On the Practical Dependency of Fresh Randomness in AES S-box with Second-Order TI

Maki Tsukahara*, Haruka Hirata*, Mingyu Yang‡, Daiki Miyahara*,
Yang Li*, Yuko Hara-Azumi‡, and Kazuo Sakiyama*
*Department of Informatics, The University of Electro-Communications, Chofu, Tokyo
Email: {tsukahara, h.haruka, miyahara, liyang, sakiyama}@uec.ac.jp
‡Department of Information and Communications Engineering, Tokyo Institute of Technology, Meguro-ku Tokyo
Email: {mingyu, hara}@cad.ict.e.titech.ac.jp

***Abstract*—Physical attacks on cryptographic hardware have become a significant threat in recent years. For example, side-channel attacks exploit information leakage, such as power consumption or processing time during encryption, to recover the secret key. Threshold implementation (TI) is a widely-used countermeasure against such attacks. In the conventional implementation of TI, the significant process for ensuring an important property called uniformity is refreshing, which re-masks intermediate values using many random bits. In this study, we demonstrate that side-channel information leaks when fresh randomness is not sufficient even if a cryptographically secure Pseudo Random Number Generator (PRNG) is appropriately implemented. More precisely, our experimental results show such leakage when the seed value given to the PRNG remains fixed for every encryption, or when the update of fresh random bits is stopped in the encryption.**

***Index Terms*—AES, Threshold Implementation, Side-channel Attack, Welch's t-test, Power Analysis, Masks and Macs**

I. Introduction

The increasing threat of security breaches, including information leakage and falsification, is a concerning issue as we handle large volumes of data in our daily lives. Cryptographic modules play a crucial role in ensuring information confidentiality for IC cards, smartphones, and other devices. However, physical attacks against cryptographic hardware have emerged as a significant problem recently. Side-channel attacks (SCAs) recover the secret key by exploiting the side-channel information such as processing time [1], the power consumption [2], [3] and electromagnetic radiations [4] during the encryption. An attacker can execute a d^{th}-order attack by probing d internal wires of a circuit at once [5]. Fault attacks (FAs) also recover the secret key by intentionally injecting faults in the cryptographic modules. Among FAs, the differential fault attack (DFA) [6] is particularly dangerous. Against these physical attacks, Mayer et al. proposed M&M (Masks and Macs) [7] at CHES 2019 as a countermeasure.

M&M is a promising countermeasure that combines masking for SCAs and infective countermeasures for DFA. A d^{th}-order masking requires splitting every sensitive variable into at least $d+1$ shares, such that an adversary probing at most d values during the computation gets no information about any sensitive variable. Masking is not unconditionally secure. A d^{th}-order masked implementation is always theoretically vulnerable to $(d+1)^{\text{th}}$-order DPA in order to exploit leakages related to $d+1$ intermediate variables at the same time.

The authors of M&M opted for a masking scheme called Threshold Implementation (TI) [8] with Consolidating Masking Schemes (CMS) [9] on the S-box of AES [10] in the original paper. In the conventional implementation of TI in AES S-box, there is a need to re-mask intermediate values, called refreshing, to satisfy an essential property in TI called uniformity. The refreshing requires a large amount of fresh random numbers.

This paper studies the impact of the fresh random bits used in re-masking the S-box with TI on the security of cryptographic hardware. In the experiment, we verify the security of second-order secure AES cryptographic hardware that implements M&M countermeasure with three shares (M&M AES) when the fresh randomness utilized in the refreshing process is not sufficient.

The contributions of this paper are summarized as follows.

1) We evaluate side-channel leakage when the seed value given to the PRNG is fixed/updated for every encryption. We observe leakage when the seed value is fixed, i.e., when random bits used in each encryption are fixed to the same value.
2) We assess side-channel leakage when fresh random bits are fixed/updated for every clock cycle in the encryption. Our experimental results show leakage when the update of fresh random bits is stopped in the encryption, i.e., when the PRNG generates and stores a sufficient number of random bits before each M&M AES operation.

The rest of this paper is organized as follows. Section II shows the preliminaries with conventional works. Section III explains the target evaluation device and the experiment setup. Section IV shows the experimental results on the dependency of fresh randomness. In Section V, we discuss the experimental results. Finally, Section VI concludes the paper.

II. Preliminaries

A. Notation

In this paper, we use shares with three elements. Given a variable x, its shared value is represented as

$$\bar{x} = \{x_1, x_2, x_3\} \quad \text{s.t.} \quad x = x_1 \oplus x_2 \oplus x_3.$$

The final published version is available at IEEE Xplore: https://doi.org/10.1109/CANDARW60564.2023.00054

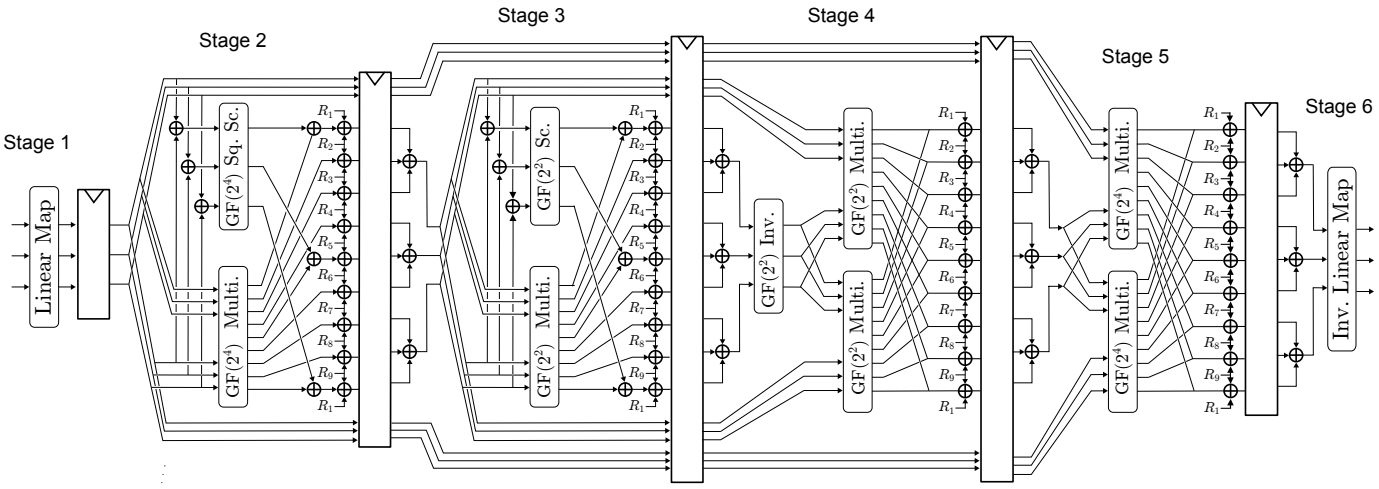


Fig. 1: M&M AES S-box with second-order security proposed in [11]

B. M&M (Masks and Macs)

M&M combines two countermeasures masking against SCAs and infective countermeasures with information-theoretic MAC tags against DFA. Masking [5], [8], [9], [12] is a widely recognized and effective countermeasure against Differential Power Analysis (DPA) [2] and Correlation Power Analysis (CPA) [3] in SCAs. It uses random numbers to randomize the internal signal, making it difficult for an attacker to infer sensitive information from observable side-channel leaks.

C. TI (Threshold Implementation)

TI is a masking method based on secret sharing and was proposed by Nikova et al. in 2006 [8]. In TI, the value x to be calculated is represented as a pair of values called a share. Additionally, a mapping $X = \psi(x)$ is considered, where X represents the final output result and x is the original input value. To perform the computation ψ securely with the share $\bar{x}$, the mapping ψ is split into three component functions, denoted as $\{\psi_1, \psi_2, \psi_3\}$:

$$\begin{aligned} X_1 &= \psi_1(x_2, x_3), \\ X_2 &= \psi_2(x_3, x_1), \\ X_3 &= \psi_3(x_1, x_2), \end{aligned} \tag{1}$$

where $\{X_1, X_2, X_3\}$ is an output share. The sharing $\{\psi_1, \psi_2, \psi_3\}$ must satisfy three essential properties: Correctness, Non-completeness, and Uniformity. These properties are essential for ensuring the security and effectiveness of TI.

1) Correctness: The sharing $\{\psi_1, \psi_2, \psi_3\}$ is said to be correct when the following is satisfied

$$\begin{aligned} \psi(x) &= X \\ &= X_1 \oplus X_2 \oplus X_3 \\ &= \psi_1(x_2, x_3) \oplus \psi_2(x_3, x_1) \oplus \psi_3(x_1, x_2). \end{aligned} \tag{2}$$

The output of three component functions ψ_1, ψ_2, and ψ_3 correctly represents the output of the original function ψ.

2) Non-Completeness: The sharing $\{\psi_1, \psi_2, \psi_3\}$ is said to be non-complete when each of the components functions ψ_1, ψ_2, and ψ_3 uses only a proper subset of the input share $\{x_1, x_2, x_3\}$. Each component function does not reveal information about the original value x.

3) Uniformity: The probability of occurrence of each share must be uniform. The probabilistic distributions of the original value and shared values are denoted by $\Pr[v]$ and $\Pr[v_1, v_2, v_3]$, where v is the original input/output value before splitting into shares. All sharing $\{v_1, v_2, v_3\}$ satisfies the uniformity if and only if, for any v, its shares occur at the same probability

$$\frac{\Pr[v]}{\alpha} = \Pr[v_1, v_2, v_3], \tag{3}$$

where α is a constant.

D. Compact S-box

In hardware implementations of the AES S-box, a common approach is to combine the calculation of the multiplicative inverse in $GF(2^8)$ and affine transformation. One popular method for the calculation of the multiplicative inverse is known as Canright's compact S-box [13]. M&M AES utilizes an S-box with six stages based on Canright's compact S-box. The S-box structure is illustrated in Fig. 1.

E. Refreshing

In the S-box calculation of M&M AES, the output shares from each stage are used as input values for the subsequent stage. For instance, if the output of a function f is used as the input to the following nonlinear function g and f is not uniform, fresh random bits need to be added to the output of f to ensure uniformity in the subsequent stage. From Fig. 1, M&M AES with second-order secure implementations opts for a re-masking technique called ring refreshing in stages 2, 3, 4, and 5 of the S-box. The values R_k $(1 \leq k \leq 9)$ in stages 2, 3, 4, and 5 are random values. Table I shows the total number of fresh random bits used for each stage. The ring refreshing structure is depicted in Fig. 2. The share $\{S_1, S_2, S_3, S_4\}$ is

TABLE I: Total number of fresh random bits per pipeline stage

S-box	(Bit width)×(Number of random units) [bits]
Stage 1	0
Stage 2	4×9
Stage 3	2×9
Stage 4	4×9
Stage 5	8×9
Stage 6	0
Total	162

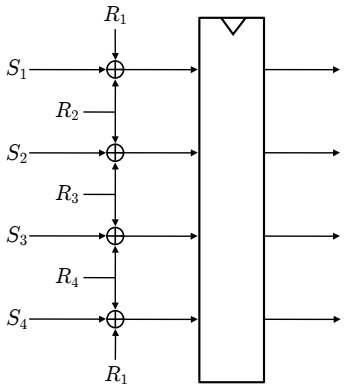


Fig. 2: Ring refreshing in [11]

the value after the operation that is not bijective, and the values R_1, R_2, R_3, and R_4 are fresh random values.

F. Welch's t-test

The t-test is a statistical method commonly used for hypothesis testing in various fields, including security evaluations for SCAs. In the evaluations for SCAs, the t-test is employed to examine the relationship between internal data and measured power consumption waveforms, assuming that the secret key and plaintext are known. The goal is not to directly extract sensitive information, such as secret keys, but to assess the level of dependence between the internal data and the power consumption to identify potential vulnerabilities. Compared to other side-channel security evaluation techniques, like power analysis, the t-test is more efficient as it does not require detailed knowledge of the cryptographic module internals or specific attack methods, intermediate values, or power models.

For the experiment conducted in this study, we prepared two datasets: one with fixed plaintexts and another with random plaintexts (fixed-vs-random t-test), following the methodology presented in reference [14], [15]. We calculate the mean $\mu[j]$ at each sample point (j) of the traces for the first-order t-test as

$$\mu[j] = \frac{1}{n}\sum_{i=0}^{n-1} l[i,j], \tag{4}$$

where $l[i,j]$ denotes the i-th trace data at the sample point (j), and n is the number of traces. The mean $\mu[j]$ at each sample point (j) of the traces for the second-order t-test is calculated as

$$\mu[j] = \frac{1}{n}\sum_{i=0}^{n-1} (l[i,j] - \bar{l}[:,j])^2, \tag{5}$$

where $\bar{l}[:,j]$ denotes the mean of all trace data at the sample point (j). The variance $v[j]$ at each sample point (j) of the traces for the first- and second-order t-test as

$$v[j] = \mu^2[j] - (\mu[j])^2, \tag{6}$$

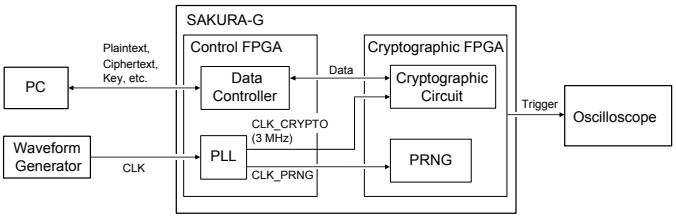
where $\mu^2[j]$ is the mean of squares, and $(\mu[j])^2$ is the square of means. Subsequently, we compute the t-value as

$$t[j] = \frac{\mu_1[j] - \mu_2[j]}{\sqrt{\frac{v_1[j]}{n_1} + \frac{v_2[j]}{n_2}}}, \tag{7}$$

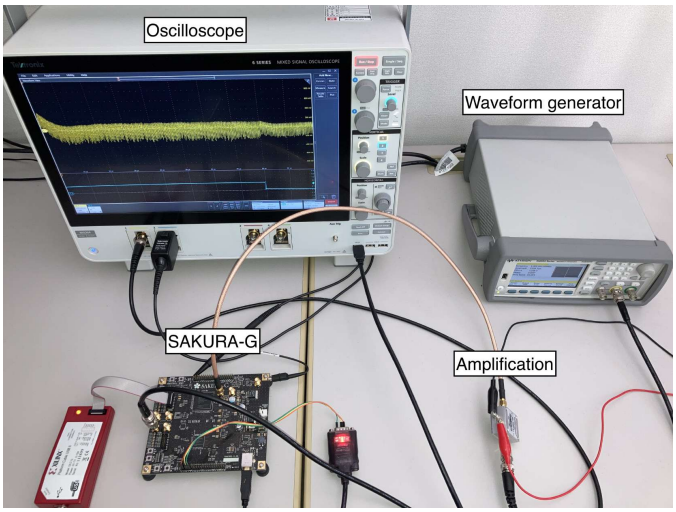
where μ, v, and n are the mean, variance, and cardinality of each dataset, and the subscripts denote each dataset.

If the t-statistic exceeds the threshold 4.5 in absolute value, it indicates a significant difference between the two datasets at the 99.999% confidence level. This is not a sufficient condition for a successful attack.

III. Target Device and Experimental Setup



(a) Experiment setup



(b) Environment of experiments

Fig. 3: Experiment setup and Environment of experiments

A. Target Device

The evaluation of the dependency of fresh randomness in the cryptographic hardware is conducted using the SAKURA-G board [16] for SCAs evaluation. This board features two Spartan-6 FPGAs: a control FPGA and a cryptographic FPGA. The encryption process starts with the PC running MATLAB

TABLE II: Devices used in experiment setup

Equipment	Product name and model number
Target Device	SAKURA-G
Control FPGA	Xilinx SPARTAN-6 XC6SLX9
Cryptographic FPGA	Xilinx SPARTAN-6 XC6SLX75
Waveform generator	Keysight 33622A
Oscilloscope	Tektronix MSO64
Amplification	ZFL-1000LN+

code, which sends an unshared plaintext, a secret key, and other relevant data to the control FPGA. The control FPGA then takes the original values and divides them into shared values. These shared values are then transmitted to the cryptographic FPGA, where the actual encryption operation takes place. By isolating the control FPGA and the cryptographic FPGA, the power consumption of the encryption process can be measured independently, reducing the impact of noise in the experiment.

B. Experimental Setup

The overview of our experimental setup is shown in Fig. 3a, and the experimental environment is shown in Fig. 3b. Detailed information on the experimental setup is shown in Table II. We supplied a clock signal CLK to control FPGA by waveform generator, and obtained two clock signals, CLK_CRYPTO and CLK_PRNG, from the phase-lock loop (PLL). The CLK_CRYPTO (3 MHz) is used for operations of the AES encryption, and the CLK_PRNG (3 MHz) is used for operations to generate random numbers for the refreshing process. We sample the power consumption at $1.25\,\mathrm{GS/s}$ with 9,587 points, which includes 23 clock cycles (1^{st} Round), using an oscilloscope with 12 bit analog to digital converter.

IV. Results

In this section, we present the result of the fixed-vs-random plaintexts t-test. To prevent bias due to the order of execution in the t-test, fixed plaintexts and random plaintexts were performed randomly. In our experiments, we used Keccak [17] as the PRNG that generates fresh random bits for refreshing. Keccak is a hash function family based on the Sponge construction [18] that outputs random numbers of arbitrary length for variable-length inputs.

Table III shows the correspondence between the experimental conditions and the results, while Figs. 4a, 4b, 4c, and 4d show the obtained results. We note that the S-box input x is split into three shared values $\{x_1, x_2, x_3\}$ and the shared values differ for each operation. As a result, we observed leakage since the t-statistic was much larger than the threshold 4.5 when the seed value was fixed for every encryption, or when the update of fresh random bits was stopped in the encryption (Figs. 4a, 4b, and 4c). On the other hand, neither the first- nor second-order t-test statistics surpassed the threshold 4.5 with up to 1 million power traces when the seed value was updated for every encryption and fresh random bits were updated for every clock cycle (Fig. 4d).

TABLE III: Correspondence between experimental conditions and experimental results

		Seed value: fixed	Seed value: updated
Random bits / clock cycle	fixed	Fig. 4a	Fig. 4c
	updated	Fig. 4b	Fig. 4d

V. Discussion

This section discusses the causes of information leakage from two perspectives. The first is the cause of information leakage when the update of fresh random bits is stopped in the encryption; The second is the cause of information leakage when the seed value is fixed for every encryption.

A. Random bits per clock cycle: Fixed

We focus on the register after the refreshing process of the stage $i\,(i = 2, 3, 4, 5)$ in the S-box shown in Fig. 1. Power consumption is proportional to the number of transition bits in the register. Therefore, it depends on the Hamming distance between the $n^{\mathrm{th}}\,(1 \le n < 16)$ and $(n+1)^{\mathrm{th}}$ byte after the refreshing process at the stage i. Fresh random bits for the refreshing are not updated for every clock cycle, so the random bits used in the stage i are the same throughout the encryption process.

Here, the values of the n^{th} and $(n+1)^{\mathrm{th}}$ byte after the refreshing process at the stage i are as

$$
\begin{aligned}
d_1^n &= s_1^n + r_1 + r_2, & d_6^n &= s_6^n + r_6 + r_7,\\
d_2^n &= s_2^n + r_2 + r_3, & d_7^n &= s_7^n + r_7 + r_8,\\
d_3^n &= s_3^n + r_3 + r_4, & d_8^n &= s_8^n + r_8 + r_9,\\
d_4^n &= s_4^n + r_4 + r_5, & d_9^n &= s_9^n + r_9 + r_1.\\
d_5^n &= s_5^n + r_5 + r_6,
\end{aligned}
$$

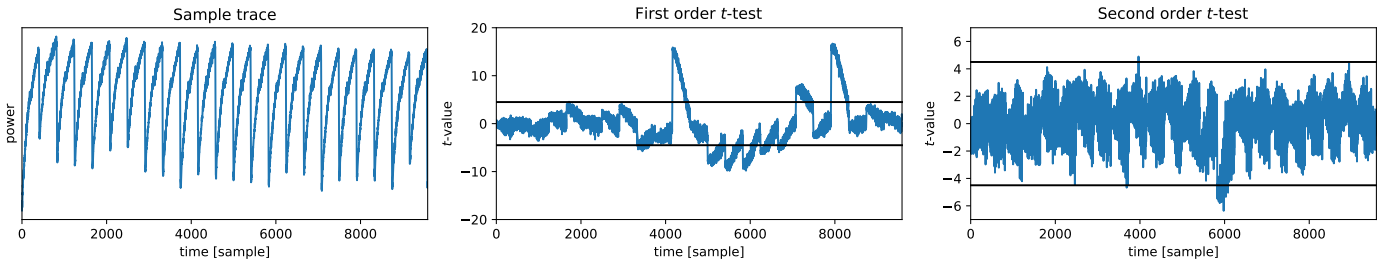
$$
\begin{aligned}
d_1^{n+1} &= s_1^{n+1} + r_1 + r_2, & d_6^{n+1} &= s_6^{n+1} + r_6 + r_7,\\
d_2^{n+1} &= s_2^{n+1} + r_2 + r_3, & d_7^{n+1} &= s_7^{n+1} + r_7 + r_8,\\
d_3^{n+1} &= s_3^{n+1} + r_3 + r_4, & d_8^{n+1} &= s_8^{n+1} + r_8 + r_9,\\
d_4^{n+1} &= s_4^{n+1} + r_4 + r_5, & d_9^{n+1} &= s_9^{n+1} + r_9 + r_1.\\
d_5^{n+1} &= s_5^{n+1} + r_5 + r_6,
\end{aligned}
$$

where $d_k\,(1 \le k \le 9)$ are values stored in the register, s_k are values before the refreshing process, r_k are random numbers, and the superscript is a byte number.

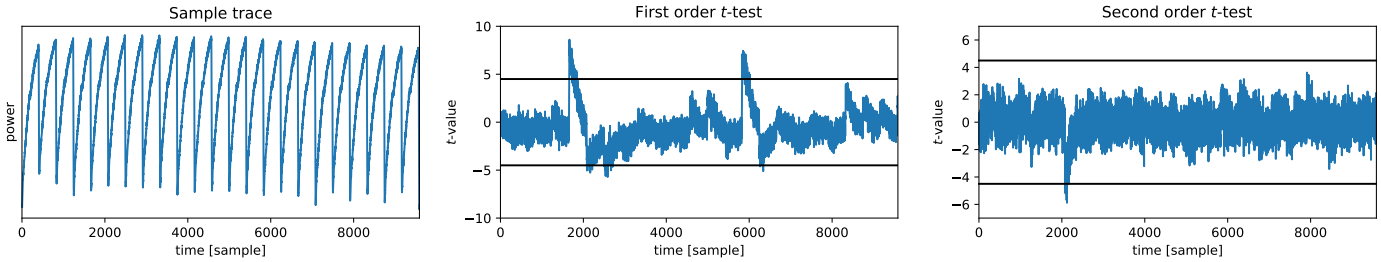
The Hamming distance of d_1^n and d_1^{n+1} is as

$$
\begin{aligned}
\mathrm{HD}(d_1^n,\, d_1^{n+1}) &= \mathrm{HD}(s_1^n + r_1 + r_2,\, s_1^{n+1} + r_1 + r_2)\\
&= \mathrm{HD}(s_1^n,\ s_1^{n+1}),
\end{aligned}
$$

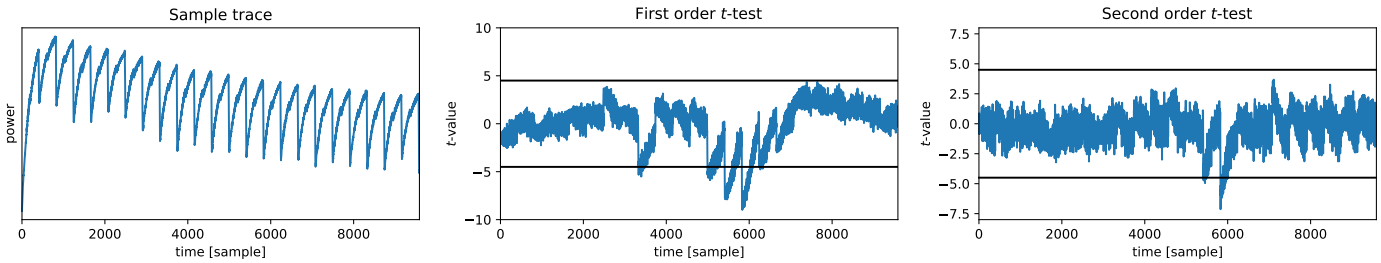
where $\mathrm{HD}(\cdot, \cdot)$ is a function to calculate the Hamming distance. Therefore, the Hamming distance of d_1^n and d_1^{n+1}



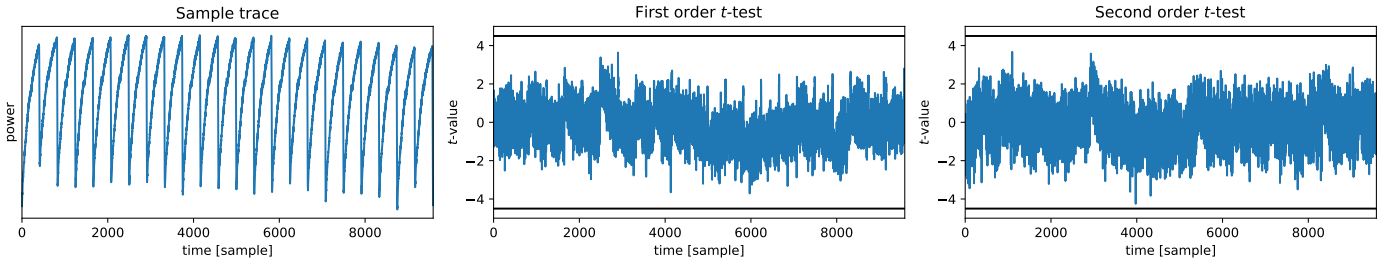
(a) Seed value: **fixed** and Random bits per clock cycle: **fixed** (862K traces)



(b) Seed value: **fixed** and Random bits per clock cycle: **updated** (300K traces)



(c) Seed value: **updated** and Random bits per clock cycle: **fixed** (1M traces)



(d) Seed value: **updated** and Random bits per clock cycle: **updated** (1M traces)

Fig. 4: Fixed-vs-random plaintexts t-test result

is equal to the Hamming distance of s_1^n and s_1^{n+1}. The same is true for other d_k. The sharing $\{s_1^n, s_2^n, \ldots, s_9^n\}$ and $\{s_1^{n+1}, s_2^{n+1}, \ldots, s_9^{n+1}\}$ do not satisfy uniformity because they are not re-masked after operations that are not bijective. Thus, the set of Hamming distances has a biased distribution that does not follow a Gaussian distribution, which affects the power consumption and causes leakage.

B. Seed Value: Fixed

Consider the case where the seed value given to the PRNG is fixed for every encryption and random bits are updated for every clock cycle. In this case, fresh random bits for the refreshing process used in the stage i are different values in the encryption process. Here, the values of the n^{th} and $(n+1)^{\text{th}}$ byte after the refreshing process at the stage i are as

$$
\begin{aligned}
d_1^n &= s_1^n + r_1^n + r_2^n, & d_6^n &= s_6^n + r_6^n + r_7^n,\\
d_2^n &= s_2^n + r_2^n + r_3^n, & d_7^n &= s_7^n + r_7^n + r_8^n,\\
d_3^n &= s_3^n + r_3^n + r_4^n, & d_8^n &= s_8^n + r_8^n + r_9^n,\\
d_4^n &= s_4^n + r_4^n + r_5^n, & d_9^n &= s_9^n + r_9^n + r_1^n.\\
d_5^n &= s_5^n + r_5^n + r_6^n,
\end{aligned}
$$

$$
\begin{aligned}
d_1^{n+1} &= s_1^{n+1} + r_1^{n+1} + r_2^{n+1}, \quad d_6^{n+1} = s_6^{n+1} + r_6^{n+1} + r_7^{n+1},\\
d_2^{n+1} &= s_2^{n+1} + r_2^{n+1} + r_3^{n+1}, \quad d_7^{n+1} = s_7^{n+1} + r_7^{n+1} + r_8^{n+1},\\
d_3^{n+1} &= s_3^{n+1} + r_3^{n+1} + r_4^{n+1}, \quad d_8^{n+1} = s_8^{n+1} + r_8^{n+1} + r_9^{n+1},\\
d_4^{n+1} &= s_4^{n+1} + r_4^{n+1} + r_5^{n+1}, \quad d_9^{n+1} = s_9^{n+1} + r_9^{n+1} + r_1^{n+1}.\\
d_5^{n+1} &= s_5^{n+1} + r_5^{n+1} + r_6^{n+1},
\end{aligned}
$$

The Hamming distance of d_1^n and d_1^{n+1} is as

$$\mathrm{HD}(d_1^n,\, d_1^{n+1}) = \mathrm{HD}(s_1^n + r_1^n + r_2^n,\, s_1^{n+1} + r_1^{n+1} + r_2^{n+1}).$$

The random numbers r_1^n, r_2^n, r_1^{n+1}, and r_2^{n+1} are each constant since the seed value is fixed for each encryption process. The same is true for other d_k. The sharing $\{d_1^n, d_2^n, \ldots, d_9^n\}$ and $\{d_1^{n+1}, d_2^{n+1}, \ldots, d_9^{n+1}\}$ do not satisfy uniformity because r_k^n and r_k^{n+1} are not random values but constant values. Similarly, even if the seed value is fixed for each encryption process, the set of Hamming distances has a skewed distribution that does not follow a Gaussian distribution, affecting power consumption and causing leakage.

VI. Conclusions and Future Work

In this paper, we evaluated the security impact of randomness in the refreshing process in cryptographic hardware. Our study involved the implementation of M&M AES hardware with second-order security on a Xilinx Spartan-6 FPGA. We assessed leakage using Welch's t-test, considering scenarios where the seed value was either fixed or updated for every encryption and where random bits were either fixed or updated for every clock cycle. In the evaluation, the t-statistic exceeded the threshold 4.5 when the same seed value was repeatedly applied to the PRNG for each encryption, conclusively confirming the presence of leakage. Additionally, we observed leakage when the fresh random bits used in each stage of the S-box were not updated for every clock cycle. This indicates that side-channel leakage still exists when the quality of randomness utilized for re-masking within the S-box is compromised even if TI prevents side-channel attacks.

In the future, we will evaluate PRNG algorithms' impact on cryptographic hardware security. Specifically, we will switch from Keccak to XSadd generator [19], which has passed the bigcrush test of TestU01 [20], and examine the side-channel leakage. In addition, we will use an ASIC with $28\,\mathrm{nm}$ CMOS process technology instead of the FPGA for evaluation boards to investigate the leakage in the same way.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers JP20H00590 and JP23H03393.

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO' 96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*. Springer, 1996, pp. 104–113.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*. Springer, 1999, pp. 388–397.

[3] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*. Springer, 2004, pp. 16–29.

[4] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001 Cannes, France, September 19–21, 2001 Proceedings*. Springer, 2001, pp. 200–210.

[5] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*. Springer, 2003, pp. 463–481.

[6] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*. Springer, 1997, pp. 513–525.

[7] L. D. Meyer, V. Arribas, S. Nikova, V. Nikov, and V. Rijmen, "M&M: Masks and macs against physical attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 25–50, 2019.

[8] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *ICICS*, vol. 4307. Springer, 2006, pp. 529–545.

[9] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*. Springer, 2015, pp. 764–783.

[10] NIST, "Advanced Encryption Standard (AES) FIPS Publication 197," 2001, https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf.

[11] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking AES with d + 1 shares in hardware," in *Cryptographic Hardware and Embedded Systems–CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Springer, 2016, pp. 194–212.

[12] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*. Springer, 2014, pp. 326–343.

[13] D. Canright, "A very compact S-box for AES," in *Ches*, vol. 3659. Springer, 2005, pp. 441–455.

[14] J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, P. Rohatgi *et al.*, "Test vector leakage assessment (TVLA) methodology in practice," in *International Cryptographic Module Conference*, vol. 20, 2013.

[15] T. Schneider and A. Moradi, "Leakage assessment methodology: A clear roadmap for side-channel evaluations," in *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*. Springer, 2015, pp. 495–513.

[16] "SAKURA hardware security project," http://satoh.cs.uec.ac.jp/SAKURA/index.html.

[17] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer, 2013, pp. 313–314.

[18] B. Guido, D. Joan, P. Michaël, and V. Gilles, "Cryptographic sponge functions," 2011.

[19] M. Saito and M. Matsumoto, "XSadd (Version 1.1).(25 March 2014)," 2014, http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/XSADD/.

[20] P. L'ecuyer and R. Simard, "TestU01: AC library for empirical testing of random number generators," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, pp. 1–40, 2007.