Side Channel Security Oriented Evaluation and Protection on Hardware Implementations of Kyber

Yiqiang Zhao, Shijian Pan, Haocheng Ma, Ya Gao, Xintong Song, Jiaji He, and Yier Jin, *Senior Member, IEEE*

***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 those designed 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 $27 \sim 1,600$ power traces or $60 \sim 2,680$ EM traces to recover the decryption key successfully. Furthermore, we propose two first-order IND-CPA Kyber decapsulation masking protected designs, and then we evaluate their securities and overheads. The experimental results demonstrate that the side channel security of masked Kyber designs has increased by more than 10x.**

***Index Terms*— CRYSTALS-Kyber, side channel analysis, Kyber hardware implementations, masking.**

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.

The primary usage of PKE/KEMs is to ensure the security of the key exchange using the public and private key pairs. 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 (INTT) 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, and SCA attacks against the 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 number theoretic transform (NTT) domain and time domain, the decryption procedure is classified into two vulnerable regions, involving multiple points of interest (PoI) like point-wise multiplication (PWM), modular reduction, and functions after INTT. Different strategies of SCA attacks are designed on these

Manuscript received 13 April 2023; revised 6 June 2023; accepted 14 June 2023. Date of publication 7 July 2023; date of current version 18 December 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3100903 and in part by the Natural Science Foundation of Tianjin under Grant 22JCQNJC00970. This article was recommended by Associate Editor P. Zhou. *(Corresponding authors: Jiaji He; Yier Jin.)*

Yiqiang Zhao, Shijian Pan, Haocheng Ma, Ya Gao, Xintong Song, and Jiaji He are with the School of Microelectronics, Tianjin University, Tianjin 300072, China (e-mail: dochejj@tju.edu.cn).

Yier Jin is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: yier.jin@ece.ufl.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2023.3288600.

Digital Object Identifier 10.1109/TCSI.2023.3288600

PoI 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. Further, potential countermeasures against SCA attacks are also discussed.

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 the leakage, designs with parallel architectures are still vulnerable to SCA attacks.
- We separate the decryption procedure into two vulnerable regions, in which all possible PoI 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 platform. Experimental results demonstrate that SCA attacks can recover secret keys from Kyber design with a few power and EM traces.
- We propose two first-order IND-CPA Kyber decapsulation masking designs based on Kyber-HDL, and then we evaluate their security and resource overheads.

The rest of the paper is organized as follows. Section II goes over the notations and description of Kyber and related works 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 PoI of Kyber. Section V demonstrates the actual results of power and EM SCA attacks on hardware designs. Section VI proposes two IND-CPA Kyber decapsulation masking designs, and security evaluation of the masked designs is performed. Finally, we conclude the paper in Section VII.

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 $\hat{s}$ 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.

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: $\hat{s} := \mathsf{Decode_{12}}(sk)$
4: $m := \mathsf{Encode_1}(\mathsf{Compress_q}(v - \mathsf{INTT}((\hat{s}^T \circ \mathsf{NTT}(\mathbf{u})), 1)$
                                   $\triangleright\ m := \mathsf{Compress_q}(v - \mathbf{s}^T\mathbf{u}, 1)$
5: return $m$
```

TABLE I
Comparison With Existing Works

Work	Target	Leakage	Scheme
Primas [3]†	INTT	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]†	INTT	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 INTT.

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 polynomial $\hat{s}$. Then, PWM between polynomial vector $\hat{s}$ and $\hat{u}$ are calculated, where the NTT process decreases time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(nlogn)$. Thereafter, the INTT 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 $\hat{s}$ 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.

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 works about SCA attacks on Kyber. Primas et al. propose the SCA attack on variable-time implementations, which targets the INTT 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 is 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 INTT by carefully chosen ciphertexts [5]. With a similar approach, Sim et al. magnify the leakage differences of the modular reduction function [12]. Both works 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, and 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 aim to recover the secret key from vulnerable regions of the decryption procedure.

D. Masking in Kyber

To mitigate the threat of SCA attacks on cryptographic hardware devices, traditional cryptography has developed various types of mature countermeasure strategies. One of the well-known strategies is masking, which can also be applied to Kyber. Table II summarizes the recent works about masking in Kyber. In the MLWE-based encryption and key exchange scheme, Schneider et al. optimize the Boolean to Arithmetic (B2A) and Arithmetic to Boolean (A2B) converters for prime modulus and propose two new masked binomial samplers, which are widely used by other researchers [14]. Bos et al. propose the first complete masked implementation for Kyber.CPAPKE decapsulation and exploit two new masked modules: masked one-bit compression and masked decompressed comparison [15]. At the same time, Fritzmann et al. published similar work [16].

For IND-CPA work, Ravi et al. exploit the efficient arithmetic properties of twiddle constants to mask NTT/INTT, and they built a general masked NTT/INTT scheme with configurable SCA resistance [17]. Hamoudi et al. discuss the difference between arithmetic masking and multiplicative masking for polynomial multiplication and propose that multiplicative masks have better performance [18]. For hardware implementations, Kamucheka et al. propose the first hardware implementation combining masking and hiding strategies [19]. They demonstrated that the proposed technology did not have a strong impact on resource utilization and performance, but its basic design is inefficient and resource intensive. However, the aforementioned masked designs do not include a more comprehensive masking cost and security assessment for the IND-CPA decapsulation. In this paper, we focus on the vulnerabilities that are demonstrated in the IND-CPA key recovery, so we implement and then compare the arithmetic masking and multiplicative masking for polynomial multiplications in the hardware. We aim to design low-overhead masked IND-CPA Kyber decapsulation designs.

TABLE II
COMPARISON WITH EXISTING MASKING WORKS

Work	Masked scheme	Scheme
Schneider [14]	A2B, B2A and binomial samplers	Software
Bos [15]	Boolean and Arithmetic	Software
Fritzmann [16]	Boolean and Arithmetic	Software
Ravi [17]	Multiplicative	Software
Hamoudi [18]	Additive and Multiplicative	Software
Kamucheka [19]	Boolean	Hardware
This work	Combined *	Hardware

* Include: multiplicative, arithmetic, and boolean masking.

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, INTT, 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 III. As shown, Kyber-HDL executes Decompress and NTT functions to obtain polynomial vector $\hat{u}_0$ 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 at a 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 $\hat{s} = (\hat{s}_0, \hat{s}_1, \hat{s}_2)$ and $\hat{u} = (\hat{u}_0, \hat{u}_1, \hat{u}_2)$. The INTT 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 9,679 LUTs, 4,113 Flip-Flops (FFs), 3 BRAMs and 0 DSPs.[1] Among all computations, hash functions occupy heavy proportions of total resource consumption containing 3,899 LUTs and 2,056 FFs.

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

TABLE III
DETAILED OPERATIONS IN KYBER-HDL DESIGN [10]

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

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 signals. The FFs and RAMs will be initialized as default state 0 instead of 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 IV shows the detailed operations, degree of parallelism, and occupied cycles of the 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 $\hat{s}_0$ and $\hat{u}_0$. The degree of parallelism ranges from 1 to 3 during those clock cycles. For Xilinx Spartan-6 FPGA series, Kyber-HLS occupies 4, 496 LUTs, 3, 888 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 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.

TABLE IV
DETAILED MULTIPLICATIONS IN KYBER-HLS DESIGN

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

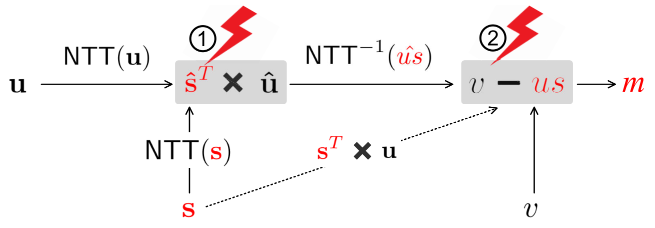


Fig. 1. Vulnerable regions in the decryption procedure.

B. First Vulnerable Region

As shown in Figure 1, the first vulnerable region is hardware implementations of $\hat{s}^T \circ \hat{u}$, 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 INTT 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], [6], [7], [9], [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).

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

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

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 [20]. We compute the leakage model of $\mathsf{point2}$ 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 INTT 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 $s_0[i]$ mod 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.

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

Note that the input variables of decryption Algorithm 1 are ciphertexts c. They are created in the encryption procedure, where $\mathsf{Compress}$ 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 $\{\lceil(3329/2^{10}) \cdot i\rfloor : i = 0, 1, \ldots, 1023\}$ and subset $\{\lceil(3329/2^4) \cdot i\rfloor : i = 0, 1, \ldots, 7\}$ 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 $1,024$ possible chosen ciphertexts for the INTT 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 $\mathsf{point3}$ as follows.

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

D. Analysis Method

We use the correlation analysis in CPA to evaluate the security of PoI 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).

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

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

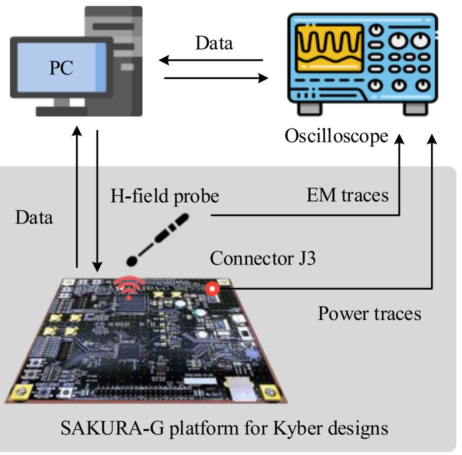


Fig. 2. The overview of the experimental setup.

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 the wrong guesses. This certain value often named measurement to disclosure (MTD), denotes the minimum traces to the disclosure of the key.

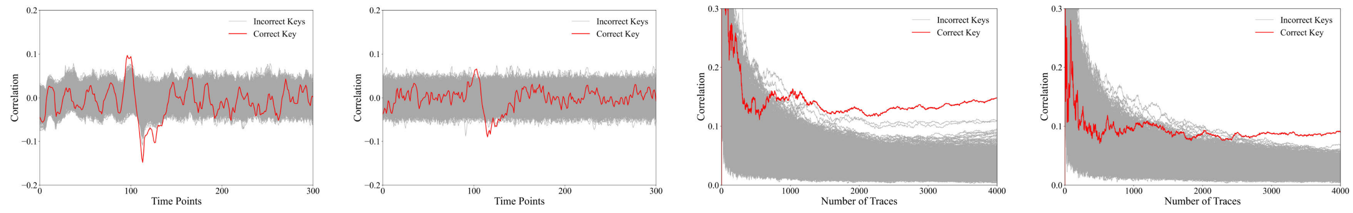
V. Experimental Results

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 power traces amplified by an 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. After locating the position, we place the probe as close to the chip surface as possible to achieve a higher SNR for the collected signals. Then we take the average of 32 measurements (with the same input stimuli) aligned in the time domain as the final data.

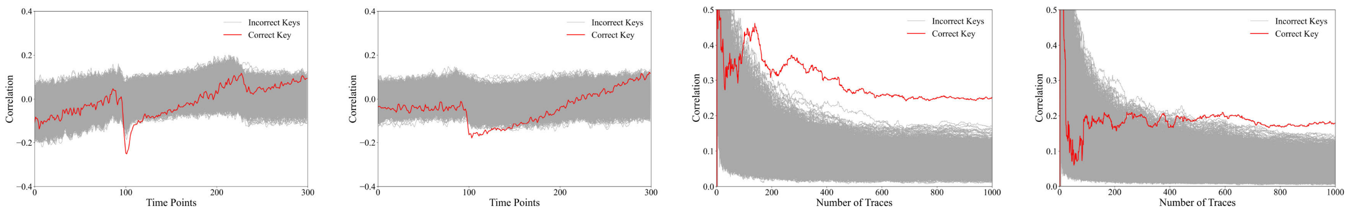
B. Result on Kyber-HDL

1) 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 $\mathsf{Point1}$. 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



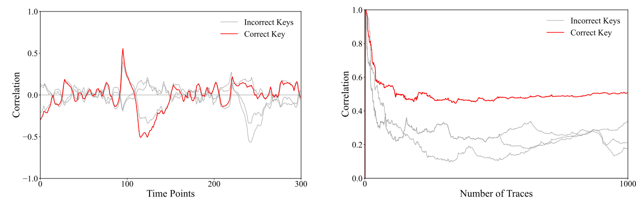
(a) Point1: correlation vs time points (b) Point2: correlation vs time points (c) Point1: correlation vs N traces (d) Point2: correlation vs N traces

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.

stand out from incorrect sub-key guesses. Results indicate that its side-channel behaviors correlate with leakage models $L(\text{point1}_i)$ used by the adversary. Hence, we can recover the correct sub-key within 1, 310 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 the incorrect key guesses until 2, 680 traces. This is because the residue is the truncation of the product output, which decreases trace discrepancy caused by data transitions.

2) Second Vulnerable Region: The INTT function of Kyber-HDL design increases degrees of parallelism, 4 sub-keys contribute to the distribution of traces together. We cannot recover the secret key from this function within 1, 024 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 the 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 V
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	1,600	0.75	9
EM	0.15	1,310	0.09	2,680	0.50	20

Table V lists the results of power SCA attacks at three PoI of Kyber-HDL design. It seems that power traces are better to reflect the secret key of Kyber compared to the 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) show attack results on the clock cycle with only one multiplication. After 196 traces, the adversary can easily recover the 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 multiple operations in parallel 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 compared with Kyber-HDL. The principal reason is function pipelining and other irrelevant operations in Kyber-HDL that obfuscate the data flow of the PWM function.

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 INTT 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 countermeasures about the NTT domain and time domain. Up to now, hardware designs of Kyber with protections are still rare [21]. We believe that our work helps better understand the vulnerabilities that exist in the hardware designs of Kyber.

VI. Masking Architecture

In this section, we introduce the algorithms and techniques applied to the implementation of masked Kyber.CPAPKE. We also discuss different masking schemes and security analysis in two vulnerable regions.

A. Arithmetic Masking

In the Kyber designs based on the MLWE scheme, the main functions are NTT, INTT, polynomial multiplication, and other polynomial operations. These functions are proved to be linear for arithmetic masking. The arithmetic shares of sensitive information $\mathrm{x} = \mathrm{x}_0 + \mathrm{x}_1$ can be calculated at the beginning, and each share is executed separately and in parallel.

In our design, we utilize a 32-bit Linear Feedback Shift Register (LFSR) to generate 128 random polynomial coefficients within $\mathbb{Z}_q$. To avoid creating additional clock overheads, we calculate the arithmetic shares $sk_0 = sk - R$, $sk_1 = R$ on the run-time of NTT(u). Each share executes the functions of PWM, $+$, INTT, $-$, and modular reduction in parallel. Note that we modify the $\mathsf{Compress}_q(v - s^T u, 1)$ in Kyber-HDL with better-masked scheme portability. As shown in Algorithm 2. The MSB returns the highest bit of the coefficient and q is the fixed prime modulus parameter of Kyber. Recent work [19] display table-based A2B, but the pre-computed table ROM is predictable huge. Hence, we employ A2B based on secure masked arithmetic addition over Boolean shares (SecAdd) refer to [16]. The SecAdd takes as inputs $\mathrm{x} = \mathrm{x}_0 \oplus \mathrm{x}_1$ and $\mathrm{y} = \mathrm{y}_0 \oplus \mathrm{y}_1$ such that $(\mathrm{B}_0 \oplus \mathrm{B}_1) = ((\mathrm{x}_0 \oplus \mathrm{x}_1) + (\mathrm{y}_0 \oplus \mathrm{y}_1))$. The A2B algorithm is shown in Algorithm 3. In [22], SecAdd has been extended to work with prime modulus. We learn that SecAdd_q makes two calls to SecAdd, which means we can implement $\mathsf{A2B}_q$ only by the SecAdd. With the exception of the A2Bq function extensional call SecAdd, working with prime modulus requires the SecAdd to replace the arithmetic operation during the Algorithm 2. The extensional call will also cause additional overhead. As a lower-overhead alternative, we employ the modified Transform2 function (see Algorithm 4) to convert coefficients to powers of two (refer to [23]), which avoids calculating in prime modulus. Compared with [23], we sacrifice the independence between (k_0,k_1) and (y_0,y_1) but retain the independence between (k_0,k_1) and the input. The introduction of too many random numbers and calculating expensive multiplications is avoided. Finally, the message can be recovered by $m = m_0 \oplus m_1$. The concrete IND-CPA Kyber first-order arithmetic masking implementation is shown in Figure 6. The addition symbol represents arithmetic sharing and the XOR symbol represents boolean sharing. The grey blocks indicate duplications of arithmetic units and storage units.

Algorithm 2 Masked Modified $\mathsf{Compress}_q(v - s^T u, 1)$

```
Input:  $\mathbf{A}^{\{0:1\}} = (A_0, A_1)$ such that $\mathbf{A} = A_0 + A_1 \bmod q$
Output: $\mathbf{M}^{\{0:1\}} = (m_0, m_1)$ such that $\mathbf{M} = m_0 \oplus m_1$
1: $A_0 := A_0 - \lfloor \frac{q}{4} \rfloor$
2: $[y_0, y_1] := \mathsf{Transform2}(A_0, A_1)$
3: $y_0 := y_0 - \lfloor \frac{q}{2} \rfloor$
4: $[y'_0, y'_1] := \mathsf{A2B}(y_0, y_1)$
5: $m_0 := \mathsf{MSB}(y'_0)$
6: $m_1 := \mathsf{MSB}(y'_1)$
7: return $(m_0, m_1)$
```

Algorithm 3 A2B Conversion

```
Input:  $\mathbf{A}^{\{0:1\}} = (A_0, A_1)$ such that $\mathbf{A} = A_0 + A_1 \bmod 2^k$
Output: $\mathbf{B}^{\{0:1\}} = (B_0, B_1)$ such that $\mathbf{B} = B_0 \oplus B_1$
1: $R_0, R_1 \leftarrow \mathbb{Z}_{2^k}$                          ▷ $k := \lceil log_2(q) \rceil + 1$
2: $x^{\{0:1\}} := (A_0 \oplus R_0, R_0)$                 ▷ $x^{\{0:1\}}$ are k bits shares
3: $y^{\{0:1\}} := (A_1 \oplus R_1, R_1)$                 ▷ $y^{\{0:1\}}$ are k bits shares
4: $B^{\{0:1\}} := \mathsf{SecAdd}(x^{\{0:1\}}, y^{\{0:1\}})$
5: return $B^{\{0:1\}}$
```

In arithmetic masking design, we not only duplicate the arithmetic unit but also the storage unit used to store multiplicative intermediate values and shares. These values and polynomial shares require a large number of registers for storage. Take the Kyber768 as an example, the private key sk requires two 128×12 bits RAMs, and the multiplicative intermediate values require one 128×48 bits RAM. In higher-order masking designs, the RAM overhead will become very large. One possible alternative is to change parallel processing to serial processing, but the clock overhead is unavoidable.

B. Combined Masking

In [18], the author discusses the feasibility of multiplicative masking in polynomial multiplications. Through masking sensitive information with $rsk = sk \times r$ and $rinv = r$, we can perform polynomial multiplication with rsk and the ciphertext alone. Note that we use the extended Euclidean algorithm to compute the multiplicative inverse root $rinv$. After computing the inverse root within $\mathbb{Z}_q$, the clock overhead of the multiplicative inverse root is at most 32 clock cycles per coefficient. By weighing the clock overhead against security, we propose

Algorithm 4 Modified Transform2 in $\text{Compress}_q(v - s^T u, 1)$

```
Input:  $\mathbf{A}^{\{0:1\}} = (A_0, A_1)$ such that $\mathbf{A} = A_0 + A_1 \bmod q$
Output: $\mathbf{B}^{\{0:1\}} = (B_0, B_1)$ such that $\mathbf{B} = B_0 + B_1 \bmod 2^k$
 1: $y_0 \leftarrow \mathbb{Z}_{2^k}$                ▷ $k := \lceil log_2(q) \rceil + 1$
 2: $y_1 := A_0 - y_0 + A_1$          ▷ $y_0, y_1$ are k bits shares
 3: $z_0 := y_0 - q$                  ▷ $z_0, z_1$ are k bits numbers
 4: $[z_0, z_1] := \text{A2B}(z_0, y_1)$
 5: $k_0 := \text{MSB}(z_0) \oplus 1$         ▷ $k_0, k_1$ are one bit numbers
 6: $k_1 := \text{MSB}(z_1)$
 7: $k := k_0 \bigoplus k_1$
 8: $B_0 := y_0 - kq$
 9: $B_1 := y_1$
10: return $(B_0, B_1)$
```

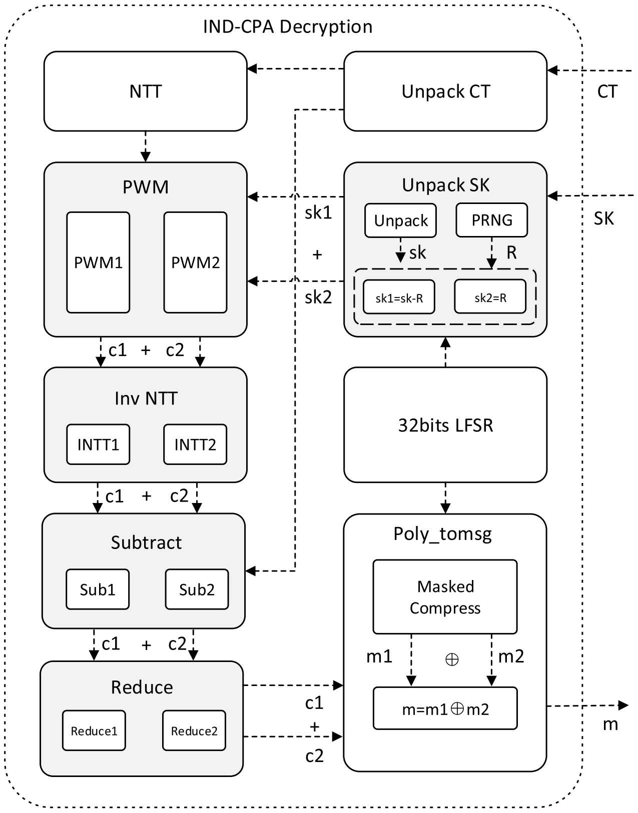


Fig. 6. First-order arithmetic masking for IND-CPA Kyber.

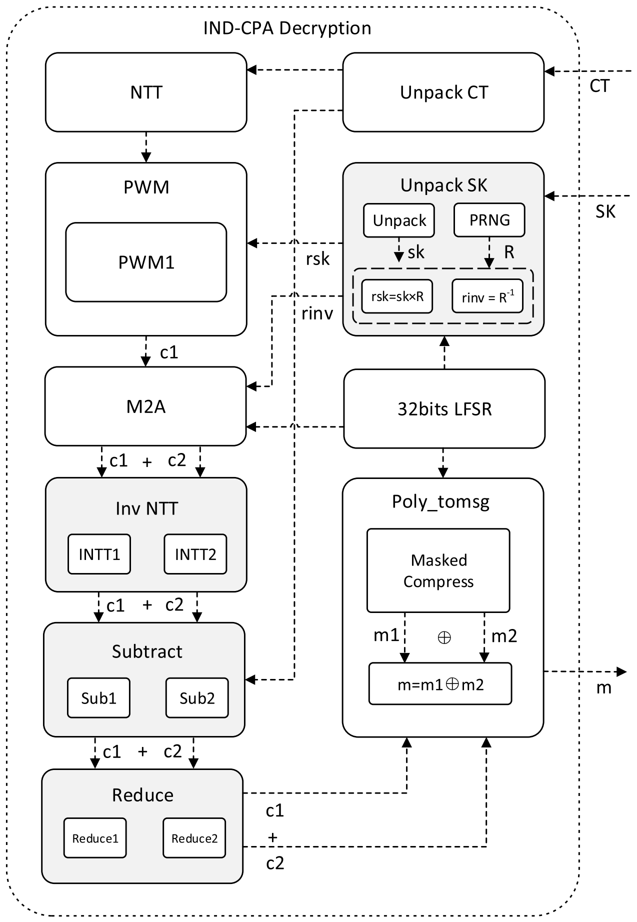


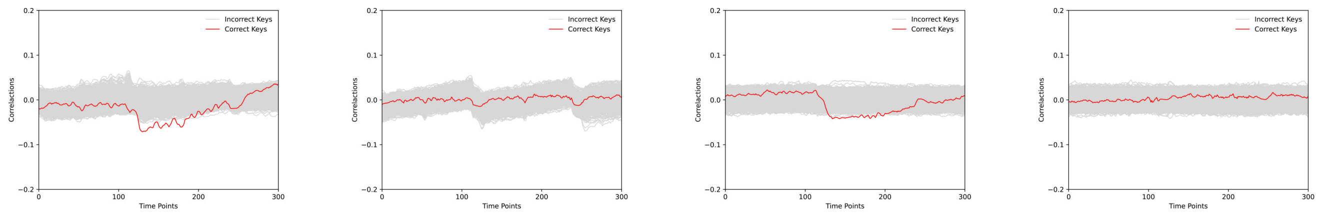
Fig. 7. First-order combined masking for IND-CPA Kyber.

creating 16 random multiplicative masks to cyclically mask 128 coefficients before the run-time of PWM. The security of the reuse of masks will be guaranteed by refreshing these masks before the masking operations. Further, we calculate $\sum_j^k (s_j \cdot u_j)$ and then employ the multiplicative to arithmetic (M2A) algorithm as described in [24]. In the subsequent INTT and compress functions, we follow the design of arithmetic masking to form our combined masking design. The implementation is shown in Figure 7. The addition symbol represents arithmetic sharing and the XOR symbol represents boolean sharing. The grey blocks indicate duplications of arithmetic units and storage units.

Compared with arithmetic masking, combined masking eliminates the duplicated arithmetic units and RAM for multiplicative intermediate values, but the method also introduces four multiplier DSPs for masking.

C. Security Analysis

1) Results on Arithmetic Masking: We first evaluate the security of 10,000 power traces at the Point1. Arithmetic masking shares take two PWM functions in parallel. Both shares are independent with the correct sub-key. Thus, we cannot recover the correct sub-key by the first-order SCA attack. Figure 8 (a) and (b) show the correlation traces of the arithmetic masking when the LFSR is off and on, respectively. The results indicate that the arithmetic masking has side-channel leakage when the LFSR is off and the leakage is eliminated when the LFSR is on. While the masking is working, $L(\text{point1}_i)$ behaves in a noise-dominated side-channel behavior and the adversary cannot recover the correct sub-key. Secondly, we evaluate the security of Point2. Figure 8 (c) and (d) show the attack results on 10,000 power traces when the LFSR is off and on, respectively. In Figure 8 (d), correct sub-key is fully covered up by the incorrect sub-key guesses. $L(\text{point2}_i)$ behaves in a noise-like side-channel behavior. In the second vulnerable region, Figure 10 (a) shows the correlation of the correct sub-key within 1,024 chosen ciphertexts. The incorrect peak leads to incorrect guessing results.



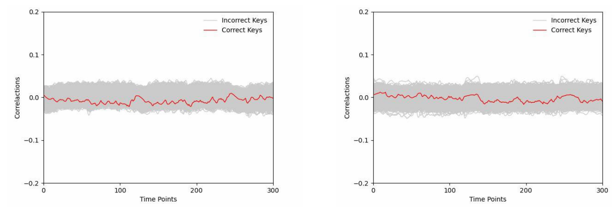
(a) Point1: correlation vs time points (b) Point1: correlation vs time points (c) Point2: correlation vs time points (d) Point2: correlation vs time points

Fig. 8. Attack results of Arithmetic Masking in the first vulnerable region.

TABLE VI
FPGA Resource Utilization and Clock Cycle Overhead

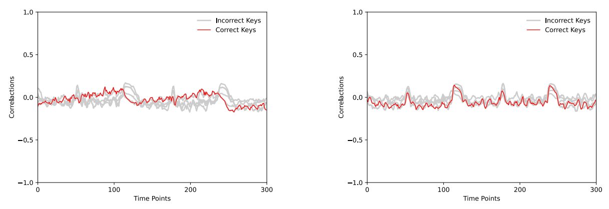
Implementation	Algorithm	FPGA	LUTs	FFs	Slices	BRAMs	Equivalent gate* (Slices+BRAMs) (thousands)	Resource overhead	Cycle count	Cycle count overhead
This work (Without Masking)	Kyber768	Spartan-6 XC6SLX75	7,653	5,074	2,313	11	481.1(222.0, 259.1)	-	10,000	-
This work (Arithmetic Masking)	Kyber768	Spartan-6 XC6SLX75	9,204	6,452	3,241	16	631.6(311.1, 320.5)	1.31x	10,000	0
This work (Combined Masking)	Kyber768	Spartan-6 XC6SLX75	9,905	6,695	3,334	11	579.1(320.0, 259.1)	1.20x	10,004	≈ 0
Kamucheka et el. (Hiding-Only) [19]	Kyber512	Virtex7 VC707	143,112	-	81,746	294	$> 7,847.6$	1.61x	126,619	1.83x
Kamucheka et el. (Hiding-Plus-Masking) [19]	Kyber512	Virtex7 VC707	152,860	-	92,977	489.5	$> 8,925.8$	1.72x	137,738	1.97x

* Equivalent gate can be caculate in the conversion of Slices = 6.4 logic cells = 96 gates and BRAMs = 4 gates/bit.



(a) Point1:correlation vs time points (b) Point2:correlation vs time points

Fig. 9. Results of Combined Masking in the first vulnerable region.



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

Fig. 10. Results of Arithmetic and Combined Masking in the second vulnerable region.

2) Results on Combined Masking: Unlike arithmetic masking which performs multiple PWM functions in parallel, combined masking only performs PWM function for masked *rsk*. Although the parallelism is reduced, the independence with *sk* still exists. The PWM function of combined masking is the same as that of the no masking implementation. Therefore, the attack result when the LFSR is off can be evaluated by Figure 3. Figure 9 (a) and (b) indicate the results of Point1 and Point2 when the LFSR is on. The incorrect sub-key guesses fully cover up the correct sub-key within 10,000 power traces. The adversary cannot recover the correct sub-key. Compared to Figure 3, the security is significantly improved. In the second vulnerable region, Figure 10 (b) shows a similar attack result as the arithmetic masking.

D. Discussion

In Table VI, we list the FPGA resource utilization and clock cycle overheads for the existing masked Kyber decapsulation hardware implementations. We include the unprotected FO transform in our design and turn it into the complete decapsulation implementation. To the best of our knowledge, Kamucheka's design has logic resource overheads of 1.6x to 1.7x and clock cycle overheads of 1.83x to 1.97x. Also, their design requires a clock count of greater than 120K at the clock frequency of 100MHz. In contrast, our design requires only a 10K clock count at a similar operating frequency. This expensive time overhead is not expected by the designer. In addition, their designs require too much logic resources and BRAM to be integrated in hardware circuits. Rather than focusing on logic resources, we are more concerned on the overhead of BRAM, which has a greater impact on area. Hence, we propose two low-area masking hardware implementations. They both perform complete masking of the IND-CPA Kyber decapsulation with 1.20x to 1.31x resource overhead and zero clock cycle counts overhead. In our design, arithmetic masking has lower logic resource overheads and higher parallelism, while combined masking add some of the logic resource overhead in exchange for 5 large BRAMs. Both masking schemes exhibit resistance under first-order SCA attacks. Our design can be easily extended to higher-order implementations and our masking techniques can be applied to other Kyber hardware implementations. The proposed attack points and protection schemes are applicable to potential

PQC algorithms because the latticed-based algorithms mostly have similar structures. Furthermore, FO transform will cause the short-term side channel leakage of the private messages directly but will not cause the long-term side channel leakage of the private key. Hence, we will extend more security enhancements to this part in our future work.

VII. 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 $27 \sim 1,600$ power traces or $60 \sim 2,680$ EM traces. Moreover, results suggest that functions after inverse NTT are more vulnerable to SCA attacks. Finally, we propose two first-order masked Kyber implementations of the decryption procedure. Result indicates that the MTD of point1/point2/point3 attack points is >10K/>10K/>1K. The increase in the MTD index demonstrates that the side channel security of our designs has increased 10x when compared with the MTD in Table V.

References

[1] P. Ravi, J. Howe, A. Chattopadhyay, and S. Bhasin, "Lattice-based key-sharing schemes: A survey," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–39, Jan. 2022.

[2] R. Avanzi et al. (2020). *CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions

[3] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *Cryptographic Hardware and Embedded Systems—CHES 2017: 19th International Conference, Taipei, Taiwan, September 25–28, 2017, Proceedings*. 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. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 307–335, Jun. 2020.

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

[6] A. Karlov and N. L. de Guertechin, "Power analysis attack on Kyber," Cryptol. ePrint Arch., Paper 2021/1311, 2021.

[7] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *Progress in Cryptology—LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings 6*. Springer, 2019, pp. 130–149.

[8] P. Ravi et al., "Drop by drop you break the rock-exploiting generic vulnerabilities in lattice-based PKE/KEMs using EM-based physical attacks," IACR Cryptol. ePrint Arch., Paper 2020/549, 2020.

[9] B. Sim et al., "Single-trace attacks on message encoding in lattice-based KEMs," *IEEE Access*, vol. 8, pp. 183175–183191, 2020.

[10] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 2, pp. 328–356, Feb. 2021.

[11] T. Zijlstra, K. Bigou, and A. Tisserand, "Lattice-based cryptosystems on FPGA: Parallelization and comparison using HLS," *IEEE Trans. Comput.*, vol. 71, no. 8, pp. 1916–1927, Aug. 2022.

[12] B. Sim, A. Park, and D. Han, "Chosen-ciphertext clustering attack on CRYSTALS-KYBER using the side-channel leakage of Barrett reduction," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21382–21397, Nov. 2022.

[13] J. Park et al., "PQC-SEP: Power side-channel evaluation platform for post-quantum cryptography algorithms," *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 527, Jan. 2022.

[14] T. Schneider, C. Paglialonga, T. Oder, and T. Güneysu, "Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto," in *Proc. IACR Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2019, pp. 534–564.

[15] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. Van Vredendaal, "Masking Kyber: First- and higher-order implementations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 4, pp. 173–214, Aug. 2021.

[16] T. Fritzmann et al., "Masked accelerators and instruction set extensions for post-quantum cryptography," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, no. 1, pp. 414–460, Nov. 2021.

[17] P. Ravi, R. Poussier, S. Bhasin, and A. Chattopadhyay, "On configurable SCA countermeasures against single trace attacks for the NTT: A performance evaluation study over Kyber and Dilithium on the arm Cortex-M4," in *Proc. 10th Int. Conf., Secur., Privacy, Appl. Cryptogr. Eng.* Kolkata, India: Springer, Dec. 2020, pp. 123–146.

[18] M. Hamoudi, A. B. Korchi, S. Guilley, S. Takarabt, K. Karray, and Y. Souissi, "Side-channel analysis of CRYSTALS-Kyber and a novel low-cost countermeasure," in *Proc. 2nd Int. Conf., Secur. Privacy (ICSP)*. Jamshedpur, India: Springer, Nov. 2021, pp. 30–46.

[19] T. Kamucheka, A. Nelson, D. Andrews, and M. Huang, "A masked pure-hardware implementation of Kyber cryptographic algorithm," Cryptol. ePrint Arch., Paper 2022/1547, 2022.

[20] 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 *Proc. IEEE 39th Int. Conf. Comput. Design (ICCD)*, Oct. 2021, pp. 583–590.

[21] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, "A configurable Crystals-Kyber hardware implementation with side-channel protection," Cryptol. ePrint Arch., Paper 2021/1189, 2021.

[22] G. Barthe et al., "Masking the GLP lattice-based signature scheme at any order," in *Proc. 37th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Tel Aviv, Israel: Springer, May 2018, pp. 354–384.

[23] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical CCA2-secure and masked ring-LWE implementation," Cryptol. ePrint Arch., Paper 2016/1109, 2016.

[24] M. Krausz, G. Land, J. Richter-Brockmann, and T. Güneysu, "Efficiently masking polynomial inversion at arbitrary order," in *Post-Quantum Cryptography: 13th International Workshop, PQCrypto 2022, Virtual Event, September 28–30, 2022, Proceedings*. Cham, Switzerland: Springer, 2022, pp. 309–326.

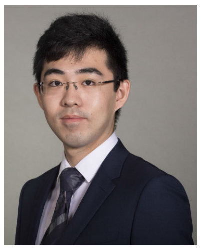
Yiqiang Zhao received the B.S. degree in semiconductor physics and device, the M.S. degree in microelectronics, and the Ph.D. degree in microelectronics and solid-state electronics from Tianjin University, Tianjin, China, in 1988, 1991, and 2006, respectively.

In 1991, he joined the Jinhang Technical Physics Institute, Tianjin, where he was responsible for analog and mixed signal circuit design. Since 2001, he has been with the School of Electronic Information Engineering and the School of Microelectronics, Tianjin University, where he is currently a Professor. His research interests include mixed-signal integrated circuits, security chips, and hardware security.

Shijian Pan received the bachelor's degree in microelectronics from Tianjin University in 2021, where he is currently pursuing the master's degree with the Microelectronics School. His current research interests include post-quantum cryptography, hardware security, and side channel analysis.

Haocheng Ma received the B.S. degree in microelectronics from Tianjin University, Tianjin, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Microelectronics. His current research interests include digital circuit design, hardware security, and EDA for security.

Ya Gao received the bachelor's degree in microelectronics from Tianjin University in 2020, where she is currently pursuing the Ph.D. degree with the Microelectronics School. Her current research interests include hardware security, side channel analysis, and machine learning.



Xintong Song received the master's degree in chemical engineering from the Georgia Institute of Technology and the master's degree in computer engineering from New York University. He is currently pursuing the Ph.D. degree with the School of Microelectronics, Tianjin University. Prior to pursuing his Ph.D. degree, he was a System Software Engineer with Sina Cooperation. His research interests include fully homomorphic encryption (FHE), post-quantum cryptography (PQC), and cryptography software and hardware co-design and optimization techniques. He was honored with the Exemplary Academic Achievement Award at the Georgia Institute of Technology in 2017. He also received the Exemplary Academic Achievement Award in 2020.

Jiaji He received the B.S. degree in electronic science and technology and the M.S. and Ph.D. degrees in microelectronics from Tianjin University, Tianjin, China, in 2013, 2015, and 2019, respectively. He was a Visiting Scholar with UCF and UF from 2016 to 2018. Before joining Tianjin University as an Associate Professor, he spent two years as a Post-Doctoral Research Fellow with the Institute of Microelectronics, Tsinghua University, from 2019 to 2021. His research interests include digital circuit design, hardware security, and EDA for security.

Yier Jin (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, China, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from Yale University in 2012. He is currently an Associate Professor and the IoT Term Professor with the Department of Electrical and Computer Engineering (ECE), University of Florida (UF). His research interests include hardware security, embedded systems design and security, trusted hardware intellectual property (IP) cores, hardware-software co-design for modern computing systems, security analysis on the Internet of Things (IoT), and wearable devices with particular emphasis on information integrity and privacy protection in the IoT era. He was a recipient of the DoE Early CAREER Award in 2016 and the ONR Young Investigator Award in 2019. He received Best Paper Award at DAC'15, ASP-DAC'16, HOST'17, ACM TODAES'18, GLSVLSI'18, and DATE'19. He is also the IEEE Council on Electronic Design Automation (CEDA) Distinguished Lecturer.