



<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2023 IEEE European Test Symposium (ETS'23)*.

Citation for the original published paper:

Ji, Y., Wang, R., Ngo, K., Dubrova, E. (2023)

A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber

In: *Proceedings of the 2023 IEEE European Test Symposium (ETS'23)*

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-324662>

A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber

Yanning Ji, Ruize Wang, Kalle Ngo, Elena Dubrova, Linus Backlund

KTH Royal Institute of Technology

Stockholm, Sweden

{yanning,ruize,kngo,dubrova,lbackl}@kth.se

Abstract—CRYSTALS-Kyber has been recently selected by the NIST as a new public-key encryption and key-establishment algorithm to be standardized. This makes it important to assess how well CRYSTALS-Kyber implementations withstand side-channel attacks. Software implementations of CRYSTALS-Kyber have already been analyzed and the discovered vulnerabilities were patched in the subsequently released versions. In this paper, we present a profiling side-channel attack on a hardware implementation of CRYSTALS-Kyber. Since hardware implementations carry out computations in parallel, they are typically more difficult to break than their software counterparts. We demonstrate a successful message (session key) recovery attack on a Xilinx Artix-7 FPGA implementation of CRYSTALS-Kyber by deep learning-based power analysis. Our results indicate that currently available hardware implementations of CRYSTALS-Kyber need better protection against side-channel attacks.

Index Terms—Post-quantum cryptography, CRYSTALS-Kyber, LWE-based KEM, side-channel attack, FPGA, power analysis, deep learning

I. INTRODUCTION

In July 2022, the National Institute of Standards and Technology (NIST) selected CRYSTALS-Kyber as a new public-key encryption and key-establishment algorithm to be standardized [1]. CRYSTALS-Kyber is a lattice-based cryptographic algorithm that is expected to be capable of protecting confidential information after the emergence of large-scale quantum computers. Even if it might take many years until large-scale quantum computers become a reality, the need for long-term security necessitates an early start of the transition. The industry is preparing to plan and budget for a shift to quantum-resistant cryptographic algorithms. The National Security Agency (NSA) has recently included CRYSTALS-Kyber in the suite of cryptographic algorithms recommended for national security systems [2].

The standardization of CRYSTALS-Kyber makes it important to assess the resistance of its implementations to side-channel attacks (SCAs). Several software implementations of CRYSTALS-Kyber have already been analyzed [3]–[6] and the discovered vulnerabilities were patched in the subsequently released versions. However, the evaluation of hardware implementations has only just begun [7], [8]. Hardware implementations are typically more difficult to break than software variants because they carry out the computations in parallel and consume less power.

Our Contributions: In this paper, we present a profiling side-channel attack on a hardware implementation of Kyber768

(CRYSTALS-Kyber with the parameter $k = 3$). Using deep learning-based power analysis, we can recover messages from a Xilinx Artix-7 FPGA implementation of CRYSTALS-Kyber from [9]. In CRYSTALS-Kyber key encapsulation mechanism (KEM), a message recovery from a properly generated ciphertext implies the session key recovery since the session key is derived from the message using hash functions. Furthermore, by recovering messages contained in seven chosen ciphertexts [10], one can extract the secret key of CRYSTALS-Kyber. Unlike the non-profiling attack on Kyber512 (CRYSTALS-Kyber with the parameter $k = 2$) presented in [8], our attack does not require any knowledge of register reference values.

The success of the presented attack is due to the new *sliced multi-bit error injection* method which is an extension of the original multi-bit error injection method introduced in [11] for side-channel analysis of software implementations of lattice-based PKE/KEMs. Software implementations do not require slicing because they execute instructions sequentially. However, our experimental results show that slicing is essential for hardware implementations. Without slicing, only 10% of messages can be recovered with enumeration up to 2^{64} . With slicing, we can recover all messages with the same enumeration.

The rest of this paper is organized as follows. Section II reviews previous work related to side-channel analysis of CRYSTALS-Kyber hardware implementations. Section III gives a background on CRYSTALS-Kyber algorithm. Section IV describes the equipment used in the experiments. Sections V and VI present the profiling and attack stages, respectively. Section VII summarizes experimental results. Section VIII concludes the paper and discusses future work.

II. PREVIOUS WORK

In this section, we describe previous works related to hardware implementations of CRYSTALS-Kyber.

A. Implementations

In [12] a hardware implementation of CRYSTALS-Kyber in FPGA is presented which takes 10 times less clock cycles than the ARM Cortex-M4 implementation of CRYSTALS-Kyber from [13].

In [9] an even smaller and faster hardware implementation of CRYSTALS-Kyber is demonstrated. With suitably designed pipelines and well-optimized architecture, this implementation

```

KYBER.CPAPKE.KeyGen()
1:  $(\rho, \sigma) \leftarrow \mathcal{U}(\{0, 1\}^{256})$ 
2:  $\mathbf{A} \leftarrow \mathcal{U}(R_q^{k \times k}; \rho)$ 
3:  $\mathbf{s}, \mathbf{e} \leftarrow \beta_{\eta_1}(R_q^{k \times 1}; \sigma)$ 
4:  $\mathbf{t} = \text{Encode}_{12}(\mathbf{A}\mathbf{s} + \mathbf{e})$ 
5:  $\mathbf{s} = \text{Encode}_{12}(\mathbf{s})$ 
6: return  $(pk = (\mathbf{t}, \rho), sk = \mathbf{s})$ 
KYBER.CPAPKE.Dec( $\mathbf{s}, c$ )
1:  $\mathbf{u} = \text{Decompress}_q(\text{Decode}_{d_u}(c_1), d_u)$ 
2:  $\mathbf{v} = \text{Decompress}_q(\text{Decode}_{d_v}(c_2), d_v)$ 
3:  $\mathbf{s} = \text{Decode}_{12}(\mathbf{s})$ 
4:  $m = \text{Encode}_1(\text{Compress}_q(\mathbf{v} - \mathbf{s}^T \mathbf{u}, 1))$ 
5: return  $m$ 
KYBER.CPAPKE.Enc( $pk = (\mathbf{t}, \rho), m, r$ )
1:  $\mathbf{t} = \text{Decode}_{12}(\mathbf{t})$ 
2:  $\mathbf{A} \leftarrow \mathcal{U}(R_q^{k \times k}; \rho)$ 
3:  $\mathbf{r} \leftarrow \beta_{\eta_1}(R_q^{k \times 1}; r)$ 
4:  $\mathbf{e}_1 \leftarrow \beta_{\eta_2}(R_q^{k \times 1}; r); \mathbf{e}_2 \leftarrow \beta_{\eta_2}(R_q^{1 \times 1}; r)$ 
5:  $\mathbf{u} = \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$ 
6:  $\mathbf{v} = \mathbf{t}^T \mathbf{r} + \mathbf{e}_2 + \text{Decompress}_q(m, 1)$ 
7:  $c_1 = \text{Encode}_{d_u}(\text{Compress}_q(\mathbf{u}, d_u))$ 
8:  $c_2 = \text{Encode}_{d_v}(\text{Compress}_q(\mathbf{v}, d_v))$ 
9: return  $c = (c_1, c_2)$ 

```

Fig. 1: Description of KYBER.CPAPKE algorithms from [14].

is capable of executing the decapsulation procedure of any version of CRYSTALS-Kyber in 14,000 clock cycles while consuming 7,500 LUTs only.

B. Attacks

The assessment of PQC KEM hardware implementations' resistance to side-channel analysis is still in an early phase. We are aware of only two previous works.

In [7] a preliminary power analysis of a hardware implementation of CRYSTALS-Kyber from [12] is presented. Leakage points are found using a t-test for the Kyber512 implementation in Xilinx Virtex-7 FPGA.

In [8], a non-profiling correlation EM analysis of an FPGA implementation of Kyber512 from [9] is presented targeting polynomial multiplication during the PKE decryption step of the decapsulation algorithm. The attack utilizes 166,620 traces to recover the secret key. However, it requires knowledge of the register reference values. Such knowledge may not be available in a real attack scenario.

III. CRYSTALS-KYBER ALGORITHM

CRYSTALS-Kyber [14] is an IND-CCA2-secure cryptographic algorithm which means that it is indistinguishable under an adaptive Chosen Ciphertext Attack (CCA). The security of CRYSTALS-Kyber relies on the hardness of the Module Learning with Errors (Mod-LWE) problem.

CRYSTALS-Kyber contains a Chosen Plaintext Attack (CPA)-secure PKE scheme, KYBER.CPAPKE, and a CCA-secure KEM scheme, KYBER.CCAKEM, based on a post-quantum version of the Fujisaki-Okamoto transform [15]. These algorithms are described in Fig. 1 and Fig. 2 respectively.

```

KYBER.CCAKEM.KeyGen()
1:  $z \leftarrow \mathcal{U}(\{0, 1\}^{256})$ 
2:  $(pk, \mathbf{s}) = \text{KYBER.CPAPKE.KeyGen}()$ 
3:  $sk = (\mathbf{s}, pk, \mathcal{H}(pk), z)$ 
4: return  $(pk, sk)$ 
KYBER.CCAKEM.Encaps( $pk$ )
1:  $m \leftarrow \mathcal{U}(\{0, 1\}^{256})$ 
2:  $\hat{K} = \mathcal{H}(m)$ 
3:  $(\hat{K}, r) = \mathcal{G}(m, \mathcal{H}(pk))$ 
4:  $c = \text{KYBER.CPAPKE.Enc}(pk, m, r)$ 
5:  $K = \text{KDF}(\hat{K}, \mathcal{H}(c))$ 
6: return  $(c, K)$ 
KYBER.CCAKEM.Decaps( $sk = (\mathbf{s}, pk, \mathcal{H}(pk), z), c$ )
1:  $m' = \text{KYBER.CPAPKE.Dec}(\mathbf{s}, c)$ 
2:  $(\hat{K}', r') = \mathcal{G}(m', \mathcal{H}(pk))$ 
3:  $c' = \text{KYBER.CPAPKE.Enc}(pk, m', r')$ 
4: if  $c = c'$  then
5:   return  $K = \text{KDF}(\hat{K}, \mathcal{H}(c))$ 
6: else
7:   return  $K = \text{KDF}(z, \mathcal{H}(c))$ 
8: end if

```

Fig. 2: Description of KYBER.CCAKEM algorithms from [14].

Let \mathbb{Z}_q denote the ring of integers modulo a prime q , and R_q denote the quotient ring $\mathbb{Z}_q[X]/(X^n + 1)$, for $n = 256$, $q = 3329$. CRYSTALS-Kyber works with vectors of ring elements in R_q^k , where k is the rank of the module used to scale the security level.

The term $x \leftarrow \chi(S; r)$ denotes sampling x from a distribution χ over a set S using seed r . The uniform distribution is denoted by \mathcal{U} . The centered binomial distribution with parameter μ is denoted by β_μ .

The functions \mathcal{G} and \mathcal{H} represent the SHA3-512 and SHA3-256 hash functions, respectively. The KDF represents the key derivation function. It is realized by SHAKE-256.

IV. EXPERIMENTAL SETUP

Our equipment consists of the ChipWhisperer-Lite board and the CW305 Artix-7 FPGA board which contains an Artix-7 XC7A100T FPGA.

The target implementation is the FPGA implementation of CRYSTALS-Kyber from [9], described in Verilog. It does not contain any countermeasures against side-channel attacks. We synthesized the Kyber768 implementation to target a Xilinx XC7A100TFTG256 FPGA device using the Vivado synthesis toolchain.

V. PROFILING STAGE

This section describes our profiling strategy.

A. Training trace acquisition

Using the equipment described in the previous section, we captured traces representing the execution of the decapsulation procedure KYBER.CCAKEM.Decaps(). For the profiling stage, the traces are captured for ciphertexts generated by KYBER.CPAPKE.Enc() for random messages. Since Kyber is a public-key cryptographic algorithm, we can create a properly generated ciphertext for any message using the corresponding

TABLE I: MLP architecture used for message recovery.

Layer type	Output shape	# Parameters
Batch Normalization 1	64	256
Dense 1	64	4160
Batch Normalization 2	64	256
ReLU 1	64	0
Dense 2	256	16640
Softmax	256	0
Total parameters:	21312	
Trainable parameters:	21056	

public key. Thus, the device under attack can be used for creating a labeled dataset for profiling [16].

Fig. 3(a) shows a trace representing the execution of the initial part of the KYBER.CCAKEM.Decaps() procedure.

B. Location of points of interest

Since for each trace \mathcal{T}_j of the profiling set \mathcal{T} , the value of the message m_j processed by KYBER.CCAKEM.Decaps() during the acquisition of \mathcal{T}_j is known, we can use Welch’s t-test [17] to analyze the leakage in order to locate intervals corresponding to the individual message bytes.

For each byte $i \in \{0, 1, \dots, 31\}$, we partition \mathcal{T} into two sets, \mathcal{T}_0 and \mathcal{T}_1 as:

$$\begin{aligned} \mathcal{T}_0 &= \{\mathcal{T}_j \in \mathcal{T} \mid m_j[i] < 128\} \\ \mathcal{T}_1 &= \{\mathcal{T}_j \in \mathcal{T} \mid m_j[i] > 128\}, \end{aligned}$$

where $m_j[i]$ is i th byte of m_j , for all $j \in \{1, 2, \dots, |\mathcal{T}|\}$.

Fig. 3(c) shows the t-test results. One can clearly see 32 groups corresponding to 32 message bytes. Fig. 3(d) gives a zoomed-in view of eight bytes. We can see two groups of t-test peaks for every byte, separated by a space. Each byte, with the exception of the ones in the beginning and at the end, overlaps with two adjacent bytes. This overlap could mean that several computations involving different message bytes are carried out in parallel. We will discuss this issue in more detail in Section VI.

C. Network architecture and training parameters

We use multilayer perceptron (MLP) neural networks with the architecture shown in Table I.

The neural networks are trained with a batch size of 1024 for a maximum of 100 epochs using early stopping with patience 20. We use Nadam optimizer with a learning rate of 0.01 and numerical stability constant $\epsilon = 1e-08$. Categorical cross-entropy is used as a loss function to evaluate the network classification error. 70% of the training set is used for training and 30% is left for validation. Only the model with the highest validation accuracy is saved.

VI. ATTACK STAGE

In this section, we describe how we adapted the multi-bit error injection method proposed in [11] to recover messages from a hardware implementation. We also show how session keys can be derived from the recovered messages.

TABLE II: Time to capture the training and test sets.

	Attack method	# Traces	Time (min)	Success rate
	Repetition	256×5	5:40	0%
Test set	MBE w/o slicing	256×5	5:40	10%
	MBE w/o slicing	256×20	13:21	10%
	4-sliced MBE	256×4×5	22:42	100%
Training set	6 hours to capture 200K traces			

A. Sliced multi-bit error injection method

In [11], a multi-bit error injection attack method is proposed and applied to a software implementation of CRYSTALS-Kyber in which the message bytes are processed sequentially. However, in the hardware implementations of CRYSTALS-Kyber from [9], computations are carried out in parallel. From Fig. 3, we can see that the t-test peaks of the adjacent bytes are overlapping. If the same error e is injected into all bytes, as in [11], it may be canceled out, reducing the attack efficiency.

We propose to overcome this problem by using the following technique which we call *slicing*. Instead of injecting an error e into all message bytes, we inject e into every fourth byte. As a result, the attack set is expanded into a (256×4)-trace set captured for the ciphertexts c_e decrypting to messages m_e whose bytes $m_e[i]$ are defined by:

$$m_e[i] = \begin{cases} m[i] \oplus e & \text{if } i \bmod 4 = x, \\ m[i] & \text{otherwise,} \end{cases} \quad (1)$$

where $x \in \{0, 1, 2, 3\}$.

Clearly, the number of slices may be different from four. For implementations with a higher overlapping between the bytes’ computations, a larger number may be more suitable.

B. Attack trace acquisition

To recover the message m from a properly generated ciphertext c , we collect the attack dataset as follows. For each $x \in \{0, 1, 2, 3\}$, we construct 256 ciphertexts c_e decrypting to messages m_e defined by (1) and capture a set of 256 traces during the decapsulation of these ciphertexts.

Eight message bytes $m[i]$ such that $(i \bmod 4) = x$ are recovered from the traces for each x by giving the segments of traces \mathcal{T}_e representing the processing of $m_e[i] = m[i] \oplus e$ as input to the MLP model \mathcal{N} trained at the profiling stage. For each \mathcal{T}_e , the model \mathcal{N} outputs a score vector $S_{i,e} = \mathcal{N}(\mathcal{T}_e)$ in which the value of the l th element, $S_{i,e}[l]$, is the probability that $m_e[i] = l$, for $l, e \in \{0, \dots, 255\}$.

The most likely label for $m[i]$ among 256 candidates is decided as:

$$\tilde{l} = \arg \max_{l \in \{0, 1, \dots, 255\}} \left(\prod_{e=0}^{255} S_{i,e}[l \oplus e] \right).$$

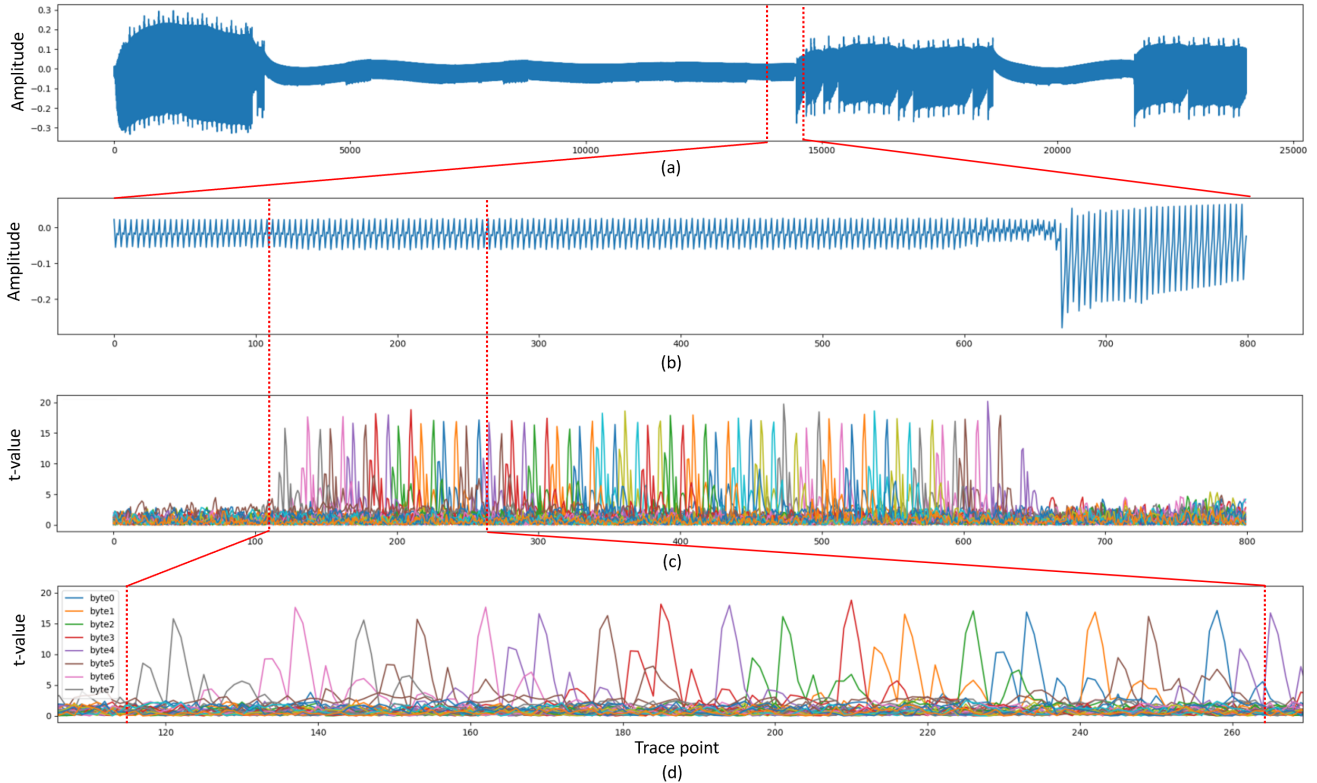


Fig. 3: (a) An average trace representing the execution of `KYBER.CCAKEM.Decaps()`; (b) a segment representing the message processing; (c) t-test results for all 32 message bytes; (d) a zoomed-in view of eight bytes.

If $\tilde{l} = m[i]$, the classification is successful. The condition $\tilde{l} = m[i]$ can be verified by checking if the rank of $m[i]$ is zero¹.

C. Session key recovery

A successful recovery of the message m from a properly generated ciphertext c trivially implies the session key recovery, since the session key can be derived as $K = \text{KDF}(\tilde{K}, \mathcal{H}(c))$ where $(\tilde{K}, r) = \mathcal{G}(m, \mathcal{H}(pk))$ (see lines 3 and 5 of `KYBER.CCAKEM.Encaps()`).

VII. EXPERIMENTAL RESULTS

Following the profiling strategy described in Section V, at the profiling stage, we capture 200K training traces during the execution of `KYBER.CCAKEM.Decaps()` procedure with input ciphertexts encrypting random messages. The training set is then expanded to 6.4M using the cut-and-join technique introduced in [16]. An MLP model with the architecture listed in Table I is trained on the expanded set.

A. Message/session key recovery attack

In this section, we compare the presented sliced multi-bit error injection method to the original multi-bit error injection method and to a direct repetition attack. All attacks are

¹A *rank* of a byte b is the number of other bytes which have a higher probability than b in the score vector.

performed on the same 10 messages selected at random. Table II shows the time it takes to capture the test set for one message in each case.

1) *Sliced multi-bit error injection attack*: For each message, we capture 256×4 traces for the 4-sliced multi-bit error injection attack. Each measurement is repeated $N = 5$ times.

Table III summarizes the results of message recovery. It shows the average and the maximum ranks of all message bytes for each message. We can see that all maximum ranks are smaller than 4. This means that we can recover the messages by enumerating the bytes with ranks 0,1,2 and 3, i.e. using up to $4^3 = 2^6$ enumerations in total, which is feasible.

2) *Multi-bit error injection attack*: To compare the presented 4-sliced method to the original multi-bit error injection method, we also capture 256 traces in which the error e is injected in all message bytes. We test two cases with a different number of repetitions: $N = 5$ and $N = 20$, in order to match the total number of traces used by 4-sliced multi-bit error injection method with $N = 5$.

Table IV summarized the results of message recovery. We can see that for both cases, only one message can be recovered with enumeration up to 2^6 . This experiment shows that slicing is essential for an attack on the hardware implementation of CRYSTALS-Kyber from [9] in which the message bytes are processed in parallel. Without slicing, only 10% of messages can be recovered with enumeration up to 2^6 . With slicing,

TABLE III: Results of the 4-sliced MBE attack using $256 \times 4 \times 5$ traces.

Byte rank	Message									
	0	1	2	3	4	5	6	7	8	9
Avg.	0.09	0.19	0.09	0.09	0.03	0.19	0.06	0.22	0.28	0.16
Max.	2	2	1	1	1	2	1	2	3	3

All messages can be recovered with enumeration $\leq 2^{64}$

TABLE IV: Results of the MBE attack without slicing, with repetition $N_1 = 5$ and $N_2 = 20$

Byte rank	Message										
	0	1	2	3	4	5	6	7	8	9	
N_1	Avg.	1.63	2.34	0.34	0.84	0.72	1.59	1.93	1.88	2.28	1.09
	Max.	15	24	2	13	5	23	25	24	26	8
N_2	Avg.	1.56	1.94	0.31	0.69	0.78	1.22	1.88	1.88	2.16	1.06
	Max.	16	19	2	11	6	15	20	23	22	9

1 out of 10 messages can be recovered with enumeration $\leq 2^{64}$.

TABLE V: Results of the repetition attack using 256×5 traces.

Byte rank	Message									
	0	1	2	3	4	5	6	7	8	9
Avg.	116	123	127	120	121	99	130	105	138	116
Max.	251	237	247	248	253	240	255	248	252	255

No messages can be recovered with enumeration $\leq 2^{64}$.

we can recover all messages with the same enumeration.

3) *Repetition attack*: To further highlight the effectiveness of the presented sliced multi-bit error injection method, we also compare it to a direct repetition attack which uses multiple traces captured for the same ciphertext c .

Table V shows the results of message recovery from 256×5 traces. It is obvious that the repetition attack does not work on the target implementation.

VIII. CONCLUSION

We demonstrated a message recovery attack on a hardware implementation of CRYSTALS-Kyber by deep learning-based power analysis. The success of the attack is due to the sliced multi-bit error injection method which we introduced in this paper. It is an extension of the original multi-bit error injection method presented in [11] for side-channel analysis of software implementations of lattice-based PKE/KEMs. Our experimental results show that slicing is essential for hardware implementations. Without slicing, only 10% of messages can be recovered with enumeration up to 2^{64} . With slicing, we can recover all messages with the same enumeration.

Future work includes re-designing the CRYSTALS-Kyber implementation to make it resistant to side-channel analysis.

IX. ACKNOWLEDGMENTS

This work was supported in part by the Swedish Civil Contingencies Agency (Grant No. 2020-11632), the Swedish Research Council (Grant No. 2018-04482) and Sweden's Innovation Agency Vinnova (Grant No. 2021-02426)

REFERENCES

- [1] D. Moody, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," *Nistir 8309*, pp. 1–27.
- [2] "Announcing the commercial national security algorithm suite 2.0."
- [3] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367.
- [4] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal, "Masking Kyber: First-and higher-order implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 173–214, 2021.
- [5] J.-P. D'Anvers, M. V. Beirendonck, and I. Verbauwhede, "Revisiting higher-order masked comparison for lattice-based cryptography: Algorithms and bit-sliced implementations," *Cryptology ePrint Archive*, Paper 2022/110, 2022.
- [6] D. Heinz, M. J. Kannwischer, G. Land, T. Pöppelmann, P. Schwabe, and D. Sprenkels, "First-order masked Kyber on ARM Cortex-M4," *Cryptology ePrint Archive*, Paper 2022/058, 2022.
- [7] T. Kamucheka, M. Fahr, T. Teague, A. Nelson, D. Andrews, and M. Huang, "Power-based side channel attack analysis on PQC algorithms," *Cryptology ePrint Archive*, Paper 2021/1021, 2021.
- [8] R. C. Rodriguez, F. Bruguier, E. Valea, and P. Benoit, "Correlation electromagnetic analysis on an fpga implementation of crystals-kyber," *Cryptology ePrint Archive*, Paper 2022/1361, 2022.
- [9] 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.
- [10] R. Ravi, S. Sinha Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, no. 3, p. 307–335, Jun. 2020.
- [11] R. Wang, K. Ngo, and E. Dubrova, "A message recovery attack on LWE/LWR-based PKE/KEMs using amplitude-modulated EM emanations," in *Proc. of 25th Annual Int. Conf. on Information Security and Cryptology*, 2022.
- [12] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse," *IEICE Electronics Express*, vol. 17, no. 17, pp. 1–6, 2020.
- [13] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of kyber on cortex-m4," in *Progress in Cryptology – AFRICACRYPT 2019*, J. Buchmann, A. Nitaj, and T. Rachidi, Eds. Cham: Springer International Publishing, 2019, pp. 209–228.
- [14] P. Schwabe *et al.*, "CRYSTALS-Kyber algorithm specifications and supporting documentation," 2020, <https://csrc.nist.gov/projects/postquantum-cryptography/round-3-submissions>.
- [15] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual international cryptology conference*. Springer, 1999, pp. 537–554.
- [16] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked IND-CCA secure Saber KEM implementation," *IACR Trans. on Cryptographic Hardware and Embedded Systems*, no. 4, pp. 676–707, 2021.
- [17] B. L. Welch, "The generalization of 'student's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.