Vulnerable PQC against Side Channel Analysis - A Case Study on Kyber

Haocheng Ma*, Shijian Pan*, Ya Gao*, Jiaji He*‡, Yiqiang Zhao*, and Yier Jin†
*School of Microelectronics, Tianjin University
†Department of Electrical and Computer Engineering, University of Florida
{hc_ma, jianjian_kk, gaoyaya, dochejj, yq_zhao}@tju.edu.cn, yier.jin@ece.ufl.edu

2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST) | 978-1-6654-6114-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASIANHOST56390.2022.10022165

***Abstract*—The emergence of quantum computing and its impact on current cryptographic algorithms has triggered the migration to post-quantum cryptography (PQC). Among the PQC candidates, CRYSTALS-Kyber is a key encapsulation mechanism (KEM) that stands out from the National Institute of Standards and Technology (NIST) standardization project. While software implementations of Kyber have been developed and evaluated recently, Kyber's hardware implementations, especially designs with parallel architecture, are rarely discussed. To help better understand Kyber hardware designs and their security against side-channel analysis (SCA) attacks, in this paper, we first adapt the two most recent Kyber hardware designs for FPGA implementations. We then perform SCA attacks against these hardware designs with different architectures, i.e., parallelization and pipelining. Our experimental results show that Kyber designs on FPGA boards are vulnerable to SCA attacks including electromagnetic (EM) and power side channels. An attacker only needs $27 \sim 1600$ power traces or $60 \sim 2680$ EM traces to recover the decryption key successfully.**

***Index Terms*—CRYSTALS-Kyber, Side Channel Analysis, Kyber Hardware Implementations**

I. Introduction

Modern cryptographic algorithms including popular public key cryptography (PKC) have faced security threats with the rapid progress of quantum computing. The upcoming large quantum computer will likely break widely adopted cryptosystems like RSA and Elliptic Curve Cryptography (ECC). Hence, it necessitates studies on next-generation PKC, i.e., post-quantum cryptography (PQC), to address these threats. Different from the integer factorization and the discrete logarithm problems, mathematical problems of PQC are thought to be intractable by classical and quantum computers [1]. To push ahead with PQC standardization, the NIST started a contest involving KEM, public key encryption (PKE) and digital signature schemes. More recently, Kyber wins three rounds of competitions and becomes the sole candidate for PKE/KEMs standard. It is a module learning-with-errors (MLWE) based protocol that outperforms other candidates for security, cost and performance aspects [2]. As existing cryptosystems transition to Kyber, the security evaluation against physical attacks such as SCA attacks is of importance. However, researchers focus more on traditional characteristics such as power and performance but often ignore security against SCA attacks.

‡Jiaji He is the corresponding author.

978-1-6654-4185-8/21/$31.00 ©2022 IEEE

The primary usage of PKE/KEMs is to ensure the security of the key exchange using the public and private key pair. SCA attack on PKE/KEMs aims to recover the long-term private key or ephemeral session key by analyzing timing delay, power consumption and EM emanation. The long-term key recovery often involves polynomial multiplication and inverse number theoretic transform (NTT) operations in the decryption procedure [3], [4], [5], [6]. While the second type of attack targets operations such as message encoding and message decoding during the encryption procedure [7], [8], [9]. Side-channel leakages about these operations can be analyzed through a variety of methods, including simple power analysis (SPA), correlation power analysis (CPA), and profiling attacks. However, most of the recent works have focused on software implementations, SCA attacks against hardware designs of Kyber are often ignored. Towards this direction, we try to help understand whether SCA attacks remain major threats to different Kyber hardware designs.

In this paper, several state-of-the-art hardware designs of Kyber are comprehensively evaluated, including manual design [10] and HLS-based design [11]. Various levels of parallel computations are used to achieve high performance with limited computational resources. Specifically, we focus on the decryption procedure of Kyber to recover the long-term key through SCA attacks. According to the NTT domain and time domain, the decryption procedure is classified into two vulnerable regions, involving multiple points of interest (PoIs) like point-wise multiplication (PWM), modular reduction and functions after inverse NTT. Different strategies of SCA attacks are designed on these PoIs for efficient key recovery. Experimental results provide a deep perception of vulnerabilities that exist in Kyber hardware implementations, and assist hardware designers to balance security, performance and cost.

The main contributions of the paper are as follows.

- We design SCA attacks on hardware implementations of Kyber to recover the long-term secret key. Though parallel computations reduce the signal-to-noise ratio (SNR) of leakage, designs with parallel architectures are still vulnerable to SCA attacks.
- We separate the decryption procedure into two vulnerable regions, in which all possible PoIs are evaluated using SCA attacks. Our results show that the security of functions in the time domain should gain more attention.
- We evaluate the efficacy of SCA attacks on the FPGA

Algorithm 1 KYBER.CPAPKE.Dec(sk, c): decryption

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

1: $\mathbf{u} := \mathsf{Decompress_q}(\mathsf{Decode_{d_u}}(c), d_u)$
2: $v := \mathsf{Decompress_q}(\mathsf{Decode_{d_v}}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \mathsf{Decode_{12}}(sk)$
4: $m := \mathsf{Encode_1}(\mathsf{Compress_q}(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u}))), 1)$
5: $\triangleright\ m := \mathsf{Compress_q}(v - \mathbf{s}^T\mathbf{u}, 1)$
6: **return** m

platform. Experimental results demonstrate that SCA attacks can recover secret keys from Kyber design with a few power and EM traces.

The rest of the paper is organized as follows. Section II goes over the notations and description of Kyber and related work on SCA attacks. In Section III, we introduce the hardware architecture of manual design and HLS-based design, respectively. In Section IV, we discuss the vulnerable regions during the decryption procedure, and give the attack methodology for multiple PoIs of Kyber. Section V demonstrates the actual results of power and EM SCA attacks on hardware designs. Finally, we conclude the paper in Section VI.

II. BACKGROUND

A. Preliminary Notation

The ring of integers modulo prime q is denoted as $\mathbb{Z}_q$. The ring of integer polynomials modulo prime q is denoted as $\mathbb{Z}_q[X]$. Then polynomials modulo both q and $X^n + 1$ form the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. Polynomial vectors with k module dimension are denoted in bold lowercase, where a single element in $\mathcal{R}_q$ is written as pure lowercase. Moreover, the polynomial vector in the time domain is denoted as $\mathbf{s}$, while $\hat{\mathbf{s}}$ denotes this polynomial vector in the NTT domain. The binomial distribution used in Kyber is denoted as $\mathcal{B}_\eta$, which samples noise in the range $[-\eta, \eta]$ with a fixed parameter n. The value of k defines three security types of Kyber, namely Kyber512, Kyber768 and Kyber1024. Here the hardware implementation and security evaluation of Kyber768 are the primary focus of this paper. In Kyber768, k, n, q and η are set to 3, 256, 3329 and 2, respectively.

B. Kyber Decryption

Kyber is a post-quantum KEM that bases its security on the MLWE problem. It follows the conventional construction method to build an IND-CPA PKE scheme and turns it into an IND-CCA KEM through the tweaked Fujisaki-Okamoto (FO) transform. For a full description of Kyber procedures, such as Kyber.CPAPKE and Kyber.CCAKEM, more details can be found at its original specification [2]. The decryption procedure gains more attention for key recovery, as internal functions act on the long-term private key.

As shown in Algorithm 1, the decryption procedure receives the secret key sk and a ciphertext c. Decode and Decompress functions convert the ciphertext c from a byte array into polynomials $\mathbf{u}$ and v, respectively. Also, the secret key is deserialized into a vector of polynomials $\hat{\mathbf{s}}$. Then, PWM between polynomial vector $\hat{\mathbf{s}}$ and $\hat{\mathbf{u}}$ are calculated, where the NTT process decreases time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(nlogn)$. Thereafter, the inverse NTT function transforms the result back to the time domain. Finally, the message is restored in the form of a binary string by Compress and Encode functions. The secret key $\hat{\mathbf{s}}$ is created by the key generation procedure, where CBD and NTT functions execute successively. CBD function samples $\mathbf{s} \in R_q^k$ from centered binomial distribution $\mathcal{B}_\eta$. Hence, each sub-key can be equal to one of the 5 different possible values in the range $[-2, 2]$ for Kyber768.

TABLE I: Comparison with existing works.

Work	Target	Leakage	Scheme.
Primas [3]†	Inverse NTT	EM	Software
Pessl [7]‡	NTT	Power	Software
Ravi [8]‡	Message decoding	EM	Software
Ravi [4]†	FO transformation	EM	Software
Sim [9]‡	Message encoding	Power	Software
Xu [5]†	Inverse NTT	EM	Software
Sim [12]†	Modular reduction	Power	Software
Karlov [6]†	PWM	Power	Software
This work†	Two vulnerable regions*	Power, EM	Hardware

† denotes key recovery and ‡ denotes message recovery.
* includes PWM, modular reduction and functions after inverse NTT.

C. SCA on Kyber

SCA attacks have been widely viewed as a threat to cryptographic hardware. The same threat applies to PQC implementations also. Table I summarizes recent work about SCA attacks on Kyber. Primas et al. propose the SCA attack on variable-time implementations, which targets the inverse NTT in the decryption procedure [3]. For constant-time deployments, they introduce a more practical attack on the NTT running in the encryption procedure [7]. Ravi et al. report vulnerabilities of the message decoding function during the decryption procedure, which are exploited by template attacks for message recovery [8]. They also check the FO transform procedure, in which side-channel vulnerabilities assist full key recovery [4]. Sim et al. exploit the sensitive bit value of the message encoding function as a determiner, recovering the whole secret message in the encryption procedure [9]. Xu et al. magnify the leakage differences from inverse NTT by carefully chosen ciphertexts [5]. With a similar approach, Sim et al. magnify the leakage differences of the modular reduction function [12]. Both work break the entire secret key for the decryption procedure. A CPA attack is proposed in [6], on the basis of polynomial multiplications in the decryption procedure. However all the above attacks target software implementations, there are only a few attempts to attack hardware implementations. Recently, Park et al. detect leakage of Kyber hardware through test vector leakage assessment (TVLA), but the actual key recovery is not involved [13]. In this paper, we perform SCA attacks on Kyber's hardware designs, which

TABLE II: Detailed operations in Kyber-HDL design [10].

operation	parallelism/cycles
receive $c = c_1 \| c_2$	$-/713$
$\mathbf{u}_0 \leftarrow \text{Decompress}(c_1),\ \hat{\mathbf{u}}_0 \leftarrow \text{NTT}(\mathbf{u}_0)$	$2, 4/576$
$acc \leftarrow \hat{\mathbf{u}}_0 \cdot \hat{\mathbf{s}}_0 + 0$	$2/256$
$\mathbf{u}_1 \leftarrow \text{Decompress}(c_1),\ \hat{\mathbf{u}}_1 \leftarrow \text{NTT}(\mathbf{u}_1)$	$2, 4/576$
$acc \leftarrow \hat{\mathbf{u}}_1 \cdot \hat{\mathbf{s}}_1 + acc$	$2/256$
$\mathbf{u}_2 \leftarrow \text{Decompress}(c_1),\ \hat{\mathbf{u}}_2 \leftarrow \text{NTT}(\mathbf{u}_2)$	$2, 4/576$
$\hat{us} \leftarrow \hat{\mathbf{u}}_2 \cdot \hat{\mathbf{s}}_2 + acc$	$2/256$
$us \leftarrow \text{INTT}(\hat{us})$	$4/448$
$v \leftarrow \text{Decompress}(c_2)$	$2/128$
$m \leftarrow \text{Encode}(\text{Compress}(v - us))$	$2/128$

aim to recover the secret key from vulnerable regions of the decryption procedure.

III. Hardware Architecture

In this section, we introduce the two most recent hardware implementations of Kyber768, including manual design and HLS-based design.

A. Direct Implementation of Kyber

We first implement Algorithm 1 through direct hardware description language (HDL) coding [10], which is denoted as Kyber-HDL in the rest of the paper. Decode and Encode functions are realized by shift registers in Kyber-HDL. Functions like Decompress, NTT, PWM, inverse NTT and Compress are regulated by two sets of butterfly units. These butterfly units have two input data pairs and corresponding output pairs, making parallel computations of the above functions feasible. Their data paths, degrees of parallelism and occupied cycles could be found in Table II. As shown, Kyber-HDL executes Decompress and NTT functions to obtain polynomial vector $\hat{\mathbf{u}}_0$ within 576 cycles. The PWM is assigned to numerous operators of two butterfly units and accomplished in 256 cycles. In particular, data pairs are computed in parallelism level of 2 by two multipliers. To avoid data overflow, the Barrett reduction is applied to these products in a pipelined manner. The same computations will repeat three times for elements of polynomial vectors $\hat{\mathbf{s}} = (\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2)$ and $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2)$. Inverse NTT function transforms 4 elements of $\hat{us}$ in parallel, and obtains the variable us after 448 cycles. With degrees of parallelism 2, Decompress and Compress functions are both complete within 128 cycles.

We also retain other operations that are less relevant for decryption, such as hash functions and quotient functions. Since these functions occur during the decryption procedure of the KEM protocol, their behaviors have an impact on the security of the decryption itself. For Xilinx Spartan-6 FPGA series, Kyber-HDL consumes 9679 LUTs, 4113 Flip-Flops (FFs), 3 BRAMs and 0 DSPs[1]. Among all computations, hash functions occupy heavy proportions of total resource consumption containing 6144 LUTs and 2042 FFs.

[1]The utilization reports of Kyber designs include the resource of the RS-232 serial communication circuit.

TABLE III: Detailed multiplications in Kyber-HLS design.

operation	parallelism/cycles
$\hat{u}_0[2i{+}1] \cdot \hat{s}_0[2i{+}1]$	$1/1$
$\hat{u}_0[2j{+}1] \cdot \hat{s}_0[2j{+}1]$	$1/1$
$\hat{u}_0[2i] \cdot \hat{s}_0[2i],\ \hat{u}_0[2i] \cdot \hat{s}_0[2i{+}1],\ \hat{u}_0[2i{+}1] \cdot \hat{s}_0[2i]$	$3/1$
$\hat{u}_0[2j] \cdot \hat{s}_0[2j]$	$1/1$
$\hat{u}_0[2j] \cdot \hat{s}_0[2j{+}1],\ \hat{u}_0[2j{+}1] \cdot \hat{s}_0[2j]$	$2/1$

B. Kyber Implementation through HLS

The second hardware implementation is built through the HLS design flow [11] and is denoted as Kyber-HLS in the rest of the paper. Kyber-HLS exploits pipeline, inline and dependence directives for various loops and functions, and applies allocation directives to limit the number of DSPs used by the implementation. The combinations of various directives provide relative optimum speed and cost trade-offs. However, there remain some issues in RTL codes that hinder proper implementation. These issues are classified into redundant logic, the conflict between conditional and assignment statements and improper initialization for FFs and RAMs. To solve these, we analyze their code styles and model respective features as regular expressions. Through reviewing the RTL code, we can easily locate all these issues in accordance with regular expressions. Then we remove large quantities of redundant logic and replace the conflicting logic of conditional statements with the equivalent signal. The FFs and RAMs will be initialized as default state 0 instead of the unknown state.

The structure of Kyber-HLS is similar to Kyber-HDL except for the PWM function. We set the initiation interval of the pipeline as 8 and limit the instance of multiplication to less than 4. Table III shows the detailed operations, degree of parallelism and occupied cycles of PWM function, where $i \in [0, 127]$ and $j = i + 128$. Multiplications between 4 elements occupy 5 clock cycles and the total PWM takes about 64 cycles for $\hat{\mathbf{s}}_0$ and $\hat{\mathbf{u}}_0$. The degree of parallelism ranges from 1 to 3 during those clock cycles. For Xilinx Spartan-6 FPGA series, Kyber-HLS occupies 4496 LUTs, 3888 FFs, slices, 6 BRAMs and 22 DSPs. Note that we do not include the Encode and Decode functions in this design. To ensure correctness, input stimuli are pre-processed and then delivered to Kyber-HLS.

IV. Security Analysis

In this section, we discuss two vulnerable regions of Kyber and corresponding security analysis methods. The vulnerable regions are defined based on the computation domain, i.e., the NTT domain and time domain.

A. Adversary Model

The attack targets long-term keys in the decryption procedure of Kyber in PKE/KEMs (see Algorithm 1). Once the secret key is revealed, the cryptographic device will be at risk of losing confidentiality. In our threat model, we assume that an adversary can forge ciphertexts and record power or EM traces from target devices. We also assume that the adversary can locate sub-traces related to target functions. Further, the adversary can determine the data flow, parallelization and

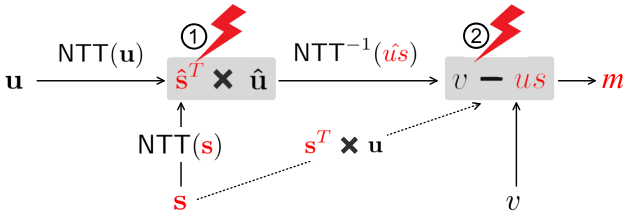


Fig. 1: Vulnerable regions in the decryption procedure.

pipeline of the hardware design. On the basis of this knowledge, the adversary will build hypothetical leakage models and apply SCA attacks around selected clock cycles.

B. First Vulnerable Region

As shown in Figure 1, the first vulnerable region is hardware implementations of $\hat{\mathbf{s}}^T \circ \hat{\mathbf{u}}$, since they are functions of the secret key and known variables. Hence, the PWM function forms the first PoI called point1 to execute SCA attacks. Though multiplications of the secret key are calculated in the NTT domain, the recovered polynomials can be converted into the normal domain by the inverse NTT function. In addition, the modular reduction function on product results constitutes the secondary PoI, denoted as point2. The above results are stored in either sequential or combinational logic, where state transitions produce exploitable side-channel leakage.

Attacks on microcontrollers always use the Hamming weight of target variables as the leakage model [3], [7], [9], [6], [12]. Here we prefer Hamming distance model to imitate the side-channel behavior of hardware implementations. For i-th PWM function, traces at point1 are formulated to the Hamming distance between front and back results, as written in Equation (1).

$$L(\mathsf{point1}_i) = \alpha \times \mathrm{HD}((\hat{\mathbf{s}}^T \circ \hat{\mathbf{u}})[i]) + \mathcal{N} \quad (1)$$

$$L(\mathsf{point2}_i) = \alpha \times \mathrm{HD}((\hat{\mathbf{s}}^T \circ \hat{\mathbf{u}} \bmod q)[i]) + \mathcal{N} \quad (2)$$

where α denotes the scaling factor and $\mathcal{N}$ represents a Gaussian noise term. Note that Hamming distance model also alleviates false positives caused by similar sub-keys (e.g., $0x06c$ and $0x1b0$) at point1. While point2 does not suffer from this issue due to the result of subtraction [14]. We compute the leakage model of point2 according to Equation (2), in which mod denotes the modular reduction operation.

C. Second Vulnerable Region

As shown in Figure 1, the second vulnerable region starts with the inverse NTT function of Kyber. It brings the result $\hat{us}$ in the NTT domain back to the time domain, in which equal values can be obtained by classical polynomial multiplication $us = \mathbf{s}^T\mathbf{u}$. The above property is defined in Equation (3), allowing adversaries to manipulate values of us by chosen ciphertexts. Taking $\mathbf{u} = (1, 0, 0)$ for instance[2], since Kyber computes us by $\sum_j^k (s_j \cdot u_j)$, the final output is equal to $\mathbf{s}_0$ in this case. Considering the logic update of i-th us, traces at this clock cycle is proportional to the Hamming distance of $s_0[i] \bmod q$ and $s_0[i] \in [-2, -1, 0, 1, 2]$. Therefore, the adversary can choose ciphertexts to remove the influence of $\mathbf{s}_1$ and $\mathbf{s}_2$, in the form of $\mathbf{u} = (a, 0, 0)$ and a denotes constant term. Assisted by SCA attacks, the adversary directly steals one-third of the secret key cycle by cycle.

$$\mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \hat{\mathbf{u}}) := \mathbf{s}^T\mathbf{u} \bmod q \quad (3)$$

Note that the input variables of decryption Algorithm 1 are ciphertexts c. They are created in the encryption procedure, where Compress and Encode functions act on $\mathbf{u}$ and v. However, due to data loss caused by compression, the decryption procedure may restore approximations of $\mathbf{u}$ and v, instead of their desired values. To ensure the bijection between desired values and received values, subset $\{\lceil (3329/2^{10}) \cdot i \rfloor : i = 0, 1, ..., 1023\}$ and subset $\{\lceil (3329/2^4) \cdot i \rfloor : i = 0, 1, ..., 7\}$ are built for coefficients of $\mathbf{u}$ and v, respectively. These two subsets provide all possible coefficient values for $\mathbf{u}$ and v in the decryption procedure. Hence there are 1024 possible chosen ciphertexts for the inverse NTT function. While for subtractions following this function, the introduction of v leads to a larger space of chosen ciphertexts. Here we describe the leakage model of subtractions namely point3 as follows.

$$L(\mathsf{point3}_i) = \alpha \times \mathrm{HD}((v - \mathbf{s}^T\mathbf{u} \bmod q)[i]) + \mathcal{N} \quad (4)$$

D. Analysis Method

We use the correlation analysis in CPA to evaluate the security of PoIs in two vulnerable regions. The Pearson correlation coefficient serves as a distinguisher that quantifies the linear relation between actual traces W_i and the leakage model H_i $(1 \le i \le N)$. For j-th key hypothesis with N traces, this distinguisher ρ_{WL_j} can be computed by Equation (5).

$$\rho_{WL_j} = \frac{N\sum_1^N W_i L_{i,j} - \sum_1^N W_i \sum_1^N L_{i,j}}{\sqrt{N\sum_1^N W_i^2 - (\sum_1^N W_i)^2}\sqrt{N\sum_1^N L_{i,j}^2 - (\sum_1^N L_{i,j})^2}} \quad (5)$$

The recovered key with the maximum Pearson correlation coefficient ρ_{max} indicates the most possible correct hypothesis. When the number of traces exceeds a certain value, the ρ_{max} of the correct hypothesis will always be greater than those of wrong guesses. This certain value often named measurement to disclosure (MTD), denotes the minimum traces to disclosure of the key.

V. Experimental Results

A. Experimental Setup

Figure 2 illustrates the overview of the experiment setup. Our experiment runs on the widely-used hardware security evaluation platform SAKURA-G. The Kyber design is deployed on the main FPGA known as a Spartan-6 XC6SLX75 FPGA. Its data communication to the PC is finished by the control Spartan-6 XC6SLX9 FPGA. Both Kyber-HDL and Kyber-HLS designs execute the decryption procedure at 20 MHz clock frequency. The Langer RF-U 5-2 probe with a PA303 pre-amplifier is used to collect EM traces. SMA connector J3 in the SAKURA-G board is exploited to monitor

[2]Coefficients $u_0[0] = 1$ and $u_0[i] = 0, i \in [1, 256)$

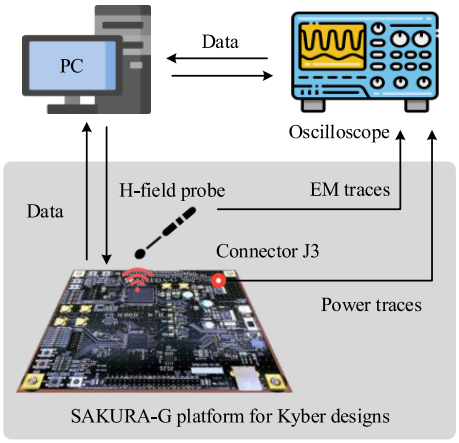


Fig. 2: The overview of the experimental setup.

power traces amplified by a AD8000 pre-amplifier. The collected signals are sampled at 2.5 GSa/s by the oscilloscope with 500 MHz bandwidth. Note that we adjust the probe location in advance until the amplitude of EM traces in the oscilloscope reaches the maximum. Then we take the average of 32 measurements (with the same input stimuli) aligned in the time domain as the final data.

B. Result on Kyber-HDL

First Vulnerable Region. As the PWM and modular reduction functions operate in parallelism 2, we attempt to attack two sub-keys respectively under parallel noise. The entire secret key can be recovered during 128 clock cycles by successive SCA attacks. We first evaluate the security of EM traces at the Point1. Figure 3 (a) shows the correlation traces as a function of time points for all sub-key candidates. The correct sub-key shows obvious peaks 0.15 that stand out from incorrect sub-key guesses. Results indicate that its side-channel behaviors correlate with leakage models $L(\mathsf{point1}_i)$ used by the adversary. Hence we can recover the correct sub-key within 1310 traces, as shown in Figure 3 (c). Then the security of Point2 in this region is evaluated. Figure 3 (b) and (d) show the attack results on EM traces. We find that the correlation peak of the correct sub-key delays 4 clock cycles compared to the peak at Point1. This result is consistent with the structure of Barrett reduction that featured 4 cascaded FFs. Moreover, the correlation peak at Point2 is relatively small and submerges in incorrect key guesses until 2680 traces. This is because the residue is the truncation of the product output, which decreases trace discrepancy caused by data transitions.

Second Vulnerable Region. The inverse NTT function of Kyber-HDL design increases degrees of parallelism, 4 sub-keys contribute the distribution of traces together. We cannot recover the secret key from this function within 1024 chosen ciphertexts. While the subtraction and Compress functions have the same data structure as the PWM function. We select the modulus operator of subtraction results as Point3 for SCA attacks. Figure 5 shows attack results targeting sub-keys with the help of chosen ciphertexts. The correlation of the correct sub-key is over 0.50, while the correlation of other sub-key candidates is below 0.34. The correct sub-key emerges from other sub-key guesses after 20 traces, as evident in Figure 5 (b). Hence a full key recovery from Point3 needs around 60 traces.

TABLE IV: Power and EM attacks on Kyber-HDL.

Kyber-HDL	Point1		Point2		Point3	
	ρ_{max}	MTD	ρ_{max}	MTD	ρ_{max}	MTD
Power	0.17	913	0.10	1600	0.75	9
EM	0.15	1310	0.09	2680	0.50	20

Table IV lists the results of power SCA attacks at three PoIs of Kyber-HDL design. It seems that power traces are better to reflect the secret key of Kyber compared to collected EM traces. This happened because the Langer RF-U 5-2 probe has a lower resolution 5 mm, which receives noisy signals from other components of the FPGA board. With high SNR, power SCA attacks could recover the full secret key using fewer traces. Nonetheless, the above results have proved the Kyber hardware with parallel design is still vulnerable to both EM and power SCA attacks.

C. Result on Kyber-HLS

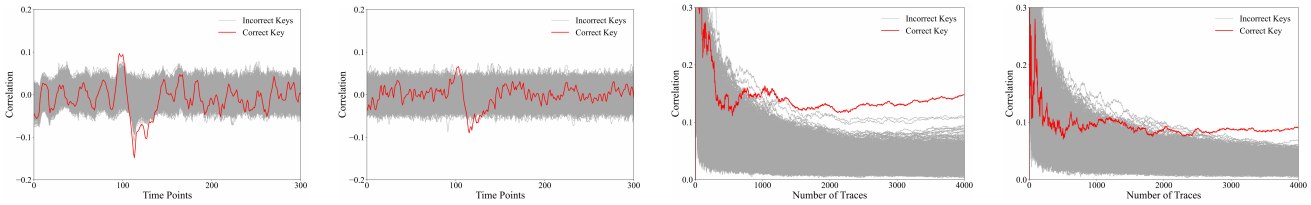
The PWM function in the Kyber-HLS design has a different structure from the Kyber-HDL design. In this case, we perform power SCA attacks on PWM, i.e., Point1 of Kyber-HLS design. Figures 4 (a) and (c) shows attack results on the clock cycle with only one multiplication. After 196 traces, the adversary can easily recover correct sub-keys with a correlation peak of 0.25. Attack results on the clock cycle that happens three multiplications are shown in Figures 4 (b) and (d). The correlation peak of the correct sub-key decreases to 0.15 and traces for key recovery increase to 515. Increased multiply operations result in extra noise and aggravate the difficulty of SCA attacks.

D. Discussion

Taking the PWM function in the NTT domain for instance, the most recent hardware implementations of Kyber improve security compared to simple microcontrollers (e.g, MTD $= 80$ in [6]). As parallel architectures result in lower SNR of leakage, Kyber-HLS with higher parallelism has increased security than the lower degree of parallelism. Moreover, the security of Kyber-HLS is comparatively low with Kyber-HDL. The principal reason is function pipelining and other irrelevant operations of Kyber-HDL, which obfuscate the data flow of PWM to some extent.

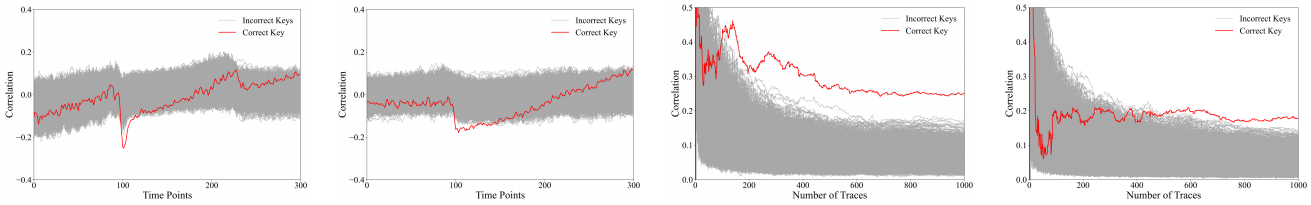
Kyber-HDL also lowers the success probability of key recovery from the time domain, compared to MTD $= 4$ in software platforms [5]. Especially for the inverse NTT function with parallelism 4, the adversary cannot recover the full key assisted by all possible ciphertexts. When the degree of parallelism keeps constant, Kyber-HDL is more vulnerable in the time domain relative to the NTT domain. This is because the property defined in Equation (3) reduces the search space of the secret key.

Although hardware designs improve security, they still leak sufficient information for SCA attacks and call for counter-measures about the NTT domain and time domain. Up to



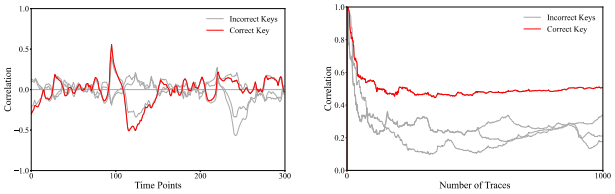
(a) Point1: correlation vs time points (b) Point2: correlation vs time points (c) Point1: correlation vs N traces (d) Point2: correlation vs N traces

Fig. 3: Attack results of Kyber-HDL in the first vulnerable region.



(a) correlation vs time points (b) correlation vs time points (c) correlation vs N traces (d) correlation vs N traces

Fig. 4: Attack results of Kyber-HLS on Point1 in the first vulnerable region.



(a) correlation vs time points (b) correlation vs N traces

Fig. 5: Results of Kyber-HDL in the second vulnerable region.

now, hardware designs of Kyber with protections are still rare [15]. We believe that our work help better understands the vulnerabilities existing in the hardware designs of Kyber.

VI. Conclusion

This paper evaluates the side-channel security of Kyber's hardware implementations with different architectures. We make a comprehensive analysis of their decryption procedure, including two vulnerable regions and multiple points of interest. Correspondingly, different types of SCA attacks are exploited to recover the secret keys of Kyber. Actual measurements demonstrate that parallelized designs can be broken with $27 \sim 1600$ power traces or $60 \sim 2680$ EM traces. Moreover, results suggest that functions after inverse NTT are more vulnerable against SCA attacks.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (Grant No. 2021YFB3100903), and the National Natural Science Foundation of China (Grant No. 62004112).

References

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

[2] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation," https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions, 2020.

[3] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 513–533.

[4] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs." *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 3, pp. 307–335, 2020.

[5] Z. Xu, O. M. Pemberton, S. S. Roy, D. Oswald, W. Yao, and Z. Zheng, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: the case study of kyber," *IEEE Transactions on Computers*, 2021.

[6] A. Karlov and N. L. de Guertechin, "Power analysis attack on Kyber," *Cryptology ePrint Archive*, 2021.

[7] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2019, pp. 130–149.

[8] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "Drop by drop you break the rock-exploiting generic vulnerabilities in lattice-based PKE/KEMs using EM-based physical attacks," *Cryptology ePrint Archive*, 2020.

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

[10] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 328–356, 2021.

[11] T. Zijlstra, K. Bigou, and A. Tisserand, "Lattice-based cryptosystems on FPGA: Parallelization and comparison using HLS," *IEEE Transactions on Computers*, 2021.

[12] B.-Y. Sim, A. Park, and D.-G. Han, "Chosen-ciphertext clustering attack on CRYSTALS-KYBER using the side-channel leakage of Barrett reduction," *IEEE Internet of Things Journal*, 2022.

[13] J. Park, N. N. Anandakumar, D. Saha, D. Mehta, N. Pundir, F. Rahman, F. Farahmandi, and M. M. Tehranipoor, "PQC-SEP: Power side-channel evaluation platform for post-quantum cryptography algorithms." *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 527, 2022.

[14] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing, "An efficient non-profiled side-channel attack on the CRYSTALS-Dilithium post-quantum signature," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*. IEEE, 2021, pp. 583–590.

[15] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, "A configurable crystals-kyber hardware implementation with side-channel protection," *Cryptology ePrint Archive*, 2021.