Power Side-Channel Leakage Assessment of Reference Implementation of SABER Key Encapsulation Mechanism

1st Leslie C Wang
School of ECE
Georgia Tech
Atlanta, USA
lwang644@gatech.edu

2nd Anupam Golder
School of ECE
Georgia Tech
Atlanta, USA
anupamgolder@gatech.edu

3rd Yan Fang
Dept. of ECE
Kennesaw State University
Kennesaw, USA
yfang9@kennesaw.edu

4th Arijit Raychowdhury
School of ECE
Georgia Tech
Atlanta, USA
arijit.raychowdhury@ece.gatech.edu

2022 Opportunity Research Scholars Symposium (ORSS) | 978-1-6654-6898-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ORSS55359.2022.9806031

***Abstract*—Traditional public-key cryptographic schemes are soon going to be replaced with Post-Quantum Cryptographic (PQC) schemes to ensure security guarantees in a Quantum Computing-enabled world. While Quantum Computing will help solve many hard problems intractable by classical computing paradigm, it will also compromise the hard problems that traditional cryptographic schemes are built upon. Among the National Institute of Standards and Technology (NIST) finalist PQC schemes, SABER Key Encapsulation Mechanism (KEM) is the only one based on Module Learning With Rounding (LWR). In this work, we have investigated the decryption procedure of SABER KEM to identify leakages in power consumption traces for the reference implementation running on an ARM Cortex-M4 microcontroller by correlating the trace samples with the decrypted message bytes, and by performing a Test Vector Leakage Assessment (TVLA). Both assessment techniques indicate that the incremental-storage steps in the reference implementation might allow an adversary to reveal information about the message and/or secret key.**

***Index Terms*—Power Side-Channel Analysis, Post Quantum Cryptography, SABER Key Encapsulation Mechanism, Correlation Analysis, Test Vector Leakage Assessment**

I. Introduction

Quantum computers are capable of finding the prime factors of an integer in polynomial time, due to Shor's algorithm [1]. As the era of quantum computing approaches, such quantum supremacy will eventually compromise the security of the current Public-Key Cryptography (PKC) which relies on the hardness of prime factorization problem. Even though Cryptographically Relevant Quantum Computer (CRQC) may take decades to become a reality, National Institute of Standards and Technology (NIST) has started the standardization process for Post-Quantum Cryptographic (PQC) schemes in 2016, which can be implemented in today's classical computers, and are quantum-resistant. In 2020, NIST announced seven finalist schemes (with five based on computationally hard problems from the Lattice theory), and by the end of this year, it plans to standardize the schemes. Out of the five Lattice Crypto schemes, three are Public-Key Encryption (PKE)/Key Encapsulation Mechanism (KEM) schemes, and two are Digital Signature Schemes. The finalist PKE schemes are SABER [2] based on Module Learning With Rounding (Module-LWR), NTRU based on Standard Learning With Errors (LWE), and CRYSTALS-KYBER based on Module-LWE. In this final round, NIST is interested in the evaluation of the finalist schemes for Power/Electromagnetic (EM) side-channel vulnerabilities, which is the premise for this work.

Side-channels, such as, timing, power or EM provide valuable information about the on-going operations and intermediate data. Power Side-Channel Analysis (SCA) [3] has proved to be one of the major side-channel vulnerabilities of traditional cryptographic schemes, such as, Advanced Encryption Standard (AES) or Rivest-Shamir-Adleman (RSA). Recent works [4], [5] on Power SCA of PQC schemes also illustrate how such vulnerabilities can be exploited to recover the encrypted message (plaintext) or the secret key. While this work closely aligns with the aforementioned works, in this work, the possible leakage points of a reference implementation of SABER KEM running on an ARM Cortex-M4 microcontroller [6] were investigated instead of devising attack techniques. The methods presented here based on Correlation Analysis and Test Vector Leakage Assessment (TVLA) are relatively easy to try out and test for a designer to check whether the implementation has any leakage.

The contributions of this work are:

- Correlation analysis was carried out to precisely identify the points in time when SABER KEM scheme performs incremental-storage based operation on decrypted message, which might allow an adversary to chop the traces into segments to launch efficient attack.
- TVLA was performed to identify the presence of leakage from the crypto core, which also demonstrates the vulnerability of the incremental-storage operation.

II. Related Works

A recent work [4] performed Chosen-Ciphertext Attack (CCA) on several LWE/LWR-based PKE/KEMs utilizing EM

This work was supported in part by the National Science Foundation (NSF) under CNS Grant No. 1935534 and 1717467, and in part by Semiconductor Research Corporation (SRC) Task 3059.001.

side-channel as a precise plaintext checking oracle. In that work, exploitation of constant-time decoding procedures of Error Correcting Codes (ECC) helped identify the EM-based side-channel vulnerabilities. Similar vulnerabilities were also identified in the Fujisaki-Okamoto (FO) transform of the decapsulation procedure, which is used to transform an IND-CPA (chosen plaintext model) scheme to IND-CCA (chosen ciphertext model) scheme. This vulnerability led to recovery of information related to the decrypted messages, bringing on complete key recovery in KEMs which do not have an ECC protocol. That work demonstrated the requirement for concrete masking countermeasures against SCA to protect IND-CCA secure Lattice Crypto schemes. More recently, a first-order masked implementation of the IND-CCA secure SABER KEM was targeted and exploited by Power SCA [5], requiring up to 24 traces containing the power consumption traces of the KEM decryption function from the target board CW308T-STM32F4, with a 32-bit ARM Cortex-M4 CPU. Feeding in those power traces to a Deep Neural Network (DNN) which was trained during the profiling phase, the attack phase successfully recovered both the overall secret key and the temporary session key. The key idea was to avoid recovering random masks to make it easy for an adversary to launch an attack by exploiting incremental-storage vulnerability. While both of these works present interesting ideas that an adversary might use, from a designer's perspective, they might be time-consuming processes to implement, and test the vulnerability of an implementation. As such, easier and fast leakage assessments are valuable to an engineer designing implementations of such schemes, which is demonstrated in our work.

III. Brief Overview of SABER KEM

SABER [2] is an IND-CCA2 (non-distinguishability under adaptive chosen ciphertext attack) secure KEM. Its security relies on the hardness of LWR problem, which is based on LWE where random errors are replaced with deterministic rounding operation. The SABER team [2] first described a key exchange protocol that is similar to the Diffie-Hellman protocol, and then transformed the protocol into an IND-CPA secure encryption scheme. In the end, the scheme was transformed into an IND-CCA2 secure KEM under a post-quantum FO transform. The three priorities in the design of SABER scheme were efficiency, simplicity, and flexibility. Therefore, the team utilized LWR to reduce bandwidth and the amount of randomness to half of what LWE-based schemes required. The choice of power-of-two for the integer moduli eliminated expensive modular reduction arithmetic and rejection sampling (which would otherwise affect performance). The module structure reused one core component for multiple security levels and thus provided flexibility. There are three security levels offered by SABER: LightSABER with security level equal to AES-128 (roughly 128 bits of security), SABER with security level equal to AES-192, and FireSABER with security level equal to AES-256. In this work, the SABER version of the scheme was the sole focus. The parameters for this version of the scheme are illustrated in Table I.

IV. Analysis and Proposal

Inspired by the recent power SCA work [5] on the masked SABER KEM implementation, this work discovered and explored the Incremental-Storage vulnerability [7] caused by the incremental update of the decrypted message in memory of SABER KEM reference implementation [6].

```
void indcpa_kem_dec(uint8_t m[SABER_KEYBYTES], const uint8_t
ciphertext[SABER_BYTES_CCA_DEC], const uint8_t sk[SABER_INDCPA_SECRETKEYBYTES])
{
    size_t i;
    uint16_t v[SABER_N];
    uint16_t s[SABER_N];
    uint16_t bp[SABER_N];
    uint16_t (*cm) = bp;

    for (i = 0; i < SABER_L; i++) {

        BS2POLmu(&sk[i * SABER_POLYSECRETBYTES], s);

        BS2POLp(&ciphertext[i * SABER_POLYCOMPRESSEDBYTES], bp);

        if (i == 0) {
            poly_mul(s, bp, v);
        } else {
            poly_mul_acc(s, bp, v);
        }
    }

    BS2POLT(ciphertext + SABER_POLYVECCOMPRESSEDBYTES, cm);

    for (i = 0; i < SABER_N; i++) {
        v[i] = (v[i] + h2 - (cm[i] << (SABER_EP - SABER_ET))) >> (SABER_EP - 1);
    }
    POLmsg2BS(m, v);
}
```

(a)

```
void POLmsg2BS(uint8_t bytes[SABER_KEYBYTES], const uint16_t data[SABER_N])
{
    size_t i, j;
    uint8_t byte;
    for (j = 0; j < SABER_KEYBYTES; j++) {
        byte = 0;
        for (i = 0; i < 8; i++) {
            byte |= ((data[j * 8 + i] & 0x01) << i);
        }
        bytes[j] = byte;
    }
}
```

(b)

Fig. 1. (a) C Code of SABER Reference Implementation of $indcpa\_kem\_dec()$ (b) C Code of Packing Decrypted Message in SABER Reference Implementation

For decryption in reference implementation of SABER, the message is only manipulated during the packing process implemented by $POLmsg2BS()$ as shown in Fig. 1(a). In $POLmsg2BS()$, the decrypted message is manipulated and filled into an empty 8-bit integer array in a sequential fashion, as can be seen from Fig. 1(b). For each of the 32 bytes of the decrypted message, every bit of that byte is being extracted separately, padded with 8 zeros, and stored into the empty message array. Therefore, each entry of the message array experiences an incremental storage from 16-bit array to 8-bit array. Based on the above observation, this work proposed to capture power traces around those 32 consecutive incremental storage operations in $indcpa\_kem\_dec()$, and then perform a correlation analysis and TVLA to verify such vulnerability and the power side-channel leakage of the message during the decryption operation.

V. Experimental Setup

A. Equipment

The measurement setup is shown in Fig. 2, which consists of the ChipWhisperer-Lite board, the CW308 UFO board, and the CW308T-STM32F4 target board. The ChipWhisperer-Lite

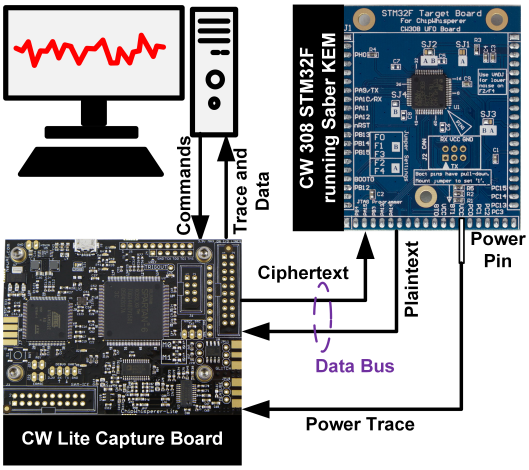

Fig. 2. Equipment for Decryption Computation and Trace Acquisition

board was used to collect power traces. It also facilitated the communication between the computer and the target board, converting analog power signals to digital signals using on-board Analog-to-Digital Converter (ADC), transporting data between the target board and the computer, and sending commands to trigger computations on the board. The power was measured over the shunt resistor placed between the power supply and the target board. The ChipWhisperer-Lite board has a maximum sampling rate of 105 MS/sec and the buffer size of $24,400$ samples. The target board CW308T-STM32F3 has an ARM 32-bit Cortex-M4F CPU (72 MHz max). The device is programmed with the C implementation of the SABER KEM Cortex-M4 implementation with some functions, such as Toom-Cook and Karatsuba multiplications replaced by their Assembly language counterparts [6].

B. Procedure

A fixed secret key and $20K$ ciphertexts were pre-generated using C testbench and sent to the target board to run the decryption algorithm. Since the serial protocol can only transport a maximum of 64 bytes data in a single transmission to the target board, we chose to keep the chunk size for transmission to 32 bytes, and the secret key was sent in 39 chunks (totaling to 1248 bytes), and each ciphertext was sent in 34 chunks (totaling to 1088 bytes). This proved to be the bottleneck of the trace capturing procedure, as most of the time was spent to send ciphertext to the target board. On average, it took about 5 seconds to capture a single trace.

Once all data required arrived the target board, SABER's $indcpa\_kem\_dec()$ was run on it. To better locate the leakage points, power traces for high impulses were collected around $POLmsg2BS()$ instead of around the entire $indcpa\_kem\_dec()$. A total of $20K$ traces were collected; so, the entire data collection took about a day to finish. The collected traces were sent back to the computer by the ChipWhisperer-Lite board. Each trace contains $15K$ samples in our setting to capture the complete operation of $POLmsg2BS()$ function. In the next section, the results of performing correlation analysis and TVLA are presented.

TABLE I
SABER PARAMETER CHART

Parameter	Description	Value
SABER_L (l)	Rank of the module	3
SABER_MU (μ)	Centered Binomial Distribution Parameter	8
SABER_N (n)	Degree of polynomials	256
SABER_KEYBYTES	No. of bytes in plaintext	32
SABER_INDCPA_SECRET KEYBYTES	No. of bytes in secret key	1248
SABER_BYTES_CCA_DEC	No. of bytes in ciphertext	1088

VI. RESULTS

$15K$ samples from each trace were used to run correlation analysis to verify the occurrence of the correlation peaks due to the incremental-storage of each byte of the message. Fig. 3 illustrates that there were 32 peaks in the correlation traces, which justifies this work's analysis on the vulnerability and leakage point. Pearson Correlation Coefficient, r_i for the i-th sample index of the trace was calculated using:

$$r_i = \frac{\sum_j ((tr(j)_i - E[tr(j)_i])(lbl(j) - E[lbl(j)])}{\sqrt{\sum_j ((tr(j)_i - E[tr(j)_i])^2} \sqrt{\sum_j (lbl(j) - E[lbl(j)])^2}} \quad (1)$$

where, $tr(j)_i$ is the sample at the i-th index of the j-th trace, and the $lbl(j)$ is the label for the j-th trace (e.g. a byte of the message, in this work), and $E[\cdot]$ denotes the expectation operator.

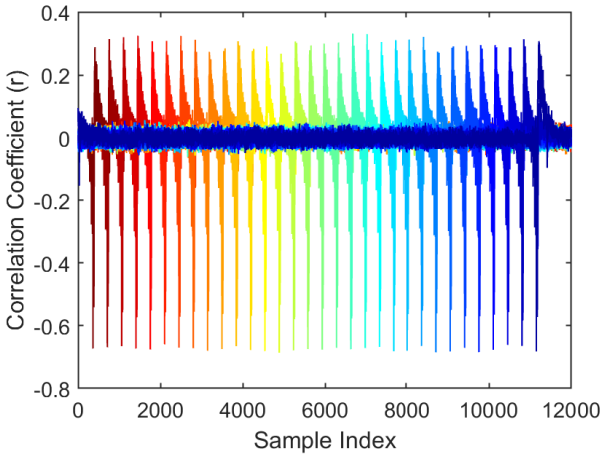

Fig. 3. Correlation Analysis: Peaks of correlation coefficient shows where the corresponding byte of the message is being produced by the decryption algorithm.

Test Vector Leakage Assessment (TVLA) [8] performs Welch's t-test over two sets of traces to identify distinguishable samples by testing for a null-hypothesis assuming that the

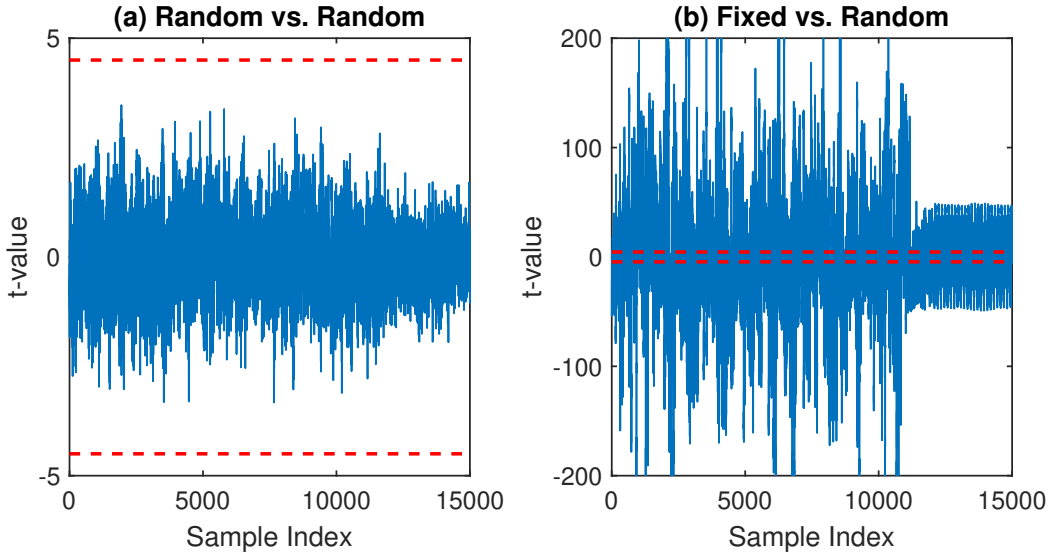

Fig. 4. (a) When ciphertexts are randomly chosen for both the variables for TVLA, t-value does not cross the threshold of 4.5 (sanity check) (b) For fixed vs. random ciphertexts, almost all the samples cross the threshold,

mean of the two sets are the same, and rejecting it with a confidence of 99.9999% if the t-test score is less than -4.5 or greater than 4.5. t-values for sample indices were computed by using the following equation:

$$t = \frac{\overline{t_1} - \overline{t_2}}{\sqrt{\frac{\sum(t_1-\overline{t_1})^2}{N_1(N_1-1)} + \frac{\sum(t_2-\overline{t_2})^2}{N_2(N_2-1)}}} \quad (2)$$

where, t_1 and t_2 are means for each sample index from the two different sets created by splitting the traces, and N_1 and N_2 are the number of traces in these sets, respectively.

Two types of comparisons were made: the first one was between two distributions of traces produced by decrypting random ciphertexts (TVLA on random vs. random set was performed as a sanity check); the second one was between the distribution of traces produced by decrypting a fixed ciphertext and the distribution of traces produced by decrypting random ciphertexts. For the first comparison, since the noise and signal are both canceled in the random ciphertext scenario, if the number of traces included in TVLA is relatively large, it is not expected to observe any leakage and the absolute value of t-value should be less then 4.5 (Fig. 4(a)). In contrast, the expected outcome for the second comparison is to observe leakage as the number of traces increases, and more and more sample mean differences should exceed the 4.5 boundary (Fig. 4(b)). Fig. 4 exhibits that both comparisons behave as expected for this implementation. It is to be noted that while TVLA certainly has merit in identifying whether there is existence of leakage, it might be not be a good candidate for points of interest selection, while correlation analysis helps to precisely point to the locations of leakages (Fig. 3).

VII. Conclusion and Future Works

The SCA we performed in this project successfully justified the exploitability of the incremental-storage vulnerability in the decryption code of SABER KEM reference implementation, leading to the leakage of decrypted messages. It is worth noting that incremental storage is also used for secret keys during decryption, but the manipulations applied to it are more complicated than packing. Therefore, more work is needed to capture the leakage points of secret keys if we plan to exploit the same vulnerability. The experimental setup in this project can be reused in the future to collect power traces to investigate any schemes of interest. Future work may focus on identifying the leakage points for the secret key, analyzing existing countermeasures, and designing new countermeasures against the Incremental-Storage vulnerability investigated in this work.

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[2] J.-P. D'Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, "Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem," in *International Conference on Cryptology in Africa*. Springer, 2018, pp. 282–305.

[3] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[4] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kem schemes," *Cryptology ePrint Archive*, 2019.

[5] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem implementation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 676–707, 2021.

[6] "Saber kem reference implementation," https://github.com/KULeuven-COSIC/SABER/tree/master/Cortex-M_Implementation_KEM/Cortex-M4, accessed: 31 March 2022.

[7] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "On exploiting message leakage in (few) nist pqc candidates for practical message recovery attacks," *IEEE Transactions on Information Forensics and Security*, 2021.

[8] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.