk), random public cybertexts (c). As shown in Figure 18a and Figure 18b, the results of the t-test and KL-divergence tests are 3283 leakage points in t-test analysis and 1128 vulnerability points, respectively.

Figure 19a and Figure 19b shows the another experiment result of the t-test and KL-divergence test for key generation step of Kyber-512 on the FPGA. In this experiment, we measure 10,000 capturing traces of the key generation step of Kyber512 KEM on a fixed secret seed (σ), fixed public seed (ρ) for each execution, and another set of 10,000 capturing traces on a fixed secret seed (σ), random public seed (ρ). As shown in Figure 19a and Figure 19b, the results of the t-test and KL-divergence tests are 1640 leakage points

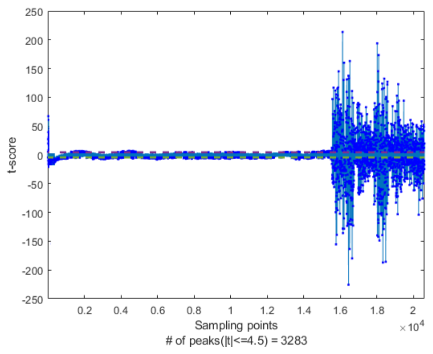


(a) TVLA for full Kyber512 KEM.

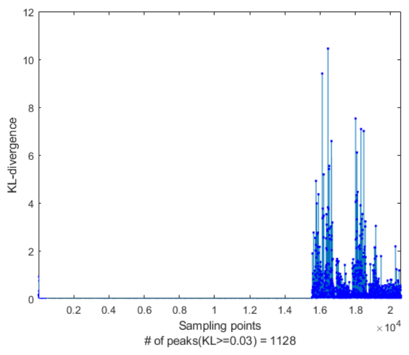


(b) KL for full Kyber512 KEM.

Figure 17: Side-channel leakage assessment of full Kyber512 KEM.



(a) TVLA for Kyber512 KEM.Decap.



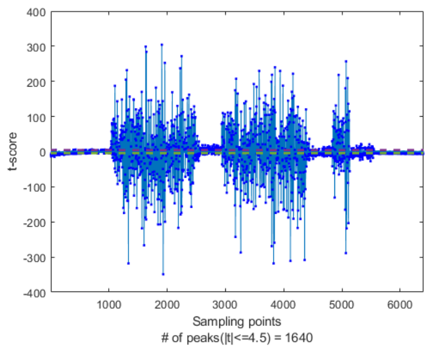
(b) KL for Kyber512 KEM.Decap.

Figure 18: Side-channel leakage assessment of decapsulation step of Kyber512 KEM.

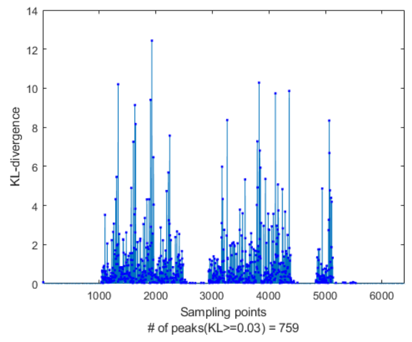
in t-test analysis and 759 vulnerability points, respectively.

SABER: We implemented the Saber instruction-set co-processor [RB20] integrated into an ARM Cortex-M4 architecture via an AHB interface on a Xilinx Kintex-7 FPGA and a stand-alone version of the Saber instruction-set architecture on a Xilinx Artix-7 FPGA. Figure 20 shows the block diagram of the Saber architecture that has a 35-bit wide instruction set consisting of a 5-bit OP code, two 10-bit input operand addresses, and a 10-bit result address, and a data memory with 64-bit words. Similarly to Saber SW analysis at Section 5.2.1, the polynomial multiplier in the Saber instruction-set architecture is scrutinized. It is based on optimized schoolbook-based multiplication and consists of parallel 256 multiplier-and-accumulate (MAC) units. For computing a matrix-vector multiplication, i.e., $\mathbf{A}\boldsymbol{s} = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,1} \\ A_{1,0} & A_{1,1} & A_{1,2} \\ A_{2,0} & A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix}$, polynomial multiplications, $A_{i,j}s_k$, are repeatedly executed. In this work, we only focus on the polynomial multiplication. For various TVLA and KL divergence tests, multiple sets are made as follows:

- Set1: a fixed key, $\boldsymbol{s_1}$ with n random $\mathbf{A}$
- Set2: a fixed key, $\boldsymbol{s_2}$ with n random $\mathbf{A}$
- Set3: a fixed key, $\boldsymbol{s_1}$ with a fixed $\mathbf{A}$
- Set4: a fixed key, $\boldsymbol{s_2}$ with a fixed $\mathbf{A}$



(a) TVLA for Kyber512 KEM.Keygen.



(b) KL for Kyber512 KEM.Keygen.

Figure 19: Side-channel leakage assessment of key generation step of Kyber512 KEM.

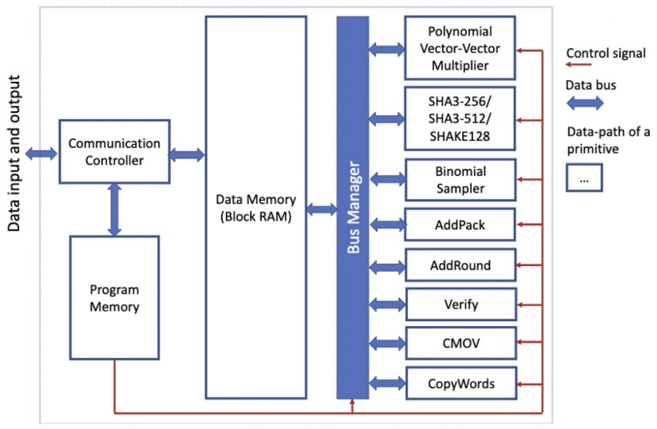
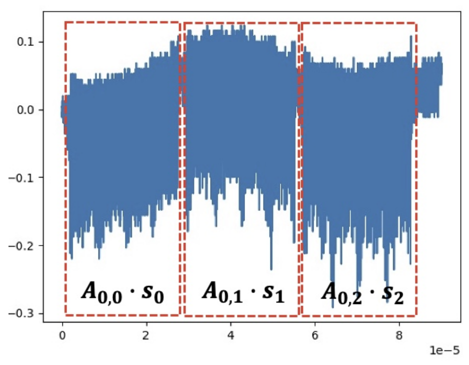


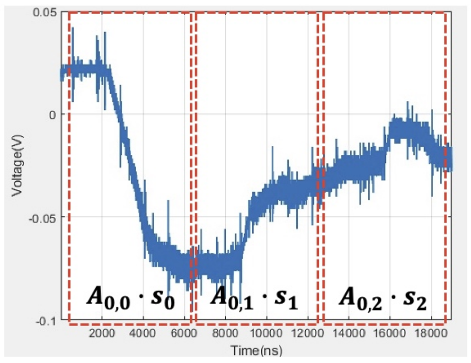
Figure 20: Block diagram of Saber instruction-set architecture [RB20].

During executing multiplications in each set, we can collect n power traces and then perform both TVLA and KL divergence tests in such a way that i) *random-vs.-random* with Set1 and Set2; ii) *random-vs.-fixed* with Set1 and Set 3; iii) *fixed-vs.-fixed* with Set3 and Set4. Two FPGAs are working at different clock frequencies; ChipWhisperer and our customized boards operate at 12 MHz and 60 MHz, respectively. Figure 21 shows a collected power trace during a polynomial multiplication in each FPGA. Measured power traces from the customized FPGA are more distorted for the high-frequency clock. Each set consists of 1000 power traces on the ChipWhisperer CW305 board. On the other hand, 10,000 power traces per set are collected from our Kintex-7 FPGA board.

Figure 22 and Figure 23 show the results based on TVLA and KL divergence metrics from two different boards, which identify many vulnerable points during polynomial multiplications. Especially, Test2, i.e., fixed-vs.-random test detects the most vulnerable points on both FPGA boards. Even if the Saber HW designs on two different boards are the same, vulnerable sampling points and quantitative leakage based on KL divergence are entirely different for the other clock frequency and integration style; a stand-alone Saber HW with a 12MHz system clock on ChipWhisperer CW board and an ARM Cortex-M4 based SoC including the Saber HW with a 60MHz system clock on our customized FPGA board. This brings us a new challenging issue, such as how to build a reference FPGA platform and measurement setups to evaluate side-channel leakages of any kind of PQC designs efficiently and accurately. We will deal with this issue again in Section 6.



(a) Power trace from ChipWhisperer CW305; freq. 12 MHz.



(b) Power trace from customized Kintex-7 FPGA; freq. 60 MHz.

Figure 21: Power trace during a polynomial multiplication of Saber HW.

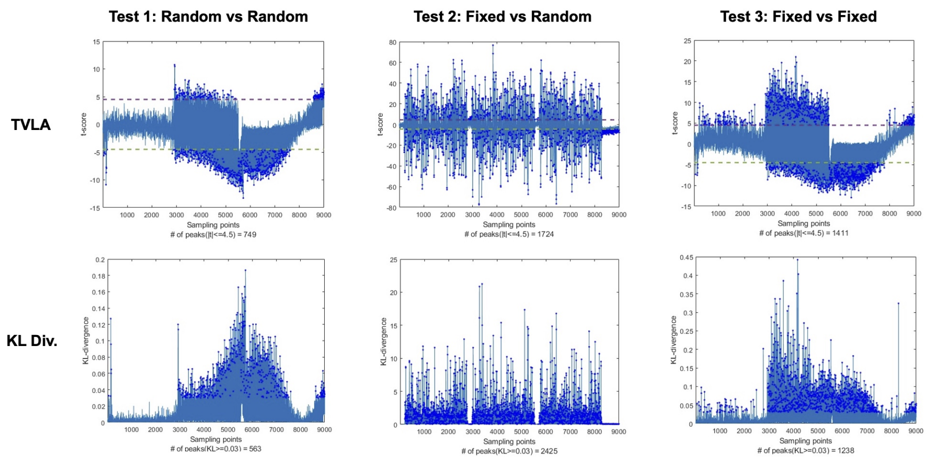


Figure 22: Side-channel leakage assessment of Saber HW on ChipWhisperer CW305 FPGA board.

Finally, Table 4 summarizes the target function, the processing time, and the results of TVLA and KL divergent tests of two Saber HW implementations.

5.2.3 AI-based SCA Attacks

In this work, extensive experiments are performed to evaluate the efficacy of the DL-based framework of side-channel attack using signal decomposition technique.

Dataset: There is no publicly available dataset on the Saber algorithm's hardware implementation. A hardware implementation of Saber is set up utilizing the CW305 Artix-7 FPGA board for testing. In all, 16000 power traces were captured from a CW305 Artix-7 FPGA board at 12 MHz clock frequency using a Tektronix MDO3102 in Figure 5 for all possible subkeys from 0x0 to 0xF, i.e. 1000 power traces per subkey. The sampling rate and bandwidth of the oscilloscope are set at 100 MS/s and 250 MHz, respectively. Each power trace corresponds to a vector multiplication in this scenario, with a total of 9000 sample points. Each subtrace corresponds to a different polynomial key s_i for $i = 0, 1$, and 2. This work focuses on extracting a coefficient of the polynomial key, i.e., a 4-bit subkey among the 256×4-bit entire key. In the Saber HW implementation, 256

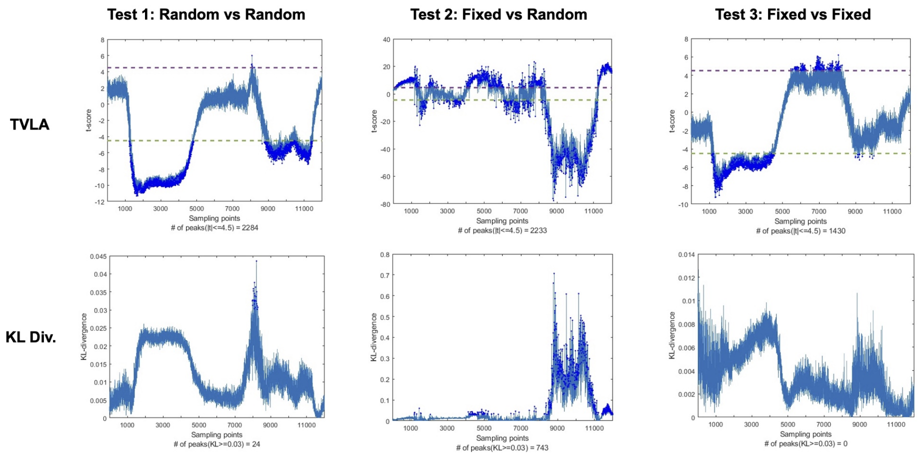


Figure 23: Side-channel leakage assessment of Saber HW on customized Kintex-7 FPGA board.

Table 4: Post-silicon side-channel leakage assessment of Saber HW implementations.

	Target	Time	TVLA (# of peaks ($\|t\| \leq 4.5$)/ # of samples)	KL (# of peaks ($D_{KL} \geq 0.3$)/ # of samples)	max(D_{KL})
Saber (CW305)	`Polynomial Multiplier`	360 us	1724/9000 = 19.15 %	2425/9000 = 26.90%	21.23
Saber (Our custom)		60 us	2233/12000 = 18.6%	743/12000 = 6.19%	0.7

multipliers with a 13-bit operand (a coefficient of A) and a 4-bit subkey (a coefficient of s) are executed simultaneously for a clock cycle. It takes 256 cycles to finish a polynomial multiplication. Figure 24 shows the structure of the polynomial multiplier consisting of 256 multiplier-and-accumulate (MAC) units. At the $(i+1)$st clock cycle, the ith coefficient of $A_{0,0}$, denoted by $a_{0,0}[i]$, is multiplied with all coefficient of a polynomial key, $s_0[j], \forall j = 0, 1, \ldots, 255$. The first part of the power traces in Figure 21a includes 256 multiplications with the target of the 4-bit subkey.

Evaluation Metrics: The performance of the methods used in experiments is measured in terms of success rate and guessing entropy (GE) in this study. The success rate is defined as the proportion of successful key recovery attempts to total key recovery attempts. When the predicted key matches the correct secret key, the attempt is successful. The GE, on the other hand, represents the average rank of the right key overall potential key-value combinations. For the GE calculation, we repeat the assault 200 times using randomly selected sub-samples of the test to determine the average number of traces necessary for key extraction. We also use a confusion matrix to visualize the key classification result properly.

Results: Figure 25a depicts the performance of the proposed technique. When the input features are varied, the performance of the method described in Section 4.3.2 is compared. When just raw traces are employed, the success percentages are the lowest. The inclusion of IMF0 to raw traces improves the performance of multiple-trace attacks by 76.59 %. EMD's first decomposed signal, IMF0, adds unique observations to the profiled attacks. In both metrics, the addition of IMF1 enhances the attack performance. The addition of decomposed traces as input features in the proposed framework ensures that the attack's unique intrinsic features are extended, which improves the attack's quality. When raw trace and four additional decomposed signals from IMF0 to IMF3 are combined as input features, the overall attack performance improves. A roughly 75% success rate is attained using several traces using this framework [3]. Although the additional complexity

[3]For 256 simultaneous multiplications, the SNR is noticeably low compared to other implementations

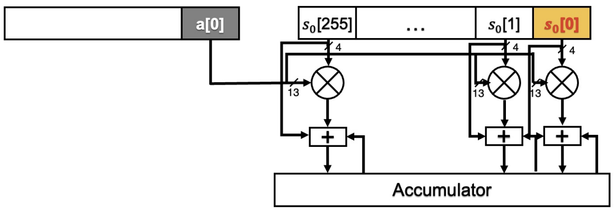
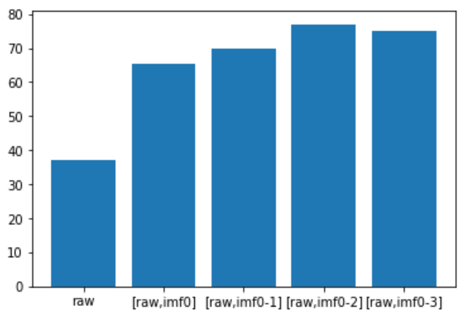
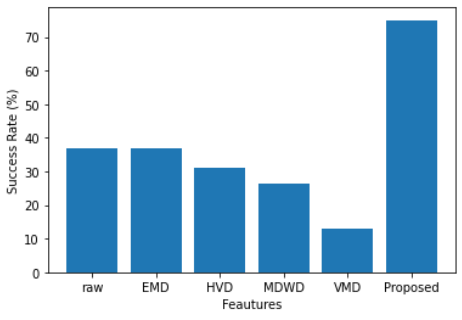


Figure 24: Polynomial multiplier architecture in Saber HW; $a[0]$ is multiplied at the first clock cycle and $s_0[0]$ is the target subkey.



(a) Performance of AI-based side-channel attacks using raw traces and different empirical modes of EMD.



(b) Performance of AI-based side-channel attacks with different signal decomposition techniques.

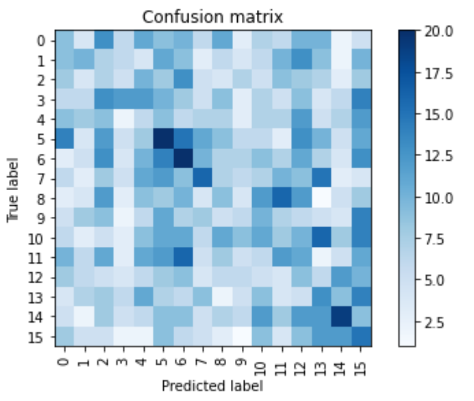
Figure 25: Performance of AI-based side-channel attacks on Saber HW.

comes at the expense of enhanced performance, the computation cost is still acceptable and not prohibitive compared to the gains in performance.

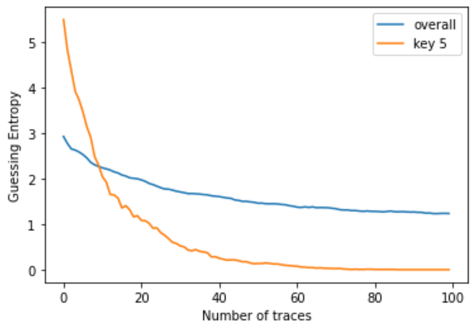
The confusion matrix for a single trace attack with the framework is shown in Figure 26a. From the figure, it can be seen that for keys 5, 6, and 14, the key detection accuracy is higher than other key classes. Guessing entropy for key 5 is plotted in Figure 26b. The figure shows that the attack phase requires less than 20 traces to find the secret key when the correct key is key 5. The average guessing entropy calculated for all secret keys is also shown in Figure 26b. From the figure, it can be noted that the overall guessing entropy is less than 2 when around 20 traces are used. That means the correct key is within the top 3 suggestions of the proposed method.

As mentioned before, the proposed approach uses EMD as the signal decomposition technique. The performance of this decomposition technique is compared to other methods when different decomposition techniques are used. In comparing methods, four different signal decomposition techniques have been used in this work. The comparing decomposition techniques are Hilbert vibration decomposition (HVD), multilevel discrete wavelet decomposition (MDWD), and variational mode decomposition (VMD). Each of these techniques focuses on different signal properties when splitting the raw traces into multiple decomposed signals and generates unique features. Figure 25b shows the comparison. From the table, it can be seen that the performance of the attack is increased in all metrics when any of these decomposition techniques are applied. Among EMD, VMD, HVD, and MDWD, the EMD approach performs better than others. When both empirical modes generated by EMD and raw traces are ensembled in the proposed methodology, the highest success rates are attained.

in Table 2.



(a) Confusion matrix for DL-based side-channel attack using EMD.



(b) Performance of single-trace side-channel attack using EMD in terms of guessing entropy.

Figure 26: Performance of single-trace side-channel attack using EMD.

6 Challenges and Future Research Directions

This section discusses the open issues and challenging problems of side-channel leakage assessment of PQC implementations and addresses high-level approaches for future research directions.

6.1 Side-channel Leakage Assessment

Although our PQC-SEP can analyze PQC IPs and a SOC, including the PQC IPs, automatically at pre-and post-silicon levels, we need to address the challenging problems as follows:

- **Need for reference evaluation platform:** In Section 5.2.2, even if the Saber HW designs on two different boards are the same, vulnerable sampling points and quantitative leakage based on KL divergence are entirely different. The side-channel leakage analysis will depend on the system clock frequency, the method to integrate the PQC design into any architecture, and measurement setups. A reference FPGA platform and measurement setup are required to evaluate the side-channel leakages of PQC designs efficiently and accurately.

- **Need for reference testing methodology:** We need a guideline to perform a side-channel leakage assessment of PQC designs, such as TVLA for AES designs. PQC algorithms have an extended length of keys and tremendous computational complexity, making the evaluation harder. For example, we chose only a pair of two keys to calculate the statistical distance between two different sets in this work. There is a considerable sample space to select a key pair, and we cannot guarantee which one is the best to estimate side-channel leakages accurately. Maybe it will be infeasible to search for a key pair as a reference for the PQC analysis.

To build the reference evaluation platform and testing methodology, tremendous experiments under various setups, e.g., different clock frequencies, other FPGA boards, various measurement instruments, and many input test patterns, should be accomplished. Since these experiments will need significant time, we plan to optimize PQC-PAT with additional FPGA evaluation boards to reduce experimental processing time.

6.2 AI-based Side-channel Attack

Although a good amount of work on side-channel attacks in AES uses machine learning and deep learning, in the case of PQC algorithms, this field has not been explored thoroughly. Nevertheless, the current works show the potential of AI in side-channel attacks on PQC protocols. The existing open challenges for AI-based side-channel attacks are described below:

- **Lack of data:** Since AI-based side-channel attack is a data-driven approach, the availability of data is crucial. Previously, in Section 4.3, existing works of AI-based side-channel attacks on PQC algorithms are described. But none of these studies make the data publicly available. Such unavailability of data makes it difficult to check the efficacy of varieties of AI algorithms to perform a successful side-channel attack on different PQC algorithms. It also makes it hard to compare the side-channel vulnerability of different PQC algorithms for a specific AI-based attack approach.

- **Lack of hardware implementation:** Among the existing studies, only in the case of AI-based side-channel attacks on Frodo and NewHope, the attack is performed on hardware implementation.

- **Lack of explainable approach:** The lack of explainability is another open challenge for AI-based side-channel attacks. Undoubtedly, neural networks have demonstrated tremendous effectiveness even in the face of advanced countermeasures. However, it is uncertain how these networks actually work. It's tough to tell if a failed attack was due to a bad countermeasure or a less effective AI strategy. Similarly, none of the existing works has been able to determine the design's specific weakness in the case of a successful attack. Without any explanation, it is difficult to devise effective countermeasures.

- **Lack of clear guidance**: Any profiled DL-based side-channel attack requires several steps to be followed sequentially: pre-processing, feature engineering, algorithm selection, and attack evaluation. There is still no proper guideline for a post-silicon or pre-silicon side-channel attack on PQC protocols for any of these steps. It is unclear what type of pre-processing and feature extraction approaches would work efficiently.

6.3 Secure PQC Implementation

Masking is commonly used to prevent higher-order side-channel attacks. Target functions, such as a polynomial multiplication in an LWE scheme, can be protected by an arithmetic masking or Boolean masking according to n-th order side-channel attack models. However, many random bits, n linear masked modules, nonlinear masked modules, and conversion modules between Boolean domain and the arithmetic domain will result in tremendous area/power overhead and lower performance. In addition, after placing and routing, different analysis results can be generated by low-level side-channel leakage analysis, i.e., PL-PAT, without additional countermeasures, e.g., hiding by inserting random noise. Only the masking technique cannot guarantee power/EM side-channel protection.

In this regard, we need a generic framework of secure PQC IP designs against power/EM side-channel attacks combined with pre-silicon side-channel leakage analysis shown in Figure 4. It consists of the following synthesizers as shown in Figure 27:

1. **Masking synthesizer (RTL)**: Based on the result of the pre-silicon side-channel leakage assessment in Section 4.1, vulnerable modules will be replaced into masked modules depending on n-th order attack models automatically by the masking synthesizer. The new top design with the generated masked modules will be analyzed

in terms of side-channel leakages. If not satisfying side-channel resistance, it will be re-masked until passing the side-channel leakage test (Loop 1).

2. **Private-cell synthesizer (gate-level):** After logic synthesis of the masked design, vulnerable cells will be identified by the GL-PAT in Figure 4. The vulnerable cells will be converted into private cells, such as WDDL or *t*-private cells, automatically by the private-cell synthesizer. Repeatedly, the above gate-level processes continue until satisfying any SCL standard and constraints, such as area overhead, power consumption, and performance (Loop 2).

3. **Physical-SCR synthesizer (layout level):** Although the gate-level design has side-channel resistance, its layout design can be vulnerable to side-channel attacks. The physical side-channel resistant (SCR) synthesizer will automatically generate a secure physical layout design by connecting physical countermeasure circuits, such as a randomizer, into vulnerable nodes based on a PL-PAT or isolating power networks of vulnerable nodes. It will be finished if the regenerated layout design passes layout-level SCL tests. Otherwise, the secure physical design will continue (Loop 3).

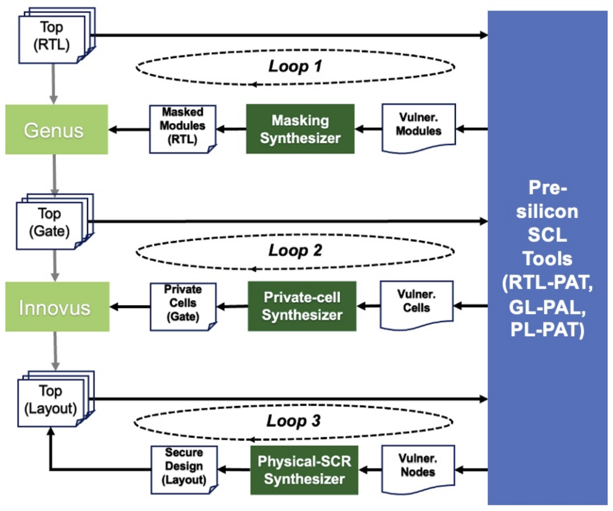


Figure 27: A framework of secure PQC IP design.

7 Conclusion

With extensive interest in the security of the PQC algorithm, it is imperative that side-channel leakage assessment be developed and deployed to thwart various side-channel attacks. Our PQC-SEP shows that it can analyze side-channel leakages of candidates, i.e., Saber and Kyber in NIST PQC competition at pre- and post-silicon levels. It primarily helps hardware developers identify vulnerability against power side-channel attacks without additional HW and SW setups to measure side-channel leakages. However, there exist various challenging problems in PQC side-channel research works. To address the issues, some high-level approaches are highlighted.

References

[AAT$^+$21] Furkan Aydin, Aydin Aysu, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. Horizontal side-channel vulnerabilities of post-quantum key exchange and encapsulation protocols. *ACM Trans. Embed. Comput. Syst.*, 20(6), oct 2021.

[ABB$^+$17] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. Bike: Bit flipping key encapsulation. 2017.

[ACC$^+$21] Amin Abdulrahman, Jiun-Peng Chen, Yu-Jia Chen, Vincent Hwang, Matthias J. Kannwischer, and Bo-Yin Yang. Multi-moduli NTTs for Saber on Cortex-M3 and Cortex-M4. *IACR Cryptol. ePrint Arch.*, page 995, 2021.

[ADSH18] N. Nalla Anandakumar, M. Prem Laxman Das, Somitra K. Sanadhya, and Mohammad S. Hashmi. Reconfigurable Hardware Architecture for Authenticated Key Agreement Protocol Over Binary Edwards Curve. *ACM Trans. Reconfigurable Technol. Syst.*, 11(2):12:1–12:19, November 2018.

[AKP$^+$20] Furkan Aydin, Priyank Kashyap, Seetal Potluri, Paul Franzon, and Aydin Aysu. Deepar-sca: Breaking parallel architectures of lattice cryptography via learning based side-channel attacks. In *International Conference on Embedded Computer Systems*, pages 262–280. Springer, 2020.

[AMBD$^+$18] Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64(5):3927–3943, 2018.

[AR21] Amund Askeland and Sondre Rønjom. A side-channel assisted attack on ntru. Cryptology ePrint Archive, Report 2021/790, 2021. https://ia.cr/2021/790.

[Bal21] Philip Ball. First quantum computer to pack 100 qubits enters crowded race, 2021. https://www.nature.com/articles/d41586-021-03476-5.

[BCD$^+$16] Joppe Bos, Craig Costello, Leo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1006–1018, New York, NY, USA, 2016. Association for Computing Machinery.

[BCLvV17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: Reducing attack surface at low cost. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*, volume 10719 of *Lecture Notes in Computer Science*, pages 235–260. Springer, 2017.

[BDH$^+$21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and defending masked polynomial comparison for lattice-based cryptography. Cryptology ePrint Archive, Report 2021/104, 2021. https://ia.cr/2021/104.

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 353–367, 2018.

[BDK+21] Michiel Van Beirendonck, Jan-Pieter D'anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of saber. *J. Emerg. Technol. Comput. Syst.*, 17(2), April 2021.

[BFM+18] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Assessing the feasibility of single trace power analysis of frodo. Cryptology ePrint Archive, Report 2018/687, 2018. https://ia.cr/2018/687.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2129–2146, New York, NY, USA, 2019. Association for Computing Machinery.

[BSP+22] Yunkai Bai, Andrew Stern, Jungmin Park, Mark Tehranipoor, and Domenic Forte. Rascv2: Enabling remote access to side-channels for mission critical and iot systems. *ACM Trans. Des. Autom. Electron. Syst.*, mar 2022. Just Accepted.

[Cad21] Cadence. Cadence voltus power integrity solution, 2021. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/voltus-ic-power-integrity-solution.html.

[CDG+13] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Pankaj Rohatgi, et al. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, volume 20, 2013.

[CDG21] Jerry Chow, Oliver Dial, and Jay Gambetta. Ibm quantum breaks the 100-qubit processor barrier, 2021. https://research.ibm.com/blog/127-qubit-quantum-processor-eagle.

[CFMR+17] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. Gemss: A great multivariate short signature. 2017.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, volume 10831 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2018.

[DZ14] Konstantin Dragomiretskiy and Dominique Zosso. Variational mode decomposition. *IEEE Transactions on Signal Processing*, 62(3):531–544, 2014.

[EKR87] J.-P Eckmann, S. Oliffson Kamphorst, and D Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters (EPL)*, 4(9):973–977, nov 1987.

[Fel06] Michael Feldman. Time-varying vibration decomposition and analysis based on the hilbert transform. *Journal of Sound and Vibration*, 295(3):518–530, 2006.

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 36, 2018.

[Gam] Jay Gambetta. Ibm's roadmap for scaling quantum technology. https://research.ibm.com/blog/ibm-quantum-roadmap.

[HCY19] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on ntru prime. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):123–151, Nov. 2019.

[HCY20] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Power analysis on NTRU prime. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):123–151, 2020.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 341–371, Cham, 2017. Springer International Publishing.

[HPN+19] Miao He, Jungmin Park, Adib Nahiyan, Apostol Vassilev, Yier Jin, and Mark Tehranipoor. RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level. In *2019 IEEE 37th VLSI Test Symposium (VTS)*, pages 1–6, 2019.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[HSL+98] Norden E Huang, Zheng Shen, Steven R Long, Manli C Wu, Hsing H Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *in Proc. the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 454(1971):903–995, 1998.

[Inca] NewAE Technology Inc. Chipwhisperer-lite (cw1173) two-part version. https://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/.

[Incb] NewAE Technology Inc. Cw1200 chipwhisperer-pro. https://rtfm.newae.com/Capture/ChipWhisperer-Pro/.

[Incc] NewAE Technology Inc. Cw305 artix fpga target. https://www.newae.com/products/NAE-CW305.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 463–481, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[KAP+20] Priyank Kashyap, Furkan Aydin, Seetal Potluri, Paul D Franzon, and Aydin 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*, 40(6):1217–1229, 2020.

[KdG21] Alexandre Karlov and Natacha Linard de Guertechin. Power analysis attack on Kyber. *IACR Cryptol. ePrint Arch.*, page 1311, 2021.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):243–268, Jun. 2020.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. Cryptology ePrint Archive, Report 2019/844, 2019. https://ia.cr/2019/844.

[McE78] Robert J. McEliece. A public key cryptosystem based on algebraic coding theory. 1978.

[NDGJ21a] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):676–707, Aug. 2021.

[NDGJ21b] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked ind-cca secure saber kem. *IACR Cryptol. ePrint Arch.*, 2021:79, 2021.

[NDJ21] Kalle Ngo, Elena Dubrova, and Thomas Johansson. Breaking masked and shuffled cca secure saber kem by power analysis. *Cryptology ePrint Archive*, 2021.

[NHPT21] Adib Nahiyan, Miao He, Jungmin Park, and Mark Tehranipoor. Chapter 7. cad for side-channel leak- age assessment, emerging topics in hardware security, 2021.

[NISa] NIST. Nist post-quantum cryptography standardization. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[NISb] NIST. Pqc standardization process: Third round candidate announcement. https://csrc.nist.gov/news/2020/pqc-third-round-candidate-announcement.

[Nog14] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014.

[PCL+19] Jungmin Park, Seongjoon Cho, Taejin Lim, Swarup Bhunia, and Mark Tehranipoor. Scr-qrng: Side-channel resistant design using quantum random number generator. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[PH16] Aesun Park and Dong-Guk Han. Chosen ciphertext simple power analysis on software 8-bit implementation of ring-lwe encryption. In *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6, 2016.

[PP19] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. Cryptology ePrint Archive, Report 2019/795, 2019. https://ia.cr/2019/795.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. Cryptology ePrint Archive, Report 2017/594, 2017. https://ia.cr/2017/594.

[PXJ$^+$18] Jungmin Park, Xiaolin Xu, Yier Jin, Domenic Forte, and Mark Tehranipoor. Power-based side-channel instruction-level disassembler. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, 2018.

[RB20] Sujoy Sinha Roy and Andrea Basso. High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 443–466, 2020.

[RBRC20] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam 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.*, page 549, 2020.

[SEG] SEGGER. J-link edu – the educational j-link. https://www.segger.com/products/debug-probes/j-link/models/j-link-edu/.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[SKB21] Hauke Malte Steffen, Lucie Johanna Kogelheide, and Timo Bartkewitz. In-depth Analysis of Side-Channel Countermeasures for CRYSTALS-Kyber Message Encoding on ARM Cortex-M4. *IACR Cryptol. ePrint Arch.*, page 1307, 2021.

[SKL$^+$20] Bo-Yeon Sim, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. Single-trace attacks on message encoding in lattice-based kems. *IEEE Access*, 8:183175–183191, 2020.

[SPH21] Bo-Yeon Sim, Aesun Park, and Dong-Guk Han. Chosen-ciphertext Clustering Attack on CRYSTALS-KYBER using the Side-channel Leakage of Barrett Reduction. *IACR Cryptol. ePrint Arch.*, page 874, 2021.

[SR51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[TV04] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '04, page 10246, USA, 2004. IEEE Computer Society.

[WO15] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation, 2015.

[XL21] Yufei Xing and Shuguo Li. A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):328–356, Feb. 2021.

[XPSR+21] Zhuang Xu, Owen Michael Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber. *IEEE Transactions on Computers*, pages 1–1, 2021.

[ZPTF21] Tao Zhang, Jungmin Park, Mark Tehranipoor, and Farimah Farahmandi. PSC-TG: RTL Power Side-Channel Leakage Assessment with Test Pattern Generation. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 709–714, 2021.

A Saber Algorithm [DKRV18]

Algorithm 3 `Saber.PKE.KeyGen()`

```
1: $seed_{\mathbf{A}} \leftarrow \mathcal{U}\{(0,1)^{256}\}$
2: $\mathbf{A} = \mathtt{gen}(seed_{\mathbf{A}}) \in \mathcal{R}_q^{l \times l}$
3: $r = \mathcal{U}(\{(0,1)^{256}\}$
4: $\boldsymbol{s} = \beta_\mu(\mathcal{R}_q^{l \times 1}; r)$
5: $\boldsymbol{b} = ((\mathbf{A}^T \boldsymbol{s} + \boldsymbol{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}^{l \times 1}$
6: Return $(pk := (seed_{\mathbf{A}}, \boldsymbol{b}), sk := (\boldsymbol{s}))$
```

Algorithm 4 `Saber.PKE.Enc`$(pk := (seed_{\mathbf{A}}, \boldsymbol{b}), m \in R_2, r)$

```
1: $\mathbf{A} = \mathtt{gen}(seed_{\mathbf{A}}) \in \mathcal{R}_q^{l \times l}$
2: if $r$ is not specified then
3:     $r = \mathcal{U}(0, 1^{256})$
4: end if
5: $\boldsymbol{s'} = \beta_\mu(\mathcal{R}_q^{l \times 1}; r)$
6: $\boldsymbol{b'} = ((\mathbf{A}\boldsymbol{s'} + \boldsymbol{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}^{l \times 1}$
7: $v' = \boldsymbol{b}^T(\boldsymbol{s'} \mod p) \in \mathcal{R}_p$
8: $c_m = (v' + h_1 - 2^{\epsilon_p - 1} m \mod p) \gg (\epsilon_q - \epsilon_T) \in \mathcal{R}_T$
9: Return $c := (c_m, \boldsymbol{b'})$
```

Algorithm 5 `Saber.PKE.Dec`$(sk = (\boldsymbol{s}), c = (c_m, \boldsymbol{b'}))$

```
1: $v = \boldsymbol{b'}^T(\boldsymbol{s} \mod p) \in \mathcal{R}_p$
2: $m' = (v - 2^{\epsilon_p - \epsilon_T} c_m + h_2 \mod p) \gg (\epsilon_p - 1) \in R_2$
3: Return $m'$
```

Algorithm 6 `Saber.KEM.KeyGen()`

```
1: $(seed_{\mathbf{A}}, \boldsymbol{b}, \boldsymbol{s})$ = Saber.PKE.KeyGen()
2: $pk = (seed_{\mathbf{A}}, \boldsymbol{b})$
3: $pkh = \mathcal{F}(pk)$
4: $z = \mathcal{U}(0, 1^{256})$
5: Return $(pk := (seed_{\mathbf{A}}, \boldsymbol{b}), sk := (\boldsymbol{s}, z, pkh))$
```

Algorithm 7 `Saber.KEM.Encaps`$(pk = (seed_{\mathbf{A}}, \boldsymbol{b}))$

```
1: $m \leftarrow \mathcal{U}(\{0,1\}^{256})$
2: $(\hat{K}, r) = \mathcal{G}(\mathcal{F}(pk), m)$
3: $c = \texttt{Saber.PKE.Enc}(pk, m; r)$
4: $K = \mathcal{F}(\hat{K}, c)$
5: Return $(c, K)$
```

Algorithm 8 `Saber.KEM.Decaps`$(sk = (\boldsymbol{s}, z, pkh), pk = (seed_{\mathbf{A}}, \boldsymbol{b}))$

```
1: $m' = \texttt{Saber.PKE.Dec}(\boldsymbol{s}, c)$
2: $(\hat{K}', r') = \mathcal{G}(\mathcal{F}(pk), m')$
3: $c' = \texttt{Saber.PKE.Enc}(pk, m'; r')$
4: if $c = c'$ then
5:     Return $K = \mathcal{H}(\hat{K}', c)$
6: else
7:     Return $K = \mathcal{H}(z, c)$
8: end if
```

B CRYSTALS-Kyber Algorithm [BDK+18]

Algorithm 9 `Kyber.CPA.KeyGen()`

```
1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} = \texttt{gen}(\rho) \in \mathcal{R}_q^{k \times k}$
3: $(\boldsymbol{s}, \boldsymbol{e}) = \texttt{gen}(\sigma) \in \beta_\eta^k \times \beta_\eta^k$
4: $\boldsymbol{t} = \texttt{Compress}_\texttt{q}(\mathbf{A}\boldsymbol{s} + \boldsymbol{e}, d_t)$
5: Return $(pk := (\boldsymbol{t}, \rho), sk := \boldsymbol{s})$
```

Algorithm 10 `Kyber.CPA.Enc`$(pk = (\boldsymbol{t}, \rho), m \in \mathcal{M})$

```
1: $r \leftarrow \{0,1\}^{256}$
2: $\boldsymbol{t} = \texttt{Decompress}_\texttt{q}(\boldsymbol{t}, d_t)$
3: $\mathbf{A} = \texttt{gen}(\rho) \in \mathcal{R}_q^{k \times k}$
4: $(\boldsymbol{r}, \boldsymbol{e}_1, e_2) = \texttt{gen}(r) \in \beta_\eta^k \times \beta_\eta^k \times_\eta$
5: $\boldsymbol{u} = \texttt{Compress}_\texttt{q}(\mathbf{A}^T \boldsymbol{r} + \boldsymbol{e}_1, d_u)$
6: $v = \texttt{Compress}_\texttt{q}(\boldsymbol{t}^T \boldsymbol{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
7: Return $c := (\boldsymbol{u}, v)$
```

Algorithm 11 `Kyber.CPA.Dec`$(sk = \boldsymbol{s}, c = (\boldsymbol{u}, v))$

```
1: $\boldsymbol{u} = \texttt{Decompress}_\texttt{q}(\boldsymbol{u}, d_u)$
2: $v = \texttt{Decompress}_\texttt{q}(v, d_v)$
3: Return $\texttt{Compress}_\texttt{q}(v - \boldsymbol{s}^T \boldsymbol{u}, 1)$
```

Algorithm 12 $\texttt{Kyber.Encaps}(pk = (\boldsymbol{t}, \rho))$

```
1: $m \leftarrow \{0,1\}^{256}$
2: $(\hat{K}, r) = \mathcal{G}(\mathcal{F}(pk), m)$
3: $(\boldsymbol{u}, v) = \texttt{Kyber.CPA.Enc}((\boldsymbol{t}, \rho), m; r)$
4: $c = (\boldsymbol{u}, v)$
5: $K = \mathcal{H}(\hat{K}, \mathcal{H}(c))$
6: Return $(c, K)$
```

Algorithm 13 $\texttt{Kyber.Decaps}(sk = (\boldsymbol{s}, z, \boldsymbol{t}, \rho), c = (\boldsymbol{u}, v))$

```
1: $m' = \texttt{Kyber.CPA.Dec}(\boldsymbol{s}, (\boldsymbol{u}, v))$
2: $(\hat{K}', r') = \mathcal{G}(\mathcal{F}(pk), m')$
3: $(\boldsymbol{u}', v') = \texttt{Kyber.CPA.Enc}((\boldsymbol{t}, \rho), m'; r')$
4: if $(\boldsymbol{u}', v') = (\boldsymbol{u}, v)$ then
5:     Return $K = \mathcal{H}(\hat{K}', \mathcal{H}(c))$
6: else
7:     Return $K = \mathcal{H}(z, \mathcal{H}(c))$
8: end if
```